Classification of Combinatorial Objects

User's Guide
Build Number 3189

Anton Betten

February 25, 2024

Contents

1 Introduction **7**
- 1.1 What is Orbiter . . . 7
- 1.2 Installing Orbiter for the First Time . . . 8
- 1.3 Updating Orbiter . . . 10

2 Getting Started **11**
- 2.1 Running and Installing Orbiter . . . 11
- 2.2 The Orbiter Session . . . 15
- 2.3 Makefiles and Shell Scripts . . . 17
- 2.4 Objects and Activities . . . 19
- 2.5 Mathematical Data . . . 24
- 2.6 Set Builder . . . 26
- 2.7 Vector Builders . . . 27
- 2.8 Vector of Group Elements . . . 32
- 2.9 Symbolic Objects . . . 34

3 Basic Algebra **53**
- 3.1 Basic Algebra and Number Theory . . . 53
- 3.2 Finite Fields . . . 65
- 3.3 Extension Fields . . . 71
- 3.4 Linear Algebra Over Finite Fields . . . 83
- 3.5 Advanced Topics in Finite Fields . . . 89
- 3.6 Basic Ring Theory . . . 97

4 Geometry **99**
- 4.1 Finite Projective Spaces . . . 99
- 4.2 Indexing Points and Lines . . . 105
- 4.3 Incidence Matrices . . . 109
- 4.4 The Grassmannian . . . 115
- 4.5 Algebraic Sets . . . 118
- 4.6 The Klein Quadric and the Plücker Map . . . 121
- 4.7 Orthogonal Spaces . . . 122
- 4.8 Hermitian Varieties . . . 130
- 4.9 Advanced Topics . . . 132
- 4.10 Geometric Objects . . . 143

5 Ring Theory **151**
- 5.1 Polynomials Over Finite Fields . . . 151
- 5.2 Multivariate Polynomial Rings . . . 159
- 5.3 Algebraic Geometry . . . 170

6 Group Theory **175**
6.1 Generic Permutation Groups . . . 175
6.2 Matrix Groups . . . 185
6.3 Subgroups . . . 205
6.4 New Actions from Old . . . 218
6.5 Group Theoretic Activities . . . 228
6.6 Group Theoretic Activities Based on Magma . . . 238
6.7 Group Theoretic Activities Based on GAP . . . 247
6.8 Linear Groups, Advanced Topics . . . 253

7 Orbit Algorithms **257**
7.1 Schreier Trees . . . 257
7.2 Poset Classification . . . 264
7.3 Orbits on Subsets . . . 269
7.4 Orbits on Subspaces . . . 273
7.5 Orbits on Set-Partitions . . . 276
7.6 Arcs and Caps in Projective Spaces . . . 278
7.7 Cubic Curves . . . 282

8 Cubic Surfaces and Quartic Curves **287**
8.1 Generalities . . . 287
8.2 Creating Cubic Surfaces . . . 290
8.3 Cubic Surface Activities . . . 296
8.4 Classification of Cubic Surfaces . . . 301
8.5 Quartic Curves . . . 316
8.6 Classification of Quartic Curves . . . 328
8.7 Interface to GAP . . . 332

9 Applications **335**
9.1 Number Theory . . . 335
9.2 Representation Theory . . . 336
9.3 Cryptography . . . 338

10 Coding Theory **347**
10.1 Introduction . . . 347
10.2 Linear Codes . . . 353
10.3 Golay Codes . . . 364
10.4 CRC Codes . . . 365
10.5 Reed-Muller Codes . . . 370
10.6 BCH Codes . . . 377
10.7 Reed-Solomon Codes . . . 384
10.8 Twisted Tensor Product Codes . . . 389
10.9 Bounds . . . 391
10.10Classification of Optimal Linear Codes . . . 395

11 Combinatorics **397**
11.1 Introduction . . . 397
11.2 Combinatorial Objects . . . 404
11.3 Combinatorial Linear Spaces . . . 412
11.4 Classification of Configurations and Geometries . . . 415
11.5 Tactical Decompositions . . . 424

12 Graph Theory **427**
12.1 Creating Graphs . . . 427
12.2 Graph Theoretic Activities . . . 435
12.3 Classification of Graphs and Tournaments . . . 440

13 Design Theory **445**
13.1 Creating Designs . . . 445
13.2 Assuming Symmetry . . . 453
13.3 Design Theory – Large Sets . . . 458
13.4 Design Theory – Delandtsheer-Doyen . . . 461

14 Finite Geometry **465**
14.1 Spreads . . . 465
14.2 Translation Planes . . . 480
14.3 Packings . . . 485
14.4 BLT-Sets . . . 490

15 Computer Science Primitives **499**
15.1 Clique Finding . . . 499
15.2 Rainbow Cliques . . . 503
15.3 Diophantine Systems . . . 506

16 Canonical Forms with Nauty **511**
16.1 Overview of Canonical Forms . . . 511
16.2 Canonical Forms of Objects in Projective Space . . . 513
16.3 Canonical Forms of Incidence Geometries . . . 522
16.4 Canonical Forms of Objects from Design Theory . . . 535
16.5 Canonical Forms of Linear Codes . . . 539
16.6 Canonical Forms of General Codes . . . 546
16.7 Canonical Forms of Graphs . . . 547
16.8 Canonical Forms of Quartic Curves . . . 552

17 Interfaces **555**
17.1 Graphical Output . . . 555
17.2 The Povray Interface . . . 559
17.3 Creating Animations . . . 572
17.4 Continuous Function Plotter . . . 573
17.5 The Gnuplot Interface . . . 577

18 Mathematical Data in Orbiter **581**
18.1 Cubic Surfaces . . . 581
18.2 BLT-Sets . . . 583

19 Miscellaneous **585**
19.1 Miscellaneous . . . 585
19.2 Limitations . . . 589

20 Orbiter on Windows **593**
20.1 Using Windows Subsystem Linux . . . 593

21 Templates **603**
21.1 Templates . . . 603

A Data on Cubic Surfaces **607**
A.1 Data on Cubic Surfaces . . . 607

B The Class Library **629**
B.1 Classes and Namespaces 629

C The Makefile **641**
C.1 The Makefile 641
References 1044

Chapter 1

Introduction

1.1 What is Orbiter

This is the user's guide of the computer algebra system Orbiter. For the most recent version of this guide, please visit

https://www.math.colostate.edu/~betten/orbiter/users_guide.pdf

For the github repository, see

https://github.com/abetten/orbiter

Orbiter is a computer algebra system for the classification of combinatorial objects. Orbiter contributes to the knowledge base of combinatorial structures, and to provide useful tools to investigate structures from various points of view, including their symmetry properties. For background on Orbiter, see [10].

Orbiter is open source, and can be downloaded and installed from github. A C++ compiler environement similar to gnu gcc and make are required. Compiler tools like bison and flex are required also. Latex is recommended.

1.2 Installing Orbiter for the First Time

How to install or update Orbiter? Here is the procedure to install Orbiter for the first time. The steps differ, depending on your operating system. On Linux/unix based systems, you will utilize a terminal window. To install Orbiter, go to github

```
abetten/orbiter
```

Click on "Code" and copy paste the address. Then type

```
git clone XXX
```

Where XXX is to be replaced with the address that you copied (simply use CTRL-P to paste).
Once the command finishes, you need to check that the prerequisites are satisfied. The two programs that need to be available are called bison and flex. You need to check if these programs are avaliable and, if so, what version number you have. Type

```
bison --version
```

If you get a message `bison not found` it means that you don't have bison. In this case you need to install bison. Otherwise, check out the version number. Orbiter requires anything above 3.8.2. If your version number is below that, you need to upgrade your bison software. Regarding flex, type

```
flex --version
```

If you get a message `flex not found` it means that you don't have flex. In this case you need to install flex. Otherwise, check out the version number. Orbiter requires anything above 2.6.4. If your version number is below that, you need to upgrade your flex software.

Next, we will show how to install bison and flex. After that, we will discuss how to upgrade either of them. Our description is taylored to a Macintosh system, using the package manager "brew." On Linux, you to utilize a different package manager.

On a Macintosh, using cshell (csh), you can install bison using "brew." Type

```
brew install bison
```

The install will go to

```
/opt/homebrew/Cellar
```

Go there and check whether a directory bison exists. If so, do

```
cd bison
```

and

```
ls
```

and note the version number. You may still have an old version of bison somewhere else. In order to activate the new bison and nor the old, you will have to change the path in your shell. For csh, make sure you add the line

```
set path = ( /opt/homebrew/Cellar/bison/3.8.2/bin $path )
```

Here, the `3.8.2.` part may has to be changed depending on your version of bison.

You can install flex using brew. Type

```
brew install flex
```

The install will go to

```
/opt/homebrew/Cellar
```

Go there and check whether a directory flex exists. If so, do

```
cd flex
```

and

```
ls
```

and note the version number. Like with bison earlier, you may still have an old version of flex somewhere else. In order to activate the new flex and not the old, you will have to change the path in your shell. For csh, make sure you add the line

```
set path = ( /opt/homebrew/Cellar/bison/2.6.4_2/bin $path )
```

Here, the `2.6.4_2` part may has to be changed depending on your version of bison.

Finally, you have to add

```
set LDFLAGS = ( -L/usr/local/Cellar/flex/2.6.4_2/lib -L/usr/local/Cellar/bison/3.8.2/lib )
```

to your `~/.cshrc` Here, the version numbers may have to be changed to match those that you used before.

After these changes, you need to read the `~/.cshrc` file again and rehash. Type

```
read ~/.cshrc
```

and

```
rehash
```

If you type `bison --version`, you should get the correct version number of bison. Likewise, if you type `flex --version`, you should get the correct version number of flex.

You are now ready to compile Orbiter. For this purpose, you need to enter the orbiter home directory. Assuming that you are still in the directory where you did git clone, type

```
cd orbiter
```

and then

```
make
```

If your computer CPU has multiple cores, you can try

```
make -j 8
```

The number 8 can be adjusted to the number of cores that you have. This multitasking reduces the time it takes to compile Orbiter. Once done, you should have an orbiter executable in the directory

```
orbiter/src/apps/orbiter
```

The executable is called

```
orbiter.out
```

Note that you are already in orbiter, so you could do

```
ls src/apps/orbiter
```

to check if `orbiter.out` appears. In either case, you can do

```
cd ..
```

to leave the orbiter branch. You should only go back in the Orbiter branch if you want to update Orbiter. Otherwise, you should not be there. In particular you should not change any files in this branch, as this would cause problems in the update.

Finally, if you have to update bison or flex, you can type

```
brew upgrade bison
```

or

```
brew upgrade flex
```

1.3 Updating Orbiter

To update Orbiter, enter the `orbiter` directory. This is the directory that was created by `git clone`.

```
cd orbiter
```

Now type

```
git pull
```

to update the Orbiter sources. If you get an error message to the effect that you should stash something first, it means that you have made changes inside the orbiter directory. Most likely, you edited a file in the Orbiter directory tree by chance. In this case, it is best to let go of these changes by issuing

```
git reset --hard
```

and then doing

```
git pull
```

again. Then type

```
make clean
```

and

```
make
```

or

```
make -j 8
```

for parallel compile (if necessary, change the 8). You should get a new Orbiter executable.

Chapter 2

Getting Started

2.1 Running and Installing Orbiter

There are two ways to run Orbiter: Native and Docker. Native means that Orbiter is compiled from scratch, using the source code from the github repository (cf. [11]). Docker [27] is a system to run preconfigured software in an encapsulated way on various platform, including Windows. We describe using Orbiter through unix *makefiles*, which are run through the tool *make* (cf. [30]). This is a software tool that allows collecting short command snipplets in the form of text files that can easily be handled. However, the conventions in the tool involve some subtleties regarding the use of whitespace, which can cause problems to novice users. We will point out possible pitfalls along the way. Note that it is not necessary to use makefiles. Another possibility would be to use shell scripts. Ultimately, it would be possible to type out all commands into a terminal window. This could be a little tedious though, considering the fact that most Orbiter commands expect lengthy parameters from the command line.

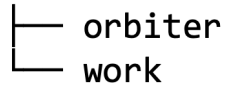
Let us start by discussing how to run Orbiter as a native application. To do so, a unix-like compile environment is required, including a modern C++ compiler and the tools *git* and *make*. Windows users may need to install Cygwin [23]. The following steps are required: Using git, *clone* the repository. Then enter the directory *orbiter* and type

```
make
```

Once compiled, the Orbiter executable is

```
src/apps/orbiter/orbiter.out
```

within the Orbiter directory. We then recommend creating a separate work directory *not within the orbiter directory*. For the following, we assume the following directory tree structure:



In the work directory, create a small makefile like so:

```
OP=../orbiter
ORBITER_PATH=$(OP)/src/apps/orbiter/

test:
  $(ORBITER_PATH)orbiter.out
```

Different directory structures can be accommodated by changing the first line. Next, typing

```
make test
```

within the work directory will invoke Orbiter. Here, *test* is the makefile "target." The makefile target must appear in the makefile. In the example above, the block

```
test:
  $(ORBITER_PATH)orbiter.out
```

is the makefile target "test." It is important that the indentation after the makefile target is done using tab characters (no spaces). There can be multiple targets in one makefile, as long as they are separated by an empty line. for more information about the syntax of makefiles, see [30].

A second way to run Orbiter is through Docker [27]. This does not require a compile environment. However, it comes at a small performance cost when running Orbiter commands that are computationally heavy. Orbiter has already been precompiled (by the Orbiter developer) into an *image*, which is a completely self-sustained copy of a unix-environment that can run by the user under the docker front-end. The image is stored on a docker server under the name `abetten/orbiter`. Docker will receive the name of the image from the command line, pull a local copy of the image, and run the image in an encapsulated environment called a *container*. A copy of the image is stored locally, so that subsequent calls to Orbiter can be satisfied using the local copy, which increases turnaround speed. For instance, the following bare-bones makefile sets up Orbiter for use through Docker:

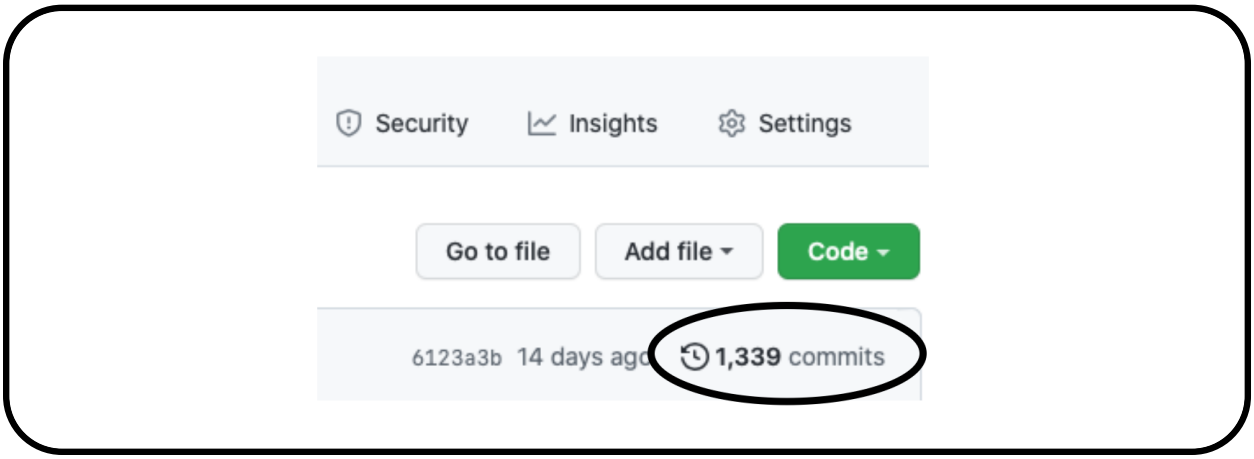
```
DOCKER_OPTIONS=run -it \
  --volume ${PWD}:/mnt -w \
  /mnt abetten/orbiter
ORBITER_PATH=docker $(DOCKER_OPTIONS)

test:
  $(ORBITER_PATH)orbiter.out
```

In this file, there is a space character in line three after `abetten/orbiter` which is important (and unfortunately cannot be seen). By typing

```
make test
```

into a terminal window, Docker starts up and pulls a copy of Orbiter to the local machine, which is then executed. Orbiter will start up, produce a few messages and then shut down. Interestingly, this will work on a Windows machine also (using *supershell* as terminal). The *make* command is passed through to the container, which contains the unix-like software environment, including make. The associated *makefile* resides on the local machine, as do input and output files.

Orbiter comes with a version numbering system called a build number. The build number should match the commit number on the github tree, shown below:



When Orbiter starts up, the build number is displayed. In order to update to a more recent version of Orbiter, Docker needs to be instructed to discard the local image. To do so, the command

```
docker rmi -f abetten/orbiter
```

is used. After that, any new invocation of Orbiter will cause Docker to pull the latest Orbiter *image* from the Docker repository. It is convenient to combine the Docker and Native compile environment into a single makefile and use the comment symbol (hash #) to switch between the two modes (the line numbers are not part of the file).

Example 1

```
ORBITER_PATH=~/orbiter
```

In the code excerpts, a tabulator character is shown as a little triangle pointing to the right. Also, the backslash signs are used to break long lines. Please make sure that there are no spaces after the backslash sign.

For use with Docker, the installation of Orbiter requires the following steps:

(a) Install Docker from `www.docker.com`, including the Linux kernel.

(b) Open a terminal window (for instance PowerShell on Windows).

(c) Type

```
docker run -it --volume ${PWD}:/mnt -w /mnt abetten/orbiter orbiter.out
```

Docker will start downloading Orbiter as a container. This can take a while, depending on the Internet speed. After that, an Orbiter session is started. As no specific commands were given, Orbiter simply starts up and quits. Once it has been downloaded, Docker will recycle the copy of orbiter and a download is no longer required. However, once Orbiter updates, Docker will update the local copy of Orbiter as well.

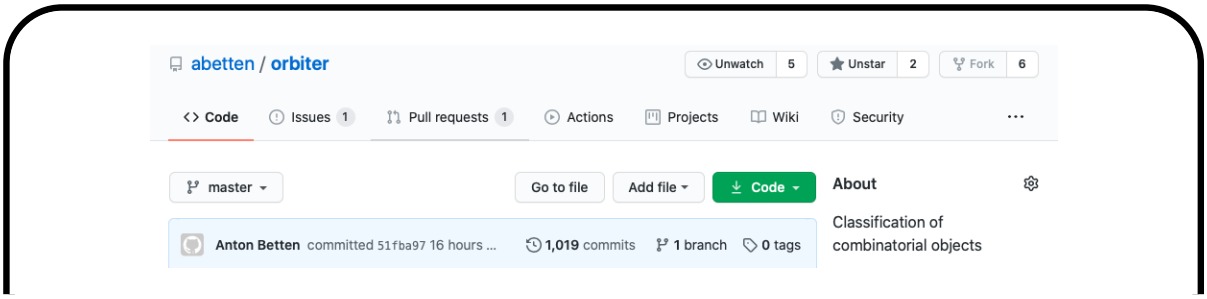
To use Orbiter in native mode, the sources have to be installed and compiled. This is more complicated on Windows machines, because the unix environment is missing. Windows users can use cygwin to install Orbiter. The installation of Orbiter requires the following steps:

(a) Ensure that `git` and the C++ development suite are installed (`gnuc` and `make`). Windows users may have to install `cygwin` (plus the extra packages `git`, `make`, `gnuc`). Macintosh users may have to install the xcode development tools from the appstore (it is free). Linux users may have to install the development packages. Orbiter often produces latex reports. In order to compile these files, make sure you have latex installed

(b) Clone the Orbiter source tree from github (abetten/orbiter). The commands are:

```
git clone <github-orbiter-path>
```

where `<github-orbiter-path>` has to be replaced by the actual address provided by github. To obtain this path, find Orbiter on github, then click on the green box that says "Code" and copy the address into the clipboard by clicking the clipboard symbol, see below:



Back in the terminal, paste this text after the `git clone` command. After cloning is complete, enter the orbiter directory (`cd orbiter`).

(c) Issue the following commands to compile Orbiter:

```
make
make install
```

These two commands compile the Orbiter source tree and copy the executables to the subdirectory bin inside the Orbiter source tree. The orbiter executable is called `orbiter.out`.

2.2 The Orbiter Session

Orbiter is a command line program. There is no graphical user interface. The Orbiter executable `orbiter.out` takes commands from the command line for execution. Once done, Orbiter terminates and the prompt is back in the terminal. Command sequences can be quite long, so it is recommended to store the command sequence in either a makefile or in a shell script. This way, long commands can be executed by short, customizable names. For more on thit topic, see Section 2.3.

Let us take a closer look at an Orbiter session. Any orbiter session is invoked through the orbiter command `orbiter.out`, which is the name of the executable. Unless the executable resides in a directory contained in the search path of the shell, a path must be given. Several options apply to the orbiter session. They are listed in Table 2.1.
To run Orbiter, we can use a makefile entry like this:

Example 2

```
test_orbiter_session:
▷  $(ORBITER)
```

Once started, the Orbiter session will produce a short welcome message:

```
Welcome to Orbiter!  Your build number is 1081.
A user's guide is available here:
https://www.math.colostate.edu/~betten/orbiter/users_guide.pdf
The sources are available here:
https://github.com/abetten/orbiter
An example makefile with many commands from the user's guide is here:
https://github.com/abetten/orbiter/tree/master/examples/users_guide/makefile
Orbiter session finished.
User time: 0:00
```

The build number is the version number of the Orbiter software, as defined by the number of submits to the Git repository. Higher numbers mean more recent versions. After this message, Orbiter will start parsing the command line arguments. Once this is done, the session will execute these commands. At the end of the session, a short message is given that specifies the processor time used up by the session.

Orbiter session commands

Command	Arguments	Purpose
-v	v	Set verbosity to v. Larger values of v lead to more text output. $v = 0$ gives minimal to no output.
-draw_options	options	Set options for drawing.
-draw_incidence_ structure_description	options	Set options for drawing of incidence structures.
-list_arguments		Prints the command line arguments.
-seed	s	Seed the pseudo random number generator with the integer value s.
-memory_debug		Turn on dynamic memory debugging.
-override_polynomial	poly	Set the override polynomial for finite fields to poly.
-orbiter_path	p	Set the orbiter path to p. This is useful in case the Orbiter session has to clone or fork new Orbiter sessions. In most cases, the orbiter path will end with a forward slash "/."
-magma_path	p	Set the magma path to p. This is useful in case the Orbiter session has to create a magma process.
-fork	L M s u d	Fork new Orbiter sessions in parallel. The new sessions will be indexed by the values $i = s + kd$, where $k = 0, 1, \ldots$ subject to $i < u$. Every occurence of the string L in the argument list is replaced by the resulting value of the loop variable i. The forked process will write to a file. The filename asis from M using by replacing %d with the value of i (similar to a printf command in C programming). All of the command line arguments after the fork command are passed through to the new Orbiter session, with all arguments L replaced by the integer value of the loop counter. The number of Orbiter sessions forked is $(u - s)/d$. The orbiter path is considered when starting the forked sessions.

Table 2.1: Orbiter session commands

2.3 Makefiles and Shell Scripts

Orbiter is a command line driven system. There is no graphical user interface. This means that commands are typed into a terminal, and executed by the operating system. In this mode of operation, Orbiter is just like any other program installed on the computer. This also means that Orbiter can be mixed with other applications, using files to share data between the processes.

The command line is entered into an application that is called *Terminal* (or *SuperShell* in Windows). Orbiter is called from the command line, and command options are given to instruct Orbiter what to do. The process that calls orbiter is the shell. There are different types of shells, but they all provide the necessary interface to allow the user to start jobs and maintain files. Shells can be programmed by means of shell scripts. Programming by mans of shell scripts is called scripting. Orbiter can be programmed using shell scripting.

Make is a software tool that allows to execute small command sequences. Commands can trigger other commands, based on dependencies. The dependencies are tied to the existence of files, which can be named. If the file does not exist or is outdate, make triggers a command whose purpose is to create that file. The dependencies are called rules. The makefile consists of rules and command sequences to be invoked based on the dependencies and the commands issued. Many commands can be collected in one makefile. Because of the convenience offered by makefiles, this user's guide will rely on a the make / makefile tool. The makefile associated with this user's guide is listed in full in Section C.1. It would also be possible to define shell scripts for each of the commands.

Make also allows to use variables, which are used by means of text substitution. A variable is defined as

```
A="I am a variable"
```

and used anywhere later using the

```
$(A)
```

syntax. Rules are defined using the following syntax

```
Label:
  Do something
```

Here, label is the name of the rule, and `Do something` is the code that is exacuted whenever make is called with the given label in the command line. For instance

```
make Label
```

will execute `Do something`. The shell will take the command and peel off the first word, which is `Do`. It will then search the system for a command called `Do`. Of course, this will result in an error because there is no command called `Do`. The remaining piece of the command line, i.e. `something` is considered as an argument to the command. For instance, suppose we have a orbiter command with several options, say

```
orbiter.out -v 3 -define F -finite_field -q 16 -end \
  -with F -do -finite_field_activity -cheat_sheet_GF -end
```

The purpose of this command is to produce a file called

```
GF_16.tex
```

which can then be processed through latex to give the report. Observe that the command is quite long, and stretches over two lines. The backslash at the end of the first line indicates that the command continues on to the next line. Using make, we can assign a label to this command. Suppose we want to call this command `F_16`. We can create a makefile like this:

Example 3

```
F_16:
▷  $(ORBITER) -v 10 \
▷  ▷  -define F -finite_field -q 16 -compute_related_fields -end \
▷  ▷  -with F -do -finite_field_activity -cheat_sheet_GF -end
▷  pdflatex GF_16.tex
▷  $(OPEN) GF_16.pdf
```

With this file present, type the terminal command `make F_16` to execute the two line Orbiter command. Windows users can use `SuperShell`. The program `make` will look for the file `makefile` in the current directory. Once found, it will search for the label `F_16` in it and execute the commands beneath it. The given commands will invoke Orbiter and produce the `GF_16.tex` containing the desired report. If we wanted to do some other Orbiter command, we could edit the makefile. We would also have a sequence of commands listed in the same target. In this case, makefile will process these commands one after the other.

Makefiles are somewhat picky when it comes to whitespacing. The command sequence needs to be indented using tab symbols. Leading spaces will cause error messages. Also, there should be no whitespace after the trailing backslash symbol. Some editors can display whitespace characters. This can be helpful when working with makefiles.

A sample makefile with all of the commands discussed in this user's guide is distributed with Orbiter (in the examples directory). The file is reproduced in Section C.1. It is advised to copy the example makefile from the orbiter tree to a location outside the orbiter distribution directory (otherwise, git update will cause error messages). It is also fine to create a new custom makefile, considering the remarks about `ORBITER_PATH` below.

One difficulty in installing Orbiter is the path of installation. In the sample makefile, there is a makefile variable called `ORBITER_PATH` which contains the path to the orbiter executable `orbiter.exe`. Depending on the local installation of orbiter, the makefile variable needs to be changed accordingly. The actual command to run the `F_16` example is as follows:

```
F_16:
  $(ORBITER_PATH)orbiter.out -v 3 -define F -finite_field -q 16 -end \
    -with F -do -finite_field_activity -cheat_sheet_GF -end
```

The orbiter installation directory `orbiter` and a second directory called `work` should be next to each other. The orbiter example makefile should be copied into the `work` directory. The top of the file should contain the line

```
MY_PATH=../orbiter
```

This will set `ORBITER_PATH` to point to the correct location of the orbiter executable. inside the `work` directory, any of the commands listed in this guide will function correctly. Another possibility is to install `orbiter.out` in a central location, where it can be found by the shell. In this case, we should change the line

```
ORBITER_PATH=$(MY_PATH)/src/apps/orbiter/
```

to

```
ORBITER_PATH=
```

in the makefile.

2.4 Objects and Activities

The majority of work in Orbiter is done by means of objects and activities. This follows the object oriented paradigm of programming, realized in the C++ programming language, which is the language in which Orbiter is written. Objects hold data and can perform tasks, which in Orbiter are called activities. This leaves two questions:

1. How are objects created?
2. What activities exist?

There are many different types of objects, and each has specific requirements. Also, the types of activities depend on the types of objects. This user's guide will answer the two questions one-by-one, by going over the different types of objects that exist.

The syntax to create an object is

`-define` *LABEL KEYWORD PARAMETERS* `-end`

Here, *LABEL* is any label under which the object is stored in the symbol table. Any object with the same label already in the symbol table will be overwritten. The *KEYWORD* can be any of the commands in Tables 2.2-2.3. The *PARAMETERS* depend on the type of object created. the command `-end` is necessary to terminate the definition command. For more details on the objects that exist, see the appropriate section listed in the table.
For instance, the command

Example 4

```
object_F_2:
▷  $(ORBITER) -v 3 -define F -finite_field -q 2 -end
```

creates a finite field object F for the field $\mathbb{F}_2$, the field with two elements. For more details, see Section 3.2. The command

Example 5

```
object_PG_3_2:
▷  $(ORBITER) \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end
```

creates the same finite field $\mathbb{F}_2$ and uses it to construct $\mathrm{PG}(3,2)$. The `-projective_space` command requires additional options to set the dimension n and the field $\mathbb{F}_q$ in $\mathrm{PG}(n,q)$. The `-n` command sets the dimension n. The `-field` command can be used to specify a praticular field. The `-q` command can be used to create a generic field $\mathbb{F}_q$.

In order to do something with an object, we need to invoke an *activity*. To select an object for an activity, the

`-with` *LABEL* `-do` *DESCRIPTION* `-end`

command sequence is used. Here, *LABEL* is the name under which the object is registered in the symbol table. *DESCRIPTION* is the activity that should be a applied. Some activities require more than one object, in which case the syntax

`-with` *LABEL1* `-and` *LABEL2* `-do` *DESCRIPTION* `-end`

Orbiter Objects (Part I)

Command	Arguments	Purpose
-finite_field	Sections 3.2 and 3.3	A finite field $\mathbb{F}_q$.
-polynomial_ring	Section 5.2	A multivariate polynomial ring.
-projective_space	Section 4.1	A projective space $\mathrm{PG}(n,q)$.
-orthogonal_space	Section 4.7	A non-degenerate orthogonal space $O^\epsilon(n,q)$.
-BLT_set_classifier	Section 14.4	An object to classify BLT-sets.
-spread_classifier	Section 14.1	An object to classify spreads.
-linear_group	Section 6.2	A linear group.
-permutation_group	Section 6.1	A generic permutation group.
-modified_group	Section 6.4	A new group from an old.
-geometric_object	Section 4.10	A geometric object.
-collection		A collection of objects.
-graph	Section 12.1	A graph.
-code	Section 10.2	A code.
-spread	Section 14.1	A spread.
-cubic_surface	Section 8.2	A cubic surface.
-quartic_curve	Section 8.5	A quartic curve.
-BLT-set	Section 14.4	A BLT-set.
-translation_plane	Section 14.2	A translation plane.
-spread_table	Section 14.3	A table of spreads.
-packing_with_symmetry_assumption	Section 14.3	A generator for packings with assumed symmetry.
-packing_choose_fixed_points	Section 14.3	A selection of fixed orbits for packings with assumed symmetry.

Table 2.2: Orbiter Objects (Part I)

Orbiter Objects (Part II)

Command	Arguments	Purpose
-packing_long_orbits	Section 14.3	A search for long orbits for packings with assumed symmetry.
-graph_classification	Section 12.3	An object which allows classifying graphs and tournaments.
-diophant	Section 15.3	A diophantine system, i.e., a system of positive integer equations.
-design	Section 13.1	A combinatorial design.
-design_table	Section 13.1	A table of designs, possibly isomorphic. It can be used to construct large sets of designs. A large set is a set of designs satisfying certain properties.
-large_set_with_symmetry_assumption	Section 13.1	An object to create a large set of designs.
-set	Section 2.6	A set of integers.
-vector	Section 2.7	A vector of elements of a finite field.
-symbolic_object	Section 2.9	A symbolic algebraic expression.
-combinatorial_object	Section 11.2	A set of combinatorial objects.
-geometry_builder	Section 11.4	An object to classify incidence geometries.
-vector_ge	Section 6.3	A vector of group elements.
-action_on_forms		The action of a projective group on homogeneous polynomials of a given degree.
-orbits	Section 7.1	An object to compute orbits of a group.
-poset_classification_control	Section 7.2	An object to set options for the poset classification algorithm.
-arc_generator_control	See Table 7.7.	An object to set options for the arc classification algorithm.
-poset_classification_activity	Section 14.4	An object to define an activity for a poset of orbits.
-crc_code	Section 10.6	An object to define CRC-code.
-mapping	See Table 5.6	An object to define a mapping.

Table 2.3: Orbiter Objects (Part II)

Orbiter Activities (Part 1)

Command	Arguments	Purpose
-finite_field_activity	Sections 3.2 and 3.3	An activity for finite fields.
-polynomial_ring_activity	Table 5.3	An activity for a polynomial ring.
-projective_space_activity	Section 4.1	An activity for a projective space.
-orthogonal_space_activity	Section 4.7	An activity for an orthogonal space.
-group_theoretic_activity	Section 6.5	An activity for a group.
-coding_theoretic_activity	Tables 10.3-10.5	An activity for a code.
-cubic_surface_activity	Section 8.2	An activity for a cubic surface.
-quartic_curve_activity	Section 8.5	An activity for a quartic curve.
-blt_set_activity	Table 14.16	An activity for a BLT-set.
-combinatorial_object_activity	Section 4.5	An activity for a combinatorial object.

Table 2.4: Orbiter Activities (Part 1)

is used. Here, *LABEL1* and *LABEL2* are the objects for which the activity is invoked. For an example of an activity requiring two objects, see Sections 14.1 and 14.2.

Tables 2.4-2.5 list the possible activities for Orbiter objects. More details about each of these activities will be given in the sections below.

Orbiter Activities (Part 2)		
Command	Arguments	Purpose
-graph_theoretic_activity	Section 12.1	An activity for a graph.
-classification_of_cubic_surfaces_with_double_sixes_activity	Section 8.2	An activity for a cubic surface.
-spread_table_activity	Section 14.3	An activity associated with a table of spreads.
-packing_with_symmetry_assumption_activity	Section 14.3	An activity related to creating packings with assumed symmetry group.
-packing_fixed_points_activity	Section 14.3	An activity related to creating packings with assumed symmetry group.
-graph_classification_activity	Section 12.3	An activity for a classification of graphs problem.
-diophant_activity	Section 15.3	An activity for a diophantine system.
-design_activity	Section 13.1	An activity for a combinatorial design.
-large_set_with_symmetry_assumption_activity	Section 13.3	An activity related to creating large sets of designs with assumed symmetry group.
-symbolic_object_activity	Table 2.12	An activity related to a symbolic object.
-BLT_set_classify_activity	Table 14.18	An activity related to a classification of BLT-sets.
-spread_classify_activity	Table 14.4	An activity related to a classification of spreads.
-spread_activity	Table 14.2	An activity related to a spread.
-translation_plane_activity	Table 14.9	An activity related to a translation plane.
-orbits_activity	Table 7.2	An activity related to orbits.

Table 2.5: Orbiter Activities (Part 2)

Mathematical Data in Orbiter

Object	Description
BLT-sets	BLT sets of $Q(4,q)$ exist for all odd prime powers. The classification of BLT-sets of $Q(4,q)$ is known to Orbiter for all $q \leq 73$.
Cubic Surfaces	Cubic surfaces with 27 lines exist for all finite fields apart from $\mathbb{F}_2, \mathbb{F}_3, \mathbb{F}_5$. Orbiter knows the classification of cubic surfaces with 27 lines for all fields $\mathbb{F}_q$ of order $q \leq 128$.
Quartic curves	Orbiter knows the classification of smooth quartic curves with 28 bitangents in projective planes over all fields $\mathbb{F}_q$ for $q = 9,\ 13,\ 19,\ 25,\ 27,\ 29,\ 31$.
Spreads	A spread is a set of $q^k + 1$ pairwise non-intersecting k-dimensional subspaces of $\mathbb{F}_q^{2k}$. Spreads are related to translation planes of order q^k. Orbiter knows the classification of spreads for $(q,k) \in \{(2,2),(3,2),(2,4),(4,2),(5,2),(3,3)\}$.
Hyperovals	A hyperoval in $\mathrm{PG}(2,2^e)$ is a set of $2^e + 2$ points, no three collinear. Orbiter knows the classification of hyperovals for $e = 3,4,5$.
Dual hyperovals	A k-dimensional dual hyperoval in an ambient space $\mathbb{F}_2^n$ is called a $\mathrm{DH}(k,n)$. Orbiter knows the classification of dual hyperovals $\mathrm{DH}(4,7)$ and $\mathrm{DH}(4,8)$.
Packings	Orbiter knows the classification of packings of $\mathrm{PG}(3,3)$.

Table 2.6: Mathematical Data in Orbiter

2.5 Mathematical Data

Orbiter serves as a repository for mathematical data. The knowledge base is concerned with classifications of geometric and combinatorial objects for small parameters. The types of objects for which a classification is available in Orbiter are listed in Table 2.6.
The mathematical objects are stored in a catalogue, together with generators for their automorphism groups. The objects are indexed by a zero-based integer, called the *Orbiter Catalogue Number* (OCN). It is possible to access any object in the catalogue. Let us consider some examples:

The command

Example 6

```
create_BLT_5_1:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 5 -end \
▷  ▷  -define O -orthogonal_space 0 5 F -end \
▷  ▷  -define BLT -BLT_set \
▷  ▷  ▷  -space O -catalogue 1 \
▷  ▷  -end \
▷  ▷  -with BLT -do -blt_set_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
```

```
▷  pdflatex BLT_catalogue_q5_iso1.tex
▷  $(OPEN) BLT_catalogue_q5_iso1.pdf
```

recalls the BLT-set with Orbiter Catalogue Number 1 in $Q(4,5)$. A latex report `catalogue_q5_iso1.tex` is written. For more details about BLT-sets, see Section 14.4.

The command

Example 7

```
create_surface_4_0:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 4 -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷  ▷  -define S4_0 -cubic_surface -space P -catalogue 0 -end \
▷  ▷  -with S4_0 -do \
▷  ▷  ▷  -cubic_surface_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex surface_catalogue_q4_iso0_report.tex
▷  $(OPEN) surface_catalogue_q4_iso0_report.pdf
```

recalls the cubic surface with Orbiter Catalogue Number 0 in $\mathrm{PG}(3,4)$. A latex report `surface_catalogue_q4_iso0_report.tex` is written. For more details about cubic surfaces, see Section 8.2.

Set Builder Commands

Command	Arguments	Purpose
-index_set_loop	a b c	creates the set $\{k \mid k = a + ic,\ i \in \mathbb{Z},\ a \leq k < c\}$
-set_builder	set	Use a set as input to a construction.
-clone_with_affine_function	a b	Apply the affine function $ax + b$ to the elements x of a set specified using -set_builder.
-affine_function	a b	Apply the affine function $ax + b$ to the elements x of a set specified using -set_builder.
-here	elements	Create the set from the given list of elements. The elements of the list are comma separated.
-file	fname	Create a set from the elements listed in the given file.
-file_orbiter_format	fname	Create a set from the elements listed in the given file.

Table 2.7: Set Builder Commands

2.6 Set Builder

Table 2.7 shows Orbiter commands to create sets.

Here is an example. We create the set S of the first six prime numbers $\{2, 3, 5, 7, 11, 13\}$:

Example 8

```
set_of_primes:
▷  $(ORBITER) -v 2 \
▷  ▷  -define S -set -here "2,3,5,7,11,13" -end \
▷  ▷  -print_symbols
```

The next command creates the integer interval $[0, 63]$. We use the -loop command to save us from typing out all elements of the set. The -loop command has three arguments: the start value s, the upper bound u, and the increment d. The numbers created have the form $s + kd$, $k = 0, 1, \ldots$ as long as they are less than u.

Example 9

```
set_interval:
▷  $(ORBITER) -v 2 -define S -set -loop 0 64 1 -end \
▷  ▷  -print_symbols
```

Vector Builder Commands

Command	Arguments	Purpose
-field	F	Assume that the vector entries are in the range $[0, q-1]$, where q is the order of the field F. An error is raised if not.
-allow_negatives		Allow negative numbers when using finite fields. A negative number will be considered modulo the characteristic of the field. For instance, in the field $\mathbb{F}_5$, a -1 is considered equal to 4.
-dense		Assume that the vector entries are given as a comma separated list (dense coding).
-compact		Assume that the vector entries are given as a continueous list without commas or other any kind of whitespace.
-repeat	pattern n	Repeat the given pattern n times.
-format	r	Assume that the data is a matrix with r rows.
-file	file	Read the data from the given csv-file.
-load_csv_no_border	file	Read the data from the given file, assuming that there are no headers.
-load_csv_data_column	file c	Read the data from column c in the given csv-file.
-sparse	data	Provide data in sparse form, i.e. as pairs value and position.
-concatenate	list	Concatenate the given vectors. List is a comma separated list of vectors that have been created before.
-loop	s u d	Create vector entries of the form $s + id < u$, $i \in \mathbb{Z}$. The upper bound u itself will not be part of the sequence.

Table 2.8: Vector Builder Commands

2.7 Vector Builders

Table 2.8 shows Orbiter commands to create vector and matrix objects. Matrices are vectors with an additional format information. By specifying the number of rows, a vector turns into a matrix. So, to create a matrix, one creates a vector, and uses the -format command. Examples will be shown below.

Vectors can be defined in two different formats, called dense and sparse, respectively. In the dense format, the coefficients are listed in order from the lowest to the highest term. The -dense command creates the vector from a list of coefficients. The sparse format can be useful for coefficient vectors with few nonzero entries. It is a list of coefficient pairs, each of which describing one entry in the vector. One pair consists of the coefficient and the index of the term. The pairs are listed in sequence. The -sparse command creates the vectors from a given list of coefficient pairs.

If the option -field is given, Orbiter will assume that the vector entries lie in the interval $[0, q-1]$, where q is the order of the finite field. If entries lie outside that interval, an error is raised. There are limits on the size of q, see Section 19.2. It is safe to assume that any number q less than $2^{31} - 1$ is fine. On 64 bit systems (with Orbiter compiled in 64 bit mode) we can have $q \leq 2^{63} - 1$.

Here is an example. We first create the field $\mathbb{F}_5$, and then create the vector $v = (0, 1, 2, 3, 4)$. The -field

option refers to the finite field created previously. The `-dense` option allows to enter the vector coefficients on the command line.

Example 10

```
vector_example1:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 5 -end \
▷  ▷  -define v -vector -field F -dense "0,1,2,3,4" -end \
▷  ▷  -print_symbols
```

Vectors can also be read from file. The `-file` option can be used to name a csv file.

A vector can also serve as a matrix. The `-format` k option can be used to specify the number k of rows. The number of columns is determined as n/k, where n is the length of the vector given. For instance, the next example creates a 2×3 matrix over $\mathbb{F}_5$:

Example 11

```
vector_example2:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 5 -end \
▷  ▷  -define v -vector -field F -format 2 -dense "0,1,2,3,4,0" -end \
▷  ▷  -print_symbols
```

Orbiter has a command to create vectors whose entries repeat. For instance, the following code creates a vector of length 11 whose entries repeat over the sequence $0, 1, 2, 3$. It is not necessary that the vector length is an integer multiple of the length of the repeating sequence.

Example 12

```
vector_example_repeat:
▷  $(ORBITER) -v 2 \
▷  ▷  -define v -vector -repeat "0,1,2,3" 11 -end \
▷  ▷  -print_symbols
```

The sequence 0,1,2,3 is repeated sufficiently often to make a vector of length 11. This creates the vector

$$(0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2).$$

In order to create a constant vector, the `-repeat` command can be used as well. Simply use a repeat sequence consisting of a single number. For instance, the following example creates the all-one vector of length 11:

Example 13

```
vector_example_all_one_11:
▷  $(ORBITER) -v 2 \
▷  ▷  -define v -vector -repeat 1 11 -end \
```

```
▷  ▷  -print_symbols
```

This code will create the all-one vector of length 11:

$$(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1).$$

A loop command can create sequences of numbers $a_0, a_1, \ldots$. It takes three arguments: s, u, and d. The values of the sequence are

$$a_i = s + id, \ldots k = 0, 1, 2, \ldots, \quad a_i < u.$$

So, the values of the sequence increases in steps of d, and are forced to be less than the upper bound u. For instance, in order to create all integers from 0 to 7, we use the command:

Example 14

```
vector_loop_8:
▷  $(ORBITER) -v 2 \
▷  ▷  -define v -vector -loop 0 8 1 -end \
```

If we wanted to create the odd integers less than or equal to 16, we could do this:

Example 15

```
vector_loop_odd_16:
▷  $(ORBITER) -v 2 \
▷  ▷  -define v -vector -loop 1 16 2 -end \
```

This command creates the vector

$$(1, 3, 5, 7, 9, 11, 13, 15).$$

Two vectors can be concatenated, as demonstrated in the following example:

Example 16

```
vector_concatenate:
▷  $(ORBITER) -v 2 \
▷  ▷  -define a -vector -compact "1,2,3" -end \
▷  ▷  -define b -vector -compact "4,5,6" -end \
▷  ▷  -define c -vector -compact "7,8,9" -end \
▷  ▷  -define abc -vector -concatenate a,b,c -end
```

The vectors $(1, 2, 3)$, $(4, 5, 6)$ and $(7, 8, 9)$ are created. After that, the concatenation command produces

$$(1, 2, 3, 4, 5, 6, 7, 8, 9).$$

The vector command can also be used to create matrices. A matrix is a vector with additional formatting information. The vector is the coefficient vector, using row-major ordering. For instance, the following command creates the vector

$$(1, 2, 3, 4, 5, 6),$$

which by means of the `-format 2` command turns into the matrix

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}.$$

Here is the command:

Example 17

```
matrix_example1:
▷  $(ORBITER) -v 2 \
▷  ▷  -define v -vector -format 2 \
▷  ▷  ▷  -dense "1,2,3,4,5,6" -end \
▷  ▷  -print_symbols
```

For large vectors, the `sparse` command can be used to enter non-zero coefficients as a list of pairs. For instance,

Example 18

```
vector_example_sparse:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 5 -end \
▷  ▷  -define v -vector -field F -format 4 -sparse 20 "1,0,1,19" -end \
▷  ▷  -print_symbols
```

creates a vector of length 20 and sets the 0-th and the 19-th coefficient to 1. So, the vector is

$$(1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1).$$

Finally, the vector is displayed as a four-rowed matrix:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Makefile variables can be used to hold the coefficients in a separate place. This helps reduce clutter. For instance, the following command creates the generator matrix of the Hamming code:

Example 19

```
HAMMING_CODE_GENERATOR="\
1,0,0,0,0,1,1, \
0,1,0,0,1,0,1, \
0,0,1,0,1,1,0, \
0,0,0,1,1,1,1"
```

Example 20

```
matrix_example2:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define v -vector -field F -format 4 \
▷  ▷  ▷  -dense $(HAMMING_CODE_GENERATOR) -end \
▷  ▷  -print_symbols
```

For matrices over small fields, the `-compact` option can be given. This works similar as the `-dense` option, but the entries are not separated by commas. This is only good for single digit data, as it arises with very small fields. For instance, the following code creates a 22×22 matrix over the binary field:

Example 21

```
CONWAY_GEN1="\
1101110001000001010000\
1111010111110100001011\
0000001000000100010101\
1111100110110001001110\
0101010000000010011101\
0000010000000100010101\
0010000000000100010101\
0001000011000000111111\
1110100100110100010011\
0000000000000110010101\
0000000000100100010101\
0110111111010011101111\
0000000000001100010101\
0000000000000100000101\
0000000001000100010101\
0000000000000100011101\
0001000110000010011010\
0000000000000000010101\
0000000000000101010101\
0000000000000100010100\
0000000000000100010111\
0000000000000100010001"
```

Example 22

```
matrix_example_co_1:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define v -vector -field F -format 22 \
▷  ▷  ▷  -compact $(CONWAY_GEN1) -end \
▷  ▷  -print_symbols
```

Vector of Group Elements Builder Commands		
Command	Arguments	Purpose
-action	A	Set the action type for the group elements.
-read_csv	fname	Read group elements from the given csv file.
-vector_data	v	Read group elements from the given vector data.

Table 2.9: Vector of Group Elements Builder Commands

2.8 Vector of Group Elements

Orbiter can create vectors of group elements. Table 2.9 shows the Orbiter commands that can be used to create a vector of group elements.
Let us consider an example. Our goal is to describe the action of the elements of class 2A in the stabilizer of the Hirschfeld surface. The group element is a Baer collineation:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_1$$

The class representative is stored in a makefile variable:

Example 23

```
CLASS_2A_REP="1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1"
```

The first command creates the conjugacy class 2A. The elements will be written to a csv file:

Example 24

```
Hirschfeld_Class_2A:
▷  $(ORBITER) -v 9 \
▷  ▷  -orbiter_path $(ORBITER_PATH) \
▷  ▷  -define G -linear_group -PGGL 4 4 \
▷  ▷  ▷  -subgroup_by_generators "Hirschfeld_Stab" \
▷  ▷  ▷  51840 6 $(HIRSCHFELD_STAB_GENERATORS) \
▷  ▷  ▷  -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -conjugacy_class_of "2A" $(CLASS_2A_REP) \
▷  ▷  -end
```

The filename is stored in a makefile variable:

Example 25

```
FILE_NAME_CLASS_2A="PGGL_4_4_Subgroup_Hirschfeld_Stab_51840_class_of_2A.csv"
```

The next command reads the elements of class 2A from this file. It then prints the action of these elements on the various objects associated with the cubic surface:

Example 26

```
surface_4_0_export_action:
▷  $(ORBITER) -v 3 \
▷  ▷  -define Class2A -vector -file $(FILE_NAME_CLASS_2A) -end \
▷  ▷  -define F -finite_field -q 4 -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷  ▷  -define S -cubic_surface -space P -catalogue 0 -end \
▷  ▷  -define G -linear_group -PGGL 4 4 -end \
▷  ▷  -define gens_ge -vector_ge \
▷  ▷  ▷  -action G \
▷  ▷  ▷  -vector_data Class2A \
▷  ▷  -end \
▷  ▷  -with S -do \
▷  ▷  -cubic_surface_activity \
▷  ▷  ▷  -export_something_with_group_element "action_on_tritangent_planes" gens_ge
\
▷  ▷  -end \
▷  ▷  -with S -do \
▷  ▷  -cubic_surface_activity \
▷  ▷  ▷  -export_something_with_group_element "action_on_double_sixes" gens_ge \
▷  ▷  -end \
▷  ▷  -with S -do \
▷  ▷  -cubic_surface_activity \
▷  ▷  ▷  -export_something_with_group_element "action_on_lines" gens_ge \
▷  ▷  -end \
```

Creating Symbolic Objects I

Command	Arguments	Purpose
-managed_variables	list	A comma-separated list of managed variables may be given. The manage variables can later be matched to the variables of a polynomial ring. The other variables can be used as parameters.
-text	text	The well-formed sequence of expressions in ascii text format, separated by commas.
-field	F	The finite field of coefficients in the expression.
-matrix	r	The expression vector is a matrix with r rows.
-determinant	s	Compute the symbolic determinant of the source object s. The object s must be a square matrix.
-characteristic_polynomial	s $lambda$	Compute the characteristic polynomial of the source object s using the variable λ. The object s must be a square matrix.
-substitute	v t s	Substitute the variables v in the expression t by the expressions given in s. v and s must be vectors of the same length.
-simplify	s	Simplify the object s.
-expand	s	Expand the object s.
-right_nullspace	s	Compute the right nullspace of the object s, which must be a fully numerical matrix.
-minor	s i j	Compute the (i,j)-minor of the matrix object s. s must be a square matrix.
-symbolic_nullspace	s	Compute the right nullspace of the object s, which must be a symbolic matrix.
-stack_matrices_vertically	list	Stack the given matrices in the list vertically. The matrices must have the same number of columns.

Table 2.10: Creating Symbolic Objects I

2.9 Symbolic Objects

Symbolic objects represent algebraic expressions, commonly known as formulae. They consist of the elementary arithmetic operations, numbers, variables and exponents. In Orbiter, algebraic expressions can be parsed from an ascii text representation. This is different from for instance Maple, where an algebraic expression composed inside the graphical user interface. Internally, an algebraic expression is represented as an abstract syntax tree. The syntax tree is an efficient way to code the expression, and allows many algebraic operations to be performed easily. The process of creating the abstract syntax tree from the ascii representation is called parsing. Orbiter can deal with vectors and matrices of symbolic expressions.

Tables 2.10-2.11 show Orbiter commands to create symbolic objects.

Table 2.12 show Orbiter activities for symbolic objects.

Let us consider some examples. The examples are over finite field $\mathbb{F}_q$. For more information about these fields, see Chapter 3. For now, it suffices to know that the elements of the finite field $\mathbb{F}_q$ are the integers

Creating Symbolic Objects II		
Command	Arguments	Purpose
-multiply_2x2_from_the_left	s $A2$ i j	Multiply the 2×2 matrix $A2$ from the left to s, applying to rows i and j.
-matrix_entry	s i j	Select the (i, j)-entry of the matrix s.
-vector_entry	s i	Select the j-th entry of the vector s. Using row-major ordering, a matrix is considered a vector also.
-collect	s t	Collect by coefficients of t in the expression s.
-do_not_simplify		Do not simplify the output object.
-write_trees_during_expand		Export trees during the expand command.

Table 2.11: Creating Symbolic Objects II

Activities for Symbolic Objects		
Command	Arguments	Purpose
-export		Export the symbolic object.
-print	F	Print the symbolic object, assuming that coefficients are from the finite field F.

Table 2.12: Activities for Symbolic Objects

from 0 to $q-1$. It is also true that 0 represents zero and 1 represents one. Numbers outside the interval 0 to $q-1$ are not allowed. In particular, we do not allow negative numbers. So, for instance, -1 is not allowed.

The first example creates the polynomial $1 - X - X^2$ over $\mathbb{F}_7$:

Example 27

```
symbolic_poly:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 7 -end \
▷  ▷  -define M -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -text "-X^2-X+1"\
▷  ▷  -end \
```

The output is

$$M = \ 1 + (6X) + (6X^2)$$

The coefficients of X and X^2 are 6, because $6 \equiv -1 \bmod 7$ and Orbiter chooses field elements of $\mathbb{F}_q$ to lie in the interval $[0, q-1]$.

The second example creates the polynomial CRC32, using the makefile variable

Example 28

```
CRC32_ETHERNET_POLY="X^32+X^26+X^23+X^22+X^16+\
X^12+X^11+X^10+X^8+X^7+X^5+X^4+X^2+X^1+1"
```

The command

Example 29

```
symbolic_CRC32:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define M -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -text $(CRC32_ETHERNET_POLY) \
▷  ▷  -end \
```

produces the following output:

$$M = \ 1 + X + X^2 + X^4 + X^5 + X^7 + X^8 + X^{10} + X^{11} + X^{12} + X^{16} + X^{22} + X^{23} + X^{26} + X^{32}$$

The second example creates the BCH polynomial Alfa, defined over the field $\mathbb{F}_{256}$, using the makefile variable

Example 30

```
BCH_POLYNOMIAL_ALFA_F256="X^2 + 167*X + 214"
```

The command

Example 31

```
symbolic_CRC_alfa:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 256 -end \
▷  ▷  -define M -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -text $(BCH_POLYNOMIAL_ALFA_F256) \
▷  ▷  -end \
```

produces the following output:

$$M = \ 214 + X^2 + (167X)$$

The next example creates the CRC polynomial bravo over the field $\mathbb{F}_{256}$:

Example 32

```
symbolic_CRC_bravo:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 256 -end \
```

```
▷  ▷  -define M -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -text $(BCH_POLYNOMIAL_BRAVO_F256) \
▷  ▷  -end \
```

The output is

$$M = \ 1 + X^4 + (27X^2) + (23X) + (213X^3)$$

Let us consider the formula $(a+b)^3$ and expand:

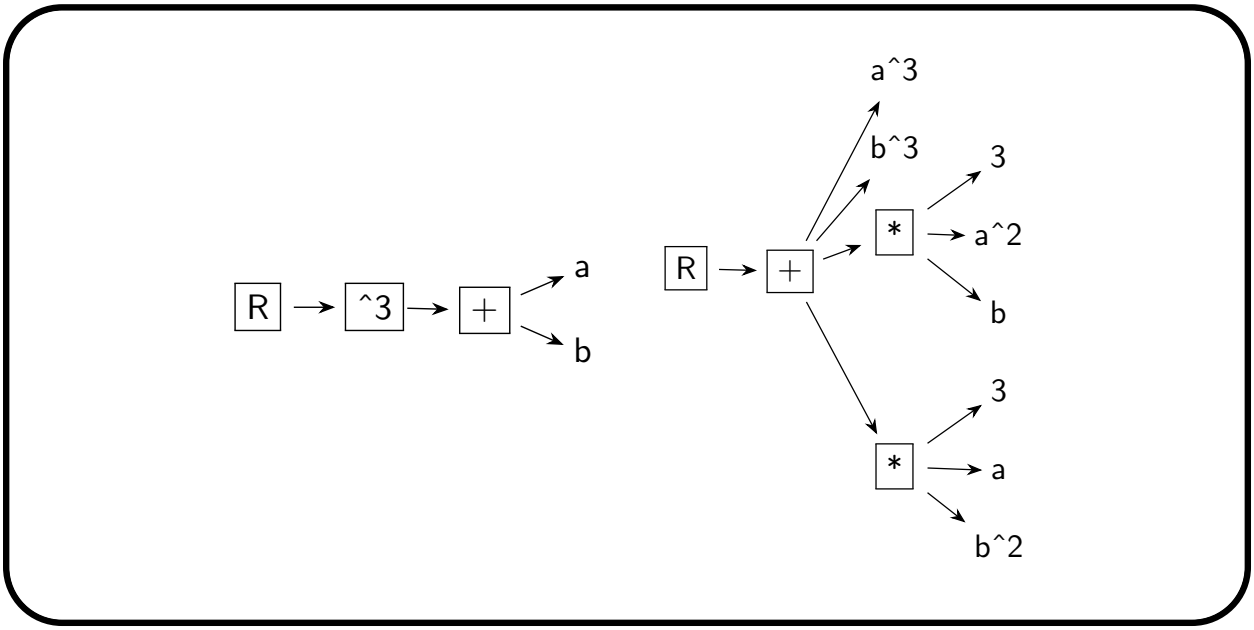
Example 33

```
symbolic_test_5_expand:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 13 -end \
▷  ▷  -define M -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -text "(a+b)^3" \
▷  ▷  -end \
▷  ▷  -define M1 -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -expand M \
▷  ▷  -end
▷  pdflatex M.tex
▷  $(OPEN) M.pdf
▷  pdflatex M1.tex
▷  $(OPEN) M1.pdf
```

The output is

$$M1 = \left[\ a^3 + b^3 + (3a^2b) + (3ab^2) \ \right]$$

It is also possible to inspect the representation of the expressions as abstract syntax trees. The node labeled R signifies the root node of the tree. The tree on the left is the syntax tree before expansion, whereas the tree on the right is the syntax tree after expansion:



The command

Example 34

```
test_expand_1:
▷  $(ORBITER) -v 3 \
▷  ▷  -define Fq -finite_field -q 31 -end \
▷  ▷  -define f -symbolic_object \
▷  ▷  ▷  -field Fq \
▷  ▷  ▷  -text "a*(x+y)" \
▷  ▷  -end \
▷  ▷  -define fe -symbolic_object \
▷  ▷  ▷  -field Fq \
▷  ▷  ▷  -expand f \
▷  ▷  ▷  -write_trees_during_expand \
▷  ▷  -end
▷  pdflatex fe_0_efsxsfscsc.tex
▷  $(OPEN) fe_0_efsxsfscsc.pdf
```

tests that

$$a(x+y) = ax + ay.$$

The command

Example 35

```
test_expand_2:
▷  $(ORBITER) -v 3 \
▷  ▷  -define Fq -finite_field -q 31 -end \
▷  ▷  -define f -symbolic_object \
▷  ▷  ▷  -field Fq \
```

```
▷  ▷  ▷  -text "a*(-(x-y))" \
▷  ▷  -end \
▷  ▷  -define fe -symbolic_object \
▷  ▷  ▷  -field Fq \
▷  ▷  ▷  -expand f \
▷  ▷  ▷  -write_trees_during_expand \
▷  ▷  -end
▷  pdflatex fe_0_efsxsfscsc.tex
▷  $(OPEN) fe_0_efsxsfscsc.pdf
```

tests that

$$a(-(x-y)) = -ax + ay.$$

The command

Example 36

```
test_expand_3:
▷  $(ORBITER) -v 3 \
▷  ▷  -define Fq -finite_field -q 31 -end \
▷  ▷  -define f -symbolic_object \
▷  ▷  ▷  -field Fq \
▷  ▷  ▷  -text "a*(-(x-y)+(x+y))" \
▷  ▷  -end \
▷  ▷  -define fe -symbolic_object \
▷  ▷  ▷  -field Fq \
▷  ▷  ▷  -expand f \
▷  ▷  ▷  -write_trees_during_expand \
▷  ▷  -end
▷  pdflatex fe_0_efs.tex
▷  $(OPEN) fe_0_efs.pdf
▷  pdflatex fe_0_efsx.tex
▷  $(OPEN) fe_0_efsx.pdf
▷  pdflatex fe_0_efsxsf.tex
▷  $(OPEN) fe_0_efsxsf.pdf
▷  pdflatex fe_0_efsxsfs.tex
▷  $(OPEN) fe_0_efsxsfs.pdf
▷  pdflatex fe_0_efsxsfsc.tex
▷  $(OPEN) fe_0_efsxsfsc.pdf
▷  pdflatex fe_0_efsxsfscsefsx.tex
▷  $(OPEN) fe_0_efsxsfscsefsx.pdf
▷  pdflatex fe_0_efsxsfscsefsxsf.tex
▷  $(OPEN) fe_0_efsxsfscsefsxsf.pdf
▷  pdflatex fe_0_efsxsfscsefsxsfscsc.tex
▷  $(OPEN) fe_0_efsxsfscsefsxsfscsc.pdf
```

tests that

$$a(-(x-y)+(x+y)) = 2ay.$$

The output is

$$fe = \quad (2ay)$$

The command shows the abstract syntax trees for the intermediate steps in the algebraic simplification.

The command

Example 37

```
test_expand_4:
▷  $(ORBITER) -v 3 \
▷  ▷  -define Fq -finite_field -q 31 -end \
▷  ▷  -define f -symbolic_object \
▷  ▷  ▷  -field Fq \
▷  ▷  ▷  -text "x*(a+(x*(a+(x*(a+x*(a+x*(a+x*(a+x)))))))" \
▷  ▷  -end \
▷  ▷  -define fx -symbolic_object \
▷  ▷  ▷  -field Fq \
▷  ▷  ▷  -expand f \
▷  ▷  ▷  -write_trees_during_expand \
▷  ▷  -end
▷  pdflatex fx_0_efsxsfscsefsxsfscsefsxsfscsefsxsfscsefsxsfscsefsxsfscsefsxsfscsefsxsfscsc.tex
▷  $(OPEN) fx_0_efsxsfscsefsxsfscsefsxsfscsefsxsfscsefsxsfscsefsxsfscsefsxsfscsefsxsfscsc.pdf
```

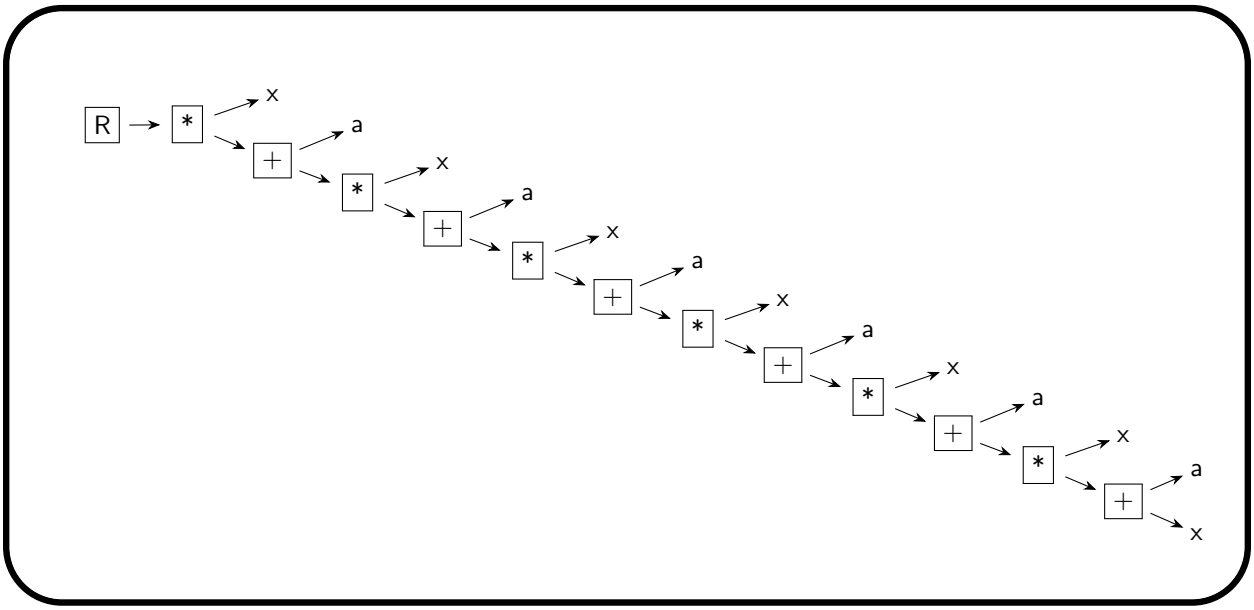
tests that

$$x*(a+(x*(a+(x*(a+x*(a+x*(a+x*(a+x))))))))=ax+ax^2+ax^3+ax^4+ax^5+ax^6+x^7.$$

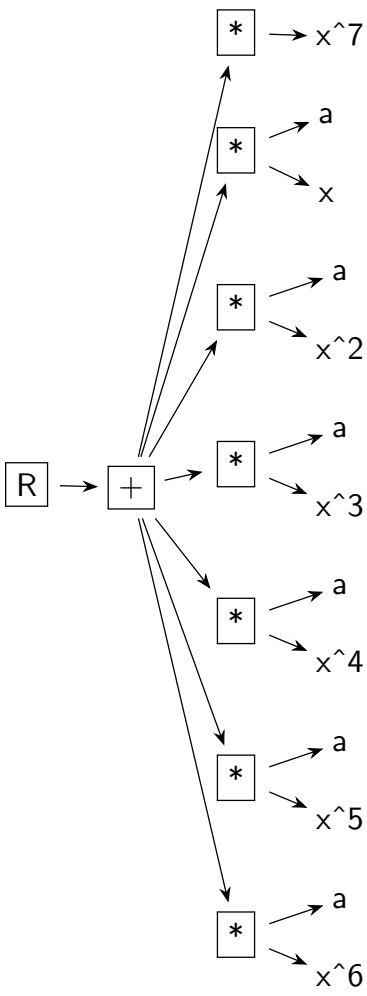
The output is

$$fx = \ x^7+(ax^2)+(ax^3)+(ax^4)+(ax^5)+(ax^6)+(ax)$$

The abstract syntax tree before expansion is:



The abstract syntax tree after expansion is:



The abstract syntax trees for the intermediate steps during expansion are avaliable also.

Objects are stored in the Orbiter symbol table. There is only one table. There can be only one object with any given name. Symbolic objects can recycle objects defined earlier using a process called implicit substitution. In this process, the object being referenced is taken from the symbol table and substituted immediately. For instance, in the next example we create an object $a = x + y$ and then another object $b = a + z$ which immediately expands into $b = x + y + z$. This is because the object a already exists. Implicit substitution is non-recursive, so the substituted syntax tree is not subjected to implicit substitution itself. This prevents unlimited recursion which would result in crashes.

Example 38

```
symbolic_objects_building_on_earlier_objects:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define a -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -text "x+y"\
▷  ▷  -end \
```

```
▷  ▷  -define b -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -text "a+z"\
▷  ▷  -end \
```

The output of the command is:

$$b = \quad x + y + z$$

Self-referential objects are allowed. The following command is legal:

Example 39

```
symbolic_objects_self_referential_1:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define x -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -text "x"\
▷  ▷  -end \
```

The output is

$$x = \quad x$$

Likewise, the following command is ok:

Example 40

```
symbolic_objects_self_referential_2:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define x -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -text "x"\
▷  ▷  -end \
▷  ▷  -define a -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -text "x+a"\
▷  ▷  -end \
```

The output is

$$a = \quad a + x$$

The implicit substitution process is limited to scalar quantities. Matrices and vectors cannot be substituted.

Let us create a Vandermonde matrix. We use the makefile variable

Example 41

```
VANDERMONDE_4X4_FORMULA="1,x0,x0^2,x0^3,1,x1,x1^2,x1^3,\
1,x2,x2^2,x2^3,1,x3,x3^2,x3^3"
```

The following command creates the matrix:

Example 42

```
symbolic_matrix1:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 13 -end \
▷  ▷  -define M -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -matrix 4 \
▷  ▷  ▷  -text $(VANDERMONDE_4X4_FORMULA) \
▷  ▷  -end
```

The output is:

$$\begin{bmatrix} 1 & x0 & x0^2 & x0^3 \\ 1 & x1 & x1^2 & x1^3 \\ 1 & x2 & x2^2 & x2^3 \\ 1 & x3 & x3^2 & x3^3 \end{bmatrix}$$

Let us create a fully symbolic 2×2 matrix:

Example 43

```
symbolic_matrix_2x2:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 13 -end \
▷  ▷  -define M -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -matrix 2 \
▷  ▷  ▷  -text "a,b,c,d" \
▷  ▷  -end
```

The output is:

$$M = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

Let us create the determinant of a symbolic 2×2 matrix:

Example 44

```
symbolic_determinant_2x2:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 13 -end \
▷  ▷  -define M -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -matrix 2 \
▷  ▷  ▷  -text "a,b,c,d" \
▷  ▷  -end \
▷  ▷  -define det -symbolic_object \
▷  ▷  ▷  -field F \
```

```
▷ ▷ ▷ -determinant M \
▷ ▷ -end \
▷ ▷ -define det1 -symbolic_object \
▷ ▷ ▷ -field F \
▷ ▷ ▷ -expand det \
▷ ▷ -end
▷ pdflatex det1.tex
▷ $(OPEN) det1.pdf
```

The output is

$$det1 = \ (ad) + (12bc)$$

Let us create the determinant of a symbolic 3×3 matrix:

Example 45

```
symbolic_determinant_3x3:
▷ $(ORBITER) -v 3 \
▷ ▷ -define F -finite_field -q 13 -end \
▷ ▷ -define M -symbolic_object \
▷ ▷ ▷ -field F \
▷ ▷ ▷ -matrix 3 \
▷ ▷ ▷ -text "a,b,c,d,e,f,g,h,i" \
▷ ▷ -end \
▷ ▷ -define det -symbolic_object \
▷ ▷ ▷ -field F \
▷ ▷ ▷ -determinant M \
▷ ▷ -end \
▷ ▷ -define det1 -symbolic_object \
▷ ▷ ▷ -field F \
▷ ▷ ▷ -expand det \
▷ ▷ -end
▷ pdflatex det1.tex
▷ $(OPEN) det1.pdf
```

The output is

$$det1 = \ (cdh) + (bfg) + (12bdi) + (12afh) + (aei) + (12ceg)$$

Let us create the determinant of the Vandermonde matrix from above:

Example 46

```
symbolic_determinant_vandermonde:
▷ $(ORBITER) -v 3 \
▷ ▷ -define F -finite_field -q 13 -end \
▷ ▷ -define M -symbolic_object \
▷ ▷ ▷ -field F \
▷ ▷ ▷ -matrix 4 \
▷ ▷ ▷ -text $(VANDERMONDE_4X4_FORMULA) \
▷ ▷ -end \
```

```
▷  ▷  -define det -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -determinant M \
▷  ▷  -end \
▷  ▷  -define det1 -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -expand det \
▷  ▷  -end
```

The output is lengthy, so it is given in text notation:

```
0 :
(12 * x1 * x2 ^ 3 * x3 ^ 2) + (12 * x1 ^ 2 * x2 * x3 ^ 3) + (x1 ^ 2 * x2 ^ 3
* x3) + (x1 ^ 3 * x2 * x3 ^ 2) + (12 * x1 ^ 3 * x2 ^ 2 * x3) + (12 * x0 * x
2 ^ 2 * x3 ^ 3) + (x0 * x2 ^ 3 * x3 ^ 2) + (x0 * x1 ^ 2 * x3 ^ 3) + (12 * x0
* x1 ^ 2 * x2 ^ 3) + (12 * x0 * x1 ^ 3 * x3 ^ 2) + (x0 * x1 ^ 3 * x2 ^ 2) +
(x0 ^ 2 * x2 * x3 ^ 3) + (12 * x0 ^ 2 * x2 ^ 3 * x3) + (12 * x0 ^ 2 * x1 *
x3 ^ 3) + (x0 ^ 2 * x1 * x2 ^ 3) + (x0 ^ 2 * x1 ^ 3 * x3) + (12 * x0 ^ 2 * x
1 ^ 3 * x2) + (12 * x0 ^ 3 * x2 * x3 ^ 2) + (x0 ^ 3 * x2 ^ 2 * x3) + (x0 ^ 3
* x1 * x3 ^ 2) + (12 * x0 ^ 3 * x1 * x2 ^ 2) + (12 * x0 ^ 3 * x1 ^ 2 * x3)
+ (x0 ^ 3 * x1 ^ 2 * x2) + (x1 * x2 ^ 2 * x3 ^ 3)
```

The next command computes the characteristic polynomial of a symbolic 2×2 matrix. When all is done, the terms of the polynomial are collected to form a coefficient vector:

Example 47

```
symbolic_characteristic_polynomial_2x2:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 13 -end \
▷  ▷  -define M -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -matrix 2 \
▷  ▷  ▷  -text "a,b,c,d" \
▷  ▷  -end \
▷  ▷  -define p -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -characteristic_polynomial lambda M \
▷  ▷  -end \
▷  ▷  -define p1 -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -expand p \
▷  ▷  ▷  -write_trees_during_expand \
▷  ▷  -end \
▷  ▷  -define coeffs -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -collect p1 lambda \
▷  ▷  -end
```

The output is:

$$coeffs = \begin{bmatrix} (ad) + (12bc) \\ (12d) + (12a) \\ 1 \end{bmatrix}$$

Let us do the characteristic polynomial of a symbolic 3×3 matrix:

Example 48

```
symbolic_characteristic_polynomial_3x3:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 13 -end \
▷  ▷  -define M -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -matrix 3 \
▷  ▷  ▷  -text "a,b,c,d,e,f,g,h,i" \
▷  ▷  -end \
▷  ▷  -define p -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -characteristic_polynomial lambda M \
▷  ▷  -end \
▷  ▷  -define p1 -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -expand p \
▷  ▷  -end \
▷  ▷  -define coeffs -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -collect p1 lambda \
▷  ▷  -end
▷  #pdflatex p1.tex
▷  #$(OPEN) p1.pdf
```

The output is:

$$coeffs = \begin{bmatrix} (cdh) + (bfg) + (12bdi) + (12afh) + (aei) + (12ceg) \\ (12ei) + (cg) + (bd) + (12ai) + (12ae) + (fh) \\ a + e + i \\ 12 \end{bmatrix}$$

Let us compare Orbiter and Maple with an example. We create an object and then substitute values over the field $\mathbb{F}_{13}$. Here is the Orbiter code:

Example 49

```
symbolic_object1:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 13 -end \
▷  ▷  -define M -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -text "a+2*bˆ2*(x-y)+3*-(2*cˆ4+3*d-(a*b*c)ˆ2)ˆ3" \
▷  ▷  -end
```

The output is:

$$M = \ a + (10((3d) + (12a^2b^2c^2) + (2c^4)^3)) + (2b^2(x + (12y)))$$

Let us substitute some values and evaluate the formula:

Example 50

```
symbolic_object1_evaluate:
▷ $(ORBITER) -v 3 \
▷ ▷ -define Fq -finite_field -q 13 -end \
▷ ▷ -define f -symbolic_object \
▷ ▷ ▷ -field Fq \
▷ ▷ ▷ -text "a+2*bˆ2*(x-y)+3*-(2*cˆ4+3*d-(a*b*c)ˆ2)ˆ3" \
▷ ▷ -end \
▷ ▷ -define assignment -symbolic_object \
▷ ▷ ▷ -field Fq \
▷ ▷ ▷ -text "1,2,3,4,5,6" \
▷ ▷ -end \
▷ ▷ -define eval -symbolic_object \
▷ ▷ ▷ -field Fq \
▷ ▷ ▷ -substitute "a,b,c,d,x,y" f assignment \
▷ ▷ -end \
```

The value obtained after substitution is:

$$eval = \begin{bmatrix} 4 \end{bmatrix}$$

Let us perform the same task in Maple:

$$f := a + 2 * b^2 * (x - y) + 3 * (-(2 * c^4 + 3 * d - (a * b * c)^2)^3)$$

$$f := a + 2b^2 (x - y) - 3 \left(-a^2b^2c^2 + 2c^4 + 3d\right)^3 \tag{2.1}$$

$$subs(a = 1, b = 2, c = 3, d = 4, x = 5, y = 6, f) \mod 13;$$

$$4 \tag{2.2}$$

Let us create the general equation of the Eckardt surface using a makefile variable:

Example 51

```
ECKARDT_SURFACE_EQN="a*X3ˆ3-a*bˆ2*(X0ˆ2+X1ˆ2+X2ˆ2)*X3+bˆ3*(aˆ2+1)*X0*X1*X2"
```

The following command parses the equation:

Example 52

```
symbolic_Eckardt_surface:
▷ $(ORBITER) -v 3 \
```

```
▷  ▷  -define F -finite_field -q 13 -end \
▷  ▷  -define M -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -managed_variables "X0,X1,X2,X3" \
▷  ▷  ▷  -text $(ECKARDT_SURFACE_EQN) \
▷  ▷  -end
```

The output is:

$$M = \ (12X3ab^2(X0^2 + X1^2 + X2^2)) + (X0X1X2b^3(1 + a^2)) + (X3^3a)$$

We specify $a = 3$ and $b = 1$ by issuing the following command:

Example 53

```
symbolic_Eckardt_surface_q13_a3_b1:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 13 -end \
▷  ▷  -define M -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -managed_variables "X0,X1,X2,X3" \
▷  ▷  ▷  -text $(ECKARDT_SURFACE_EQN) \
▷  ▷  -end \
▷  ▷  -define L -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -text "3,1" \
▷  ▷  -end \
▷  ▷  -define M1 -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -substitute "a,b" M L \
▷  ▷  -end \
▷  ▷  -define M2 -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -expand M1 \
▷  ▷  ▷  -write_trees_during_expand \
▷  ▷  -end
▷  pdflatex M2.tex
▷  $(OPEN) M2.pdf
```

The output is:

$$M2 = \Big[\ (10X2^2X3) + (10X1^2X3) + (10X0X1X2) + (10X0^2X3) + (3X3^3) \ \Big]$$

Let us consider the 4 parameter normal form of a cubic surface from [14]. A makefile variable can be used to define the equation in text form:

Example 54

```
F_abcd_eqn="-(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*(b - d)*X0^2*X2 \
+ (a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*(a + b - c - d)*X0*X1*X2 \
```

```
+ (a^2*c - a^2*d - a*c^2 + b*c^2 + a*d - b*c)*(b - d)*X0*X1*X3 \
- (a*d - b*c)*(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*X0*X2^2 \
- (a^2*c*d - a*b*c^2 - a^2*d + a*b*d + b*c^2 - b*c*d)*(b - d)*X0*X2*X3 \
- (a - c)*(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*X1^2*X2 \
- (a - c)*(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*X1^2*X3 \
+ (a*d - b*c)*(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*X1*X2^2 \
+ ((1+1)*a^2*b*c*d - a^2*b*d^2 - (1+1)*a^2*c*d^2 \
- (1+1)*a*b^2*c^2 + a*b^2*c*d + (1+1)*a*b*c^2*d + a*b*c*d^2 \
- b^2*c^2*d - a^2*b*c + a^2*c*d + a^2*d^2 + a*b^2*c + a*b*c^2 \
- (1+1+1+1)*a*b*c*d - a*c^2*d + a*c*d^2 + b^2*c^2)*X1*X2*X3 \
+ c*a*(a*d - b*c - a + b + c - d)*(b - d)*X1*X3^2"
```

The following command parses the equation.

Example 55

```
symbolic_Fabcd:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 13 -end \
▷  ▷  -define M -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -managed_variables "X0,X1,X2,X3" \
▷  ▷  ▷  -text $(F_abcd_eqn) \
▷  ▷  -end
```

The output in text form is: `0 :`

```
(X0 * X1 * X2 * (a + b + (12 * c) + (12 * d)) * ((12 * a * b * d) + (12 * a
* c * d) + (b * c * d) + (a * d) + (12 * b * c) + (a * b * c))) + (X0 * X1 *
X3 * (b + (12 * d)) * ((12 * a ^ 2 * d) + (12 * a * c ^ 2) + (b * c ^ 2) +
(a * d) + (12 * b * c) + (a ^ 2 * c))) + (12 * X0 * X2 ^ 2 * ((12 * a * b *
d) + (12 * a * c * d) + (b * c * d) + (a * d) + (12 * b * c) + (a * b * c))
* ((12 * b * c) + (a * d))) + (12 * X0 * X2 * X3 * (b + (12 * d)) * ((12 * a
* b * c ^ 2) + (12 * a ^ 2 * d) + (a * b * d) + (b * c ^ 2) + (12 * b * c *
d) + (a ^ 2 * c * d))) + (12 * X1 ^ 2 * X2 * ((12 * a * b * d) + (12 * a *
c * d) + (b * c * d) + (a * d) + (12 * b * c) + (a * b * c)) * (a + (12 * c)
)) + (12 * X1 ^ 2 * X3 * ((12 * a * b * d) + (12 * a * c * d) + (b * c * d)
+ (a * d) + (12 * b * c) + (a * b * c)) * (a + (12 * c))) + (X1 * X2 ^ 2 * (
(12 * a * b * d) + (12 * a * c * d) + (b * c * d) + (a * d) + (12 * b * c) +
(a * b * c)) * ((12 * b * c) + (a * d))) + (X1 * X2 * X3 * ((12 * a ^ 2 * b
* d ^ 2) + (11 * a ^ 2 * c * d ^ 2) + (11 * a * b ^ 2 * c ^ 2) + (a * b ^ 2
* c * d) + (2 * a * b * c ^ 2 * d) + (a * b * c * d ^ 2) + (12 * b ^ 2 * c
^ 2 * d) + (12 * a ^ 2 * b * c) + (a ^ 2 * c * d) + (a ^ 2 * d ^ 2) + (a * b
^ 2 * c) + (a * b * c ^ 2) + (9 * a * b * c * d) + (12 * a * c ^ 2 * d) + (
a * c * d ^ 2) + (b ^ 2 * c ^ 2) + (2 * a ^ 2 * b * c * d))) + (X1 * X3 ^ 2
* a * c * (b + c + (12 * a) + (12 * b * c) + (12 * d) + (a * d)) * (b + (12
* d))) + (12 * X0 ^ 2 * X2 * (b + (12 * d)) * ((12 * a * b * d) + (12 * a *
c * d) + (b * c * d) + (a * d) + (12 * b * c) + (a * b * c)))
```

The next command expands the equation. The domain of coefficients is the finite field $\mathbb{F}_{31}$.

Example 56

```
symbolic_Fabcd_expand:
▷  $(ORBITER) -v 3 \
▷  ▷  -define Fq -finite_field -q 31 -end \
▷  ▷  -define Fabcd -symbolic_object \
▷  ▷  ▷  -field Fq \
▷  ▷  ▷  -managed_variables "X0,X1,X2,X3" \
▷  ▷  ▷  -text $(F_abcd_eqn) \
▷  ▷  -end \
▷  ▷  -define Fabcd_e -symbolic_object \
▷  ▷  ▷  -field Fq \
▷  ▷  ▷  -managed_variables "X0,X1,X2,X3" \
▷  ▷  ▷  -expand Fabcd \
▷  ▷  ▷  -write_trees_during_expand \
▷  ▷  -end
```

The expanded equation has 127 terms.

Over the field $\mathbb{F}_{31}$, let us substitute $(a, b, c, d) = (25, 5, 5, 25)$ in the general equation. The following command does the job:

Example 57

```
symbolic_Fabcd_q31_E18:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 31 -end \
▷  ▷  -define M -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -managed_variables "X0,X1,X2,X3" \
▷  ▷  ▷  -text $(F_abcd_eqn) \
▷  ▷  -end \
▷  ▷  -define L -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -text "25,5,5,25" \
▷  ▷  -end \
▷  ▷  -define M1 -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -substitute "a,b,c,d" M L \
▷  ▷  -end \
▷  ▷  -define M2 -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -expand M1 \
▷  ▷  ▷  -write_trees_during_expand \
▷  ▷  -end
▷  pdflatex M2.tex
▷  $(OPEN) M2.pdf
```

The output is:

$$M2 = \Big[\ (13X1X2X3) + (22X1X2^2) + (22X1^2X3) + (22X1^2X2) + (9X0X2^2) + (9X0^2X2) + (22X1X3^2) \ \Big]$$

It is also possible to use implicit substitution to achieve the same effect, as shown in the following command:

Example 58

```
symbolic_Fabcd_q31_E18_with_implicit_substitution:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 31 -end \
▷  ▷  -define a -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -text "25" \
▷  ▷  -end \
▷  ▷  -define b -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -text "5" \
▷  ▷  -end \
▷  ▷  -define c -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -text "5" \
▷  ▷  -end \
▷  ▷  -define d -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -text "25" \
▷  ▷  -end \
▷  ▷  -define M -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -managed_variables "X0,X1,X2,X3" \
▷  ▷  ▷  -text $(F_abcd_eqn) \
▷  ▷  -end \
▷  ▷  -define M2 -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -expand M \
▷  ▷  -end
▷  pdflatex M2.tex
▷  $(OPEN) M2.pdf
```

The output is again

$$M2 = \ (13X1X2X3) + (22X1X2^2) + (22X1^2X3) + (22X1^2X2) + (9X0X2^2) + (9X0^2X2) + (22X1X3^2)$$

We can use the option `-do_not_simplify` to see the syntax tree exactly after the moment of substitution, before the nodes are evaluated: So, the command

Example 59

```
symbolic_Fabcd_q31_E18_with_implicit_substitution_do_not_simplify:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 31 -end \
▷  ▷  -define a -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -text "25" \
▷  ▷  -end \
▷  ▷  -define b -symbolic_object \
▷  ▷  ▷  -field F \
```

```
▷ ▷ ▷ -text "5" \
▷ ▷ -end \
▷ ▷ -define c -symbolic_object \
▷ ▷ ▷ -field F \
▷ ▷ ▷ -text "5" \
▷ ▷ -end \
▷ ▷ -define d -symbolic_object \
▷ ▷ ▷ -field F \
▷ ▷ ▷ -text "25" \
▷ ▷ -end \
▷ ▷ -define M -symbolic_object \
▷ ▷ ▷ -field F \
▷ ▷ ▷ -managed_variables "X0,X1,X2,X3" \
▷ ▷ ▷ -text $(F_abcd_eqn) \
▷ ▷ ▷ -do_not_simplify \
▷ ▷ -end
```

produces the output

```
0 :
((30 * ((25 * 5 * 5) + (30 * (25 * 5 * 25)) + (30 * (25 * 5 * 25)) + (5 * 5
* 25) + (25 * 25) + (30 * (5 * 5)))) * (5 + (30 * 25)) * X0 ^ 2 * X2) + (((2
5 * 5 * 5) + (30 * (25 * 5 * 25)) + (30 * (25 * 5 * 25)) + (5 * 5 * 25) + (2
5 * 25) + (30 * (5 * 5))) * (25 + 5 + (30 * 5) + (30 * 25)) * X0 * X1 * X2)
+ (((25 ^ 2 * 5) + (30 * (25 ^ 2 * 25)) + (30 * (25 * 5 ^ 2)) + (5 * 5 ^ 2)
+ (25 * 25) + (30 * (5 * 5))) * (5 + (30 * 25)) * X0 * X1 * X3) + (30 * (((2
5 * 25) + (30 * (5 * 5))) * ((25 * 5 * 5) + (30 * (25 * 5 * 25)) + (30 * (25
* 5 * 25)) + (5 * 5 * 25) + (25 * 25) + (30 * (5 * 5))) * X0 * X2 ^ 2)) + (
30 * (((25 ^ 2 * 5 * 25) + (30 * (25 * 5 * 5 ^ 2)) + (30 * (25 ^ 2 * 25)) +
(25 * 5 * 25) + (5 * 5 ^ 2) + (30 * (5 * 5 * 25))) * (5 + (30 * 25)) * X0 *
X2 * X3)) + (30 * ((25 + (30 * 5)) * ((25 * 5 * 5) + (30 * (25 * 5 * 25)) +
(30 * (25 * 5 * 25)) + (5 * 5 * 25) + (25 * 25) + (30 * (5 * 5))) * X1 ^ 2 *
X2)) + (30 * ((25 + (30 * 5)) * ((25 * 5 * 5) + (30 * (25 * 5 * 25)) + (30
* (25 * 5 * 25)) + (5 * 5 * 25) + (25 * 25) + (30 * (5 * 5))) * X1 ^ 2 * X3)
) + (((25 * 25) + (30 * (5 * 5))) * ((25 * 5 * 5) + (30 * (25 * 5 * 25)) + (
30 * (25 * 5 * 25)) + (5 * 5 * 25) + (25 * 25) + (30 * (5 * 5))) * X1 * X2 ^
2) + ((((1 + 1) * 25 ^ 2 * 5 * 5 * 25) + (30 * (25 ^ 2 * 5 * 25 ^ 2)) + (30
* ((1 + 1) * 25 ^ 2 * 5 * 25 ^ 2)) + (30 * ((1 + 1) * 25 * 5 ^ 2 * 5 ^ 2))
+ (25 * 5 ^ 2 * 5 * 25) + ((1 + 1) * 25 * 5 * 5 ^ 2 * 25) + (25 * 5 * 5 * 25
^ 2) + (30 * (5 ^ 2 * 5 ^ 2 * 25)) + (30 * (25 ^ 2 * 5 * 5)) + (25 ^ 2 * 5
* 25) + (25 ^ 2 * 25 ^ 2) + (25 * 5 ^ 2 * 5) + (25 * 5 * 5 ^ 2) + (30 * ((1
+ 1 + 1 + 1) * 25 * 5 * 5 * 25)) + (30 * (25 * 5 ^ 2 * 25)) + (25 * 5 * 25 ^
2) + (5 ^ 2 * 5 ^ 2)) * X1 * X2 * X3) + (5 * 25 * ((25 * 25) + (30 * (5 * 5
)) + (30 * 25) + 5 + 5 + (30 * 25)) * (5 + (30 * 25)) * X1 * X3 ^ 2)
```

Chapter 3

Basic Algebra

3.1 Basic Algebra and Number Theory

In Table 3.1, global Orbiter commands for Algebra are summarized.

Table 3.2 shows Orbiter commands for basic number theory, including integer factor rings and the Euclidean algorithm.

In Table 3.3, some number theoretic commands are shown.

To compute primitive roots, there exists a randomized fast algorithm. On the other hand, there is also an algorithm that finds the smalles primitive root. For instance,

Example 60

```
PR101:
▷  $(ORBITER) -v 5 -primitive_root 101
```

finds a primitive root modulo 101 using the randomized algorithm. The command find the root 75. On the other hand, the command

Example 61

```
PR101_smallest:
▷  $(ORBITER) -v 5 -smallest_primitive_root 101
```

finds the smallest primitive root mot 101, which turns out to be 2.

Example 62

```
PR29:
▷  $(ORBITER) -v 1 -smallest_primitive_root 29
```

computes a primitive root modulo 29. The answer in this case is 2.

It is possible to compute primitive roots for all prime numbers in a given interval. The command

Global Algebra Commands

Command	Arguments	Purpose
-primitive_root	p	Computes a primitive root modulo p.
-smallest_primitive_root	p	Computes the smallest primitive root modulo p.
-smallest_primitive_root_interval	m1 m2	Computes the smallest primitive root modulo m for all m with $m1 \le m \le m2$.
-number_of_primitive_roots_interval	m1 m2	Counts the number of primitive roots modulo m for all m with $m1 \le m \le m2$.
-inverse_mod	a p	Computes the modular inverse of a modulo p, i.e. an integer b with $ab \equiv 1$ mod p.
-extended_gcd	a b	Computes integers g, u, and v such that $g = \gcd(a, b) = ua + vb$.
-power_mod	a k n	Compute the power a^k mod n.
-discrete_log	y a m	Computes x such that $a^x \equiv y$ mod m. Employs a brute force algorithm.
-square_root	n	computes $\lfloor \sqrt{a} \rfloor$ over the integers.
-square_root_mod	a p	Computes an integer b such that $b^2 \equiv a$ mod p. Here, p must be a prime number.
-all_square_roots_mod_n	a n	Computes all integers b such that $b^2 \equiv a$ mod n.
-count_subprimitive	Q_max H_max	Count subprimitive polynomials for $q \le Q\_max$
-character_table_symmetric_group	n	Compute the character table of the symmetric group of degree n.
-make_A5_in_PSL_2_q	q	Compute generators for a subgroup of $\mathrm{PSL}(2, q)$ isomorphic to $\mathrm{Alt}(5)$.
-order_of_q_mod_n	q $n_{\min}$ $n_{\max}$	Computes the order $\mathrm{ord}(q, n)$ of q modulo n for all n with $n_{\min} \le n \le n_{\max}$ for which $\gcd(n, q) = 1$. Also computes $\varphi(n)$ and $\varphi(n)/\mathrm{ord}(q, n)$.
-eulerfunction_interval	n1 n2	Compute Euler's totient function $\varphi(n)$ for all n with $n1 \le n \le n2$.
-young_symmetrizer	n	Compute the Young symmetrizer in the symmetric group of degree n.
-young_symmetrizer_sym_4		Compute the Young symmetrizer in the symmetric group of degree 4.
-draw_mod_n	descr	Draw a diagram with n vertices equally spaced out on a circle.
-power_function_mod_n	k n	
-all_rational_normal_forms	F d	Compute all rational normal forms of elements in $\mathrm{GL}(d, F)$, where F is a field.
-eigenstuff	F n coeffs fname	Compute the eigenvalues and eigenvectors of a matrix.
-jacobi	a p	Computes the Jacobi symbol $\left(\frac{a}{p}\right)$
-smith_normal_form	M	Compute the Smith Normal Form of the matrix M.

Table 3.1: Global Algebra Commands

Basic Number Theory Commands

Command	Arguments	Purpose
-draw_mod_n	descr	Draws the integers modulo n on a circle.
-Chinese_remainders	R M	Solves a system of congruences with remainders R and moduli M. R and M must be vectors whose labels are given.

Table 3.2: Basic Number Theory Commands

Number Theoretic Commands

Command	Arguments	Purpose
-sift_smooth	a n primes	Computes all smooth numbers in the interval $[a, a+n-1]$. Smooth means that they factor completely over the list of primes given.
-random	n fname	Creates n random numbers and writes them to the csv file fname
-random_last	n	Creates n random numbers prints the last one
-affine_sequence	a b p	Splits the interval $[0, p-1]$ into affine sequences of the form $x_{n+1} = ax_n + b \mod p$

Table 3.3: Number Theoretic Commands

Example 63

```
PR_2_1000:
▷ $(ORBITER) -v 1 -smallest_primitive_root_interval 2 1000
▷ $(ORBITER) -v 1 -csv_file_latex 1 \
▷ ▷ primitive_element_table_2_1000.csv
▷ pdflatex primitive_element_table_2_1000.tex
▷ $(OPEN) primitive_element_table_2_1000.pdf
```

finds all primitive roots for all primes between 2 and 1000.

The command

Example 64

```
PE_number_2_1000:
▷ $(ORBITER) -v 1 -number_of_primitive_roots_interval 2 1000
▷ $(ORBITER) -v 1 -csv_file_latex 1 table_number_of_pe_2_1000.csv
▷ pdflatex table_number_of_pe_2_1000.tex
▷ $(OPEN) table_number_of_pe_2_1000.pdf
```

determines the number of primitive elements for all primes between 2 and 1000.

The command

Example 65

```
PR_915839:
▷  $(ORBITER) -v 5 -primitive_root 915839
```

computes a primitive root modulo 915839. The answer is 43085. Let us check this result. The command

Example 66

```
PR_915839_check:
▷  $(ORBITER) -v 5 -power_mod 43085 49842 915839
```

computes

$$43085^{49842} \mod 915839$$

which is 487320.

The command `-discrete_log` can be used to compute the discrete logarithm of a modulo p with respect to b. This means, a number k is computed such that

$$b^k \equiv a \mod p.$$

For instance, the discrete log of 487320 with respect to the base 43085 modulo 915839 is 49842, based on the previous example. We can compute the discrete logarithm using the command

Example 67

```
DL_915839:
▷  $(ORBITER) -v 5 -discrete_log 487320 43085 915839
```

This command can be quite expensive.

Computing inverses modulo a prime p is possible using the `-inverse_mod` command. The command

Example 68

```
IM:
▷  $(ORBITER) -v 5 -inverse_mod 1865025205 2147483647
```

computes the inverse of 1865025205 modulo 2147483647 which is 579785381.

A different way of computing the inverse is by using the 1-trick. This approach computes the gcd of two numbers a and b, say, and writes

$$\gcd(a, b) = ua + vb$$

for some $u, v \in \mathbb{Z}$. The `-extended_gcd` command can be used. For instance, the following command computes the gcd of $a = 2147483647$ and $b = 1865025205$.

Example 69

```
IM_gcd:
▷  $(ORBITER) -v 5 -extended_gcd 1865025205 2147483647
```

The output is

$$1 = -503526232 * 2147483647 + 579785381 * 1865025205,$$

from which we see that $\gcd(a, b) = 1$ and $u = -503526232$ and $v = 579785381$. which is the gcd written as a lattice combination of the input arguments. The inverse of 1865025205 mod 2147483647 is $v = 579785381$.

In order to compute the modular power

$$a^e \mod n,$$

the `-power_mod` command can be used. For instance,

Example 70

```
PM3a:
▷  $(ORBITER) -v 5 -power_mod 16807 1073741823 2147483647
```

computes 16807 raised to the power 1073741823 modulo 2147483647, which is 2147483646.

The modular square root of a modulo p is any x such that

$$x^2 \equiv a \mod p.$$

The command `-square_root_mod` can be used to compute modular square roots using the algorithm of Tonelli and Shanks (cf. [21]). For instance,

Example 71

```
sqrt_mod:
▷  $(ORBITER) -v 2 -square_root_mod 33 41
```

finds that the square root of 33 mod 41 is 19, i.e.

$$19^2 \equiv 33 \mod 41.$$

The command `order_of_q_mod_n` computes $\mathrm{ord}(q, n)$, the order of q modulo n, for all n with $n_{\min} \le n \le n_{\max}$ and $\gcd(n, q) = 1$. It also computes Euler's totient function $\varphi(n)$ and the cofactor $\varphi(n)/\mathrm{ord}(q, n)$. For instance,

Example 72

```
order_of_2_mod_n:
▷  $(ORBITER) -v 3 -order_of_q_mod_n 2 3 51
▷  $(ORBITER) -v 1 -csv_file_latex 1 \
▷  ▷  order_of_q_mod_n_q2_3_51.csv
▷  pdflatex order_of_q_mod_n_q2_3_51.tex
▷  $(OPEN) order_of_q_mod_n_q2_3_51.pdf
```

produces the output shown below

N	ORD	PHI	COF	N	ORD	PHI	COF	N	ORD	PHI	COF
3	2	2	1	53	52	52	1	103	51	102	2
5	4	4	1	55	20	40	2	105	12	48	4
7	3	6	2	57	18	36	2	107	106	106	1
9	6	6	1	59	58	58	1	109	36	108	3
11	10	10	1	61	60	60	1	111	36	72	2
13	12	12	1	63	6	36	6	113	28	112	4
15	4	8	2	65	12	48	4	115	44	88	2
17	8	16	2	67	66	66	1	117	12	72	6
19	18	18	1	69	22	44	2	119	24	96	4
21	6	12	2	71	35	70	2	121	110	110	1
23	11	22	2	73	9	72	8	123	20	80	4
25	20	20	1	75	20	40	2	125	100	100	1
27	18	18	1	77	30	60	2	127	7	126	18
29	28	28	1	79	39	78	2	129	14	84	6
31	5	30	6	81	54	54	1	131	130	130	1
33	10	20	2	83	82	82	1	133	18	108	6
35	12	24	2	85	8	64	8	135	36	72	2
37	36	36	1	87	28	56	2	137	68	136	2
39	12	24	2	89	11	88	8	139	138	138	1
41	20	40	2	91	12	72	6	141	46	92	2
43	14	42	3	93	10	60	6	143	60	120	2
45	12	24	2	95	36	72	2	145	28	112	4
47	23	46	2	97	48	96	2	147	42	84	2
49	21	42	2	99	30	60	2	149	148	148	1
51	8	32	4	101	100	100	1	151	15	150	10

The command

Example 73

```
Eulerfunction_150:
▷  $(ORBITER) -v 1 -eulerfunction_interval 1 150
▷  $(ORBITER) -v 1 -csv_file_latex 1 \
▷  ▷  table_eulerfunction_1_150.csv
▷  pdflatex table_eulerfunction_1_150.tex
▷  $(OPEN) table_eulerfunction_1_150.pdf
```

computes Euler's totient function for all integers n with $1 \leq n \leq 150$. The result is shown below:

N	PHI	N	PHI	N	PHI	N	PHI	N	PHI	N	PHI
1	1	26	12	51	32	76	36	101	100	126	36
2	1	27	18	52	24	77	60	102	32	127	126
3	2	28	12	53	52	78	24	103	102	128	64
4	2	29	28	54	18	79	78	104	48	129	84
5	4	30	8	55	40	80	32	105	48	130	48
6	2	31	30	56	24	81	54	106	52	131	130
7	6	32	16	57	36	82	40	107	106	132	40
8	4	33	20	58	28	83	82	108	36	133	108
9	6	34	16	59	58	84	24	109	108	134	66
10	4	35	24	60	16	85	64	110	40	135	72
11	10	36	12	61	60	86	42	111	72	136	64
12	4	37	36	62	30	87	56	112	48	137	136
13	12	38	18	63	36	88	40	113	112	138	44
14	6	39	24	64	32	89	88	114	36	139	138
15	8	40	16	65	48	90	24	115	88	140	48
16	8	41	40	66	20	91	72	116	56	141	92
17	16	42	12	67	66	92	44	117	72	142	70
18	6	43	42	68	32	93	60	118	58	143	120
19	18	44	20	69	44	94	46	119	96	144	48
20	8	45	24	70	24	95	72	120	32	145	112
21	12	46	22	71	70	96	32	121	110	146	72
22	10	47	46	72	24	97	96	122	60	147	84
23	22	48	16	73	72	98	42	123	80	148	72
24	8	49	42	74	36	99	60	124	60	149	148
25	20	50	20	75	40	100	40	125	100	150	40

A power map sends a to a^k for some fixed k. Orbiter can compute power maps modulo p. For instance, the following command computes the function $a \mapsto a^k \mod 11$:

Example 74

```
power_function_2_mod_11:
▷ $(ORBITER) -v 5 -power_function_mod_n 2 11
▷ $(ORBITER) -v 1 -csv_file_latex 1 power_function_k2_n11.csv
▷ pdflatex power_function_k2_n11.tex
▷ $(OPEN) power_function_k2_n11.pdf
```

The result is shown below:

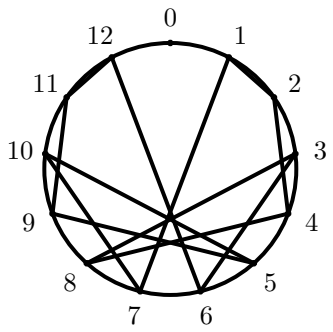
A	APOWK
0	0
1	1
2	4
3	9
4	5
5	3
6	3
7	5
8	9
9	4
10	1

It is sometimes helpful to draw the elements modulo n on a circle, using the vertices of an n-gon to represent the field elements. For instance, for the command

Example 75

```
draw_mod_13:
▷  $(ORBITER) -v 2 \
▷  ▷  -draw_options -embedded -end \
▷  ▷  -draw_mod_n \
▷  ▷  ▷  -n 13 \
▷  ▷  ▷  -file mod_13 \
▷  ▷  ▷  -power_cycle 2 \
▷  ▷  -end
▷  pdflatex mod_13_draw.tex
▷  $(OPEN) mod_13_draw.pdf
```

uses a 13-gon to represent the elements modulo 13 as shown below



Consecutive powers of 2 mod 13 are connected.

The command

Example 76

```
inverse_mod_a:
▷  $(ORBITER) -v 2 -inverse_mod 18059241 58014043
```

computes the inverse of 18059241 modulo 58014043.

Orbiter can compute the Smith-Normal-Form of matrices with integer entries. The command

Example 77

```
smith_normal_form_1:
▷  $(ORBITER) -v 3 \
▷  ▷  -define M -vector \
▷  ▷  ▷  -dense "1,2,3,4,5,6,7,8,9" \
▷  ▷  ▷  -format 3 \
▷  ▷  -end \
▷  ▷  -smith_normal_form M
```

computes the Smith-Normal-Form of the matrix

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

The Legendre symbol evaluates if a number a is a square modulo an odd prime p. By definition,

$$\left(\frac{a}{p}\right) = \begin{cases} 1 & \text{if there exists } r \text{ s.t. } r^2 \equiv a \mod p \\ -1 & \text{if there does not exist } r \text{ s.t. } r^2 \equiv a \mod p \\ 0 & \text{if } p \text{ divides } a. \end{cases}$$

The Jacobi symbol generalizes the Legendre symbol to allow non-prime bottom arguments. By definition,

$$\left(\frac{a}{b}\right) = \prod_{i=1}^{k} \left(\frac{a}{r_i}\right)^{e_i},$$

where

$$b = \prod_{i=1}^{k} r_i^{e_i}$$

is the prime factorization of b with pairwise distinct primes r_i. The Jacobi symbol agrees with the Legendre symbol whenever the bottom argument b is an odd prime. Because there is no ambiguity, the same notation is used for the Jacobi symbol as for the Legendre symbol. Orbiter can compute Jacobi symbols. For instance, the command

Example 78

```
jacobi_a:
▷  $(ORBITER) -v 5 -jacobi 2221 7817
```

computes the Jacobi symbol

$$\left(\frac{2221}{7817}\right).$$

In the Jacobi symbol, the denominator p has to be a positive odd integer. This command creates the file `jacobi_2221_7817.tex` which contains a detailed step-by-step description of the computation. After reformatting, the description looks like this:

$$\begin{aligned}
&\left(\tfrac{2221}{7817}\right)\\
&= \left(\tfrac{7817}{2221}\right) \cdot (-1)^{\frac{2221-1}{2}\cdot\frac{7817-1}{2}}\\
&= \left(\tfrac{7817}{2221}\right)\\
&= \left(\tfrac{1154}{2221}\right)\\
&= \left(\tfrac{2}{2221}\right) \cdot \left(\tfrac{577}{2221}\right)\\
&= (-1)^{\frac{2221^2-1}{8}} \cdot \left(\tfrac{577}{2221}\right)\\
&= (-1) \cdot \left(\tfrac{577}{2221}\right)\\
&= (-1) \cdot \left(\tfrac{2221}{577}\right) \cdot (-1)^{\frac{577-1}{2}\cdot\frac{2221-1}{2}}\\
&= (-1) \cdot \left(\tfrac{2221}{577}\right)\\
&= (-1) \cdot \left(\tfrac{490}{577}\right)\\
&= (-1) \cdot \left(\tfrac{2}{577}\right) \cdot \left(\tfrac{245}{577}\right)\\
&= (-1) \cdot (-1)^{\frac{577^2-1}{8}} \cdot \left(\tfrac{245}{577}\right)\\
&= (-1) \cdot \left(\tfrac{245}{577}\right)\\
&= (-1) \cdot \left(\tfrac{577}{245}\right) \cdot (-1)^{\frac{245-1}{2}\cdot\frac{577-1}{2}}\\
&= (-1) \cdot \left(\tfrac{577}{245}\right)\\
&= (-1) \cdot \left(\tfrac{87}{245}\right)\\
&= (-1) \cdot \left(\tfrac{245}{87}\right) \cdot (-1)^{\frac{87-1}{2}\cdot\frac{245-1}{2}}\\
&= (-1) \cdot \left(\tfrac{245}{87}\right)\\
&= (-1) \cdot \left(\tfrac{71}{87}\right)\\
&= (-1) \cdot \left(\tfrac{87}{71}\right) \cdot (-1)^{\frac{71-1}{2}\cdot\frac{87-1}{2}}\\
&= \left(\tfrac{87}{71}\right)\\
&= \left(\tfrac{16}{71}\right)\\
&= \left(\tfrac{2}{71}\right)^4 \cdot \left(\tfrac{1}{71}\right)\\
&= \left((-1)^{\frac{71^2-1}{8}}\right)^4 \cdot \left(\tfrac{1}{71}\right)\\
&= \left(\tfrac{1}{71}\right)\\
&= 1
\end{aligned}$$

The answer 1 tells us that 2221 is a square modulo 7817. Because 7817 is prime, the Jacobi symbol and the Legendre symbol agree on this input pair. We can use the `square_root_mod` command to compute a square root of 2221 modulo 7817 and verify this fact. The command

Example 79

```
sqrt_mod_7817:
▷  ▷  $(ORBITER) -v 2 -square_root_mod 2221 7817
```

yields that 7634 is a square root. Indeed,

$$7634^2 \equiv 2221 \mod 7817.$$

The next command illustrates how to solve a system of congruences with coprime moduli using the Chinese remainder theorem. Suppose we want to find the integer x such that

$$\begin{aligned} x &\equiv 2 \mod 5 \\ x &\equiv 2 \mod 6 \\ x &\equiv 5 \mod 7 \end{aligned}$$

The following command creates one vector for the remainders and one for the moduli and then invokes the `-Chinese_remainders` command.

Example 80

```
Chinese_remainders_A:
▷  $(ORBITER) -v 2 \
▷  ▷  -define R -vector -dense "2,2,5" -end \
▷  ▷  -define M -vector -dense "5,6,7" -end \
▷  ▷  -Chinese_remainders R M
```

The answer is $x \equiv 152$ modulo 210.

The next example shows that the Chinese remainder algorithm is safe for large numbers. We pick two 10 digit prime numbers as moduli and solve

$$\begin{aligned} x &\equiv 2 \mod 2147483647 \\ x &\equiv 3 \mod 5915587277 \end{aligned}$$

using the command

Example 81

```
Chinese_remainders_C2:
▷  $(ORBITER) -v 2 \
▷  ▷  -define R -vector -dense "2,3" -end \
▷  ▷  -define M -vector -dense "2147483647,5915587277" -end \
▷  ▷  -Chinese_remainders R M
```

The answer is

$$x \equiv 5684294357108828365 \mod 12703626939758759219.$$

A quick check with Maple shows that

$$\begin{aligned} 5684294357108828365 \mod 2147483647 &\equiv 2 \\ 5684294357108828365 \mod 5915587277 &\equiv 3 \end{aligned}$$

as required.

Creating Finite Fields		
Command	Arguments	Purpose
-q	q	Specify the order of the field. Here, $q = p^k$ for some prime p and some positive integer k.
-override_polynomial	n	Specify the polynomial used to create the finite field. The polynomial is given as integer, using the base p representation. See Section 3.3.
-without_tables		Create the field without precomputing the tables.
-compute_related_fields		Compute information about all subfields.
-symbol	s	Use s as a symbol for the primitive element during printing.

Table 3.4: Creating Finite Fields

3.2 Finite Fields

Let $\mathbb{F}_q$ denote the finite field with q elements. Up to isomorphism, there is only one field of order q. In Orbiter, the elements of $\mathbb{F}_q$ are represented by the integers in the interval $[0, q-1]$.

Table 3.4 lists Orbiter commands for creating a finite field. For instance,

Example 82

```
F_2:
▷ $(ORBITER) -v 3 -list_arguments \
▷ ▷ -define F -finite_field -q 2 -end \
▷ ▷ -with F -do -finite_field_activity -cheat_sheet_GF -end
▷ pdflatex GF_2.tex
▷ $(OPEN) GF_2.pdf
```

creates the finite field $\mathbb{F}_2$ and produces a report for it.

Table 3.5 lists basic Orbiter activities for finite fields. More activities will follow in Section 3.3.
The command

Example 83

```
F_7:
▷ $(ORBITER) -v 3 \
▷ ▷ -define F -finite_field -q 7 -end \
▷ ▷ -with F -do -finite_field_activity -cheat_sheet_GF -end
▷ pdflatex GF_7.tex
▷ $(OPEN) GF_7.pdf
```

creates a cheat sheet for $\mathbb{F}_7$ as shown below. The element α is a primitive element.

Finite Field Activities

Command	Arguments	Purpose
-cheat_sheet_GF		Produce a cheat sheet in latex which shows information about the field, including addition and multiplication tables.
-export_tables		Export addition and multiplication table to a csv-file.
-polynomial_division	A B	Perform polynomial division: Divide B into A.
-extended_gcd_for_polynomials	A B	Compute the GCD of G of two polynomials A and B. Determine two polynomials U and V such that $G = UA + VB$.
-polynomial_mult_mod	A B M	Perform modular polynomial multiplication. Compute $A \cdot B \mod M$.
-polynomial_power_mod	A n M	Perform modular polynomial exponentiation. Compute $A^n \mod M$.
-Berlekamp_matrix	A	Compute the Berlekamp matrix associated with the polynomial A.
-polynomial_find_roots	A	Computes the roots of the polynomial A.
-product_of	v	Compute the product of all field elements in the vector v.
-sum_of	v	Compute the sum of all field elements in the vector v.
-negate	v	Negate each field element in the vector v.
-inverse	v	Compute the multiplicative inverse of each field element in the vector v.
-power_map	k v	Compute the k-th power of each field element in the vector v.
-parse_and_evaluate	label managed-vars formula parameters	Parse the given formula with the given managed variables and evaluate at the given values of the parameters.
-evaluate	label parameters	Parse the given formula and evaluate at the given values of the parameters.

Table 3.5: Finite Field Activities

$Z_i = \log_\alpha(1 + \alpha^i)$

i	γ_i	$-\gamma_i$	γ_i^{-1}	$\log_\alpha(\gamma_i)$	α^i	Z_i
0	$0 = 0$	0	DNE	DNE	1	2
1	$1 = 1$	6	1	0	3	4
2	$2 = \alpha^2$	5	4	2	2	1
3	$3 = \alpha$	4	5	1	6	DNE
4	$4 = \alpha^4$	3	2	4	4	5
5	$5 = \alpha^5$	2	3	5	5	3
6	$6 = \alpha^3$	1	6	3	1	2

$+$	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6
1	1	2	3	4	5	6	0
2	2	3	4	5	6	0	1
3	3	4	5	6	0	1	2
4	4	5	6	0	1	2	3
5	5	6	0	1	2	3	4
6	6	0	1	2	3	4	5

$\cdot$	1	2	3	4	5	6
1	1	2	3	4	5	6
2	2	4	6	1	3	5
3	3	6	2	5	1	4
4	4	1	5	2	6	3
5	5	3	1	6	4	2
6	6	5	4	3	2	1

$3^0 \equiv 1$
$3^1 \equiv 3$
$3^2 \equiv 2$
$3^3 \equiv 6$
$3^4 \equiv 4$
$3^5 \equiv 5$
$3^6 \equiv 1$

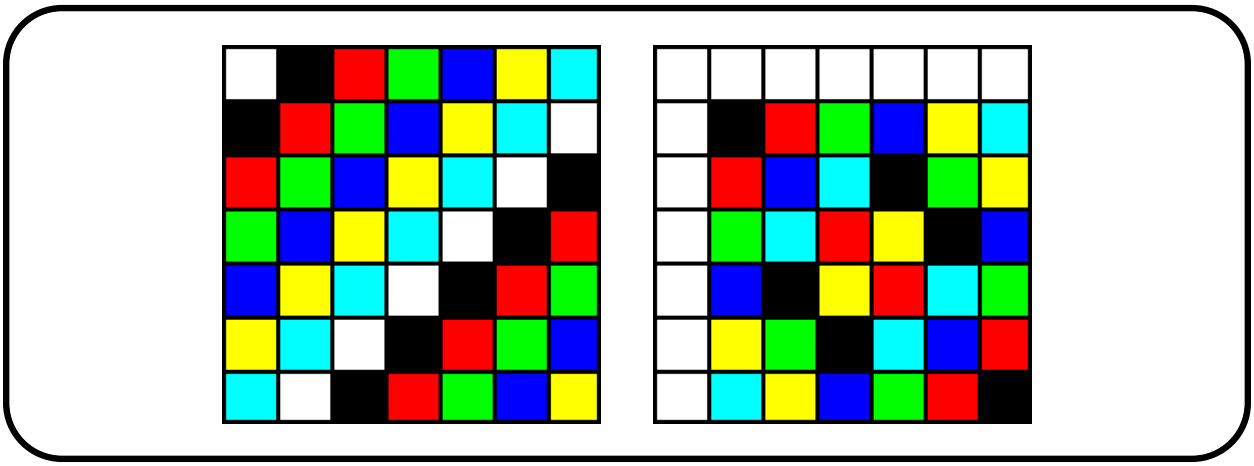
The command

Example 84

```
F_7_tables:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 7  -end \
▷  ▷  -with F -do -finite_field_activity -export_tables -end
▷  $(ORBITER) -v 3 \
▷  ▷  -define all_one -vector -repeat 1 7 -end \
▷  ▷  -draw_matrix -input_csv_file GF_q7_table_add.csv \
▷  ▷  ▷  -box_width 40 -bit_depth 24 \
▷  ▷  ▷  -partition 3 all_one all_one \
▷  ▷  -end \
```

```
▷  ▷  -draw_matrix -input_csv_file GF_q7_table_mul.csv \
▷  ▷  ▷  -box_width 40 -bit_depth 24 \
▷  ▷  ▷  -partition 3 all_one all_one \
▷  ▷  -end
▷  convert GF_q7_table_add_draw.bmp GF_q7_table_add_draw.png
▷  convert GF_q7_table_mul_draw.bmp GF_q7_table_mul_draw.png
```

creates the addition and multiplication tables for the field, shown below



Suppose we want to check Wilson's theorem:

$$\prod_{a \in \mathbb{F}_q^\times} a = -1.$$

The following command verifies this statement for $q = 11$. We first create a vector of all nonzero field elements, which we take as the integers from 1 to 10. After that, we use the `-product_of` finite field activity. This computes the product of the elements in the given set.

Example 85

```
F_11_product_of_all_nonzero_elements:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 11 -end \
▷  ▷  -define S -vector -field F -loop 1 11 1 -end \
▷  ▷  -with F -do -finite_field_activity -product_of S -end
```

The answer is 10, which is congruent to -1 mod 11 :

Suppose we want to create the Vandermonde matrix whose entries are x_i^j. Here $x_0, \ldots, x_{q-1}$ are the elements of the field $\mathbb{F}_q$ and j ranges from 0 to $q-1$. The following command does so for $q = 7$. The command also computes the inverse of the Vandermonde matrix.

Example 86

```
F_7_vandermonde:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 7 -end \
▷  ▷  -with F -do -finite_field_activity \
▷  ▷  ▷  -Vandermonde_matrix \
▷  ▷  -end
```

The output is shown below. The first matrix is $V = (x_i^j)$. The second matrix is V^{-1}:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 1 & 2 & 4 & 1 \\ 1 & 3 & 2 & 6 & 4 & 5 & 1 \\ 1 & 4 & 2 & 1 & 4 & 2 & 1 \\ 1 & 5 & 4 & 6 & 2 & 3 & 1 \\ 1 & 6 & 1 & 6 & 1 & 6 & 1 \end{bmatrix}, \quad \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 6 & 3 & 2 & 5 & 4 & 1 \\ 0 & 6 & 5 & 3 & 3 & 5 & 6 \\ 0 & 6 & 6 & 1 & 6 & 1 & 1 \\ 0 & 6 & 3 & 5 & 5 & 3 & 6 \\ 0 & 6 & 5 & 4 & 3 & 2 & 1 \\ 6 & 6 & 6 & 6 & 6 & 6 & 6 \end{bmatrix}$$

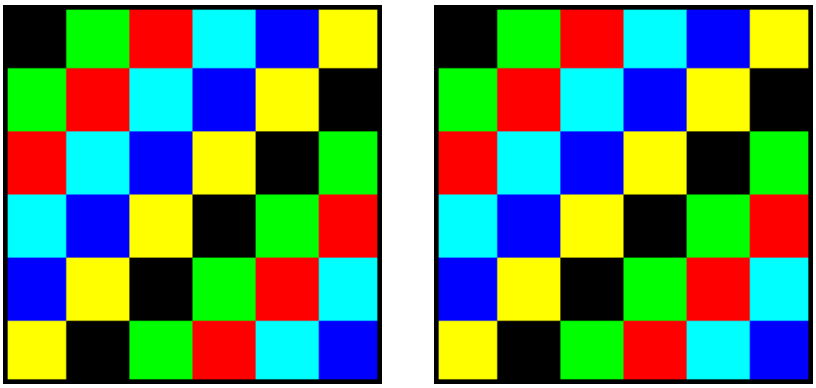
There is a second ordering of the elements which is used occasionally. In this labeling, every non-zero element is written as a power of a fixed primitive element. So, if α is a primitive element, we arrange the elements of $\mathbb{F}_p$ as

$$0, 1, \alpha, \alpha^2, \ldots, \alpha^{q-2}.$$

The cheat sheet contains this list of field elements at the very end. It is sometimes convenient to arrange the elements in the order of powers of a primitive element. In $\mathbb{F}_7$, 3 is a primitive element. The powers of 3 are

$$0, 3^0, 3^1, 3^2, \ldots, 3^6 = 0, 1, 3, 2, 6, 4, 5, 1 \mod 7.$$

With respect to this ordering of elements, the addition and multiplication tables become



For small field orders, the Orbiter employs precomputed tables for the arithmetic operations such as addition and multiplication and computing inverses. Precomputing these tables can be time-consuming. The option

`-without_tables` can be given to avoid precomputing tables. Of course, not by not precomputing the tables, arithmetic operations are somewhat slower than before. Here is an example. We create the field $\mathbb{F}_{101}$ without precomputed tables:

Example 87

```
F_101_wo:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 101 -without_tables -end \
▷  ▷  -with F -do -finite_field_activity -cheat_sheet_GF -end
▷  pdflatex GF_101.tex
▷  $(OPEN) GF_101.pdf
```

See Section 19.2 for a list of limitations of Orbiter. For instance, the possible range of q is restricted to the machine integers.

3.3 Extension Fields

Let E and F be fields. We say that E is an extension of F if F is a subfield of E. n this case, we write E/F. The degree of E/F is the dimension of E as a vector space over F. An example of a field extension is a field of the form $E = F(\alpha)$, where α is any element over F. Here, $F(\alpha)$ is the smallest field which contains F and α. If $\gamma \in E$ satisfies a polynomial equation with coefficients in F, then γ is called algebraic over F. The minimum polynomial of an element γ in E over F is the monic polynomial in $F[X]$ of lowest degree which has γ as a root. A field extension E/F is algebraic if every element in E is algebraic over F. In particular, $F(\alpha)$ is algebraic over F if α is. The degree of $F(\alpha)/F$ is the degree of the minimum polynomial of α over F.

In this section, we will consider algebraic extensions of finite fields. If $F = \mathbb{F}_q$ is a field of order q, then any algebraic extension E of F has order q^e where e is the degree of E over F. If $E = F(\alpha)$ is algebraic, the degree of E over F is the degree of the minimum polynomial of E over F. If $F = \mathbb{F}_q$ and $E = F(\alpha)$ is algebraic of degree e, then $|F| = q^e$. Every finite field E is of this form, where $F = \mathbb{F}_p$ and p is the characteristic of E.

Any such E can be constructed as a polynomial factorring of the ring $\mathbb{F}_p[X]$. For a polynomial $m(X)$ we consider the ideal

$$\mathbf{I}(m) = m(X)\mathbb{F}_p[X] = \{m(X)k(X) \mid k(X) \in \mathbb{F}_p[X]\}$$

of all polynomial multiples of $m(X)$. Under the assumption that $m(X)$ has degree $e > 1$ and is irreducible, the residue class ring

$$\mathbb{F}_p[X]/\mathbf{I}(m)$$

is a field with $q = p^e$ elements. Each residue class has a canonical representative. The canonical representative is the unique element in the residue class modulo $\mathbf{I}(m)$ which has degree less than e and leading coefficient one. So, for instance, for $q = 4 = 2^2$, we can pick the irreducible polynmial $m(X) = X^2 + X + 1$ over $\mathbb{F}_2$. We have four standard representatives modulo $\mathbf{I}(m)$, namely

$$\begin{gathered} 0, \\ 1, \\ X, \\ X + 1. \end{gathered}$$

Together, these make up a complete set of representatives of the residue classes modulo $\mathbf{I}(m)$, and hence can be identified with the elements of $\mathbb{F}_4$:

$$\mathbb{F}_4 = \{0,\ 1,\ X,\ X + 1\}.$$

The addition of polynomials is as in $\mathbb{F}_2[X]$, so

	0	1	X	$X+1$
0	0	1	X	$X+1$
1	1	0	$X+1$	X
X	X	$X+1$	0	1
$X+1$	$X+1$	X	1	0

To compute the multiplication table for the field $\mathbb{F}_4$. We can use polynomial arithmetic modulo $m(X)$: It is clear how multiplication by 0 or 1 works, so we need to focus on the polynomials X and $X + 1$:

$$\begin{array}{llll} X & \cdot X & = X^2 & \equiv X + 1 \quad \mod X^2 + X + 1, \\ X & \cdot (X + 1) & = X^2 + X & \equiv 1 \quad \mod X^2 + X + 1, \\ (X + 1) & \cdot X & = X^2 + X & \equiv 1 \quad \mod X^2 + X + 1, \\ (X + 1) & \cdot (X + 1) & = X^2 + 1 & \equiv X \quad \mod X^2 + X + 1, \end{array}$$

More Finite Field Activities

Command	Arguments	Purpose
`-trace`		Computes the partition of the field elements according to the value of their absolute trace.
`-norm`		Computes the partition of the field elements according to the value of their absolute norm.
`-normal_basis`	d	Computes a normal basis for $\mathbb{F}_{q^d}$ over $\mathbb{F}_q$.
`-nth_roots`	n	Compute the minimum polynomials of all n-th roots over F.
`-field_reduction`	label q m n v	Compute the field reduction from F to the subfield $\mathbb{F}_q$ of the $m \times n$ matrix whose coefficients are given in v.

Table 3.6: More Finite Field Activities

The multiplication table of $\mathbb{F}_4$ is

	0	1	X	$X+1$
0	0	0	0	0
1	0	1	X	$X+1$
X	0	X	$X+1$	1
$X+1$	0	$X+1$	1	X

Table 3.6 lists Orbiter activities for finite fields. This extends Table 3.5 in Section 3.3.

The isomorphism type of the resulting field only depends on the order q of the field, and not on the choice of the polynomial. However, for practical computations, the choice of the polynomial matters. For instance, results can only be shared between different computer algebra systems if the same polynomials are used. Orbiter has a large collection of polynomials built in. Besides these, a polynomial can be specified. The polynomials that Orbiter offers are in fact primitive, which means that the root α is a primitive element for the field $\mathbb{F}_q$. This just means that it is a generator of the multiplicative group. So, any non-zero element in $\mathbb{F}_q$ is a suitable power of α.

If $\mathbb{F}_q$ is an extension of the prime field $\mathbb{F}_p$, we use a different labeling. This time, we exploit the fact that $\mathbb{F}_q$ is a vector space over $\mathbb{F}_p$. Let α be a root of the irreducible polynomial $m(X) \in \mathbb{F}_p[X]$ used to create the field. Suppose that $q = p^e$, i.e., the degree of $m(X)$ is e. An $\mathbb{F}_p$-basis for the vector space $\mathbb{F}_q$ over $\mathbb{F}_p$ is given by the powers α^i, for $0 \le i < e$. Therefore, any element γ of $\mathbb{F}_q$ has a unique expression of the form

$$\gamma = \sum_{h=0}^{e-1} a_i \alpha^i, \quad 0 \le a_i < p \text{ for all } i.$$

The associated integer rank of γ is obtained by replacing α by p in this expression and evaluating the expression over the integers. So, the rank of γ is

$$\sum_{h=0}^{e-1} a_i p^i.$$

As γ ranges over all field element in $\mathbb{F}_q$, the rank values take on every value in the interval $[0, q-1]$. The ordering of elements of $\mathbb{F}_q$ according to these ranks is called the lexicographical ordering. The numerical rank of zero is 0 and the numerical rank of one is 1. The numerical rank of α, the primitive element, is p. The numerical ranks of the elements of the prime subfield are exactly the elements of $[0, p-1]$.

Orbiter Primitive Polynomials

q	Polynomial	Numerical	Relation
4	X^2+X+1	7	$\omega^2=\omega+1$
8	X^3+X^2+1	13	$\gamma^3=\gamma^2+1$
9	X^2+X+2	14	
16	X^4+X^3+1	25	$\delta^4=\delta^3+1$
25	X^2+X+2	22	
27	X^3+2X+1	34	
32	X^5+X^2+1	37	$\eta^5=\eta^2+1$
49	X^2+X+3	59	
64	X^6+X^5+1	97	
81	X^4+X^3+2	110	
121	X^2+4X+2	167	
125	X^3+X^2+X+2	86	
128	X^7+X^6+1	193	$\zeta^7=\zeta^6+1$
169	X^2+X+2	184	
243	X^5+2X+1	250	
256	$X^8+X^4+X^3+X^2+1$	285	
289	X^2+X+3	309	
343	X^3+3X+2	366	
361	X^2+X+2	382	
512	X^9+X^4+1	529	
529	X^2+2X+5	580	
625	X^4+X^3+X+2	326	
729	X^6+X^5+2	974	
841	X^2+5X+2	988	
961	X^2+2X+3	1026	
1024	$X^{10}+X^3+1$	1033	

Table 3.7: Orbiter Primitive Polynomials

The primitive polynomials used by Orbiter to create small finite fields are listed in Table 3.7. The relation is given using the Greek letter that is used in orbiter cheat sheets for that particular field.

Let us look at a few examples. The command

Example 88

```
F_4:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 4 -end \
▷  ▷  -with F -do -finite_field_activity -cheat_sheet_GF -end
▷  pdflatex GF_4.tex
▷  $(OPEN) GF_4.pdf
```

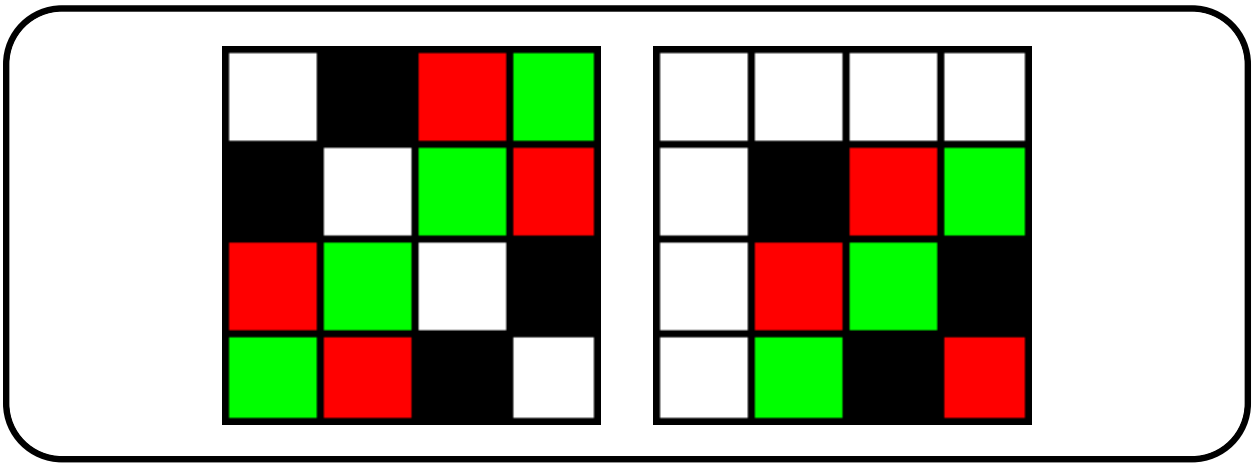
creates a report for the field $\mathbb{F}_4$.

The command

Example 89

```
F_4_tables:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 4 -end \
▷  ▷  -with F -do -finite_field_activity -export_tables -end
▷  $(ORBITER) -v 3 \
▷  ▷  -define all_one -vector -repeat 1 4 -end \
▷  ▷  -draw_matrix -input_csv_file GF_q4_table_add.csv \
▷  ▷  ▷  -box_width 40 -bit_depth 24 \
▷  ▷  ▷  -partition 3 all_one all_one \
▷  ▷  -end \
▷  ▷  -draw_matrix -input_csv_file GF_q4_table_mul.csv \
▷  ▷  ▷  -box_width 40 -bit_depth 24 \
▷  ▷  ▷  -partition 3 all_one all_one \
▷  ▷  -end
▷  convert GF_q4_table_add_draw.bmp GF_q4_table_add_draw.png
▷  convert GF_q4_table_mul_draw.bmp GF_q4_table_mul_draw.png
```

creates the addition and multiplication tables for the field:



The command

Example 90

```
F_16:
▷  $(ORBITER) -v 10 \
▷  ▷  -define F -finite_field -q 16 -compute_related_fields -end \
▷  ▷  -with F -do -finite_field_activity -cheat_sheet_GF -end
▷  pdflatex GF_16.tex
▷  $(OPEN) GF_16.pdf
```

creates a cheat sheet for $\mathbb{F}_{16}$.

polynomial: $X^4 + X^3 + 1 = 25$
$Z_i = \log_\alpha(1 + \alpha^i)$
Subfields:

Subfield	Polynomial	Numerical Rank
$\mathbb{F}_4$	$X^2 + X + 1$	7

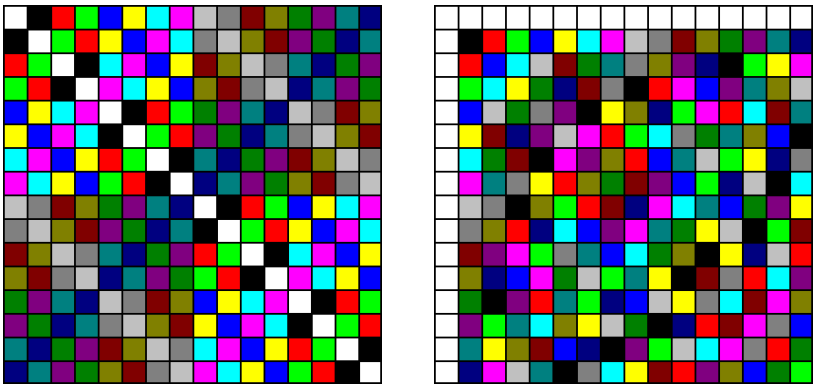
i	γ_i	$-\gamma_i$	γ_i^{-1}	$\log_\alpha(\gamma_i)$	α^i	Z_i	$\phi(\gamma_i)$	$T(\gamma_i)$	$N(\gamma_i)$	$T_2(\gamma_i)$	$N_2(\gamma_i)$
0	$0 = 0$	0	DNE	DNE	1	DNE	0	0	0	0	0
1	$1 = 1$	1	1	15	2	12	1	0	1	0	1
2	$\alpha = \delta$	2	12	1	4	9	4	1	1	11	11
3	$\alpha + 1 = \delta^{12}$	3	8	12	8	4	5	1	1	11	1
4	$\alpha^2 = \delta^2$	4	6	2	9	3	9	1	1	10	10
5	$\alpha^2 + 1 = \delta^9$	5	15	9	11	10	8	1	1	10	1
6	$\alpha^2 + \alpha = \delta^{13}$	6	4	13	15	8	13	0	1	1	11
7	$\alpha^2 + \alpha + 1 = \delta^7$	7	14	7	7	13	12	0	1	1	11
8	$\alpha^3 = \delta^3$	8	3	3	14	6	15	1	1	11	1
9	$\alpha^3 + 1 = \delta^4$	9	13	4	5	2	14	1	1	11	11
10	$\alpha^3 + \alpha = \delta^{10}$	10	11	10	10	5	11	0	1	0	11
11	$\alpha^3 + \alpha + 1 = \delta^5$	11	10	5	13	14	10	0	1	0	10
12	$\alpha^3 + \alpha^2 = \delta^{14}$	12	2	14	3	1	6	0	1	1	10
13	$\alpha^3 + \alpha^2 + 1 = \delta^{11}$	13	9	11	6	7	7	0	1	1	10
14	$\alpha^3 + \alpha^2 + \alpha = \delta^8$	14	7	8	12	11	2	1	1	10	10
15	$\alpha^3 + \alpha^2 + \alpha + 1 = \delta^6$	15	5	6	1	DNE	3	1	1	10	1

The command

Example 91

```
F_16_tables:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 16 -end \
▷  ▷  -with F -do -finite_field_activity -export_tables -end
▷  $(ORBITER) -v 3 \
▷  ▷  -define all_one -vector -repeat 1 16 -end \
▷  ▷  -draw_matrix -input_csv_file GF_q16_table_add.csv \
▷  ▷  ▷  -box_width 40 -bit_depth 24 \
▷  ▷  ▷  -partition 3 all_one all_one \
▷  ▷  -end \
▷  ▷  -draw_matrix -input_csv_file GF_q16_table_mul.csv \
▷  ▷  ▷  -box_width 40 -bit_depth 24 \
▷  ▷  ▷  -partition 3 all_one all_one \
▷  ▷  -end
▷  convert GF_q16_table_add_draw.bmp GF_q16_table_add_draw.png
▷  convert GF_q16_table_mul_draw.bmp GF_q16_table_mul_draw.png
```

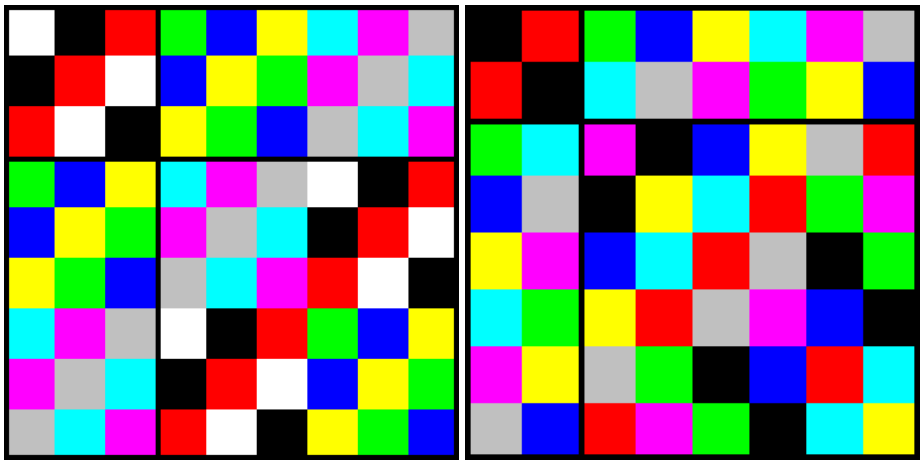
creates the addition and multiplication tables for the field:



Unlike other GAP [31] and Magma [16], Orbiter does not use Conway polynomials to create field extensions. Instead, it provides the option to override the polynomial used to create the finite field. For subfield relationships, the cheat sheet will indicate the irreducible polynomials of all subfields for a given field. The table below is taken from a cheat sheet for the field $\mathbb{F}_{64}$ generated by the polynomial $X^6 + X^5 + 1$, whose numerical rank is 97:

Subfield	Polynomial	Numerical rank
$\mathbb{F}_4$	$X^2 + X + 1$	7
$\mathbb{F}_8$	$X^3 + X + 1$	11

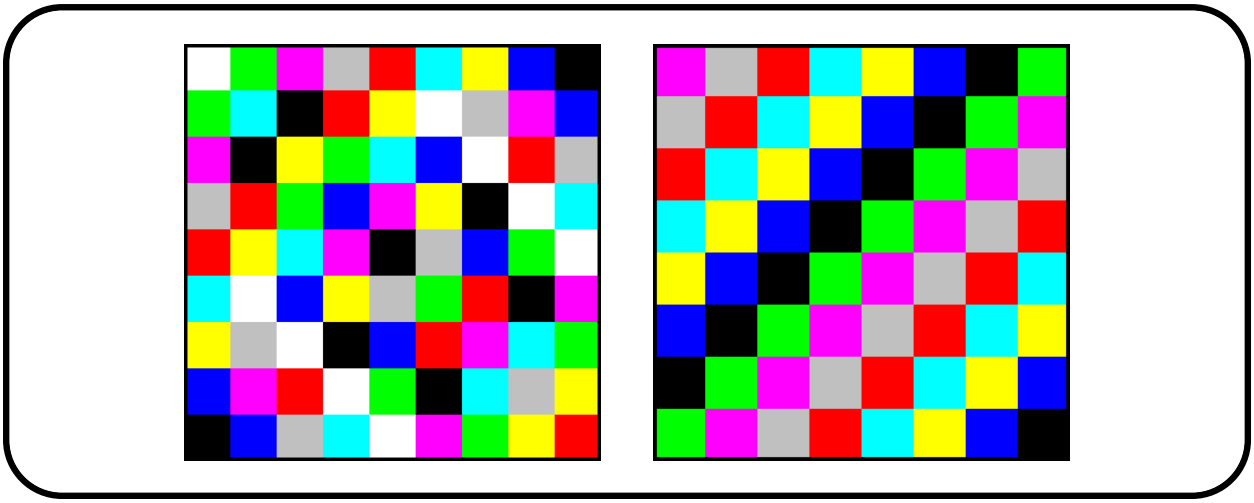
The lexicographic ordering has an interesting side-effect for the ordering of elements in extension fields. The elements of the prime subfield are always listed before any other elements in the extension field. For this reason, the addition and multiplication tables of the extension field contain the respective table of the prime field in the upper left corner, as shown below for the field $\mathbb{F}_9$:



Here, we omit the zero element in the multiplication table.

Orbiter uses primitive polynomials for creating extension fields. Because of this, the element α is always primitive. Since the numerical rank of α is p, this means that the rank p always represents a primitive

element in an extension field. For the addition and multiplication tables of $\mathbb{F}_9$ arranged with respect to powers of a primitive element, see below:



The command

Example 92

```
nth_roots_63:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -with F -do -finite_field_activity \
▷  ▷  ▷  -nth_roots 63 \
▷  ▷  -end
▷  pdflatex Nth_roots_q2_n63.tex
▷  $(OPEN) Nth_roots_q2_n63.pdf
```

creates the minimum polynomials of all elements in $\mathbb{F}_{64}^{\times}$. This corresponds to all irreducible polynomials over $\mathbb{F}_2$ of degree dividing 6, apart from the polynomial X.

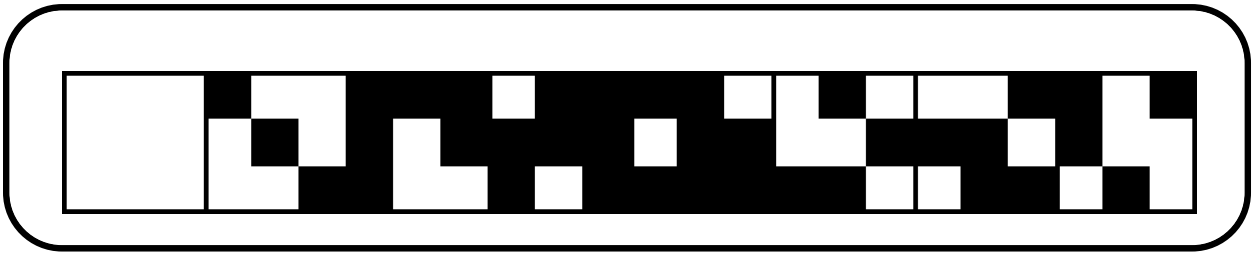
i	r_i	Cyc(r_i)	$m_{\beta^{r_i}}(X)$	$m_{\beta^{r_i}}(X)$	rank
0	0	(0)	$(100000)X^0 + (100000)X^1$	$X+1$	3
1	1	(1, 2, 4, 8, 16, 32)	$(100000)X^0+(000000)X^1+(000000)X^2+(000000)X^3+(000000)X^4+(100000)X^5+(100000)X^6$	X^6+X^5+1	97
2	3	(3, 6, 12, 24, 48, 33)	$(100000)X^0+(000000)X^1+(100000)X^2+(000000)X^3+(100000)X^4+(100000)X^5+(100000)X^6$	$X^6+X^5+X^4+X^2+1$	117
3	5	(5, 10, 20, 40, 17, 34)	$(100000)X^0+(100000)X^1+(000000)X^2+(000000)X^3+(100000)X^4+(100000)X^5+(100000)X^6$	$X^6+X^5+X^4+X+1$	115
4	7	(7, 14, 28, 56, 49, 35)	$(100000)X^0+(000000)X^1+(000000)X^2+(100000)X^3+(000000)X^4+(000000)X^5+(100000)X^6$	X^6+X^3+1	73
5	9	(9, 18, 36)	$(100000)X^0+(100000)X^1+(000000)X^2+(100000)X^3$	X^3+X+1	11
6	11	(11, 22, 44, 25, 50, 37)	$(100000)X^0+(100000)X^1+(000000)X^2+(100000)X^3+(100000)X^4+(000000)X^5+(100000)X^6$	$X^6+X^4+X^3+X+1$	91
7	13	(13, 26, 52, 41, 19, 38)	$(100000)X^0+(000000)X^1+(100000)X^2+(100000)X^3+(000000)X^4+(100000)X^5+(100000)X^6$	$X^6+X^5+X^3+X^2+1$	109
8	15	(15, 30, 60, 57, 51, 39)	$(100000)X^0+(100000)X^1+(100000)X^2+(000000)X^3+(100000)X^4+(000000)X^5+(100000)X^6$	$X^6+X^4+X^2+X+1$	87
9	21	(21, 42)	$(100000)X^0 + (100000)X^1 + (100000)X^2$	X^2+X+1	7
10	23	(23, 46, 29, 58, 53, 43)	$(100000)X^0+(100000)X^1+(100000)X^2+(000000)X^3+(000000)X^4+(100000)X^5+(100000)X^6$	$X^6+X^5+X^2+X+1$	103
11	27	(27, 54, 45)	$(100000)X^0+(000000)X^1+(100000)X^2+(100000)X^3$	X^3+X^2+1	13
12	31	(31, 62, 61, 59, 55, 47)	$(100000)X^0+(100000)X^1+(000000)X^2+(000000)X^3+(000000)X^4+(000000)X^5+(100000)X^6$	X^6+X+1	67

The command

Example 93

```
F8_F2_field_reduction:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 8 -end \
▷  ▷  -define v -vector -loop 0 8 1 -end \
▷  ▷  -define C -code -field F -Reed_Solomon 7 3 -end \
▷  ▷  -with F -do \
▷  ▷  -finite_field_activity \
▷  ▷  -field_reduction "F8_to_F2" \
▷  ▷  ▷  2 1 8 v \
▷  ▷  -end
▷  $(ORBITER) -v 2 \
▷  ▷  -draw_matrix -input_csv_file F8_to_F2.csv \
▷  ▷  -box_width 40 -bit_depth 24 \
▷  ▷  -partition 4 "3" "3,3,3,3,3,3,3,3" -end
▷  convert F8_to_F2_draw.bmp F8_to_F2_draw.png
▷  $(OPEN) F8_to_F2_draw.png
```

performs field reduction on the elements $0, \ldots, 7$ in $\mathbb{F}_8$ down to $\mathbb{F}_2$. Here is a graphical representation of the matrices arising this way:



Let us discuss field embeddings in Orbiter. This is important for working in field extensions, as there is a restriction on the polynomials used to create the fields. Suppose we have fields $\mathbb{F}_q$ and $\mathbb{F}_Q$ with

$$\mathbb{F}_q \leq \mathbb{F}_Q,$$

i.e.

$$Q = q^e$$

for some positive integer e. Suppose α is a primitive element in $\mathbb{F}_Q$ and β is a primitive element in $\mathbb{F}_q$. Then Orbiter assumes that

$$\beta = \alpha^{\frac{Q-1}{q-1}},$$

where

$$\frac{Q-1}{q-1}$$

is the index of $\mathbb{F}_q^\times$ in the multiplicative group $\mathbb{F}_Q^\times$. With respect to this embedding, the minimum polynomial of β is determined. Let us consider and example. The command

Example 94

```
F_4096:
▷ $(ORBITER) -v 4 \
▷ ▷ -define F -finite_field -q 4096 \
▷ ▷ ▷ -compute_related_fields \
▷ ▷ -end \
▷ ▷ -with F -do -finite_field_activity -cheat_sheet_GF -end
▷ pdflatex GF_4096.tex
▷ $(OPEN) GF_4096.pdf
```

creates the field $\mathbb{F}_{2^{12}}$. Orbiter chooses the polynomial

$$X^{12} + X^6 + X^4 + X + 1$$

to create the field. This field has non-trivial subfields $\mathbb{F}_4$, $\mathbb{F}_8$, $\mathbb{F}_{16}$ and $\mathbb{F}_{64}$. The report shows the polynomials for the primitive elements of the subfields:

Subfield	Polynomial	Numerical Rank
$\mathbb{F}_4$	X^2+X+1	7
$\mathbb{F}_8$	X^3+X+1	11
$\mathbb{F}_{16}$	X^4+X+1	19
$\mathbb{F}_{64}$	X^6+X^5+1	97

For each subfield, including the prime field, Orbiter computes a subfield basis and the embedding of the elements in the subfield. We use the convention that $d = \frac{Q-1}{q-1}$. Here is the output from the report:

Subfield $\mathbb{F}_2$ generated by polynomial 3:
Subfield of Order 2:
polynomial:
Field basis:

$$(1, 83, 342, 598, 1143, 518, 1492, 938, 1881, 3381, 1159, 227)$$

The power table shows $(\alpha^d)^i$ for $i = 0, 1, \ldots, e$ where $q = p^e$:

i	$i \cdot d$	α^{id}	vector
0	0	1	$(0,0,0,0,0,0,0,0,0,0,0,1)$
1	4095	1	$(0,0,0,0,0,0,0,0,0,0,0,1)$

Subfield $\mathbb{F}_4$ generated by polynomial 7:
Subfield of Order 4:
polynomial: 7
Field basis:

$$(1, 64, 83, 1171, 342, 1183)$$

The power table shows $(\alpha^d)^i$ for $i = 0, 1, \ldots, e$ where $q = p^e$:

i	$i \cdot d$	α^{id}	vector
0	0	1	$(0,0,0,0,0,0,0,0,0,0,0,1)$
1	1365	70	$(0,0,0,0,0,1,0,0,0,1,1,0)$
2	2730	71	$(0,0,0,0,0,1,0,0,0,1,1,1)$

Subfield $\mathbb{F}_8$ generated by polynomial 11:
Subfield of Order 8:
polynomial: 11
Field basis:

$$(1, 16, 256, 83)$$

The power table shows $(\alpha^d)^i$ for $i = 0, 1, \ldots, e$ where $q = p^e$:

i	$i \cdot d$	α^{id}	vector
0	0	1	$(0,0,0,0,0,0,0,0,0,0,0,0,1)$
1	585	937	$(0,0,1,1,1,0,1,0,1,0,0,1)$
2	1170	1198	$(0,1,0,0,1,0,1,0,1,1,1,0)$
3	1755	936	$(0,0,1,1,1,0,1,0,1,0,0,0)$

Subfield $\mathbb{F}_{16}$ generated by polynomial 19:
Subfield of Order 16:
polynomial: 19
Field basis:

$$(1, 8, 64)$$

The power table shows $(\alpha^d)^i$ for $i = 0, 1, \ldots, e$ where $q = p^e$:

i	$i \cdot d$	α^{id}	vector
0	0	1	$(0,0,0,0,0,0,0,0,0,0,0,1)$
1	273	1971	$(0,1,1,1,1,0,1,1,0,0,1,1)$
2	546	2037	$(0,1,1,1,1,1,1,1,0,1,0,1)$
3	819	3214	$(1,1,0,0,1,0,0,0,1,1,1,0)$
4	1092	1970	$(0,1,1,1,1,0,1,1,0,0,1,0)$

Subfield $\mathbb{F}_{64}$ generated by polynomial 97:
Subfield of Order 64:
polynomial: 97
Field basis:

$$(1, 4)$$

The power table shows $(\alpha^d)^i$ for $i = 0, 1, \ldots, e$ where $q = p^e$:

i	$i \cdot d$	α^{id}	vector
0	0	1	$(0,0,0,0,0,0,0,0,0,0,0,1)$
1	65	396	$(0,0,0,1,1,0,0,0,1,1,0,0)$
2	130	1068	$(0,1,0,0,0,0,1,0,1,1,0,0)$
3	195	1315	$(0,1,0,1,0,0,1,0,0,0,1,1)$
4	260	1615	$(0,1,1,0,0,1,0,0,1,1,1,1)$
5	325	811	$(0,0,1,1,0,0,1,0,1,0,1,1)$
6	390	810	$(0,0,1,1,0,0,1,0,1,0,1,0)$

The command

Example 95

```
F_4489:
▷  $(ORBITER) -v 4 \
▷  ▷  -define F -finite_field -q 4489 -without_tables -end \
```

creates the field $\mathbb{F}_{67^2}$. For the sake of speed, the field tables are not computed.

Linear Algebra Activities

Command	Arguments	Purpose
-nullspace	$m\ n\ L$	Compute a basis for the right nullspace of the $m \times n$ matrix L
-RREF	$m\ n\ L$	Compute the RREF of the $m \times n$ matrix L over $\mathbb{F}_q$
-normalize_from_the_left		Compute left normalized RREF or nullspace.
-normalize_from_the_right		Compute right normalized RREF or nullspace.
-RREF_random_matrix	$m\ n$	Produce a random $m \times n$ matrix over F and perform RREF.
-Walsh_matrix	n	Create the Walsh matrix of order n.
-Vandermonde_matrix		Create the Vandermonde matrix of order q and compute its inverse. The (i,j) entry is γ_i^j $(i,j = 0,\ldots,q-1)$. Here, $\gamma_0,\ldots,\gamma_{q-1}$ is the list of elements in $\mathbb{F}_q$ in Orbiter ordering. The matrix and its inverse are written to file.

Table 3.8: Linear Algebra Activities

3.4 Linear Algebra Over Finite Fields

In Table 3.8, some finite field activities regarding linear algebra are shown. For instance, the command

Example 96

```
RREF:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define v -vector -field F -format 2 \
▷  ▷  ▷  -dense "1,1,1,1,0,1,1,0,0,1" \
▷  ▷  -end \
▷  ▷  -with F -do -finite_field_activity \
▷  ▷  ▷  -RREF v \
▷  ▷  -end
▷  pdflatex v_rref.tex
▷  $(OPEN) v_rref.pdf
```

computes the RREF form of the matrix

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

over $\mathbb{F}_2$. The output is the matrix

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}.$$

The -RREF command produces a latex log of the steps. This can be used to follow the algorithm along. For a somewhat longer example, consider the Vandermonde matrix over the field $\mathbb{F}_7$. Suppose we want to compute the inverse matrix directly. We can use the following command to do so. Notice how we first create the matrix and an identity matrix next to it. After that we apply the -RREF command:

Example 97

```
V7_VANDERMONDE_EXTENDED="\
1,0,0,0,0,0,0,1,0,0,0,0,0,0, \
1,1,1,1,1,1,1,0,1,0,0,0,0,0, \
1,2,4,1,2,4,1,0,0,1,0,0,0,0, \
1,3,2,6,4,5,1,0,0,0,1,0,0,0, \
1,4,2,1,4,2,1,0,0,0,0,1,0,0, \
1,5,4,6,2,3,1,0,0,0,0,0,1,0, \
1,6,1,6,1,6,1,0,0,0,0,0,0,1"
```

Example 98

```
RREF_V7:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 7 -end \
▷  ▷  -define V7 -vector -format 7 \
▷  ▷  ▷  -dense $(V7_VANDERMONDE_EXTENDED) \
▷  ▷  -end \
▷  ▷  -with F -do -finite_field_activity \
▷  ▷  ▷  -RREF V7 \
▷  ▷  -end
▷  pdflatex V7_rref.tex
▷  $(OPEN) V7_rref.pdf
```

The following (shortened) output is produced. Observe how the inverse matrix appears in the second half once the -RREF algorithm is finished:

A matrix over the field $\mathbb{F}_7$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 4 & 1 & 2 & 4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 3 & 2 & 6 & 4 & 5 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 4 & 2 & 1 & 4 & 2 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 5 & 4 & 6 & 2 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 6 & 1 & 6 & 1 & 6 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Position $(i,j) = (0,0)$, found pivot in column 0

$$\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 2 & 4 & 1 & 2 & 4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 3 & 2 & 6 & 4 & 5 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 4 & 2 & 1 & 4 & 2 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 5 & 4 & 6 & 2 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 6 & 1 & 6 & 1 & 6 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}$$

After making pivot 1:

$$\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 2 & 4 & 1 & 2 & 4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 3 & 2 & 6 & 4 & 5 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 4 & 2 & 1 & 4 & 2 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 5 & 4 & 6 & 2 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 6 & 1 & 6 & 1 & 6 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}$$

output truncated

After elimination above pivot 0 in position (0,0):

$$\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & 3 & 2 & 5 & 4 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 6 & 5 & 3 & 3 & 5 & 6 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 6 & 6 & 1 & 6 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 6 & 3 & 5 & 5 & 3 & 6 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 6 & 5 & 4 & 3 & 2 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 6 & 6 & 6 & 6 & 6 & 6 & 6
\end{bmatrix}$$

The inverse matrix agrees with the output obtained in Section 3.2.

Another task is computing the nullspace of a matrix. The command

Example 99

```
nullspace:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F2 -finite_field -q 2 -end \
▷  ▷  -define v -vector -field F2 -format 2 \
▷  ▷  ▷  -dense "1,1,1,1,0,1,1,0,0,1" \
▷  ▷  -end \
▷  ▷  -with F2 -do \
▷  ▷  -finite_field_activity \
▷  ▷  ▷  -nullspace v \
▷  ▷  -end
▷  pdflatex v_nullspace.tex
▷  $(OPEN) v_nullspace.pdf
```

computes the right nullspace of the matrix from the first example. The output is the matrix

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}.$$

Orbiter can compute eigenvalues and eigenvectors of matrices over finite fields. For instance, the command

Example 100

```
eigenstuff:
▷  $(ORBITER) -v 6 \
▷  ▷  -define F -finite_field -q 5 -end \
▷  ▷  -eigenstuff F 4 "0,1,0,2,0,1,2,1,4,2,3,1,2,0,4,3"
```

computes all eigenvectors and eigenvalues of the matrix

$$\begin{bmatrix} 0 & 1 & 0 & 2 \\ 0 & 1 & 2 & 1 \\ 4 & 2 & 3 & 1 \\ 2 & 0 & 4 & 3 \end{bmatrix}$$

over $\mathbb{F}_5$.

Orbiter can produce a list of all conjugacy classes of endomorphisms of $\mathbb{F}_q^d$ by means of their rational normal forms. For instance

Example 101

```
classes_GL_3_2:
▷  $(ORBITER) -v 7 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -all_rational_normal_forms F 3
▷  pdflatex Class_reps_GL_3_2.tex
▷  $(OPEN) Class_reps_GL_3_2.pdf
```

produces a list of all conjugacy classes of $\mathrm{GL}(3,2)$. There are 6 of them. The report includes the order of the centralizer and the order of the conjugacy class. The order of the centralizer is computed using Kung's formula [45]. This command relies on the Orbiter catalogue of irreducible polynomials. For an introduction to the rational normal form of endomorphisms, see [51].

Conjugacy Classes of $\mathrm{GL}(3,2)$

The number of conjugacy classes of $\mathrm{GL}(3,2)$ is 6:

$$\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Class 0 / 6

3, 1, 0

$$\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

centralizer order 7
class size 24
Class 1 / 6
2, 1, 0

$$\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

centralizer order 7
class size 24
Class 2 / 6
0, 1, 0; 1, 1, 0

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

centralizer order 3
class size 56
Class 3 / 6
0, 3, 0

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

centralizer order 4
class size 42
Class 4 / 6
0, 3, 1

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

centralizer order 8
class size 21
Class 5 / 6
$0, 3, 2$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

centralizer order 168
class size 1

Finite Field Activities (Part I)

Command	Arguments	Purpose
-write_code_for_division	fname A B	Write C++ source code for the polynomial division of A by B. See Section 10.4.
-weight_enumerator	A	Computes the weight enumerator of the code whose generator matrix is A.
-Walsh_Hadamard_transform	fname n	Computes the Walsh-Hadamard transform for the n-variable boolean function in the given file.
-algebraic_normal_form_of_ boolean_function	fname n	Computes the algebraic normal form for the n-variable boolean function in the given file.
-algebraic_normal_form	n input	Computes the algebraic normal form for the n-variable function in the given file.
-apply_trace_function	fname	Applies the absolute trace function to the function in the given file.
-apply_power_function	fname d	Applies the raise-to-the-power-d function to the function in the given file.
-identity_function	fname_csv	Creates the identity function and stores in the given csv file.

Table 3.9: Finite Field Activities (Part I)

3.5 Advanced Topics in Finite Fields

Let us now look at some advanced topics in the theory of finite fields.

First, in Tables 3.9-3.10, a summary of finite field activities is shown.

A normal basis for a field extension $\mathbb{F}_{q^d}$ over $\mathbb{F}_q$ is a basis of $\mathbb{F}_{q^d}$ as vector space over $\mathbb{F}_q$ which consists of one cycle of the Frobenius automorphism of $\mathbb{F}_{q^d}$ over $\mathbb{F}_q$. For instance, the command

Example 102

```
normal_basis_2_3:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -with F -do -finite_field_activity \
▷  ▷  -normal_basis 3 -end
▷  pdflatex normal_basis_q2_d3.tex
▷  $(OPEN) normal_basis_q2_d3.pdf
```

computes a normal basis of $\mathbb{F}_8$ over $\mathbb{F}_2$. Using the polynomial $X^3 + X^2 + 1$, the normal basis in terms of the standard polynomial basis $1, X, X^2, \ldots$ is given by the columns of the matrix

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}.$$

Reading the columns as cofficient vectors with respect to the standard basis, the normal basis is

$$b_1 = 1 + X + X^2, \quad b_2 = X, \quad b_3 = X^2.$$

Finite Field Activities (Part II)

Command	Arguments	Purpose
-transversal	$L1$ $L2$ P	Computes the unique transversal to the lines $L1$ and $L2$ through the point P in $\mathrm{PG}(3,q)$. The linese are given by a basis consisting of 8 field elements.
-intersection_of_two_lines	$L1$ $L2$	Computes the intersection of two lines in $\mathrm{PG}(3,q)$. The linese are given by a basis consisting of 8 field elements.
-rank_point_in_PG	P	Computes the orbiter point rank corresponding to the point P in $\mathrm{PG}(n,q)$. P is a label of a vector, which is the coefficient vector.
-unrank_point_in_PG	r	Computes the coordinate vector of the Orbiter point in $\mathrm{PG}(n,q)$ corresponding to the Orbiter rank value r.
-inverse_isomorphism_klein_quadric	$L36$	
-NTT	k n	Computes the Number-theoretic transform for $n = 2^k$, which must divide $q-1$.

Table 3.10: Finite Field Activities (Part II)

Let us apply the Frobenius mapping Φ to the elements of the normal bases:

$$b_1^\Phi = (1+X+X^2)^2 = 1+X^2+X^4 = 1+X^2+X^3+X = 1+X+X^2+X^2+1 = X = b_2,$$
$$b_2^\Phi = X^2 = b_3,$$
$$b_3^\Phi = X^4 = X^3+X = X^2+X+1 = b_1.$$

Thus,

$$b_1 \mapsto b_2 \mapsto b_3 \mapsto b_1$$

under Φ, as required.

A field is a vector space over any of its subfields. Using a field basis, the elements of the large field can be identified with invertible matrices. So, for $\mathbb{F}_{q^r}$ over $\mathbb{F}_q$, and for $a \in \mathbb{F}_{q^r}$, we consider the $\mathbb{F}_q$-linear map

$$\mathbb{F}_{q^r} \to \mathbb{F}_{q^r}, x \mapsto ax.$$

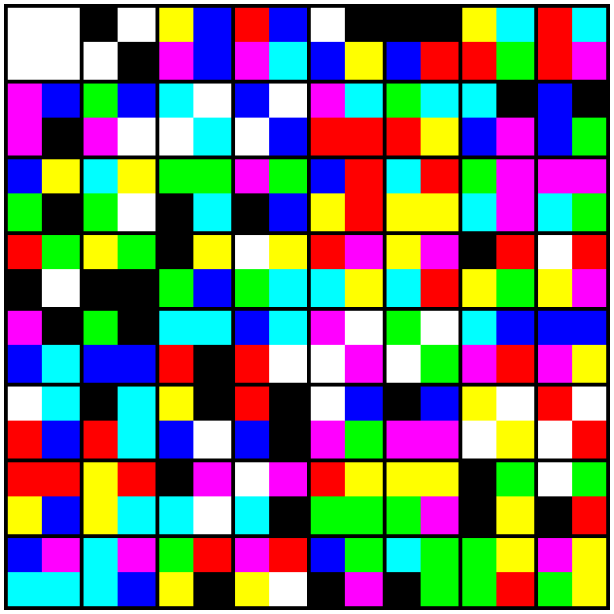
The following code computes the field reduction from $\mathbb{F}_{64}$ to $\mathbb{F}_8$. Elements in the small field are represented as colors. The (i,j)-th block is the matrix of $a = i8+j$ in the chosen basis.

Example 103

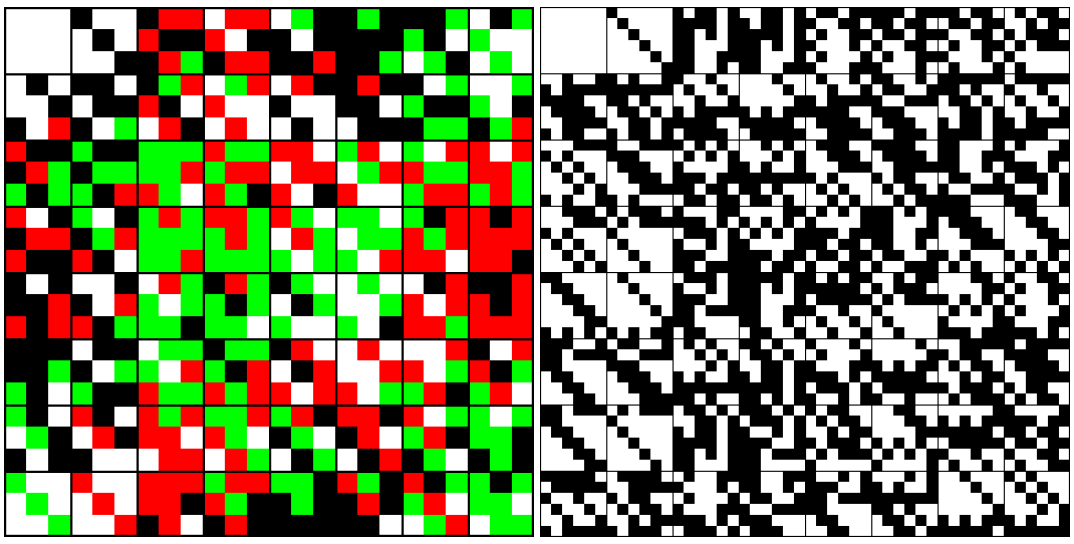
```
F_64_over_F8_field_reduction:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 64 -end \
▷  ▷  -define elts -vector -field F -loop 0 64 1 -end \
▷  ▷  -with F -do \
▷  ▷  -finite_field_activity -field_reduction "F64_over_F8" 8 8 8 \
▷  ▷  ▷  elts -end
▷  $(ORBITER) -v 2 -draw_matrix \
▷  ▷  -input_csv_file F64_over_F8.csv \
▷  ▷  -box_width 40 -bit_depth 24 \
```

```
▷ ▷  -partition 4 "2,2,2,2,2,2,2,2" "2,2,2,2,2,2,2,2" -end
▷ $(OPEN) F64_over_F8_draw.bmp
▷ #pdflatex field_reduction_Q64_q8_8_8.tex
▷ #$(OPEN) field_reduction_Q64_q8_8_8.pdf
```

The output is shown below:



Note that the dimension of the vector space is 2, so the block matrices are 2×2. Observe that $\mathbb{F}_{64}$ has many subfields. The reduction from $\mathbb{F}_{64}$ to $\mathbb{F}_4$ and to $\mathbb{F}_2$ is also possible:



Here, the block matrices have size 3×3 when reducing to $\mathbb{F}_4$ (left picture) and 6×6 when reducing to $\mathbb{F}_2$.

The minimum polynomials associated with the n-th roots over $\mathbb{F}_q$ can be computed using the `-nth_roots` command, which is a finite field activity. The activity is aplied to the field $\mathbb{F}_q$ over which the n-th roots are defined. The command constructs the field extension $\mathbb{F}_{q^m}$ where m is the order of q modulo n. This field extension contains the n-th roots of unity. Let α be a primitive element of $\mathbb{F}_{q^m}$ and let β be a generator of the subgroup of n-th roots. Also, let γ be the generator of the subgroup of $q-1$ th roots, which are the elements of the multiplicative group of $\mathbb{F}_q$. The output lists the n-th roots first, generated by β. After that, the $q-1$th roots are shown, generated by γ. Finally, a table is produced which shows the irreducible polynomials over $\mathbb{F}_q$ associated with the n-th roots of unity. For instance, the following command computes the minimum polynomials of all 21st roots of unity over $\mathbb{F}_8$:

Example 104

```
F_8_Nth_roots_21:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 8 -override_polynomial 11 -end \
▷  ▷  -with F -do -finite_field_activity \
▷  ▷  ▷  -nth_roots 21 \
▷  ▷  -end
▷  pdflatex Nth_roots_q8_n21.tex
▷  $(OPEN) Nth_roots_q8_n21.pdf
```

The output is:

Let α be a primitive element of GF(64). Let β be a primitive 21-th root in GF(64), so $\beta = \alpha^3$.
$\beta^0 = 100000 = 1$
$\beta^1 = 000100 = \alpha^3$
$\beta^2 = 100001 = \alpha^5 + 1$
$\beta^3 = 111101 = \alpha^5 + \alpha^3 + \alpha^2 + \alpha + 1$
$\beta^4 = 011111 = \alpha^5 + \alpha^4 + \alpha^3 + \alpha^2 + \alpha$
$\beta^5 = 101010 = \alpha^4 + \alpha^2 + 1$
$\beta^6 = 110100 = \alpha^3 + \alpha + 1$
$\beta^7 = 100111 = \alpha^5 + \alpha^4 + \alpha^3 + 1$
$\beta^8 = 101101 = \alpha^5 + \alpha^3 + \alpha^2 + 1$
$\beta^9 = 011101 = \alpha^5 + \alpha^3 + \alpha^2 + \alpha$
$\beta^{10} = 011011 = \alpha^5 + \alpha^4 + \alpha^2 + \alpha$
$\beta^{11} = 001011 = \alpha^5 + \alpha^4 + \alpha^2$
$\beta^{12} = 001001 = \alpha^5 + \alpha^2$
$\beta^{13} = 111000 = \alpha^2 + \alpha + 1$
$\beta^{14} = 000111 = \alpha^5 + \alpha^4 + \alpha^3$
$\beta^{15} = 101001 = \alpha^5 + \alpha^2 + 1$
$\beta^{16} = 111100 = \alpha^3 + \alpha^2 + \alpha + 1$
$\beta^{17} = 100110 = \alpha^4 + \alpha^3 + 1$
$\beta^{18} = 010100 = \alpha^3 + \alpha$
$\beta^{19} = 100011 = \alpha^5 + \alpha^4 + 1$
$\beta^{20} = 001100 = \alpha^3 + \alpha^2$

Let γ be a primitive 7-th root in GF(64), so $\gamma = \alpha^9$.
$\gamma^0 = 100000 = 1$
$\gamma^1 = 111101 = \alpha^5 + \alpha^3 + \alpha^2 + \alpha + 1$

$\gamma^2 = 110100 = \alpha^3 + \alpha + 1$
$\gamma^3 = 011101 = \alpha^5 + \alpha^3 + \alpha^2 + \alpha$
$\gamma^4 = 001001 = \alpha^5 + \alpha^2$
$\gamma^5 = 101001 = \alpha^5 + \alpha^2 + 1$
$\gamma^6 = 010100 = \alpha^3 + \alpha$

The q-cyclotomic set for $q = 8$ are:
{ 0 }
{ 1, 8 }
{ 2, 16 }
{ 3 }
{ 4, 11 }
{ 5, 19 }
{ 6 }
{ 7, 14 }
{ 9 }
{ 10, 17 }
{ 12 }
{ 13, 20 }
{ 15 }
{ 18 }

Subfield basis, a basis for GF(8) inside GF(64):

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

The irreducible polynomials associated with the 21-th roots over GF(8) are:

i	r_i	Cyc(r_i)	$m_{\beta^{r_i}}(X)$	$m_{\beta^{r_i}}(X)$
0	0	(0)	$(100000)X^0 + (100000)X^1$	$X + 1$
1	1	(1, 8)	$(011101)X^0 + (101001)X^1 + (100000)X^2$	$X^2 + 7X + 3$
2	2	(2, 16)	$(010100)X^0 + (011101)X^1 + (100000)X^2$	$X^2 + 3X + 5$
3	3	(3)	$(111101)X^0 + (100000)X^1$	$X + 2$
4	4	(4, 11)	$(101001)X^0 + (010100)X^1 + (100000)X^2$	$X^2 + 5X + 7$
5	5	(5, 19)	$(111101)X^0 + (001001)X^1 + (100000)X^2$	$X^2 + 6X + 2$
6	6	(6)	$(110100)X^0 + (100000)X^1$	$X + 4$
7	7	(7, 14)	$(100000)X^0 + (100000)X^1 + (100000)X^2$	$X^2 + X + 1$
8	9	(9)	$(011101)X^0 + (100000)X^1$	$X + 3$
9	10	(10, 17)	$(110100)X^0 + (111101)X^1 + (100000)X^2$	$X^2 + 2X + 4$
10	12	(12)	$(001001)X^0 + (100000)X^1$	$X + 6$
11	13	(13, 20)	$(001001)X^0 + (110100)X^1 + (100000)X^2$	$X^2 + 4X + 6$
12	15	(15)	$(101001)X^0 + (100000)X^1$	$X + 7$
13	18	(18)	$(010100)X^0 + (100000)X^1$	$X + 5$

In Section 3.2, we have considered the Vandermonde matrix over $\mathbb{F}_7$. Let us do the same for the field $\mathbb{F}_8$ instead. We use the following command:

Example 105

```
F_8_vandermonde:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 8 -end \
▷  ▷  -with F -do -finite_field_activity \
▷  ▷  ▷  -Vandermonde_matrix \
▷  ▷  -end
```

The output is shown below. Again, the first matrix is $V = (x_i^j)$. The second matrix is V^{-1}:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 5 & 7 & 3 & 6 & 1 \\ 1 & 3 & 5 & 2 & 6 & 7 & 4 & 1 \\ 1 & 4 & 7 & 6 & 2 & 5 & 3 & 1 \\ 1 & 5 & 6 & 4 & 3 & 2 & 7 & 1 \\ 1 & 6 & 3 & 7 & 5 & 4 & 2 & 1 \\ 1 & 7 & 2 & 3 & 4 & 6 & 5 & 1 \end{bmatrix}, \quad \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 6 & 4 & 3 & 7 & 2 & 5 \\ 0 & 1 & 3 & 7 & 5 & 2 & 4 & 6 \\ 0 & 1 & 7 & 6 & 2 & 3 & 5 & 4 \\ 0 & 1 & 5 & 2 & 6 & 4 & 7 & 3 \\ 0 & 1 & 4 & 5 & 7 & 6 & 3 & 2 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Let us now do a somewhat larger example of the same problem. The next command computes the Vandermonde matrix and its inverse over the field $\mathbb{F}_{1024}$:

Example 106

```
F_1024_vandermonde:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 1024 -end \
▷  ▷  -with F -do -finite_field_activity \
▷  ▷  ▷  -Vandermonde_matrix \
▷  ▷  -end
▷  #rm Vandermonde_1024.csv
▷  #rm Vandermonde_inv_1024.csv
```

This command takes a bit of time to execute. The matrix is not shown. It would be too big to be printed. In order to save disc space, we delete the output files, using the `rm` command.

Orbiter can create code for the number theoretic transform. This is the discrete Fourier transform performed over finite fields. The generated code can be compiled with the Orbiter library. Compiling code requires additional makefile options are necessary. Because of this, we define the following makefile variables at the top of the makefile.

Example 107

```
SRC=$(ORBITER_PATH)/src
MY_CPP = g++
MY_CC = gcc
CPPFLAGS = -Wall -I../../DEV.23/orbiter/src/lib -std=c++14
LIB = $(SRC)/lib/liborbiter.a -lpthread
LFLAGS = -lm -Wl,-rpath -Wl,/usr/local/gcc-8.2.0/lib64
```

Suppose we want to create the number theoretic transform for the 16th roots of unity inside the field $\mathbb{F}_{17}$. Here is the command to generate the Orbiter source code:

Example 108

```
NTT_k4_q17.cpp:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 17 -end \
▷  ▷  -with F -do -coding_theoretic_activity \
▷  ▷  ▷  -NTT 4 17 \
▷  ▷  -end
```

This produces a C++ file `NTT_k4_q17.cpp`. This file should be compiled and linked against the Orbiter library. The command

Example 109

```
F_17_NTT_compile: NTT_k4_q17.cpp
▷  $(MY_CPP)  NTT_k4_q17.cpp $(CPPFLAGS) \
```

```
▷ ▷  $(LIB) $(LFLAGS) -o NTT_k4_q17.out
▷  ./NTT_k4_q17.out
```

can be used to compile the code and run it. Note the dependency on the file `NTT_k4_q17.cpp`. This means that `make` would automatically invoke the first command if only the second one was issued.

3.6 Basic Ring Theory

Orbiter can deal with multivariate polynomial rings with coefficients over finite fields. Orbiter creates the homogeneous components only (so it is technically not a ring).

The following command creates the homogeneous component of degree 3 in a polynomial ring in 4 variables. The variables are named. They are x_0, x_1, x_2, x_3. Note that two sets of names are defined using the `-variables` command. The first is the labels for regular text output. The second is the set of names for latex output. Here is the command:

Example 110

```
Polynomial_ring:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 4 -end \
▷  ▷  -define R -polynomial_ring -field F \
▷  ▷  ▷  -number_of_variables 4 \
▷  ▷  ▷  -homogeneous_of_degree 3 \
▷  ▷  ▷  -variables "x0,x1,x2,x3" "x_0,x_1,x_2,x_3" \
▷  ▷  -end
```

For more on rings, see Chapter 5.

Chapter 4

Geometry

4.1 Finite Projective Spaces

Orbiter commands to create a finite projective space $\mathrm{PG}(n,q)$ are listed in Table 4.1. The command also creates the group of the space, which is $\mathrm{P\Gamma L}(n+1,q)$. Let us look at a very simple example. Suppose we want to create $\mathrm{PG}(3,2)$. The following command sequence creates the finite field $\mathbb{F}_2$ and stores the object in the symbol table, using the label F. After that, the projective space $\mathrm{PG}(3,F)$ is created, using the field F. The projective space is stored in the symbol table under the label P.

Example 111

```
PG_3_2_easy:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define P -projective_space -n 3 -field F -end
```

Orbiter offers indexing for points and for subspaces o allf $\mathrm{PG}(n,q)$ of a fixed dimension. The incidence matrix between points and lines with respect to this ordering can be computed. The indexing is used to establish the permutation representations of the projective group, as will be described in Section 6.2. The indexing of points is not the lexicographic ordering. It emphasizes the role of frames in the geometry by assigning the

Creating a Projective Space

Command	Arguments	Purpose
-n	n	Specify the projective dimension n.
-q	q	Specify the order q of the underlying finite field $\mathbb{F}_q$. A default version of the field $\mathbb{F}_q$ will be created.
-field	F	Specify the underlying field F.
-use_projectivity_subgroup		Create the projectivity group instead of the collineation group.
-v	k	Set verbosity to k during creation of the projective space. Higher values of k give more output. $k=0$ is silent mode.

Table 4.1: Options for Creating a Projective Space

smallest rank values to the members of th standard frame. After that, the other points are listed.

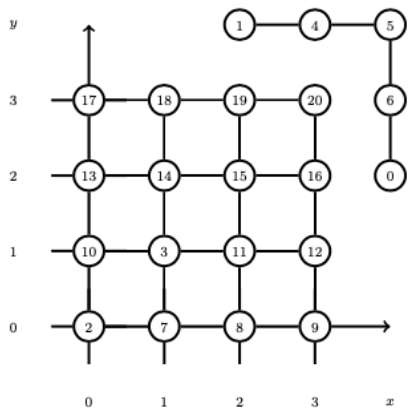
Orbiter can create a cheat sheet for PG(n, q). The cheat sheet lists the objects in the space, and basic properties. Here are two examples. The first is a projective plane. The second is a projective 3-space.

The following command creates a cheat sheet for PG$(2, 4)$:

Example 112

```
PG_2_4:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 4 -end \
▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
▷  ▷  -with P -do -projective_space_activity \
▷  ▷  ▷  -cheat_sheet \
▷  ▷  -end
▷  pdflatex PG_2_4.tex
▷  $(OPEN) PG_2_4.pdf
```

The cheat sheet contains a drawing of the plane:



The affine plane is shown in the cartesian plane, while the line at infinity is wrapped around the top right corner. The cheat sheet continues by listing the points, including the canonical Baer subgeometry PG$(2, 2)$. After that, the points are listed again, but with left-normalized vectors. Finally, the lines are shown.

PG$(2, 4)$ has 21 points:

$P_0 = (1,0,0) = (1,0,0)$
$P_1 = (0,1,0) = (0,1,0)$
$P_2 = (0,0,1) = (0,0,1)$
$P_3 = (1,1,1) = (1,1,1)$
$P_4 = (1,1,0) = (1,1,0)$
$P_5 = (2,1,0) = (\alpha,1,0)$
$P_6 = (3,1,0) = (\alpha^2,1,0)$
$P_7 = (1,0,1) = (1,0,1)$
$P_8 = (2,0,1) = (\alpha,0,1)$
$P_9 = (3,0,1) = (\alpha^2,0,1)$
$P_{10} = (0,1,1) = (0,1,1)$
$P_{11} = (2,1,1) = (\alpha,1,1)$
$P_{12} = (3,1,1) = (\alpha^2,1,1)$
$P_{13} = (0,2,1) = (0,\alpha,1)$
$P_{14} = (1,2,1) = (1,\alpha,1)$
$P_{15} = (2,2,1) = (\alpha,\alpha,1)$
$P_{16} = (3,2,1) = (\alpha^2,\alpha,1)$
$P_{17} = (0,3,1) = (0,\alpha^2,1)$
$P_{18} = (1,3,1) = (1,\alpha^2,1)$
$P_{19} = (2,3,1) = (\alpha,\alpha^2,1)$
$P_{20} = (3,3,1) = (\alpha^2,\alpha^2,1)$

Baer subgeometry:

$P_0 = (1,0,0)$
$P_1 = (0,1,0)$
$P_2 = (0,0,1)$
$P_3 = (1,1,1)$
$P_4 = (1,1,0)$
$P_7 = (1,0,1)$
$P_{10} = (0,1,1)$

There are 7 elements in the Baer subgeometry.
Normalized from the left:

$P_0 = (1,0,0)$
$P_1 = (0,1,0)$
$P_2 = (0,0,1)$
$P_3 = (1,1,1)$
$P_4 = (1,1,0)$
$P_5 = (1,3,0)$
$P_6 = (1,2,0)$
$P_7 = (1,0,1)$
$P_8 = (1,0,3)$
$P_9 = (1,0,2)$
$P_{10} = (0,1,1)$
$P_{11} = (1,3,3)$
$P_{12} = (1,2,2)$
$P_{13} = (0,1,3)$
$P_{14} = (1,2,1)$
$P_{15} = (1,1,3)$
$P_{16} = (1,3,2)$
$P_{17} = (0,1,2)$
$P_{18} = (1,3,1)$
$P_{19} = (1,2,3)$
$P_{20} = (1,1,2)$

The Lines of PG$(2,4)$. PG$(2,4)$ has 21 1-subspaces:

$$L_0 = \begin{bmatrix} 100 \\ 010 \end{bmatrix} \quad L_1 = \begin{bmatrix} 100 \\ 011 \end{bmatrix} \quad L_2 = \begin{bmatrix} 100 \\ 012 \end{bmatrix} \quad L_3 = \begin{bmatrix} 100 \\ 013 \end{bmatrix} \quad L_4 = \begin{bmatrix} 100 \\ 001 \end{bmatrix} \quad L_5 = \begin{bmatrix} 101 \\ 010 \end{bmatrix} \quad L_6 = \begin{bmatrix} 101 \\ 011 \end{bmatrix}$$

$$L_7 = \begin{bmatrix} 101 \\ 012 \end{bmatrix} \quad L_8 = \begin{bmatrix} 101 \\ 013 \end{bmatrix} \quad L_9 = \begin{bmatrix} 110 \\ 001 \end{bmatrix} \quad L_{10} = \begin{bmatrix} 102 \\ 010 \end{bmatrix} \quad L_{11} = \begin{bmatrix} 102 \\ 011 \end{bmatrix} \quad L_{12} = \begin{bmatrix} 102 \\ 012 \end{bmatrix} \quad L_{13} = \begin{bmatrix} 102 \\ 013 \end{bmatrix}$$

$$L_{14} = \begin{bmatrix} 120 \\ 001 \end{bmatrix} \quad L_{15} = \begin{bmatrix} 103 \\ 010 \end{bmatrix} \quad L_{16} = \begin{bmatrix} 103 \\ 011 \end{bmatrix} \quad L_{17} = \begin{bmatrix} 103 \\ 012 \end{bmatrix} \quad L_{18} = \begin{bmatrix} 103 \\ 013 \end{bmatrix} \quad L_{19} = \begin{bmatrix} 130 \\ 001 \end{bmatrix} \quad L_{20} = \begin{bmatrix} 010 \\ 001 \end{bmatrix}$$

The following command creates a cheat sheet for PG$(3,2)$.

Example 113

```
PG_3_2:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷  ▷  -with P -do -projective_space_activity \
▷  ▷  ▷  -cheat_sheet \
▷  ▷  -end
▷  pdflatex PG_3_2.tex
▷  $(OPEN) PG_3_2.pdf
```

The cheat sheet shows points, lines and planes. The lines are shown together with their Plücker coordinates. The lines whose Plücker coordinates are unit vectors are shown separately.

The projective space PG$(3,2)$

$q = 2$
$p = 2$
$e = 1$
$n = 3$
Number of points = 15
Number of lines = 35
Number of lines on a point = 7
Number of points on a line = 3

The points of PG$(3,2)$

PG$(3,2)$ has 15 points:

$P_0 = (1,0,0,0)$	$P_4 = (1,1,1,1)$	$P_8 = (1,1,1,0)$	$P_{12} = (0,0,1,1)$
$P_1 = (0,1,0,0)$	$P_5 = (1,1,0,0)$	$P_9 = (1,0,0,1)$	$P_{13} = (1,0,1,1)$
$P_2 = (0,0,1,0)$	$P_6 = (1,0,1,0)$	$P_{10} = (0,1,0,1)$	$P_{14} = (0,1,1,1)$
$P_3 = (0,0,0,1)$	$P_7 = (0,1,1,0)$	$P_{11} = (1,1,0,1)$	

Normalized from the left:

$P_0 = (1,0,0,0)$	$P_4 = (1,1,1,1)$	$P_8 = (1,1,1,0)$	$P_{12} = (0,0,1,1)$
$P_1 = (0,1,0,0)$	$P_5 = (1,1,0,0)$	$P_9 = (1,0,0,1)$	$P_{13} = (1,0,1,1)$
$P_2 = (0,0,1,0)$	$P_6 = (1,0,1,0)$	$P_{10} = (0,1,0,1)$	$P_{14} = (0,1,1,1)$
$P_3 = (0,0,0,1)$	$P_7 = (0,1,1,0)$	$P_{11} = (1,1,0,1)$	

The lines of $\mathrm{PG}(3,2)$

$\mathrm{PG}(3,2)$ has 35 1-subspaces:

$$L_0 = \begin{bmatrix} 1000 \\ 0100 \end{bmatrix} = \mathbf{Pl}(1,0,0,0,0,0)$$

$$L_1 = \begin{bmatrix} 1000 \\ 0110 \end{bmatrix} = \mathbf{Pl}(1,0,1,0,0,0)$$

$$L_2 = \begin{bmatrix} 1000 \\ 0101 \end{bmatrix} = \mathbf{Pl}(1,0,0,0,1,0)$$

$$L_3 = \begin{bmatrix} 1000 \\ 0111 \end{bmatrix} = \mathbf{Pl}(1,0,1,0,1,0)$$

$$L_4 = \begin{bmatrix} 1000 \\ 0010 \end{bmatrix} = \mathbf{Pl}(0,0,1,0,0,0)$$

$$L_5 = \begin{bmatrix} 1000 \\ 0011 \end{bmatrix} = \mathbf{Pl}(0,0,1,0,1,0)$$

$$\vdots$$

$$L_{34} = \begin{bmatrix} 0010 \\ 0001 \end{bmatrix} = \mathbf{Pl}(0,1,0,0,0,0)$$

Lines sorted by Pluecker coordinates

$$0 = \mathbf{Pl}(1,0,0,0,0,0) = L_0 = \begin{bmatrix} 1000 \\ 0100 \end{bmatrix}$$

$$1 = \mathbf{Pl}(0,1,0,0,0,0) = L_{34} = \begin{bmatrix} 0010 \\ 0001 \end{bmatrix}$$

$$2 = \mathbf{Pl}(0,0,1,0,0,0) = L_4 = \begin{bmatrix} 1000 \\ 0010 \end{bmatrix}$$

$$3 = \mathbf{Pl}(0,0,0,1,0,0) = L_{30} = \begin{bmatrix} 0100 \\ 0001 \end{bmatrix}$$

$$4 = \mathbf{Pl}(0,0,0,0,1,0) = L_6 = \begin{bmatrix} 1000 \\ 0001 \end{bmatrix}$$

$$5 = \mathbf{Pl}(0,0,0,0,0,1) = L_{28} = \begin{bmatrix} 0100 \\ 0010 \end{bmatrix}$$

$$\vdots$$

$$34 = \mathbf{Pl}(0,1,1,1,1,1) = L_{26} = \begin{bmatrix} 1101 \\ 0011 \end{bmatrix}$$

$\mathrm{PG}(3,2)$ has the following low weight Pluecker lines:

$$L_0 = \begin{bmatrix} 1000 \\ 0100 \end{bmatrix} = \mathbf{Pl}(1,0,0,0,0,0)$$

$$L_4 = \begin{bmatrix} 1000 \\ 0010 \end{bmatrix} = \mathbf{Pl}(0,0,1,0,0,0)$$

$$L_6 = \begin{bmatrix} 1000 \\ 0001 \end{bmatrix} = \mathbf{Pl}(0,0,0,0,1,0)$$

$$L_{28} = \begin{bmatrix} 0100 \\ 0010 \end{bmatrix} = \mathbf{Pl}(0,0,0,0,0,1)$$

$$L_{30} = \begin{bmatrix} 0100 \\ 0001 \end{bmatrix} = \mathbf{Pl}(0,0,0,1,0,0)$$

$$L_{34} = \begin{bmatrix} 0010 \\ 0001 \end{bmatrix} = \mathbf{Pl}(0,1,0,0,0,0)$$

The planes of $\mathrm{PG}(3,2)$

$\mathrm{PG}(3,2)$ has 15 2-subspaces:

$$L_0 = \begin{bmatrix} 1000 \\ 0100 \\ 0010 \end{bmatrix}$$

$$L_1 = \begin{bmatrix} 1000 \\ 0100 \\ 0011 \end{bmatrix}$$

$$\vdots$$

$$L_{14} = \begin{bmatrix} 0100 \\ 0010 \\ 0001 \end{bmatrix}$$

The polynomial rings associated with $\mathrm{PG}(3,2)$

h	monomial	vector
0	X_0	$(1,0,0,0)$
1	X_1	$(0,1,0,0)$
2	X_2	$(0,0,1,0)$
3	X_3	$(0,0,0,1)$

Ranking Points in Small Projective Spaces

PG(2, 2)	PG(3, 2)	PG(2, 3)
$P_0 = \mathbf{P}(1,0,0)$	$P_0 = \mathbf{P}(1,0,0,0)$	$P_0 = \mathbf{P}(1,0,0)$
$P_1 = \mathbf{P}(0,1,0)$	$P_1 = \mathbf{P}(0,1,0,0)$	$P_1 = \mathbf{P}(0,1,0)$
$P_2 = \mathbf{P}(0,0,1)$	$P_2 = \mathbf{P}(0,0,1,0)$	$P_2 = \mathbf{P}(0,0,1)$
$P_3 = \mathbf{P}(1,1,1)$	$P_3 = \mathbf{P}(0,0,0,1)$	$P_3 = \mathbf{P}(1,1,1)$
$P_4 = \mathbf{P}(1,1,0)$	$P_4 = \mathbf{P}(1,1,1,1)$	$P_4 = \mathbf{P}(1,1,0)$
$P_5 = \mathbf{P}(1,0,1)$	$P_5 = \mathbf{P}(1,1,0,0)$	$P_5 = \mathbf{P}(2,1,0)$
$P_6 = \mathbf{P}(0,1,1)$	$P_6 = \mathbf{P}(1,0,1,0)$	$P_6 = \mathbf{P}(1,0,1)$
	$P_7 = \mathbf{P}(0,1,1,0)$	$P_7 = \mathbf{P}(2,0,1)$
	$P_8 = \mathbf{P}(1,1,1,0)$	$P_8 = \mathbf{P}(0,1,1)$
	$P_9 = \mathbf{P}(1,0,0,1)$	$P_9 = \mathbf{P}(2,1,1)$
	$P_{10} = \mathbf{P}(0,1,0,1)$	$P_{10} = \mathbf{P}(0,2,1)$
	$P_{11} = \mathbf{P}(1,1,0,1)$	$P_{11} = \mathbf{P}(1,2,1)$
	$P_{12} = \mathbf{P}(0,0,1,1)$	$P_{12} = \mathbf{P}(2,2,1)$
	$P_{13} = \mathbf{P}(1,0,1,1)$	
	$P_{14} = \mathbf{P}(0,1,1,1)$	

Table 4.2: Ranking Points in Small Projective Spaces

4.2 Indexing Points and Lines

The enumerator for points establishes a bijection between the set of points and the integers on the interval $[0, \theta_n(q) - 1]$, where

$$\theta_n(q) = \frac{q^{n+1} - 1}{q - 1}.$$

In order to facilitate the bijection, Orbiter enumerates representative vectors for the one-dimensional subspaces. The conditions on the vectors are summarized below:

1. The vector is not the zero vector.

2. The rightmost nonzero entry in the vector is one. If it is not, we nomalize the vector so that the rightmost nonzero vector is indeed one. This operation does not change the projective point which is associated with the vector.

The second condition ensures that we list each projective point exactly once. We require two functions, RANK and UNRANK. The function RANK takes a vector $\mathbf{x} \in \mathbb{F}_q^n$, not zero, and maps it to the element in $\mathbb{Z}_N$ representing the projective point $\mathbf{P}(\mathbf{x})$. A frame in $\mathrm{PG}(n, q)$ is a set of $n+2$ points, no $n+1$ in a hyperplane. We assume that the coordinates of a vector are indexed by the elements of $\mathbb{Z}_n$. Also, we let $\mathbf{e}_i$ be the i-th unit vector. A frame for $\mathrm{PG}(n, q)$ is

$$\mathbf{e}_0, \ldots, \mathbf{e}_{n-1}, \mathbf{e}_0 + \cdots + \mathbf{e}_{n-1}.$$

This is the *standard frame.* We start the labeling of points with the standard frame. After these $n+2$ points, we list the remaining points in lexicographic ordering (utilizing right-normalized representative). The points in the small spaces $\mathrm{PG}(2, 2)$, $\mathrm{PG}(3, 2)$ and $\mathrm{PG}(2, 3)$ are listed by their ranks in Table 4.2.

It is possible to request ranks of specific points given by means of homogeneous coordinate vectors. Conversely, given a point rank, the coordinate vector can be produced. The following command computes the rank of

$$\mathbf{P}(3,3,1) = \mathbf{P}(\omega+1, \omega+1, 1)$$

in PG$(2,4)$:

Example 114

```
PG_2_4_rank_point:
▷  $(ORBITER) -v 2 \
▷  ▷  -define v -vector -dense "3,3,1" -format 1 -end \
▷  ▷  -define F -finite_field -q 4 -end \
▷  ▷  -with F -do -finite_field_activity \
▷  ▷  ▷  -rank_point_in_PG v -end
```

The rank turns out to be 20. Conversely, runnnig

Example 115

```
PG_2_4_unrank_point:
▷  $(ORBITER) -v 2 \
▷  ▷  -define v -vector -dense "20" -end \
▷  ▷  -define F -finite_field -q 4 -end \
▷  ▷  -with F -do -finite_field_activity \
▷  ▷  ▷  -unrank_point_in_PG 2 v -end
```

shows that the point with rank 20 is $\mathbf{P}(3,3,1)$.

Line ranking will be discussed next. The lines in the small spaces PG$(2,2)$ and PG$(2,3)$ are listed by their ranks in Table 4.3.

The command

Example 116

```
PG_2_2_rank_lines:
▷  $(ORBITER) -v 2 \
▷  ▷  -define v -vector -format 4 \
▷  ▷  -dense "1,0,0, 0,1,0, 1,0,0, 0,0,1, 1,1,1, 0,1,0, 1,1,1, 0,0,1" \
▷  ▷  -end \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
▷  ▷  -with P -do \
▷  ▷  -projective_space_activity \
▷  ▷  ▷  -rank_lines_in_PG v \
▷  ▷  -end \
```

computes the Orbiter ranks of 4 lines in PG$(2,2)$. The lines are given by two points each. The data is formatted as a matrix with 4 rows of size 6. Each row represents one line, with the coordinates of the two points written one after the other.

Ranking Lines in Small Projective Spaces

PG(2, 2)

$$L_0 = \mathbf{L}\begin{bmatrix}100\\010\end{bmatrix}\{0,1,4\}$$
$$L_1 = \mathbf{L}\begin{bmatrix}100\\011\end{bmatrix}\{0,3,6\}$$
$$L_2 = \mathbf{L}\begin{bmatrix}100\\001\end{bmatrix}\{0,2,5\}$$
$$L_3 = \mathbf{L}\begin{bmatrix}101\\010\end{bmatrix}\{1,3,5\}$$
$$L_4 = \mathbf{L}\begin{bmatrix}101\\011\end{bmatrix}\{4,5,6\}$$
$$L_5 = \mathbf{L}\begin{bmatrix}110\\001\end{bmatrix}\{2,3,4\}$$
$$L_6 = \mathbf{L}\begin{bmatrix}010\\001\end{bmatrix}\{1,2,6\}$$

PG(2, 3)

$$L_0 = \mathbf{L}\begin{bmatrix}100\\010\end{bmatrix}\{0,1,4,5\}$$
$$L_1 = \mathbf{L}\begin{bmatrix}100\\011\end{bmatrix}\{0,3,8,9\}$$
$$L_2 = \mathbf{L}\begin{bmatrix}100\\012\end{bmatrix}\{0,10,11,12\}$$
$$L_3 = \mathbf{L}\begin{bmatrix}100\\001\end{bmatrix}\{0,2,6,7\}$$
$$L_4 = \mathbf{L}\begin{bmatrix}101\\010\end{bmatrix}\{1,3,6,11\}$$
$$L_5 = \mathbf{L}\begin{bmatrix}101\\011\end{bmatrix}\{5,6,8,12\}$$
$$L_6 = \mathbf{L}\begin{bmatrix}101\\012\end{bmatrix}\{4,6,9,10\}$$
$$L_7 = \mathbf{L}\begin{bmatrix}110\\001\end{bmatrix}\{2,3,4,12\}$$
$$L_8 = \mathbf{L}\begin{bmatrix}102\\010\end{bmatrix}\{1,7,9,12\}$$
$$L_9 = \mathbf{L}\begin{bmatrix}102\\011\end{bmatrix}\{4,7,8,11\}$$
$$L_{10} = \mathbf{L}\begin{bmatrix}102\\012\end{bmatrix}\{3,5,7,10\}$$
$$L_{11} = \mathbf{L}\begin{bmatrix}120\\001\end{bmatrix}\{2,5,9,11\}$$
$$L_{12} = \mathbf{L}\begin{bmatrix}010\\001\end{bmatrix}\{1,2,8,10\}$$

Table 4.3: Ranking Lines in Small Projective Spaces

The command

Example 117

```
PG_2_2_unrank_lines:
▷  $(ORBITER) -v 2 \
▷  ▷  -define v -vector -format 4 \
▷  ▷  -dense "0,1,2,3,4,5,6" \
▷  ▷  -end \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
▷  ▷  -with P -do \
▷  ▷  -projective_space_activity \
▷  ▷  ▷  -unrank_lines_in_PG v \
▷  ▷  -end \
```

lists the 7 lines of PG$(2,2)$ by their Orbiter ranks from 0 to 6.

4.3 Incidence Matrices

The projective spaces $\mathrm{PG}(2,q)$ are examples of a more general structure called projective planes. The $\mathrm{PG}(2,F)$, F a field, are distinguished in the class of projective planes by the fact that the theorem of Desargues always holds. They are called the desarguesian projective planes. Other classes of projective planes are the translation planes, see Section 14.2.

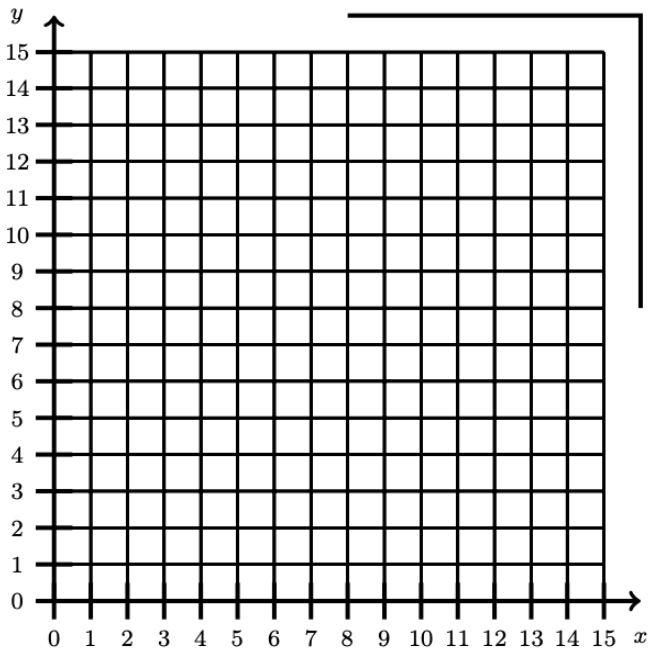
The points in the dearguesian projective plane $\mathrm{PG}(2,q)$ have the coordinates $\mathbf{P}(x,y,z)$, with $x,y,z \in \mathbb{F}_q$, not all three zero. We distingish one line called the line at infinity. The points not on this line form an affine plane $\mathrm{AG}(2,q)$.

The command

Example 118

```
PG_2_16:
▷  $(ORBITER) -v 2 \
▷  ▷  -draw_options -xin 20000 -yin 20000 \
▷  ▷  -radius 200 -line_width 0.3 -nodes_empty -end \
▷  ▷  -define F -finite_field -q 16 -end \
▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
▷  ▷  -with P -do -projective_space_activity \
▷  ▷  ▷  -cheat_sheet \
▷  ▷  -end
▷  pdflatex PG_2_16.tex
▷  $(OPEN) PG_2_16.pdf
```

produces the drawing of $\mathrm{PG}(2,16)$ shown below



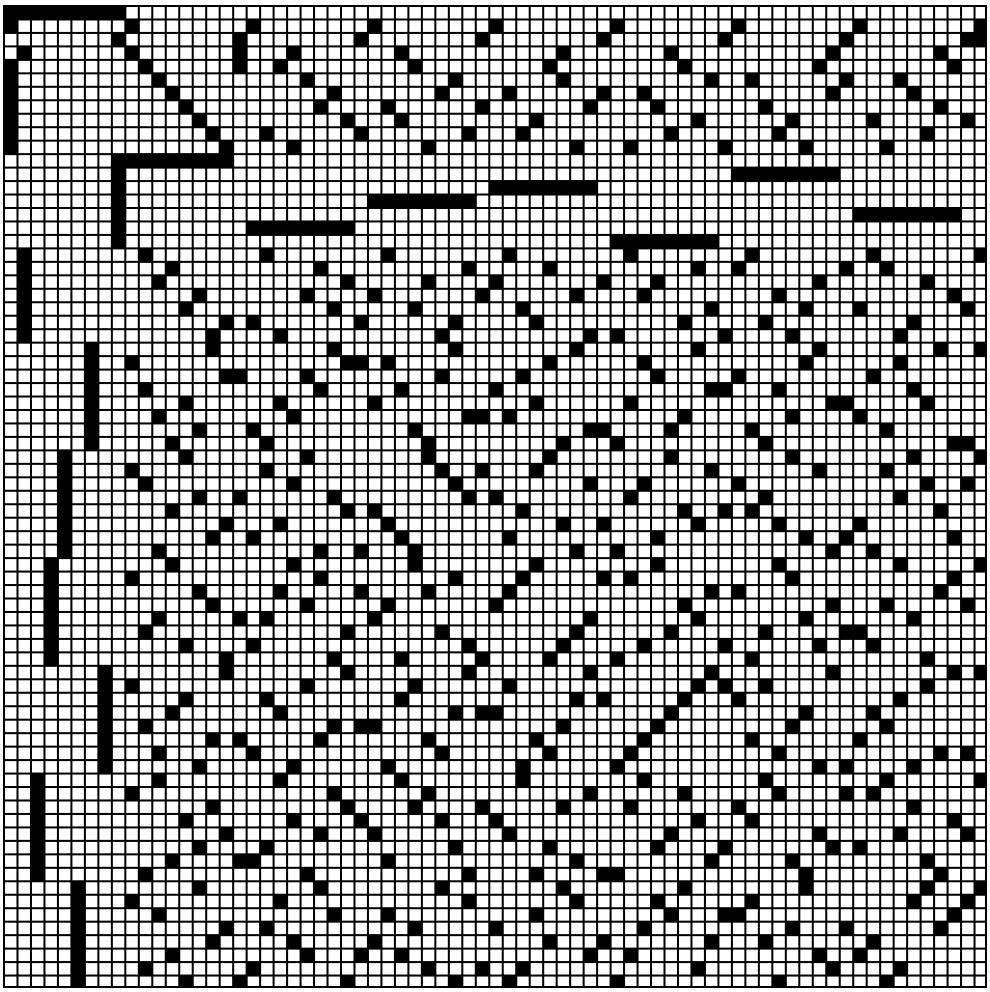
The `-nodes_empty` command is used to suppress the drawing of the nodes. The `-xin 20000` and `-yin 20000` options double the input coordinate system (recall from Table 17.2 that the default values are 10000), which has the effect that the text appears smaller relative to the grid.

It is possible to export the incidence matrix of a projective space to a file. The following example creates $\mathrm{PG}(2, 8)$ and exports the incidence matrix to a csv file. After that, a graphical representation is produced.

Example 119

```
PG_2_8_incidence_matrix:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 8 -end \
▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
▷  ▷  -with P -do -projective_space_activity \
▷  ▷  ▷  -export_point_line_incidence_matrix \
▷  ▷  -end
▷  $(ORBITER) -v 2 \
▷  ▷  -define all_one -vector -repeat 1 73 -end \
▷  ▷  -draw_matrix \
▷  ▷  ▷  -input_csv_file PG_n2_q8_incidence_matrix.csv \
▷  ▷  ▷  -box_width 20 -bit_depth 8 \
▷  ▷  ▷  -partition 3 \
▷  ▷  ▷  ▷  all_one all_one \
▷  ▷  -end
▷  $(OPEN) PG_n2_q8_incidence_matrix_draw.bmp
```

The incidence matrix is shown below



The rows and columns correspond to points and lines, respectively, and are ordered according to their respective Orbiter ranks.

Projective spaces admit a cyclic group action on points and hyperplanes. Such a group is often called a Singer cycle. It is generated from a projectivity defined by the companion matrix of an irreducible polynomial. Let us look at an example. The following command creates a Singer cycle of PG$(2, 4)$

Example 120

```
PG_2_4_with_decomposition:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 4 -end \
▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
▷  ▷  -with P -do -projective_space_activity \
▷  ▷  ▷  -decomposition_by_element_PG \
▷  ▷  ▷  1 "0,1,0, 0,0,1, 2,1,1, 0" \
▷  ▷  ▷  "PG_2_4_singer" \
▷  ▷  -end
▷  pdflatex PG_2_4_singer.tex
▷  $(OPEN) PG_2_4_singer.pdf
```

The output is shown below:

Considering the cyclic group generated by

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ \omega & 1 & 1 \end{bmatrix}_0 = \begin{bmatrix} 010 \\ 001 \\ 211 \end{bmatrix}_0$$

The group is transitive on points and on lines.
Orbits on points:
There are 1 orbits, the orbit lengths are 21
Orbits on lines:
There are 1 orbits, the orbit lengths are 21
Fixed points:
Fixed lines:
Row scheme:

$\rightarrow$	21
21	5

Column scheme:

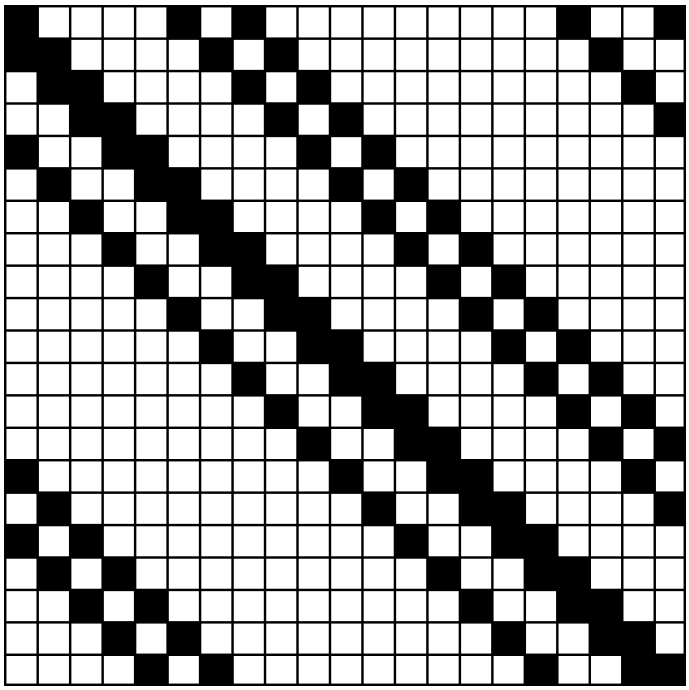
$\downarrow$	21
21	5

The command produces a csv file containing the cyclic incidence matrix, which can be drawn using the follwing command:

Example 121

```
PG_2_4_incma_cyclic:
▷ $(ORBITER) -v 2 \
▷ ▷ -list_arguments \
▷ ▷ -define R -vector -repeat 1 21 -end \
▷ ▷ -define C -vector -repeat 1 21 -end \
▷ ▷ -draw_matrix \
▷ ▷ -input_csv_file PG_2_4_singer_incma_cyclic.csv \
▷ ▷ -box_width 40 -bit_depth 24 \
▷ ▷ -partition 3 R C \
▷ ▷ -end
▷ $(OPEN) PG_2_4_singer_incma_cyclic_draw.bmp
```

The cyclic incidence matrix is

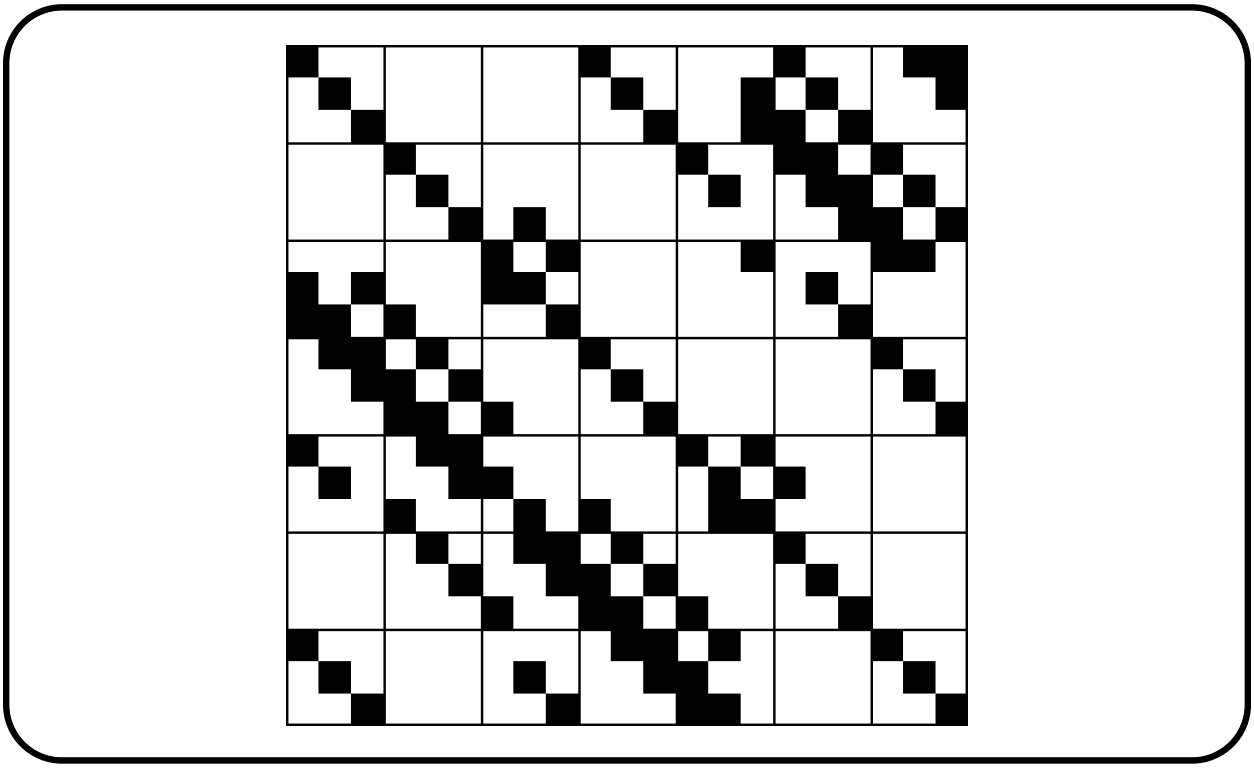


If the number of points is not a prime power, the group acts imprimitively. By considering various subgroups of the Singer group, tactical decompositions are created. For instance, for PG$(2, 4)$, with 21 points, we can consider a subgroup the Singer cycle of order 3, which induces a partition with 7 classes of size 3 on both points and lines:

Example 122

```
PG_2_4_incma_singer_sub_3:
▷  $(ORBITER) -v 2 \
▷  ▷  -list_arguments \
▷  ▷  -define R -vector -repeat 3 7 -end \
▷  ▷  -define C -vector -repeat 3 7 -end \
▷  ▷  -draw_matrix \
▷  ▷  -input_csv_file PG_2_4_singer_incma_subgroup_index_3.csv \
▷  ▷  -box_width 40 -bit_depth 24 \
▷  ▷  -partition 3 R C \
▷  ▷  -end
▷  $(OPEN) PG_2_4_singer_incma_subgroup_index_3_draw.bmp
```

The tactical decomposition of the incidence matrix is



4.4 The Grassmannian

Let V be a finite dimensional vector space and let $\mathfrak{Gr}_k(V)$ be the Grassmannian of k-dimensional subspaces of V. If $\dim(V) = n$, the notation $\mathfrak{Gr}_{n,k}$ is used for $\mathfrak{Gr}_k(V)$. If $V = \mathbb{F}_q^n$, the notation $\mathfrak{Gr}_{n,k,q}$ is used for $\mathfrak{Gr}_k(V)$. The order of the set $\mathfrak{Gr}_{n,k,q}$ can be computed as

$$\begin{bmatrix} n \\ k \end{bmatrix}_q = \prod_{i=0}^{k-1} \frac{q^{n-i} - 1}{q^{k-i} - 1},$$

using the q-binomial coefficient.

Orbiter has an enumerator for the Grassmannian. The purpose of this enumerator is to establish a bijection between the Grassmannian and the integers in the interval $[0, N-1]$, where $N = \begin{bmatrix} n \\ k \end{bmatrix}_q$. In order to do so, Orbiter picks a basis for each subspace. By writing the elements of the basis in the rows of a matrix, a $k \times n$ matrix is obtained. In order to make the matrix unique, we assume it to be in RREF. In coding theory, such a matrix is called a generator matrix.

The Orbiter cheat sheets for $\mathrm{PG}(n, q)$ (see Section 4.1) contain lists of all Grassmannians, provided they are not too big. It is also possible to create cheat sheets specifically for one Grassmannian. For instance, the command

Example 123

```
GR_3_2_2:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
▷  ▷  -with P -do \
▷  ▷  -projective_space_activity \
▷  ▷  ▷  -report_Grassmannian 2 \
▷  ▷  -end
▷  pdflatex Gr_3_2_2.tex
▷  $(OPEN) Gr_3_2_2.pdf
```

produces a list of 2-dimensional subspaces of $\mathbb{F}_2^3$, i.e. the lines of $\mathrm{PG}(2, 2)$:

$$L_0 = \begin{bmatrix} 100 \\ 010 \end{bmatrix} \quad L_3 = \begin{bmatrix} 101 \\ 010 \end{bmatrix} \quad L_6 = \begin{bmatrix} 010 \\ 001 \end{bmatrix}$$
$$L_1 = \begin{bmatrix} 100 \\ 011 \end{bmatrix} \quad L_4 = \begin{bmatrix} 101 \\ 011 \end{bmatrix}$$
$$L_2 = \begin{bmatrix} 100 \\ 001 \end{bmatrix} \quad L_5 = \begin{bmatrix} 110 \\ 001 \end{bmatrix}$$

The following command illustrates how to rank lines. In the example, we consider lines in $\mathrm{PG}(3, 3)$. The lines are given as vectors of length 8. Three lines are given in `v1` and three lines are given in `v2`.

Example 124

```
PG_3_3_rank_lines:
▷  $(ORBITER) -v 2 \
▷  ▷  -define v1 -vector -format 3 \
▷  ▷  -dense "1,0,2,2,0,1,1,2, 1,0,2,0,0,1,1,2, 1,0,2,2,0,1,2,1" \
▷  ▷  -end \
▷  ▷  -define v2 -vector -format 3 \
▷  ▷  -dense "1,0,0,0,0,1,0,0, 1,0,0,0,0,0,0,1, 0,1,0,0,0,0,2,1" \
▷  ▷  -end \
▷  ▷  -define F -finite_field -q 3 -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷  ▷  -with P -do \
▷  ▷  -projective_space_activity \
▷  ▷  ▷  -rank_lines_in_PG v1 \
▷  ▷  -end \
▷  ▷  -with P -do \
▷  ▷  -projective_space_activity \
▷  ▷  ▷  -rank_lines_in_PG v2 \
▷  ▷  -end
```

In the next example, we unrank six lines in $\mathrm{PG}(3,5)$.

Example 125

```
PG_3_5_unrank_lines:
▷  $(ORBITER) -v 2 \
▷  ▷  -define v -vector \
▷  ▷  ▷  -dense "0,36,72,108,144,805" \
▷  ▷  -end \
▷  ▷  -define F -finite_field -q 5 -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷  ▷  -with P -do \
▷  ▷  -projective_space_activity \
▷  ▷  ▷  -unrank_lines_in_PG v \
▷  ▷  -end
```

The following command produces a list of planes through a line. In the example, the line is 0. The projective space is $\mathrm{PG}(3,8)$

Example 126

```
planes_in_pencil:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 8 -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷  ▷  -with P -do \
▷  ▷  -projective_space_activity \
▷  ▷  ▷  -planes_through_line 0 \
```

```
▷ ▷ -end
```

4.5 Algebraic Sets

A set of points V in $\mathrm{PG}(n,q)$ is algebraic if there is a set of homogeneous polynomials $p_1,\ldots,p_r$ whose roots over $\mathbb{F}_q$ are the given set. In this case, we write $V = \mathbf{v}(p_1,\ldots,p_r)$. The set V is often called the variety of $p_1,\ldots,p_r$.

Conversely, given a set of points V in $\mathrm{PG}(n,q)$, the ideal $I(V)$ is the set of homogeneous polynomials in $\mathbb{F}_q[X_0,\ldots,X_n]$ which vanish on all of V. This set is an ideal in the polynomial ring. In $\mathrm{PG}(n,q)$, every set is algebraic of degree at most $(n+1)(q-1)$ [32]. The associated polynomial is unique and known as the algebraic normal form of the set.

In order to work with algebraic sets, multivariate polynomial rings are required. For details, see Section 5.2.

Suppose we are interested in $\mathbb{F}_{11}$-rational points of the elliptic curve $y^2 = x^3+x+3$. We write $x^3+3-y^2+x = 0$. Homogenizing yields $X^3 + 3Z^3 - Y^2Z + XZ = 0$. Using X_0, X_1, X_2 instead of X, Y, Z yields

$$X_0^3 + 3X_2^3 + 10X_1^2X_2 + X_0X_2^2 = 0.$$

Using the indexing of monomials from Table 5.4, we record the coefficient vector of the equation as sequence

$$(1,0,3,0,0,0,10,1,0,0).$$

The Orbiter command

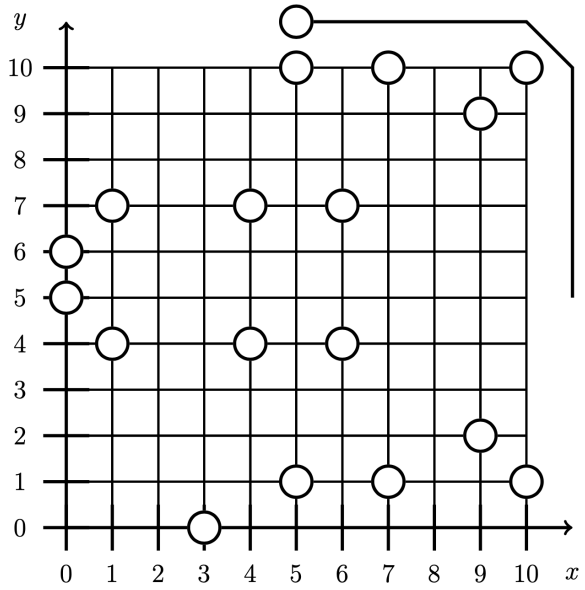
Example 127

```
EC_11_EQUATION="1,0,3,0,0,0,10,1,0,0"
```

Example 128

```
EC_11.txt:
▷ $(ORBITER) -v 2 \
▷ ▷ -define F -finite_field -q 11 -end \
▷ ▷ -define R -polynomial_ring -field F \
▷ ▷ ▷ -number_of_variables 3 \
▷ ▷ ▷ -homogeneous_of_degree 3 \
▷ ▷ ▷ -end \
▷ ▷ -define P -projective_space -n 2 -field F -v 0 -end \
▷ ▷ -define EC -geometric_object P \
▷ ▷ ▷ -projective_variety R \
▷ ▷ ▷ ▷ "EC_11" "EC\_11" \
▷ ▷ ▷ ▷ $(EC_11_EQUATION) \
▷ ▷ ▷ -end \
▷ ▷ -with EC -do -combinatorial_object_activity -save \
▷ ▷ -end
```

creates the algebraic set associated to the cubic curve $y^2 = x^3 + x + 3$ in $\mathrm{PG}(2,11)$. It turns out that there are exactly 18 points over $\mathbb{F}_{11}$. A picture is shown below:



Suppose we want to create the Hirschfeld surface with equation

$$X_0^2X_3 + X_1^2X_2 + X_1X_2^2 + X_0X_3^2 = 0.$$

Based on the partition ordering of Figure 5.5, the equation is coded as coefficient vector

$$(0,0,0,0,0,0,1,0,1,0,0,1,0,1,0,0,0,0,0,0).$$

The following command can be used to create the variety over $\mathbb{F}_4$:

Example 129

```
HIRSCHFELD_SURFACE_EQUATION="0,0,0,0,0,0,1,0,1,0, 0,1,0,1,0,0,0,0,0,0"
```

Example 130

```
Hirschfeld_surface_q4.txt:
▷ $(ORBITER) -v 2 \
▷ ▷ -define F -finite_field -q 4 -end \
▷ ▷ -define R -polynomial_ring -field F \
▷ ▷ ▷ -number_of_variables 4 \
▷ ▷ ▷ -homogeneous_of_degree 3 \
▷ ▷ ▷ -end \
▷ ▷ -define P -projective_space -n 3 -field F -v 0 -end \
▷ ▷ -define H4 -geometric_object P \
▷ ▷ ▷ -projective_variety R \
▷ ▷ ▷ ▷ "Hirschfeld_surface_q4" \
▷ ▷ ▷ ▷ "Hirschfeld\_surface\_q4" \
▷ ▷ ▷ ▷ $(HIRSCHFELD_SURFACE_EQUATION) \
▷ ▷ ▷ -end \
▷ ▷ -with H4 -do -combinatorial_object_activity -save \
```

```
▷ ▷ -end
```

A file called `Hirschfeld_surface_q4.txt` is created. The file contains the Orbiter ranks of the 45 points on the surface.

4.6 The Klein Quadric and the Plücker Map

Orbiter can work with the Grassmannian over a finite field. In particular, Orbiter offers indexing for subspaces. For the special case of the Grassmannian $\mathfrak{G}r_{4,2}(V)$, Plücker coordinates can be used to identify $\mathfrak{G}r_{4,2}(V)$ with the $Q^+(5,q)$ (Klein) quadric. Here is an example. The command

Example 131

```
GR_4_2_2:
▷ $(ORBITER) -v 2 \
▷ ▷ -define F -finite_field -q 2 -end \
▷ ▷ -define P -projective_space -n 3 -field F -v 0 -end \
▷ ▷ -with P -do \
▷ ▷ -projective_space_activity \
▷ ▷ ▷ -report_Grassmannian 2 \
▷ ▷ -end
▷ pdflatex Gr_4_2_2.tex
▷ $(OPEN) Gr_4_2_2.pdf
```

creates the elements of $\mathfrak{G}r_{4,2,2}$ and lists them together with their Plücker coordinates. The following list is produced (output shortened):

There are 35 lines:

$$L_0 = \begin{bmatrix} 1000 \\ 0100 \end{bmatrix} = \mathbf{Pl}(1,0,0,0,0,0) \qquad L_2 = \begin{bmatrix} 1000 \\ 0101 \end{bmatrix} = \mathbf{Pl}(1,0,0,0,1,0)$$

$$L_1 = \begin{bmatrix} 1000 \\ 0110 \end{bmatrix} = \mathbf{Pl}(1,0,1,0,0,0) \qquad \vdots$$

$$L_{34} = \begin{bmatrix} 0010 \\ 0001 \end{bmatrix} = \mathbf{Pl}(0,1,0,0,0,0)$$

The Plücker coordinates satisfy

$$p_{12}p_{34} - p_{13}p_{24} + p_{14}p_{23} = 0$$

and hence belong to the Klein quadric $Q^+(5,q)$. Orthogonal spaces and quadrics will be discussed in Section 4.7.

The Orbiter labeling of points of the $Q^+(5,q)$ quadric (see Section 4.7) can then be used to enumerate the lines of $\mathrm{PG}(3,q)$ in a second, different way. In the example of $\mathrm{PG}(3,2)$, this yields the following list (output shortened):

$$0 = \mathbf{Pl}(1,0,0,0,0,0) = L_0 = \begin{bmatrix} 1000 \\ 0100 \end{bmatrix} \qquad 2 = \mathbf{Pl}(0,0,1,0,0,0) = L_4 = \begin{bmatrix} 1000 \\ 0010 \end{bmatrix}$$

$$1 = \mathbf{Pl}(0,1,0,0,0,0) = L_{34} = \begin{bmatrix} 0010 \\ 0001 \end{bmatrix} \qquad \vdots$$

$$34 = \mathbf{Pl}(0,1,1,1,1,1) = L_{26} = \begin{bmatrix} 1101 \\ 0011 \end{bmatrix}$$

Types of Quadrics

Type	Quadratic Form	# Points
$Q^+(n,q)$ Hyperbolic (n is odd)	$\sum_{i=0}^{\frac{n-1}{2}-1} X_{2i}X_{2i+1}$	$\frac{(q^{(n+1)/2}-1)(q^{(n-1)/2}+1)}{q-1}$
$Q^-(n,q)$ Elliptic (n is odd)	$p(X_{n-1},X_n)+\sum_{i=0}^{\frac{n-1}{2}-1} X_{2i}X_{2i+1}$	$\frac{(q^{(n+1)/2}+1)(q^{(n-1)/2}-1)}{q-1}$
$Q(n,q)$ Parabolic (n is even)	$X_0^2+\sum_{i=0}^{\frac{n}{2}-1} X_{2i+1}X_{2i+2}$	$\frac{q^n-1}{q-1}$

Table 4.4: Types of Quadrics

Creating an Orthogonal Space

Command	Arguments	Purpose
`-label_txt`	L	Set the ascii-label of the space. The label is used for things like file names etc. A default label will be used if this option is not given.
`-label_tex`	L	Set the tex-label of the space. The label is used within latex reports. A default label will be used if this option is not given.
`-without_group`		Do not create the orthogonal group.

Table 4.5: Creating an Orthogonal Space

4.7 Orthogonal Spaces

Orbiter can create and work with orthogonal spaces and their groups. An orthogonal space is created by a quadratic form. We assume that the form is nondegenerate. There are three types of nondegenerate quadratic forms in $\mathrm{PG}(n,q)$. Two when n is odd (hyperbolic and elliptic) and one if n is even (parabolic). Basic information about these quadrics and their representative quadratic forms in Orbiter is given in Table 4.4. Here, $p(X,Y)=c_1X^2+c_2XY+c_3Y^2\in\mathbb{F}_q[X,Y]$ is irreducible over $\mathbb{F}_q$.
To create an orthogonal space, the

`-orthogonal_space` ϵ d q `-end`

command can be used. Here, $d=n+1$, q is the order of the finite field, and

$$\epsilon=\begin{cases} 1 & \text{hyperbolic type} \quad Q^+(d-1,q), \quad d \text{ even} \\ 0 & \text{elliptic type} \quad Q(d-1,q), \quad d \text{ odd} \\ -1 & \text{hyperbolic type} \quad Q^-(d-1,q), \quad d \text{ even} \end{cases}$$

In Table 4.5, Orbiter command options for creating orthogonal spaces are shown.

For instance, the following command creates $Q(3,2)$ together with its group $PGO^+(4,2)$:

Example 132

```
Op_4_2:
▷  $(ORBITER) -v 2 \
```

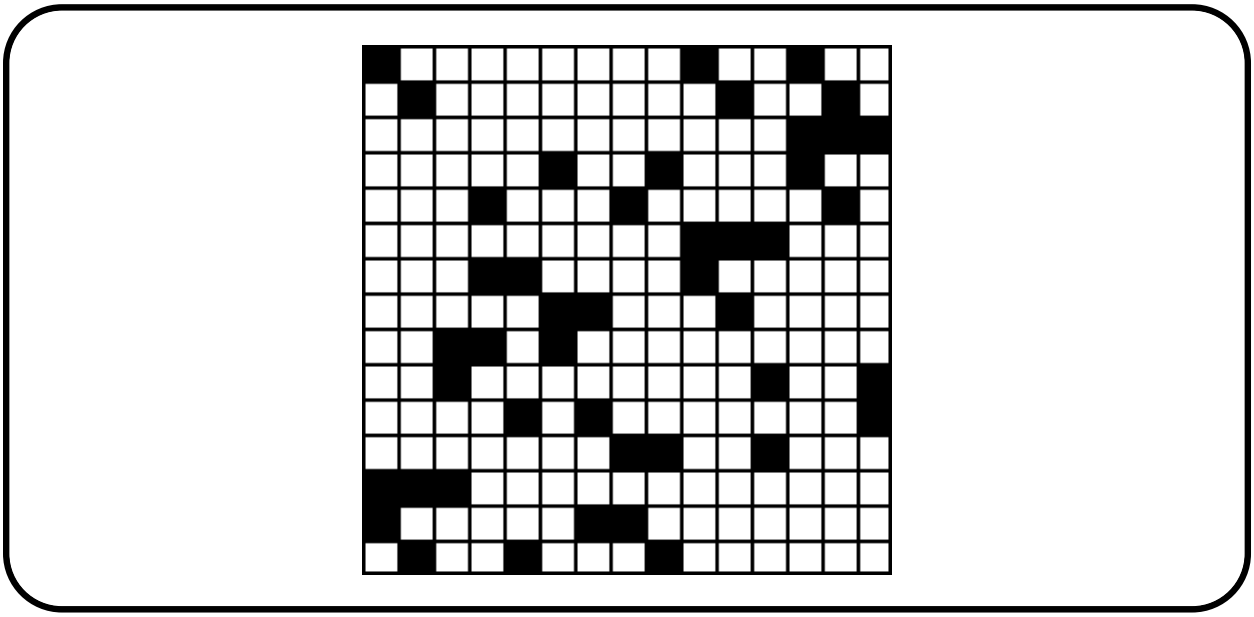
```
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define O -orthogonal_space 1 4 F -without_group -end \
▷  ▷  -with O -do -orthogonal_space_activity \
▷  ▷  ▷  -cheat_sheet_orthogonal -end
▷  pdflatex O_1_4_2_report.tex
▷  $(OPEN) O_1_4_2_report.pdf
```

The next command creates $Q(4,2)$ together with its group $PGO(5,2)$. There are 15 points and 15 lines. The geometry is a configuration 15_3 which is also known as the Cremona-Richmond configuration.

Example 133

```
O_5_2_incidence_matrix.csv:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define O -orthogonal_space 0 5 F -without_group -end \
▷  ▷  -with O -do -orthogonal_space_activity \
▷  ▷  ▷  -export_point_line_incidence_matrix \
▷  ▷  -end
▷  $(ORBITER) -v 2 \
▷  ▷  -define all_one_r -vector -repeat 1 15 -end \
▷  ▷  -define all_one_c -vector -repeat 1 15 -end \
▷  ▷  -draw_matrix \
▷  ▷  ▷  -input_csv_file O_5_2_incidence_matrix.csv \
▷  ▷  ▷  -box_width 20 -bit_depth 8 \
▷  ▷  ▷  -partition 2 \
▷  ▷  ▷  ▷  all_one_r all_one_c \
▷  ▷  -end
▷  $(OPEN) O_5_2_incidence_matrix_draw.bmp
```

The command also creates a bitmap drawing of the incidence matrix between points and lines of $Q(4,2)$.



The Orbiter indexing for points and lines of quadrics is used to address rows and columns.

By default, the orthogonal space is created together with the orthogonal group $\mathrm{P\Gamma O}(n+1,q)$. When q is prime, the group $\mathrm{PGO}(n+1,q)$ is created instead (the groups are isomorphic in this case, and $\mathrm{PGO}(n+1,q)$ is a bit more efficient). For large orthogonal spaces, creating the group is expensive in terms of time and memory. The a command `-without_group` can be used to prevent the group from being created. For instance

```
-define O -orthogonal_space 1 6 2 -end
```

creates an object O of type $Q^+(5,2)$. In Table 4.6, Orbiter activities for orthogonal spaces are shown.

The command

Example 134

```
Op_6_2:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define O -orthogonal_space 1 6 F -without_group -end \
▷  ▷  -with O -do -orthogonal_space_activity \
▷  ▷  ▷  -cheat_sheet_orthogonal -end
▷  pdflatex O_1_6_2_report.tex
▷  $(OPEN) O_1_6_2_report.pdf
```

produces a cheat sheet for the quadric $Q^+(5,2)$. This is the Klein quadric from Section 4.6. Orbiter produces the following output. At the top is the tactical decomposition of the incidence matrix between points and lines with respect to a hyperbolic pair. After that, the points and lines are listed (output shortened):

Orthogonal Space Activities

Command	Arguments	Purpose
-create_BLT_set	descr	Creates a BLT-set of $Q(4,q)$. See Section 14.4.
-cheat_sheet_orthogonal		Create a cheat sheet.
-print_points	v	Print the points whose ranks are given in the vector v.
-print_lines	v	Print the lines whose ranks are given in the vector v.
-unrank_line_through_two_points	$p1$ $p2$	Determine the rank of the line through the points whose ranks are $p1$ and $p2$.
-lines_on_point	p	Create the ranks of all lines through the point whose rank is p.
-perp	L	Determine the common perp of a set of points. The point ranks are given in the list L.
-export_point_line_incidence_matrix		Create a csv file with the point line incidence matrix of the space.
-intersect_with_subspace	M	Find the points in the intersection of the quadric with the subspace whose generating matrix has label M.

Table 4.6: Orthogonal Space Activities

$\rightarrow$	9	36	18	18	6	9	9
6	3	6	0	0	0	0	0
9	0	4	4	0	0	1	0
9	0	4	0	4	0	0	1
9	1	0	2	2	2	1	1
1	0	0	0	0	0	9	0
1	0	0	0	0	0	0	9

$\downarrow$	9	36	18	18	6	9	9
6	2	1	0	0	0	0	0
9	0	1	2	0	0	1	0
9	0	1	0	2	0	0	1
9	1	0	1	1	3	1	1
1	0	0	0	0	0	1	0
1	0	0	0	0	0	0	1

The number of points is 35
points:
$P_0 = (1,0,0,0,0,0)$
$P_1 = (0,1,0,0,0,0)$
$P_2 = (0,0,1,0,0,0)$
$P_3 = (1,0,1,0,0,0)$
$P_4 = (0,1,1,0,0,0)$

$P_5 = (0,0,0,1,0,0)$
$P_6 = (1,0,0,1,0,0)$
$P_7 = (0,1,0,1,0,0)$
$P_8 = (1,1,1,1,0,0)$
$P_9 = (0,0,0,0,1,0)$
$P_{10} = (1,0,0,0,1,0)$
$P_{11} = (0,1,0,0,1,0)$
$P_{12} = (0,0,1,0,1,0)$
$P_{13} = (1,0,1,0,1,0)$
$P_{14} = (0,1,1,0,1,0)$
$P_{15} = (0,0,0,1,1,0)$
$P_{16} = (1,0,0,1,1,0)$
$P_{17} = (0,1,0,1,1,0)$
$P_{18} = (1,1,1,1,1,0)$
$P_{19} = (0,0,0,0,0,1)$
$P_{20} = (1,0,0,0,0,1)$
$P_{21} = (0,1,0,0,0,1)$
$P_{22} = (0,0,1,0,0,1)$
$P_{23} = (1,0,1,0,0,1)$
$P_{24} = (0,1,1,0,0,1)$
$P_{25} = (0,0,0,1,0,1)$
$P_{26} = (1,0,0,1,0,1)$
$P_{27} = (0,1,0,1,0,1)$
$P_{28} = (1,1,1,1,0,1)$
$P_{29} = (1,1,0,0,1,1)$
$P_{30} = (1,1,1,0,1,1)$
$P_{31} = (1,1,0,1,1,1)$
$P_{32} = (0,0,1,1,1,1)$
$P_{33} = (1,0,1,1,1,1)$
$P_{34} = (0,1,1,1,1,1)$
The number of lines is 105

$$L_0 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix} \{P_0, P_{32}, P_{33}\}$$

$$L_1 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix} \{P_1, P_{32}, P_{34}\}$$

$$L_{104} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \{P_8, P_9, P_{18}\}$$

Orbiter has enumerators for points and lines in orthogonal spaces. For small spaces, the cheat sheet lists points and lines in the Orbiter ordering. Creating the groups can be expensive. For large spaces, it may be necessary to disable the group using the `-without_group` option. The command

Example 135

```
Op_6_64_line_rank:
▷  $(ORBITER) -v 4 \
▷  ▷  -define F -finite_field -q 64 -end \
▷  ▷  -define O -orthogonal_space 1 6 F -without_group -end \
▷  ▷  -with O -do -orthogonal_space_activity \
▷  ▷  ▷  -unrank_line_through_two_points 15447347 15225451 \
▷  ▷  -end
```

computes the Orbiter rank of the line through the points with rank 15447347 and 15225451, respectively. The rank of the line is 16767254. These ranks refer to the orthogonal geometry. They are different from the ranks of points and lines in projective spaces.

It is possible to create reports for orthogonal spaces without group. In this case, the group information will be skipped. For instance, the following command creates a report for $Q(5, 64)$:

Example 136

```
Op_6_64_report:
▷ $(ORBITER) -v 4 \
▷ ▷ -define F -finite_field -q 64 -end \
▷ ▷ -define O -orthogonal_space 1 6 F -without_group -end \
▷ ▷ -with O -do -orthogonal_space_activity \
▷ ▷ ▷ -cheat_sheet_orthogonal \
▷ ▷ -end
▷ pdflatex O_1_6_64_report.tex
▷ $(OPEN) O_1_6_64_report.pdf
```

The report does not show information about the group. However, it still contains the tactical decomposition with respect to a hyperbolic pair. The printing of points is restricted to small spaces only.

The group is not available.
The quadratic form is:

$$X_0X_1 + X_2X_3 + X_4X_5 = 0$$

→	16769025	1090252800	532350	532350	130	4225	4225
16511040	65	4160	0	0	0	0	0
266175	0	4096	128	0	0	1	0
266175	0	4096	0	128	0	0	1
4225	3969	0	126	126	2	1	1
1	0	0	0	0	0	4225	0
1	0	0	0	0	0	0	4225

↓	16769025	1090252800	532350	532350	130	4225	4225
16511040	64	63	0	0	0	0	0
266175	0	1	64	0	0	63	0
266175	0	1	0	64	0	0	63
4225	1	0	1	1	65	1	1
1	0	0	0	0	0	1	0
1	0	0	0	0	0	0	1

The number of points is 17047617
Too many points to print.
The number of lines is 1108095105

To study BLT-sets in $Q(4, q)$, see Section 14.4.

According to Table 4.4, Orbiter uses the equation

$$X_0X_1 + X_2X_3 + X_4X_5 = 0$$

to define the Klein quadric. An elliptic quadric is an ovoid of the Klein quadric that is obtained by intersecting the quadric with a suitable solid. In PG$(5, 5)$, the subspace generated by the rows of the matrix

$$\begin{bmatrix} 1 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

is such a subspace. The ordering of columns corresponds to the natural ordering of the variables as $X_0, X_1, X_2, X_3, X_4, X_5$. The following command produces a list of points of an elliptic quadric ovoid in $Q^+(5, 5)$.

Example 137

```
elliptic_quadric_subspace:
▷ $(ORBITER) -v 3 \
▷ ▷ -define F -finite_field -q 5 -end \
▷ ▷ -define v -vector -format 4 \
▷ ▷ ▷ -dense "1,3,0,0,0,0, 0,0,1,0,0,0, 0,0,0,1,0,0, 0,0,0,0,1,1" \
▷ ▷ -end \
▷ ▷ -define O -orthogonal_space 1 6 F -end \
▷ ▷ -with O -do -orthogonal_space_activity \
▷ ▷ ▷ -intersect_with_subspace v \
▷ ▷ -end
```

The elliptic quadric has 26 points.

The coding of points and line in orthogonal spaces is different from the coding of points in projective spaces. We will create and print a set called BLT set (after [4]). This is a set of $q + 1$ points on the $Q(4, q)$ quadric satisfying a special geometric property. According to Table 4.4, Orbiter uses the equation

$$X_0^2 + X_1X_2 + X_3X_4 = 0$$

to define the $Q(4, q)$ quadric. The following example creates the BLT-set with Orbiter catalogue number #1 in $Q(4, 7)$:

Example 138

```
BLT_database_7_1:
▷ $(ORBITER) -v 2 \
▷ ▷ -define F -finite_field -q 7 -end \
▷ ▷ -define P -projective_space -n 4 -field F -v 0 -end \
▷ ▷ -define S -geometric_object P \
▷ ▷ ▷ -BLT_database 1 \
▷ ▷ -end \
▷ ▷ -with S -do -combinatorial_object_activity -save \
▷ ▷ -end
```

The set is stored in a file. The next command reads the file and prints the elements of the set:

Example 139

```
BLT_database_7_1_print:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 7 -end \
▷  ▷  -define O -orthogonal_space 0 5 F -without_group -end \
▷  ▷  -define S -set -file_orbiter_format BLT_7_1.txt -end \
▷  ▷  -with O -do -orthogonal_space_activity \
▷  ▷  ▷  -print_points S -end
▷  pdflatex S_set_report.tex
▷  $(OPEN) S_set_report.pdf
```

The command produces the following list of points, comprising the second BLT-set over $\mathbb{F}_7$.

A set of points of size 8
The Points:
$0 : P_0 = (0, 1, 0, 0, 0)$
$1 : P_1 = (0, 0, 1, 0, 0)$
$2 : P_{40} = (0, 1, 2, 6, 2)$
$3 : P_{41} = (0, 1, 4, 3, 1)$
$4 : P_{225} = (1, 6, 6, 5, 1)$
$5 : P_{270} = (1, 5, 5, 4, 4)$
$6 : P_{241} = (1, 2, 2, 2, 1)$
$7 : P_{340} = (1, 1, 1, 4, 3)$

More on BLT-sets can be found in Section 14.4.

The next command prints points and lines of the $W(2)$, also known as the Doily. It is an example of a generalized quadrangle.

Example 140

```
Doily_W_2:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define O -orthogonal_space 0 5 F -without_group -end \
▷  ▷  -define W2_points -set -loop 0 15 1 -end \
▷  ▷  -define W2_lines -set -loop 0 15 1 -end \
▷  ▷  -with O -do \
▷  ▷  -orthogonal_space_activity \
▷  ▷  ▷  -print_points W2_points \
▷  ▷  -end \
▷  ▷  -with O -do \
▷  ▷  -orthogonal_space_activity \
▷  ▷  ▷  -print_lines W2_lines \
▷  ▷  -end
▷  pdflatex W2_points_set_report.tex
▷  $(OPEN) W2_points_set_report.pdf
▷  pdflatex W2_lines_set_of_lines_report.tex
▷  $(OPEN) W2_lines_set_of_lines_report.pdf
```

4.8 Hermitian Varieties

Orbiter has enumerators for points of the hermitian variety $\mathrm{H}(k, Q)$. Here, Q is a square, and so $q = \sqrt{Q}$ is an integer. The equation of the variety is

$$\sum_{i=0}^{k} X_i^{q+1} = 0.$$

The command

Example 141

```
H_2_4:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 4 -end \
▷  ▷  -with F -do -finite_field_activity \
▷  ▷  ▷  -cheat_sheet_hermitian 2 -end
▷  pdflatex H_2_4.tex
▷  $(OPEN) H_2_4.pdf
```

produces a cheat sheet for the variety $H(2, 4)$:

The Hermitian variety $\mathrm{H}(2, 4)$ contains 9 points:

$P_0 = (1, 1, 0) = 4$
$P_1 = (2, 1, 0) = 5$
$P_2 = (3, 1, 0) = 6$
$P_3 = (1, 0, 1) = 7$
$P_4 = (2, 0, 1) = 8$
$P_5 = (3, 0, 1) = 9$
$P_6 = (0, 1, 1) = 10$
$P_7 = (0, 2, 1) = 13$
$P_8 = (0, 3, 1) = 17$

All points: (4, 5, 6, 7, 8, 9, 10, 13, 17)

The command

Example 142

```
H_3_4:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 4 -end \
▷  ▷  -with F -do -finite_field_activity \
▷  ▷  ▷  -cheat_sheet_hermitian 3 -end
▷  pdflatex H_3_4.tex
▷  $(OPEN) H_3_4.pdf
```

produces a cheat sheet for the variety $H(3, 4)$.

The Hermitian variety $\mathrm{H}(3,4)$ contains 45 points:

$P_0 = (1,1,0,0) = 5$
$P_1 = (2,1,0,0) = 6$
$P_2 = (3,1,0,0) = 7$
$P_3 = (1,0,1,0) = 8$
$P_4 = (2,0,1,0) = 9$
$P_5 = (3,0,1,0) = 10$
$P_6 = (0,1,1,0) = 11$
$P_7 = (0,2,1,0) = 15$
$P_8 = (0,3,1,0) = 19$
$P_9 = (1,0,0,1) = 23$
$P_{10} = (2,0,0,1) = 24$
$P_{11} = (3,0,0,1) = 25$
$P_{12} = (0,1,0,1) = 26$
$P_{13} = (0,2,0,1) = 30$
$P_{14} = (0,3,0,1) = 34$
$P_{15} = (1,1,1,1) = 4$
$P_{16} = (2,1,1,1) = 43$
$P_{17} = (3,1,1,1) = 44$
$P_{18} = (1,2,1,1) = 46$
$P_{19} = (2,2,1,1) = 47$
$P_{20} = (3,2,1,1) = 48$
$P_{21} = (1,3,1,1) = 50$
$P_{22} = (2,3,1,1) = 51$
$P_{23} = (3,3,1,1) = 52$
$P_{24} = (0,0,1,1) = 38$
$P_{25} = (1,1,2,1) = 58$
$P_{26} = (2,1,2,1) = 59$
$P_{27} = (3,1,2,1) = 60$
$P_{28} = (1,2,2,1) = 62$
$P_{29} = (2,2,2,1) = 63$
$P_{30} = (3,2,2,1) = 64$
$P_{31} = (1,3,2,1) = 66$
$P_{32} = (2,3,2,1) = 67$
$P_{33} = (3,3,2,1) = 68$
$P_{34} = (0,0,2,1) = 53$
$P_{35} = (1,1,3,1) = 74$
$P_{36} = (2,1,3,1) = 75$
$P_{37} = (3,1,3,1) = 76$
$P_{38} = (1,2,3,1) = 78$
$P_{39} = (2,2,3,1) = 79$
$P_{40} = (3,2,3,1) = 80$
$P_{41} = (1,3,3,1) = 82$
$P_{42} = (2,3,3,1) = 83$
$P_{43} = (3,3,3,1) = 84$
$P_{44} = (0,0,3,1) = 69$

All points: (5, 6, 7, 8, 9, 10, 11, 15, 19, 23, 24, 25, 26, 30, 34, 4, 43, 44, 46, 47, 48, 50, 51, 52, 38, 58, 59, 60, 62, 63, 64, 66, 67, 68, 53, 74, 75, 76, 78, 79, 80, 82, 83, 84, 69)

Coincidentally, this Hermitian variety is the Hirschfeld cubic surface over $\mathbb{F}_4$.

4.9 Advanced Topics

The Orbiter commands associated with projective space objects are summarized in Tables 4.7-4.10.

Table 4.11 lists Orbiter global commands related to projective geometries. These commands do not need an object of type projective space in order to be invoked.

Suppose we want to study the fix structure of a collineation in projective space. Suppose we want to do so for the Baer collineation

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_1 .$$

It fixes a subgeometry PG(3, 2). The command

Example 143

```
fixed_objects_2A:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 4 -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷  ▷  -with P -do \
▷  ▷  -projective_space_activity \
▷  ▷  ▷  -report_fixed_objects \
▷  ▷  ▷  ▷  "1,0,0,0, 0,1,0,0, 0,0,1,0, 0,0,0,1, 1" \
▷  ▷  ▷  ▷  fix_structure_2A.tex \
▷  ▷  -end
▷  pdflatex fix_structure_2A.tex
▷  $(OPEN) fix_structure_2A.pdf
```

can be used. The output is shown below. There are 15 fixed points, 35 fixed lines, and 15 fixed planes. These objects form a Baer subgeometry.

The element

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_1$$

has the following fixed objects:
There are 15 / 85 fixed points, they are:

0 / 15 = 0 : (1, 0, 0, 0)
1 / 15 = 1 : (0, 1, 0, 0)
2 / 15 = 2 : (0, 0, 1, 0)
3 / 15 = 3 : (0, 0, 0, 1)
4 / 15 = 4 : (1, 1, 1, 1)
5 / 15 = 5 : (1, 1, 0, 0)
6 / 15 = 8 : (1, 0, 1, 0)
7 / 15 = 11 : (0, 1, 1, 0)
8 / 15 = 12 : (1, 1, 1, 0)

Projective Space Activities (Part 1)		
Command	Arguments	Purpose
-export_point_line_incidence_matrix		Create a csv file of the point line incidence matrix.
-export_restricted_point_line_incidence_matrix	points lines	Create a csv file of the incidence matrix between the given set of points (rows) and lines (columns).
-export_cubic_surface_line_vs_line_incidence_matrix		
-export_cubic_surface_line_tritangent_plane_incidence_matrix		
-export_double_sixes		
-table_of_cubic_surfaces_compute_properties	fname q_0 coloffset	Computes the properties of cubic surfaces over a different field.
-cubic_surface_properties_analyze	fname q_0	Explores the properties of cubic surfaces over a different field.
-canonical_form_of_code	label m n matrix	Compute the automorphism group of a linear code using Nauty. See Section 10.
-map	formula parameters	Evaluate a formula using the given parameters
-affine_map	R formula parameters	Evaluate an affine map given by a formula using the given parameters over the ring R.
-analyze_del_Pezzo_surface	label parameters	
-decomposition_by_element_PG	power elt fname	Analyze the orbit structure of the cyclic group generated by the given element in the action on $\mathrm{PG}(n,q)$. The command also produces a latex report.
-decomposition_by_subgroup	label descr	Analyze the orbit structure of the subgroup H in the action on $\mathrm{PG}(n,q)$. The subgroup must be a linear group, and the description of H must come from the commands from Section 6.2. Produce a latex report.
-table_of_quartic_curves		Export the classification of quartic curves to a csv file.
-table_of_cubic_surfaces		Export the classification of cubic surfaces to a csv file.

Table 4.7: Projective Space Activities (Part 1)

Projective Space Activities (Part 2)

Command	Arguments	Purpose
-sweep	fname	
-sweep_4_15_lines	fname surface-descr	
-sweep_F_beta_9_lines	fname surface-descr	
-sweep_6_9_lines	fname surface-descr	
-sweep_4_27	fname surface-descr	
-sweep_4_L9_E4	fname surface-descr	
-cheat_sheet		Produce a cheat sheet for $\mathrm{PG}(n, q)$
-set_stabilizer	k fname-mask N col-label fname-out	Compute canonical form of sets using the substructure algorithm.
-conic_type	t set	Compute the conic type of the given set (given by its label). Record intersections of size $\geq t$ only.
-arc_with_given_set_as_s_lines_after_dualizing	$sz\ d\ d_{\min}\ s$	Finds arcs with the given set as s-lines.
-arc_with_two_given_sets_of_lines_after_dualizing	$sz\ d\ d_{\min}\ s\ t\ T$	Finds arcs with the two given sets as s-lines and t-lines, respectively.
-arc_with_three_given_sets_of_lines_after_dualizing	$sz\ d\ d_{\min}\ s\ t\ T\ u\ U$	Finds arcs with the three given sets as s-lines and t-lines and u-lines, respectively.
-dualize_hyperplanes_to_points		Turns ranks of hyperplanes into ranks of points.
-dualize_points_to_hyperplanes		Turns ranks of points into ranks of hyperplanes.
-dualize_rank_k_subspaces	k	Turns ranks of k-subspaces into ranks of $n-k$ subspaces.

Table 4.8: Projective Space Activities (Part 2)

Projective Space Activities (Part 3)

Command	Arguments	Purpose
-lines_on_point_but_within_a_plane	pt-rk plane-rk	Compute the lines through a given point contained in a given plane.
-rank_lines_in_PG	M	Rank lines. The generating systems for the lines are given in rows of the matrix M.
-unrank_lines_in_PG	v	Unrank lines. The ranks are given in the vector v.
-move_two_lines_in_ hyperplane_stabilizer	$l1$ $l2$ $m1$ $m2$	Find the unique transvection fixing the hyperplane at infinity moving $l1$ and $l2$ to $m1$ and $m2$.
-move_two_lines_in_ hyperplane_stabilizer_text	$l1$ $l2$ $m1$ $m2$	Find the unique transvection fixing the hyperplane at infinity moving $l1$ and $l2$ to $m1$ and $m2$.
-planes_through_line	l	Find all planes through the line l.
-restricted_incidence_matrix	$t1$ $t2$ R C fname	Compute the restricted incidence matrix between objects R of type $t1$ and objects C of type $t2$. Write output to the given file in csv format.
-plane_intersection_type_ of_klein_image	t input	Compute the plane intersection type of the Klein image of the lines given in input. Use a threshold of t to filter planes. Only planes which intersect in at least t points on the Klein quadric are considered.
-report_Grassmannian	k	Produce a report on the Grassmannian of k-simensional subspaces.
-report_fixed_objects	data fname	Produce a report about the fixed points of the group element given by data in the action on objects of various types of the projective space. Writes a latex report using the given file name.

Table 4.9: Projective Space Activities (Part 3)

Projective Space Activities (Part 4)

Command	Arguments	Purpose
-classify_surfaces_with_double_sixes	label control	Classify cubic surfaces using the approach of double sixes. See Section 8.4.
-classify_surfaces_through_arcs_and_two_lines		Classify cubic surfaces using the approach of six-arcs and two skew lines. See Section 8.4.
-classify_surfaces_through_arcs_and_trihedral_pairs		See Section 8.4.
-test_nb_Eckardt_points	e	Restrict to e Eckardt points. See Section 8.4. To be used in conjunction with -classify_surfaces_through_arcs_and_trihedral_pairs.
-trihedra1_control	poset-control	For -classify_surfaces_through_arcs_and_trihedral_pairs
-trihedra2_control	poset-control	For -classify_surfaces_through_arcs_and_trihedral_pairs
-control_six_arcs	poset-control	For -classify_surfaces_through_arcs_and_trihedral_pairs
-six_arcs_not_on_conic		Classify six-arcs not on a conic in a plane.
-filter_by_nb_Eckardt_points	e	Filter for the number of Eckardt points to be equal to e. Used in conjunction with -six_arcs_not_on_conic.
-classify_quartic_curves_nauty	fname-mask N fname	Classify quartic curves using Nauty.
-classify_quartic_curves_with_substructure	fname-mask N k d fname	Classify quartic curves using substructure algorithm.
-classify_arcs	descr	Classify arcs.
-classify_semifields	descr	Classify semifields.
-classify_bent_functions	n	Classify bent functions in n variables.

Table 4.10: Projective Space Activities (Part 4)

Global Commands for Projective Geometry

Command	Arguments	Purpose
-create_points_on_quartic	ϵ	Create a table of points on a specific quartic curve. Consecutive points are no more than ϵ apart.
-create_points_on_parabola	ϵ a b c	Create a table of points on the parabola $y = ax^2 + bx + c$. Consecutive points are no more than ϵ apart.
-smooth_curve	ϵ N b $t_{\min}$ $t_{\max}$ function	Create at least N points on a continuous curve given by "function". Consecutive points are no more than ϵ apart. The function must be in terms of a parameter t. The values of t are taken from the interval $[t_{\min}, t_{\max}]$.
-make_table_of_surfaces		Produce a latex table summarizing the surfaces in the Orbiter catalogue.
-create_surface_reports	field-orders	Produce reports for all surfaces in the Orbiter catalogue over the give field orders.
-create_surface_atlas	$q_{\max}$	Produce reports for all surfaces in the Orbiter catalogue for field orders $q \le q_{\max}$.
-create_dickson_atlas		Produce reports of Dickson surfaces.

Table 4.11: Global Commands for Projective Geometry

9 / 15 = 23 : (1, 0, 0, 1)
10 / 15 = 26 : (0, 1, 0, 1)
11 / 15 = 27 : (1, 1, 0, 1)
12 / 15 = 38 : (0, 0, 1, 1)
13 / 15 = 39 : (1, 0, 1, 1)
14 / 15 = 42 : (0, 1, 1, 1)
There are 35 / 357 fixed subspaces of rank 2, they are:

$$0 \;/\; 35 = 0 : \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$1 \;/\; 35 = 1 : \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

$$2 \;/\; 35 = 4 : \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

$$3 \;/\; 35 = 5 : \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

$$4 \;/\; 35 = 16 : \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$5 \;/\; 35 = 17 : \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

$$6 \;/\; 35 = 20 : \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$7 \,/\, 35 = 21 : \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$8 \,/\, 35 = 22 : \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

$$9 \,/\, 35 = 25 : \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

$$10 \,/\, 35 = 26 : \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

$$11 \,/\, 35 = 37 : \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$12 \,/\, 35 = 38 : \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

$$13 \,/\, 35 = 41 : \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$14 \,/\, 35 = 84 : \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$15 \,/\, 35 = 85 : \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

$$16 \,/\, 35 = 88 : \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

$$17 \,/\, 35 = 89 : \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

$$18 \,/\, 35 = 100 : \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$19 \,/\, 35 = 101 : \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

$$20 \,/\, 35 = 104 : \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$21 \,/\, 35 = 105 : \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$22 \,/\, 35 = 106 : \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

$$23 \,/\, 35 = 109 : \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

$$24 \,/\, 35 = 110 : \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

$$25 \,/\, 35 = 121 : \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$26 \,/\, 35 = 122 : \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

$$27 \,/\, 35 = 125 : \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$28 \,/\, 35 = 336 : \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$29 \,/\, 35 = 337 : \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

$$30 \,/\, 35 = 340 : \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$31 \,/\, 35 = 341 : \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$32 \,/\, 35 = 342 : \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

$$33 \,/\, 35 = 345 : \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$34 \,/\, 35 = 356 : \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

There are 15 / 85 fixed subspaces of rank 3, they are:

$$0 \,/\, 15 = 0 : \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$1 \,/\, 15 = 1 : \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

$$2 \,/\, 15 = 4 : \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$3 \,/\, 15 = 5 : \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$4 \,/\, 15 = 6 : \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

$$5 \,/\, 15 = 9 : \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$6 \,/\, 15 = 20 : \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$7 \,/\, 15 = 21 : \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$
\begin{aligned}
8 \,/\, 15 = 22 : &\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \\
9 \,/\, 15 = 25 : &\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
10 \,/\, 15 = 26 : &\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \\
11 \,/\, 15 = 27 : &\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \\
12 \,/\, 15 = 30 : &\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
13 \,/\, 15 = 41 : &\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
14 \,/\, 15 = 84 : &\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}
$$

Suppose we want to study the fix structure of a collineation group in projective space. Suppose we want to do so for the elementary abelian group or order 4 generated by the two elements

$$
\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}_0, \quad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}_0.
$$

It fixes no points, 5 lines, and no plane. The following command sequence can be used. It uses two makefile variables: First, we set a label for the group:

Example 144

```
H1_LABEL="H1"
```

Now we define the subgroup:

Example 145

```
H1_GENS=-PGGL 4 4 \
▷  -subgroup_by_generators "H1" 4 2 \
▷  "1,0,0,0,1,1,0,0,0,0,1,0,0,0,1,1,0, \
▷  1,0,0,0,0,1,0,0,1,0,1,0,0,1,0,1,0" \
▷  -end
```

And finally the command itself:

Example 146

```
fix_structure_G1:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 4 -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷  ▷  -with P -do \
▷  ▷  -projective_space_activity \
▷  ▷  ▷  -decomposition_by_subgroup \
▷  ▷  ▷  ▷  $(H1_LABEL) $(H1_GENS)  \
▷  ▷  -end
▷  pdflatex PGGL_4_4_Subgroup_$(H1_LABEL)_4_decomp.tex
▷  $(OPEN) PGGL_4_4_Subgroup_$(H1_LABEL)_4_decomp.pdf
```

The decomposition based on orbit lengths is shown below:

Decomposition based on orbit lengths:
Row scheme:

→	3	2	48	16	16	144	128
1	3	2	0	16	0	0	0
12	1	0	4	0	4	12	0
8	0	1	0	0	4	0	16
64	0	0	3	1	0	9	8

Column scheme:

↓	3	2	48	16	16	144	128
1	1	1	0	1	0	0	0
12	4	0	1	0	3	1	0
8	0	4	0	0	2	0	1
64	0	0	4	4	0	4	4

The decomposition of orbits is shown as well, but it is too large to be shown.

Suppose we are looking for a projectivity of PG$(3, 16)$ fixing the plane $v(X_3)$ pointwise and mapping a pair of skew lines not in that plane to another pair of skew lines not in that plane. For instance, suppose we want to map

$$M_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mapsto N_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M_2 = \begin{bmatrix} 1 & 1 & 0 & \delta \\ 0 & 0 & 1 & 0 \end{bmatrix} \mapsto N_2 = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

The command

Example 147

```
trans:
▷  $(ORBITER) -v 5 \
▷  ▷  -define F -finite_field -q 16 -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷  ▷  -with P -do \
▷  ▷  -projective_space_activity \
▷  ▷  ▷  -move_two_lines_in_hyperplane_stabilizer_text \
▷  ▷  ▷  ▷  "1,0,0,0, 0,0,0,1" "1,1,0,2, 0,0,1,0" \
▷  ▷  ▷  ▷  "1,0,0,0, 0,0,0,1" "0,1,0,1, 0,0,1,0" \
▷  ▷  -end
```

computes a projectivity (transvection) to do so:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \delta^{14} & 0 & 0 & \delta^{14} \end{bmatrix}$$

Here, δ is the primitive element in the built-in field $\mathbb{F}_{16}$, satisfying $\delta^4 = \delta^3 + 1$.

4.10 Geometric Objects

Orbiter can create objects in projective space. To do so, define an object of type `-geometric_object`. The definition of a geometric object requires a projective geometry object. For this reason, the definition requires an extra argument, which is the label of a previously created projective geometry object. After that, one of the commands shown in Tables 4.12 and 4.13 can be issued.
The following command creates an elliptic quadric ovoid on PG(3, 8):

Example 148

```
elliptic_quadric_ovoid_q8:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 8 -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷  ▷  -define O -geometric_object P \
▷  ▷  ▷  -elliptic_quadric_ovoid \
▷  ▷  -end \
▷  ▷  -with O -do -combinatorial_object_activity -save \
▷  ▷  -end
```

The next command creates the Suzuki-Tits ovoid in PG(3, 8):

Example 149

```
ovoid_ST_q8:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 8 -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷  ▷  -define O -geometric_object P \
▷  ▷  ▷  -ovoid_ST \
▷  ▷  -end \
▷  ▷  -with O -do -combinatorial_object_activity -save \
▷  ▷  -end
```

The following command creates the Baer subplane of PG(2, 4):

Example 150

```
Baer_PG_2_4:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 4 -end \
▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
▷  ▷  -define O -geometric_object P \
▷  ▷  ▷  -Baer_substructure \
▷  ▷  -end \
▷  ▷  -with O -do -combinatorial_object_activity -save \
▷  ▷  -end
```

The next command creates the Baer subgeometry of PG(3, 4):

Geometric Objects (Part 1)

Command	Arguments	Purpose
-hyperoval		To create a hyperoval
-subiaco_oval	f_short	Create the Subiaco oval
-subiaco_hyperoval		Create the Subiaco hyperoval
-adelaide_hyperoval		Create the Adalaide hyperoval
-translation	i	Create the translation hyperoval with exponent i
-Segre		Create the Segre hyperoval
-Payne		Create the Payne hyperoval
-Cherowitzo		Create the Cherowitzo hyperoval
-OKeefe_Penttila		Create the O'Keefe, Penttila hyperoval
-BLT_database	k	Create the kth BLT-set of order q from the database ($k = 0, 1, \ldots$)
-elliptic_quadric_ovoid		Create an elliptic quadric ovoid in $\mathrm{PG}(3, q)$.
-ovoid_ST		Create the Suzuki Tits ovoid in $\mathrm{PG}(3, q)$. Here, $q = 2^{2r+1}$.
-Baer		Create the (standard) Baer subgeometry
-orthogonal	ϵ	Create the $Q^{\epsilon}(n, q)$ quadric
-hermitian		Create the Hermitian variety given by $\sum_{i=0}^{n} X_i^{\sqrt{q}+1} = 0$
-cuspidal_cubic		Create the cuspidal cubic (s^3, ts^2, t^3) in $\mathrm{PG}(2, q)$
-twisted_cubic		Create a twisted cubic (s^3, s^2t, st^2, t^3) in $\mathrm{PG}(3, q)$
-elliptic_curve	a b	Create the elliptic curve $y^2 = x^3 + ax + b$
-ttp_construction_A		Create the twisted tensor product code of type A [8]
-ttp_construction_A_hyperoval		Create the twisted tensor product code of type A [8]
-ttp_construction_B		Create the twisted tensor product code of type B [8]

Table 4.12: Geometric Objects (Part 1)

Geometric Objects (Part 2)

Command	Arguments	Purpose
-unital_XXq_YZq_ZYq		Create the unital with equation $XX^q + YZ^q + ZY^q = 0$
-desarguesian_line_spread_in_PG_3_q		Create the desarguesian line spread in $\mathrm{PG}(3, q)$ as a set of 2-subspaces
-Buekenhout_Metz		Create the Buekenhout Metz unital
-Uab	a b	Create the Buekenhout Metz unital in the form of Barwick and Ebert [6]
-whole_space		Create the whole space
-hyperplane	pt	Create the hyperplane given by dual coordinates associated with the given point
-segre_variety	a b	Create the Segre variety
-Maruta_Hamada_arc		Create the Maruta Hamada arc
-projective_variety	lab_ascii lab_tex d coeffs	Create a projective variety of degree d from an equation. By default, the coefficients of the equation are listed in the partition ordering. A different ordering can be specified. A label for the variety in ascii and in tex is required. See Section 4.5.
-intersection_of_zariski_open_sets	l d n $\mathcal{C}_1$ ... $\mathcal{C}_n$	Create the intersection of the Zariski open sets given by equations $\mathcal{C}_1, \ldots \mathcal{C}_n$ of degree d with label l, see Section 4.5.
-projective_curve	l r d $\mathcal{C}$	Create the projective curve of degree d with label l, with coefficient vector $\mathcal{C}$ in r variables

Table 4.13: Geometric Objects (Part 2)

Example 151

```
Baer_PG_3_4:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 4 -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷  ▷  -define O -geometric_object P \
▷  ▷  ▷  -Baer_substructure \
▷  ▷  -end \
▷  ▷  -with O -do -combinatorial_object_activity -save \
▷  ▷  -end
```

Let us consider BLT-sets. The following command recalls the unique BLT-set of order 5 from the database:

Example 152

```
BLT_database_5_0:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 5 -end \
▷  ▷  -define P -projective_space -n 4 -field F -v 0 -end \
▷  ▷  -define O -geometric_object P \
▷  ▷  ▷  -BLT_database 0 \
▷  ▷  -end \
▷  ▷  -with O -do -combinatorial_object_activity -save \
▷  ▷  -end
```

Let us pull the first BLT-set of order 7 and print it to latex:

Example 153

```
BLT_database_7_0_print:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 7 -end \
▷  ▷  -define P -projective_space -n 4 -field F -v 0 -end \
▷  ▷  -define Obj -geometric_object P \
▷  ▷  ▷  -BLT_database 0 \
▷  ▷  -end \
▷  ▷  -with Obj -do -combinatorial_object_activity -save \
▷  ▷  -end \
▷  ▷  -define O -orthogonal_space 0 5 F -without_group -end \
▷  ▷  -define BLT_7_0 -set -file_orbiter_format BLT_7_0.txt -end \
▷  ▷  -with O -do -orthogonal_space_activity \
▷  ▷  ▷  -print_points BLT_7_0 -end
▷  pdflatex BLT_7_0_set_report.tex
▷  $(OPEN) BLT_7_0_set_report.pdf
```

The output is shown below:

A set of points of size 8
The Points:
$0 : P_0 = (0, 1, 0, 0, 0)$
$1 : P_1 = (0, 0, 1, 0, 0)$
$2 : P_{40} = (0, 1, 2, 6, 2)$
$3 : P_{41} = (0, 1, 4, 3, 1)$
$4 : P_{42} = (0, 1, 1, 2, 3)$
$5 : P_{43} = (0, 1, 1, 5, 4)$
$6 : P_{44} = (0, 1, 4, 4, 6)$
$7 : P_{45} = (0, 1, 2, 1, 5)$

In Section 4.7, we have created the generalized quadrangle $W(2)$ (the Doily). We can search for disjoint sets of size 5 in the collineation graph:

Example 154

```
Doily_disjoint_sets_graph_cliques_5:
▷  echo $(DOILY) >doily.csv
▷  $(ORBITER) -v 2 \
▷  ▷  -define G -graph -disjoint_sets_graph \
▷  ▷  ▷  doily.csv \
▷  ▷  -end \
▷  ▷  -with G -do \
▷  ▷  ▷  -graph_theoretic_activity \
▷  ▷  ▷  -find_cliques \
▷  ▷  ▷  ▷  -target_size 5 \
▷  ▷  ▷  ▷  -output_file doily_cliques_5 \
▷  ▷  ▷  -end \
▷  ▷  -end \
▷  ▷  -print_symbols
```

There are exactly 6 cliques.

The Edge curve is given by the equation

$$X^4 - Y^4 - Z^4 + 2f^2Y^2Z^2 + 4fX^2YZ = 0$$

where f is a primitive element of $\mathbb{F}_q$. Let us pick $q = 17$. The next example creates the Edge curve in $\mathrm{PG}(2, 17)$ and saves it to file. The equation is encoded using the ordering of quartic monomials from Table 5.4.

Example 155

```
EDGE_CURVE_Q17_EQUATION="1,16,16,0,0,0,0,0,0,0,0,1,12,0,0"
```

Example 156

```
EDGE_CURVE_Q17_AS_POINTS="4, 7, 16, 19, 20, 23, 32, 35, 89, 100, 244, 251"
```

Example 157

```
FILE_Q17="orbit,curve,pts_on_curve,bitangents,go\
\n0,\"$(EDGE_CURVE_Q17_EQUATION)\",\"$(EDGE_CURVE_Q17_AS_POINTS)\",\"\",-1\
\nEND"
```

Example 158

```
Edge_curve_17:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 17 -end \
▷  ▷  -define R -polynomial_ring -field F \
▷  ▷  ▷  -number_of_variables 3 \
▷  ▷  ▷  -homogeneous_of_degree 4 \
▷  ▷  ▷  -end \
▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
▷  ▷  -define C -geometric_object P \
▷  ▷  ▷  -projective_variety R \
▷  ▷  ▷  ▷  "Edge_q17" "Edge\_q17" \
▷  ▷  ▷  ▷  $(EDGE_CURVE_Q17_EQUATION) \
▷  ▷  -end \
▷  ▷  -with C -do -combinatorial_object_activity -save \
▷  ▷  -end
```

The following command computes the line type of the Edge curve:

Example 159

```
Edge_curve_17_line_type:
▷  echo $(FILE_Q17) >edge_q17.csv
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 17 -end \
▷  ▷  -define R -polynomial_ring -field F \
▷  ▷  ▷  -number_of_variables 3 \
▷  ▷  ▷  -homogeneous_of_degree 4 \
▷  ▷  ▷  -end \
▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
▷  ▷  -define C -geometric_object P \
▷  ▷  ▷  -projective_variety R \
▷  ▷  ▷  ▷  "Edge_q17" "Edge\_q17" \
▷  ▷  ▷  ▷  $(EDGE_CURVE_Q17_EQUATION) \
▷  ▷  ▷  -end \
▷  ▷  -with C -do \
▷  ▷  -combinatorial_object_activity \
▷  ▷  ▷  -line_type_old \
▷  ▷  -end \
▷  ▷  -print_symbols
```

The line type is

$$(4^6, 2^{30}, 1^{132}, 0^{139})$$

This means that there are 6 4-secants, 30 2-secants, 132 tangent lines, and 139 external lines to the curve.

Chapter 5

Ring Theory

5.1 Polynomials Over Finite Fields

For p prime, the finite field $\mathbb{F}_p$ of order p can be constructed as factorring of the integers moulo p. In this section, we will consider polynomials over $\mathbb{F}_p$. The ring of polynomials in one variable with coefficients in $\mathbb{F}_p$ is denoted as $\mathbb{F}_p[X]$. Table 5.1 lists Orbiter finite field activities related to polynomials. For instance, the command

Example 160

```
poly_division:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -with F -do \
▷  ▷  -finite_field_activity \
▷  ▷  -polynomial_division "1,0,0,0,0,0,0,0,0,0,1" "1,0,1,1" -end
```

computes the polynomial long division of $A(X)$ by $B(X)$ over $\mathbb{F}_2$ where

$$A(X) = X^{10} + 1, \quad B(X) = X^3 + X^2 + 1.$$

The result is $Q(X)$ and $R(X)$ with

$$A(X) = Q(X) \cdot B(X) + R(X)$$

with

$$Q(X) = X^7 + X^6 + X^5 + X^3 + 1, \quad R(X) = X^2.$$

The coefficient lists in the arguments are from the lowest term up.

It is perhaps more convenient to create the polynomials as vectors, as in Section 2.7. The following example uses vectors named A and B. After that, the division command is called.

Example 161

```
poly_division2:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define A -vector -field F -sparse 11 "1,0,1,10" -end \
▷  ▷  -define B -vector -field F -dense "1,0,1,1" -end \
```

Finite Field Activities Related to Polynomials

Command	Arguments	Purpose
-polynomial_division	$A(X)$ $B(X)$	Polynomial division of $A(X)$ by $B(X)$ over $\mathbb{F}_q$. $A(X)$ and $B(X)$ are given as coefficient list, starting from the lowest coefficient.
-extended_gcd_for_polynomials	$A(X)$ $B(X)$	Extended gcd for polynomials $A(X)$ and $B(X)$ over $\mathbb{F}_q$. $A(X)$ and $B(X)$ are given as coefficient list, starting from the lowest coefficient.
-polynomial_mult_mod	$A(X)$ $B(X)$ $M(X)$	Multiply the polynomials $A(X)$ and $B(X)$ modulo $M(X)$ in $\mathbb{F}_q[X]$.
-Berlekamp_matrix	$A(X)$	Computes the rank of the Berlekamp matrix associated to the polynomial $A(X)$ over $\mathbb{F}_q$. The polynomial $A(X)$ is irreducible over $\mathbb{F}_q$ if the Berlekamp matrix has rank $d-1$, where d is the degree of $A(X)$. The Berlekamp matrix is $F-I$ where F is the Frobenius matrix and I is the identity matrix. The Frobenius matrix is the matrix of the Frobenius endomorphism with respect to the standard basis of the polynomial ring: $1, X, X^2, \ldots, X^{d-1}$.
-polynomial_find_roots	$A(X)$	Find the roots of $A(X)$ over $\mathbb{F}_q$.
-get_primitive_polynomial	d	Get a primitive polynomial of degree d over $\mathbb{F}_q$.
-get_primitive_polynomial_in_range	$d_{\min}$ $d_{\max}$	Get a primitive polynomial of degree d over $\mathbb{F}_q$ for all $d \in [d_{\min}, d_{\max}]$.
-make_table_of_irreducible_polynomials	d	Produces a list of all irreducible polynomials of degree d over $\mathbb{F}_q$.

Table 5.1: Finite Field Activities Related to Polynomials

```
▷  ▷  -with F -do \
▷  ▷  -finite_field_activity \
▷  ▷  -polynomial_division A B -end
```

The command `-extended_gcd_for_polynomials` takes two polynomials $A(X)$ and $B(X)$ and computes polynomials $U(X)$ and $V(X)$ and $G(X)$ such that $G(X)$ is the greatest common divisor of $A(X)$ and $B(X)$ and

$$G(X) = U(X) \cdot A(X) + V(X) \cdot B(X).$$

For instance,

Example 162

```
poly_gcd:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -with F -do \
▷  ▷  -finite_field_activity \
▷  ▷  -extended_gcd_for_polynomials "1,0,0,0,0,0,0,0,0,0,1" "1,0,1,1" -end
```

computes

$$U(X) = X + 1, \quad V(X) = X^8 + X^5 + X^4 + X^3 + X, \quad G(X) = 1.$$

The next command computes

$$(3X^2 + 2X + 1) \cdot (5X^2 + 4X + 3) \mod (X^3 + 6) \mod 7.$$

Example 163

```
poly_mult_mod1:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 7 -end \
▷  ▷  -with F -do \
▷  ▷  -finite_field_activity \
▷  ▷  -polynomial_mult_mod "1,2,3" "3,4,5" "6,0,0,1" -end
```

which has a result of

$$X^2 + 4X + 4.$$

The coefficients are given from the lowest to the highest term. For the opposite order, the following command computes

$$(2X^2 + X + 3) \cdot (4X^2 + 3X + 5) \mod (X^3 + 6) \mod 7.$$

Example 164

```
poly_mult_mod2:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 7 -end \
▷  ▷  -with F -do \
```

```
▷  ▷  -finite_field_activity \
▷  ▷  -polynomial_mult_mod "3,1,2" "5,3,4" "6,0,0,1" -end
```

The result is

$$4X^2 + X + 4.$$

The finite field $\mathbb{F}_4$ can be defined by using polynomial arithmetic over $\mathbb{F}_2$ modulo $X^2 + X + 1$. Here is a command that computes the three non-trivial products of polynomials:

Example 165

```
poly_mult_mod_F4:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -with F -do \
▷  ▷  -finite_field_activity \
▷  ▷  -polynomial_mult_mod "1,1" "1,1" "1,1,1" -end
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -with F -do \
▷  ▷  -finite_field_activity \
▷  ▷  -polynomial_mult_mod "0,1" "1,1" "1,1,1" -end
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -with F -do \
▷  ▷  -finite_field_activity \
▷  ▷  -polynomial_mult_mod "0,1" "0,1" "1,1,1" -end
```

It is possible to use numerical values for polynomials, using the representation in radix q. The following command computes the product of the polynomials 5 and 7 over $\mathbb{F}_2$:

Example 166

```
mult_polynomials_2_5_7:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -with F -do \
▷  ▷  -finite_field_activity -mult_polynomials 5 7 -end
▷  pdflatex polynomial_mult_5_7.tex
▷  $(OPEN) polynomial_mult_5_7.pdf
```

The next command performs polynomial long division based on numerical polynomials:

Example 167

```
polynomial_division_ranked_2_27_13:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
```

```
▷  ▷  -with F -do \
▷  ▷  -finite_field_activity \
▷  ▷  ▷  -polynomial_division_ranked 27 13 \
▷  ▷  -end
▷  pdflatex polynomial_division_27_13.tex
▷  $(OPEN) polynomial_division_27_13.pdf
```

Here is a somewhat larger example for numerical arguments. We wish to multiply 999 by 997 modulo 1033. The first command performs multiplication:

Example 168

```
mult_polynomials_1024_999_997:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -with F -do \
▷  ▷  -finite_field_activity \
▷  ▷  ▷  -mult_polynomials 999 997 \
▷  ▷  -end
▷  pdflatex polynomial_mult_999_997.tex
▷  $(OPEN) polynomial_mult_999_997.pdf
```

The next command performs division with remainder:

Example 169

```
polynomial_division_ranked_2_349147_1033:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -with F -do \
▷  ▷  -finite_field_activity \
▷  ▷  -polynomial_division_ranked 349147 1033 \
▷  ▷  -end
▷  pdflatex polynomial_division_349147_1033.tex
▷  $(OPEN) polynomial_division_349147_1033.pdf
```

Orbiter allows polynomial arithmetic modulo a factor polynomial. The coefficient vector of the polynomial can be created as a vector. Here is an example which performs arithmetic modulo the CRC32 polynomial. The goal is to compute the multiplicative inverse of X. In order to do so, we use the fact that the CRC32 polynomial is irreducible, and hence the factor ring is a finite field of order 2^{32}. The inverse of a polynomial can be computed by raising to the power of $2^{32} - 2$:

Example 170

```
CRC32_SPARSE="1,32,1,26,1,23,1,22,1,16,1,12,1,11,\
1,10,1,8,1,7,1,5,1,4,1,2,1,1,1,0"
```

Example 171

```
TWO_TO_THE_32_MINUS_2=4294967294
```

Example 172

```
power_mod_inverse:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define M -vector -field F -sparse 33 $(CRC32_SPARSE) -end \
▷  ▷  -define A -vector -field F -sparse 2 "1,1" -end \
▷  ▷  -with F -do \
▷  ▷  -finite_field_activity \
▷  ▷  -polynomial_power_mod A $(TWO_TO_THE_32_MINUS_2) M \
▷  ▷  -end
```

This command produces the polynomial

$$B(X) = X^{31} + X^{25} + X^{22} + X^{21} + X^{15} + X^{11} + X^{10} + X^9 + X^7 + X^6 + X^4 + X^3 + X + 1$$

In order to test that this polynomial really is the multiplicative inverse of X modulo CRC32, we perform the following command:

Example 173

```
INVERSE_SPARSE="1,31,1,25,1,22,1,21,1,15,\
1,11,1,10,1,9,1,7,1,6,1,4,1,3,1,1,1,0"
```

Example 174

```
mult_mod_to_get_one:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define M -vector -field F -sparse 33 $(CRC32_SPARSE) -end \
▷  ▷  -define A -vector -field F -sparse 2 "1,1" -end \
▷  ▷  -define B -vector -field F -sparse 33 $(INVERSE_SPARSE) -end \
▷  ▷  -with F -do \
▷  ▷  -finite_field_activity \
▷  ▷  ▷  -polynomial_mult_mod A B M \
▷  ▷  -end
```

The product is indeed 1.

The Berlekamp matrix can be used to test if a polynomial is irreducible over a given finite field. The polynomial is irreducible if and only if the rank of the Berlekamp matrix is $d - 1$, where d is the degree of the polynomial. For instance, the command

Example 175

```
Berlekamp_matrix_2_3:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define v -vector -field F -dense "1,1,0,1" -end \
▷  ▷  -with F -do \
▷  ▷  -finite_field_activity \
▷  ▷  -Berlekamp_matrix v -end
▷  pdflatex Berlekamp_matrix_q2_d3.tex
▷  $(OPEN) Berlekamp_matrix_q2_d3.pdf
```

computes the Berlekamp matrix associated with the polynomial $X^3 + X + 1$ over $\mathbb{F}_2$. The matrix is

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}.$$

Since the matrix has rank 2, the polynomial is irreducible.

Orbiter can compute irreducible and primitive polynomials. The following command creates a list of all irreducible polynomials of degree 3 over $\mathbb{F}_4$:

Example 176

```
irred_3_4:
▷  $(ORBITER) -v 6 \
▷  ▷  -define F -finite_field -q 4 -end \
▷  ▷  -with F -do \
▷  ▷  -finite_field_activity \
▷  ▷  -make_table_of_irreducible_polynomials 3 -end
▷  pdflatex Irred_q4_d3.tex
▷  $(OPEN) Irred_q4_d3.pdf
```

The output is:

```
There are 20 irreducible polynomials of degree 3 over the field F4:
0 : 1123 : 91
1 : 1031 : 77
2 : 1213 : 103
3 : 1323 : 123
4 : 1322 : 122
5 : 1222 : 106
6 : 1021 : 73
7 : 1101 : 81
8 : 1333 : 127
9 : 1232 : 110
10 : 1113 : 87
11 : 1233 : 111
12 : 1301 : 113
13 : 1003 : 67
```

```
14 : 1112 : 86
15 : 1002 : 66
16 : 1312 : 118
17 : 1011 : 69
18 : 1132 : 94
19 : 1201 : 97
```

The following command creates a primitive polynomial of degre 10 over $\mathbb{F}_2$:

Example 177

```
F2_get_primitive_poly_of_deg_10:
▷  $(ORBITER) -v 6 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -with F -do \
▷  ▷  -finite_field_activity \
▷  ▷  -get_primitive_polynomial 10 -end
```

The following command creates one primitive polynomial over $\mathbb{F}_2$ for each degree in the range $[2, 10]$:

Example 178

```
F2_get_primitive_poly_range_2_10:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -with F -do \
▷  ▷  -finite_field_activity \
▷  ▷  -get_primitive_polynomial_in_range 2 10 -end
```

The unix commnd `grep` is used to filter the output for lines containing the given pattern "//". This yields the list

```
"7", // X^{2} + X + 1
"13", // X^{3} + X^{2} + 1
"25", // X^{4} + X^{3} + 1
"37", // X^{5} + X^{2} + 1
"97", // X^{6} + X^{5} + 1
"193", // X^{7} + X^{6} + 1
"285", // X^{8} + X^{4} + X^{3} + X^{2} + 1
"529", // X^{9} + X^{4} + 1
"1033", // X^{10} + X^{3} + 1
```

Primitive polynomials over the base field $\mathbb{F}_s$ are converted into integers, using the base-s representation of integers. For instance, the polynomial $X^2 + X + 1$ is read as binary string 111, which in turn translates to the integer 7 (we use $s = 2$).

Creating Multivariate Polynomial Rings		
Command	Arguments	Purpose
-field	label	Specify the field of coefficients.
-homogeneous_of_degree	d	Specify the degree d of polynomials.
-number_of_variables	n	Specify the number n of variables.
-monomial_ordering_partition		Set monomial ordering to partition ordering.
-monomial_ordering_lex		Set monomial ordering to lexicographic ordering.
-variables	label-txt label-tex	Specify variable labels in ascii and in latex.

Table 5.2: Creating Multivariate Polynomial Rings

Multivariate Polynomial Ring Activities		
Command	Arguments	Purpose
-cheat_sheet		Create a report about the polynomial ring.
-ideal	label_txt label_tex label_set	Compute the ideal of the set of points.
-apply_transformation	eqn elt	Apply the transformation given by the group element to the given equation.
-set_variable_names	txt tex	Choose new variable names in text and tex format.
-print_equation	equation	Print the given equation.

Table 5.3: Multivariate Polynomial Ring Activities

5.2 Multivariate Polynomial Rings

Orbiter can work with multivariate polynomial rings. Table 5.2 lists the commands for creating a multivariate polynomial ring.
Table 5.3 lists the activities for a multivariate polynomial ring.

There are two orderings of the monomials which can be chosen:

1. The partition ordering is grouping terms according to the partition that results from the degrees of the variables first, and then applies the lexicographic ordering as a tie breaker.

2. The lexicographic ordering orders the monomials lexicographically.

By default, the partition ordering is used. Table 5.4 shows the monomials in the partition ordering for degrees 1, 2, 3 and 4 in a plane.

Table 5.5 shows the partition ordering monomials of degree at most 3 in $\mathrm{PG}(3, q)$.

The following example shows how a Cremona map can be defined. At first, we define 4 polynomials as makefile variables. After that, we invoke Orbiter to create a polynomial ring and to evaluate the map.

Monomial Ordering of Partition Type in the Plane

Degrees 1 and 2

h	mon	vector
0	X_0	$(1,0,0)$
1	X_1	$(0,1,0)$
2	X_2	$(0,0,1)$

h	mon	vector
0	X_0^2	$(2,0,0)$
1	X_1^2	$(0,2,0)$
2	X_2^2	$(0,0,2)$
3	X_0X_1	$(1,1,0)$
4	X_0X_2	$(1,0,1)$
5	X_1X_2	$(0,1,1)$

Degree 3

h	mon	vector
0	X_0^3	$(3,0,0)$
1	X_1^3	$(0,3,0)$
2	X_2^3	$(0,0,3)$
3	$X_0^2X_1$	$(2,1,0)$
4	$X_0^2X_2$	$(2,0,1)$
5	$X_0X_1^2$	$(1,2,0)$
6	$X_1^2X_2$	$(0,2,1)$
7	$X_0X_2^2$	$(1,0,2)$
8	$X_1X_2^2$	$(0,1,2)$
9	$X_0X_1X_2$	$(1,1,1)$

Degree 4

h	mon	vector
0	X_0^4	$(4,0,0)$
1	X_1^4	$(0,4,0)$
2	X_2^4	$(0,0,4)$
3	$X_0^3X_1$	$(3,1,0)$
4	$X_0^3X_2$	$(3,0,1)$
5	$X_0X_1^3$	$(1,3,0)$
6	$X_1^3X_2$	$(0,3,1)$
7	$X_0X_2^3$	$(1,0,3)$
8	$X_1X_2^3$	$(0,1,3)$
9	$X_0^2X_1^2$	$(2,2,0)$
10	$X_0^2X_2^2$	$(2,0,2)$
11	$X_1^2X_2^2$	$(0,2,2)$
12	$X_0^2X_1X_2$	$(2,1,1)$
13	$X_0X_1^2X_2$	$(1,2,1)$
14	$X_0X_1X_2^2$	$(1,1,2)$

Table 5.4: Monomial Ordering of Partition Type in the Plane

Monomial Ordering of Partition Type in Three-Space

Degree 1

h	mon	vector
0	X_0	$(1,0,0,0)$
1	X_1	$(0,1,0,0)$
2	X_2	$(0,0,1,0)$
3	X_3	$(0,0,0,1)$

Degree 2

h	mon	vector
0	X_0^2	$(2,0,0,0)$
1	X_1^2	$(0,2,0,0)$
2	X_2^2	$(0,0,2,0)$
3	X_3^2	$(0,0,0,2)$
4	X_0X_1	$(1,1,0,0)$
5	X_0X_2	$(1,0,1,0)$
6	X_0X_3	$(1,0,0,1)$
7	X_1X_2	$(0,1,1,0)$
8	X_1X_3	$(0,1,0,1)$
9	X_2X_3	$(0,0,1,1)$

Degree 3

h	mon	vector
0	X_0^3	$(3,0,0,0)$
1	X_1^3	$(0,3,0,0)$
2	X_2^3	$(0,0,3,0)$
3	X_3^3	$(0,0,0,3)$
4	$X_0^2X_1$	$(2,1,0,0)$
5	$X_0^2X_2$	$(2,0,1,0)$
6	$X_0^2X_3$	$(2,0,0,1)$
7	$X_0X_1^2$	$(1,2,0,0)$
8	$X_1^2X_2$	$(0,2,1,0)$
9	$X_1^2X_3$	$(0,2,0,1)$
10	$X_0X_2^2$	$(1,0,2,0)$
11	$X_1X_2^2$	$(0,1,2,0)$
12	$X_2^2X_3$	$(0,0,2,1)$
13	$X_0X_3^2$	$(1,0,0,2)$
14	$X_1X_3^2$	$(0,1,0,2)$
15	$X_2X_3^2$	$(0,0,1,2)$
16	$X_0X_1X_2$	$(1,1,1,0)$
17	$X_0X_1X_3$	$(1,1,0,1)$
18	$X_0X_2X_3$	$(1,0,1,1)$
19	$X_1X_2X_3$	$(0,1,1,1)$

Table 5.5: Monomial Ordering of Partition Type in Three-Space

Example 179

```
CREMONA_MAP_Y0="3*y0^5*y2+4*y0^3*y1^2*y2\
+2*y0^3*y2^3+y0*y1^4*y2\
+6*y0*y1^2*y2^3+9*y0*y2^5"
```

Example 180

```
CREMONA_MAP_Y1="y0^5*y1+5*y0^3*y1^3\
+12*y0^3*y1*y2^2+3*y0*y1^5\
+5*y0*y1^3*y2^2+y0*y1*y2^4"
```

Example 181

```
CREMONA_MAP_Y2="10*y0^6+11*y0^4*y1^2\
+11*y0^4*y2^2+4*y0^2*y1^4\
+9*y0^2*y1^2*y2^2+4*y0^2*y2^4"
```

Example 182

```
CREMONA_MAP_Y3="0"
```

Example 183

```
Cremona_map:
▷  $(ORBITER) -v 6 \
▷  ▷  -define F -finite_field -q 13 -end \
▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
▷  ▷  -define R -polynomial_ring \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -number_of_variables 3 \
▷  ▷  ▷  -homogeneous_of_degree 6 \
▷  ▷  ▷  -monomial_ordering_lex \
▷  ▷  ▷  -variables "y0,y1,y2" "y_0,y_1,y_2" \
▷  ▷  ▷  -end \
▷  ▷  -define Y0 -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -text $(CREMONA_MAP_Y0) \
▷  ▷  ▷  -end \
▷  ▷  -define Y1 -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -text $(CREMONA_MAP_Y1) \
▷  ▷  ▷  -end \
▷  ▷  -define Y2 -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -text $(CREMONA_MAP_Y2) \
▷  ▷  ▷  -end \
```

```
▷  ▷  -define Cremona -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -text "Y0,Y1,Y2" \
▷  ▷  ▷  -end \
▷  ▷  -with P -do \
▷  ▷  -projective_space_activity \
▷  ▷  ▷  -map R Cremona "" \
▷  ▷  -end
```

Next, we will consider ideals. As an application, we will classify arcs in a projective plane and see which conics we get. The next command classifies the $(5, 2)$-arcs in $\mathrm{PG}(2, 11)$:

Example 184

```
arcs_5_2_q11:
▷  $(ORBITER) -v 4 \
▷  ▷  -define F -finite_field -q 11 -end \
▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
▷  ▷  -define Control -poset_classification_control \
▷  ▷  ▷  -problem_label arcs_5_2_q11 \
▷  ▷  ▷  -W -depth 5 \
▷  ▷  -end \
▷  ▷  -with P -do \
▷  ▷  -projective_space_activity \
▷  ▷  ▷  -classify_arcs \
▷  ▷  ▷  ▷  -control Control \
▷  ▷  ▷  ▷  -target_size 5 \
▷  ▷  ▷  ▷  -d 2 \
▷  ▷  ▷  -end \
▷  ▷  -end
▷  #pdflatex arcs_5_2_q11_poset.tex
▷  #$(OPEN) arcs_5_2_q11_poset.pdf
```

It finds exactly two isomorphism types of arcs. The representative sets are

$$\{0, 1, 2, 3, 37\}, \quad \{0, 1, 2, 3, 49\}.$$

They are stored in the file `arcs_5_2_q11_lvl_5`. Let us now create the ideal in the quadratic component of the polynomial ring in three variables over $\mathbb{F}_{11}$:

Example 185

```
arcs_5_2_q11_ideal:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 11 -end \
▷  ▷  -define R -polynomial_ring \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -number_of_variables 3 \
▷  ▷  ▷  -homogeneous_of_degree 2 \
▷  ▷  ▷  -monomial_ordering_lex \
```

```
▷ ▷ ▷ -variables "x0,x1,x2" "x_0,x_1,x_2" \
▷ ▷ ▷ -end \
▷ ▷ -define C -combinatorial_object \
▷ ▷ ▷ -label arcs_5_2_q11_lvl_5 arcs\_5\_2\_q11\_lvl\_5 \
▷ ▷ ▷ -file_of_points arcs_5_2_q11_lvl_5 \
▷ ▷ -end \
▷ ▷ -with C -do \
▷ ▷ -combinatorial_object_activity \
▷ ▷ ▷ -ideal R \
▷ ▷ -end
```

The ideals are generated by

$$7*x0*x1 + 5*x0*x2 + 10*x1*x2$$

and

$$4*x0*x1 + 8*x0*x2 + 10*x1*x2,$$

respectively.

Let us consider a smooth cubic surface with 9 lines and 4 Eckardt points. Suppose we have the set of points and we wish to determine the equation of the object. To do so, we first define the object from the given set of points.

Example 186

```
PTS_OF_SURFACE_ORBIT211_Q3_L9_E4="\
0,1,2,5,7,8,10,14,9,12, \
15,3,16,37,31,34,20,19,17,32,36,33"
```

Then, we create a ring and compute the ideal:

Example 187

```
surface_9lines_4E_ideal:
▷ $(ORBITER) -v 2 \
▷ ▷ -define Pts -vector -dense \
▷ ▷ ▷ $(PTS_OF_SURFACE_ORBIT211_Q3_L9_E4) \
▷ ▷ ▷ -end \
▷ ▷ -define F -finite_field -q 3 -end \
▷ ▷ -define R -polynomial_ring \
▷ ▷ ▷ -field F \
▷ ▷ ▷ -number_of_variables 4 \
▷ ▷ ▷ -homogeneous_of_degree 3 \
▷ ▷ ▷ -monomial_ordering_lex \
▷ ▷ ▷ -variables "x0,x1,x2,x3" "x_0,x_1,x_2,x_3" \
▷ ▷ ▷ -end \
▷ ▷ -with R -do \
▷ ▷ ▷ -ring_theoretic_activity \
▷ ▷ ▷ -ideal "surf_eqn" "surf\_eqn" Pts \
▷ ▷ ▷ -end
```

We find a two-dimensional ideal. Generators are:

x0*x0*x1 + 2*x0*x1*x1 + 2*x0*x1*x3 and 2*x2*x2*x3 + 2*x2*x3*x3.

Let us take the sum of the two polynomials and create the cubic surface:

Example 188

```
SURFACE_F_9="x0*x0*x1 - x0*x1*x1 -x0*x1*x3 -x2*x2*x3 - x2*x3*x3"
```

Example 189

```
F_9_q7:
▷  $(ORBITER) -v 10 \
▷  ▷  -define F -finite_field -q 7 -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷  ▷  -define R -polynomial_ring \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -number_of_variables 4 \
▷  ▷  ▷  -homogeneous_of_degree 3 \
▷  ▷  ▷  -monomial_ordering_partition \
▷  ▷  ▷  -variables "x0,x1,x2,x3" "x_0,x_1,x_2,x_3" \
▷  ▷  ▷  -end \
▷  ▷  -define F_9 -cubic_surface -space P \
▷  ▷  ▷  -by_equation R "F_9" \
▷  ▷  ▷  "\DF_9\D" "x0,x1,x2,x3" \
▷  ▷  ▷  $(SURFACE_F_9) \
▷  ▷  ▷  "" \
▷  ▷  ▷  "\Dno parameters\D" "" \
▷  ▷  -end \
▷  ▷  -with F_9 -do \
▷  ▷  -cubic_surface_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex surface_equation_F_9_q7_report.tex
▷  $(OPEN) surface_equation_F_9_q7_report.pdf
```

In the next example, we wish to explore the relationship between conics and $(5, 2)$-arcs. We consider the plane $\mathrm{PG}(2, 11)$. Instead of classification, we will try random generation this time. Since there are 133 points, we create a number of 5-subsets of a set of size 133. In this case, we create 20 sets at random:

Example 190

```
random_k_subsets_PG_2_11:
▷  $(ORBITER) -v 4 \
▷  ▷  -create_random_k_subsets 133 5 20
```

The sets are stored in the file `random_k_subsets_n133_k5_nb20.csv`. Now, let's compute the line type of these subsets, to see which ones are arcs:

Example 191

```
line_type_in_PG_2_11:
▷  $(ORBITER) -v 3 \
▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
▷  ▷  -define F -finite_field -q 11 -end \
▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
▷  ▷  -define C -combinatorial_object \
▷  ▷  ▷  -label random_sets random_sets \
▷  ▷  ▷  -file_of_points random_k_subsets_n133_k5_nb20.csv \
▷  ▷  -end \
▷  ▷  -with C -do \
▷  ▷  -combinatorial_object_activity \
▷  ▷  ▷  line_type_old \
▷  ▷  -end
```

It turns out that the second set is an arc. It is the set $\{3, 33, 40, 83, 102\}$. We create the conic through these 5 points:

Example 192

```
random_arc_5_2_q11_ideal:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 11 -end \
▷  ▷  -define R -polynomial_ring \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -number_of_variables 3 \
▷  ▷  ▷  -homogeneous_of_degree 2 \
▷  ▷  ▷  -monomial_ordering_lex \
▷  ▷  ▷  -variables "x0,x1,x2" "x_0,x_1,x_2" \
▷  ▷  ▷  -end \
▷  ▷  -define C -combinatorial_object \
▷  ▷  ▷  -label random_arc random\_arc \
▷  ▷  ▷  -set_of_points "3,33,40,83,102" \
▷  ▷  -end \
▷  ▷  -with C -do \
▷  ▷  -combinatorial_object_activity \
▷  ▷  ▷  -ideal R \
▷  ▷  -end
```

The ideal is generated by

```
10*x0*x0 + 3*x0*x1 + 8*x0*x2 + 2*x1*x1 + 10*x2*x2.
```

The conic contains the following 12 points:

$$\{3, 15, 19, 33, 40, 42, 46, 50, 83, 88, 102, 108\}.$$

The next command creates the Endrass surface over $\mathbb{F}_7$. The surface is defined as a makefile variable in sparse form.

Example 193

```
ENDRASS_SPARSE="\
6,0,4,4,2,7,5,9,6,20,6,23,1,25,3,30,1,32,3,34,4,56,6,59,1,61,6,66, \
2,68,6,70,3,77,2,79,6,83,6,120,2,123,5,125,3,130,1,132,3,134,3,141, \
2,143,6,147,3,156"
```

Example 194

```
Endrass_F7.txt:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 7 -end \
▷  ▷  -define R -polynomial_ring -field F \
▷  ▷  ▷  -number_of_variables 4 \
▷  ▷  ▷  -homogeneous_of_degree 8 \
▷  ▷  ▷  -end \
▷  ▷  -define eqn -vector -field F -sparse 165 \
▷  ▷  ▷  $(ENDRASS_SPARSE) -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷  ▷  -define Endrass_F7 -geometric_object P \
▷  ▷  ▷  -projective_variety R \
▷  ▷  ▷  ▷  "Endrass_F7" \
▷  ▷  ▷  ▷  "Endrass\_F7" \
▷  ▷  ▷  ▷  eqn \
▷  ▷  -end \
▷  ▷  -with Endrass_F7 -do \
▷  ▷  -combinatorial_object_activity -save \
▷  ▷  -end
```

Suppose we want to create the monomials of degree 8 in 4 variables. We use an diophantine system to do so. The following command creates the system and solves it. After that, it applies the unix sort command to sort the monomials:

Example 195

```
octic_prepare:
▷  $(ORBITER) -v 4 \
▷  ▷  -define A -vector -format 1 -dense "1,1,1,1" -end \
▷  ▷  -define D -diophant \
▷  ▷  ▷  -label octic_monomials \
▷  ▷  ▷  -coefficient_matrix A \
▷  ▷  ▷  -RHS "mult=1,EQ=8" \
▷  ▷  ▷  -x_min_global 0 -x_max_global 8 \
▷  ▷  -end \
▷  ▷  -with D -do \
▷  ▷  ▷  -diophant_activity -solve_mckay \
▷  ▷  -end
▷  sort -r octic_monomials_sol.csv >octic_monomials_sorted.txt
```

There are 165 monomials. They are listed in the file `octic_monomials_sorted.txt`.

The following command creates the permutation polynomial associated with the map

$$\sigma : \mathbb{F}_5 \to \mathbb{F}_5, \begin{bmatrix} 0 & 1 & 2 & 3 & 4 \\ 3 & 0 & 4 & 1 & 2 \end{bmatrix}$$

Example 196

```
permutation_polynomial_F5:
▷  $(ORBITER) -v 6 \
▷  ▷  -define F -finite_field -q 5 -end \
▷  ▷  -define P -projective_space -n 1 -field F -v 0 -end \
▷  ▷  -define R -polynomial_ring \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -number_of_variables 2 \
▷  ▷  ▷  -homogeneous_of_degree 4 \
▷  ▷  ▷  -monomial_ordering_lex \
▷  ▷  ▷  -variables "x,y" "x,y" \
▷  ▷  ▷  -end \
▷  ▷  -define Y0 -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -text "3*(1-x^4)+4*(1-(x-2)^4)+1*(1-(x-3)^4)+2*(1-(x-4)^4)" \
▷  ▷  ▷  -end \
▷  ▷  -define Y -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -text "Y0" \
▷  ▷  ▷  -end \
▷  ▷  -with P -do \
▷  ▷  -projective_space_activity \
▷  ▷  ▷  -affine_map R Y "" \
▷  ▷  -end \
▷  ▷  -define Yx -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -expand Y \
▷  ▷  ▷  -end \
```

The code utilizes the basis functions

$$\delta_{x,i} = 1 - (x - i)^{q-1}$$

of $\mathbb{F}_5$, which are one if $x = i$ and zero otherwise. Because Orbiter offers homogeneous polynomials only, a polynomial ring in two-variables is utilized. The polynomials are homogeneous of degree $q - 1 = 4$. We can write

$$\begin{aligned} \sigma &= \sum_{i=0}^{4} \sigma(i)\delta_{x,i} \\ &= \sigma(0)\delta_{x,0} + \sigma(1)\delta_{x,1} + \sigma(2)\delta_{x,2} + \sigma(3)\delta_{x,3} + \sigma(4)\delta_{x,4} \\ &= 3\delta_{x,0} + 0\delta_{x,1} + 4\delta_{x,2} + 1\delta_{x,3} + 2\delta_{x,4}, \\ &= 3(1 - x^4) + 4(1 - (x-2)^4) + (1 - (x-3)^4) + 2(1 - (x-4)^4), \end{aligned}$$

because

$$\sigma(0) = 3, \sigma(1) = 0, \sigma(2) = 4, \sigma(3) = 1, \sigma(4) = 2.$$

After expansion, the permutation polynomial associated with σ turns out to be

$$3 + x^3 + (3x^2) + (3x).$$

Mappings Between Projective Spaces

Command	Arguments	Purpose
-domain	D	Set the domain to D (a projective space).
-codomain	C	Set the codomain to C (a projective space).
-ring	R	Set the coordinate ring for D.
-formula	f	Use the symbolic object f to define the mapping.
-substitute	s	Substitute s into the equation before doing the mapping.
-affine		Treat the domain as an affine space.
-object_in_codomain	S	Set the codomain to the cubic surface object S and map to the points on the surface.

Table 5.6: Mappings Between Projective Spaces

5.3 Algebraic Geometry

Table 5.6 lists Orbiter commands for mappings between projective spaces.
The first example creates the Veronese variety of PG(1, 7) in PG(3, 7):

Example 197

```
Veronese_1_3_q7:
▷  $(ORBITER) -v 5 \
▷  ▷  -define Fq -finite_field -q 7 -end \
▷  ▷  -define C -symbolic_object \
▷  ▷  ▷  -field Fq \
▷  ▷  ▷  -managed_variables "y0,y1" \
▷  ▷  ▷  -text "y1^3,y0*y1^2,y0^2*y1,y0^3" \
▷  ▷  -end \
▷  ▷  -define P1 -projective_space -n 1 -field Fq -v 0 -end \
▷  ▷  -define P3 -projective_space -n 3 -field Fq -v 0 -end \
▷  ▷  -define R2 -polynomial_ring \
▷  ▷  ▷  -field Fq \
▷  ▷  ▷  -number_of_variables 2 \
▷  ▷  ▷  -homogeneous_of_degree 3 \
▷  ▷  ▷  -monomial_ordering_partition \
▷  ▷  ▷  -variables "y0,y1" "y_0,y_1" \
▷  ▷  ▷  -end \
▷  ▷  -define map -mapping \
▷  ▷  ▷  -domain P1 \
▷  ▷  ▷  -codomain P3 \
▷  ▷  ▷  -ring R2 \
▷  ▷  ▷  -formula C \
▷  ▷  ▷  -substitute "" \
▷  ▷  -end
```

The image is written as a set of points in Orbiter ranks:

$$(3, 0, 4, 268, 336, 184, 224, 364)$$

For a deeper discussion of the combinatorial and group-theoretic properties of this object, see Section 16.3.

Our next example is the Hermitian curve, with equation

$$X^4 + Y^4 + Z^4$$

in PG(2, 9). Orbiter can compute the variety over a specific field, as well as some of its properties, like automorphism group and tactical decomposition. We first need a makefile variable which holds the equation in algebraic form:

Example 198

```
FILE_HERMITIAN_CURVE="Line,equation\n0,\"X^4+Y^4+Z^4\"\nEND\n\n"
```

The following command copies the makefile variable into a file and then performs the relevant Orbiter computations to create the curve and to investigate it.

Example 199

```
variety_hermitian_curve:
▷  echo $(FILE_HERMITIAN_CURVE) >input.csv
▷  $(ORBITER) -v 8 \
▷  ▷  -define F -finite_field -q 9 -end \
▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
▷  ▷  -define R -polynomial_ring \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -number_of_variables 3 \
▷  ▷  ▷  -homogeneous_of_degree 4 \
▷  ▷  ▷  -monomial_ordering_partition \
▷  ▷  ▷  -variables "X,Y,Z" "X,Y,Z" \
▷  ▷  -end \
▷  ▷  -define O -orbits -classification_by_canonical_form \
▷  ▷  ▷  -space P \
▷  ▷  ▷  -ring R \
▷  ▷  ▷  -input_fname_mask input.csv \
▷  ▷  ▷  -nb_files 1 \
▷  ▷  ▷  -output_fname hermitian_curve_q9 \
▷  ▷  ▷  -label_equation "equation" \
▷  ▷  ▷  -algorithm_nauty \
▷  ▷  ▷  -end \
▷  ▷  -end \
▷  -with O -do -orbits_activity \
▷  ▷  -report \
▷  ▷  -report_options \
▷  ▷  ▷  -fname hermitian_curve_q9 \
▷  ▷  -end \
▷  -end
▷  pdflatex hermitian_curve_q9_orbits.tex
▷  $(OPEN) hermitian_curve_q9_orbits.pdf
```

The command produces a latex report, shown below:

Classification
$q = 9$
Number of isomorphism classes: 1
Automorphism group order statistic: 12096

The isomorphism classes are:
Isomorphism class 0 / 1 is input 0:
Automorphism group order 12096
Number of points 28
Equation `X^4+Y^4+Z^4`
Equation (1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
Points:
0: $3 = (1,1,1)$, 1: $6 = (3,1,0)$, 2: $7 = (4,1,0)$, 3: $9 = (6,1,0)$, 4: $11 = (8,1,0)$, 5: $14 = (3,0,1)$, 6: $15 = (4,0,1)$, 7: $17 = (6,0,1)$, 8: $19 = (8,0,1)$, 9: $21 = (2,1,1)$, 10: $24 = (5,1,1)$, 11: $26 = (7,1,1)$, 12: $29 = (1,2,1)$, 13: $30 = (2,2,1)$, 14: $33 = (5,2,1)$, 15: $35 = (7,2,1)$, 16: $37 = (0,3,1)$, 17: $46 = (0,4,1)$, 18: $56 = (1,5,1)$, 19: $57 = (2,5,1)$, 20: $60 = (5,5,1)$, 21: $62 = (7,5,1)$, 22: $64 = (0,6,1)$, 23: $74 = (1,7,1)$, 24: $75 = (2,7,1)$, 25: $78 = (5,7,1)$, 26: $80 = (7,7,1)$, 27: $82 = (0,8,1)$
Number of equations with the same set of points 1

Automorphism group:
Strong generators for a group of order 12096:
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}_1, \begin{bmatrix} 7 & 0 & 0 \\ 0 & 7 & 0 \\ 0 & 0 & 1 \end{bmatrix}_1, \begin{bmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 1 \end{bmatrix}_1, \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}_0, \begin{bmatrix} 8 & 0 & 0 \\ 0 & 5 & 7 \\ 0 & 1 & 1 \end{bmatrix}_1, \begin{bmatrix} 6 & 4 & 0 \\ 4 & 6 & 0 \\ 0 & 0 & 1 \end{bmatrix}_1, \begin{bmatrix} 2 & 8 & 2 \\ 1 & 5 & 3 \\ 3 & 5 & 1 \end{bmatrix}_1$$
1,0,0,0,1,0,0,0,1,1,
1,0,0,0,1,0,0,0,5,1,
1,0,0,0,5,0,0,0,1,1,
1,0,0,0,0,1,0,1,0,0,
1,0,0,0,8,4,0,6,6,1,
1,7,0,7,1,0,0,0,8,1,
1,4,1,2,7,6,6,7,2,1,

The TDO decomposition is

$\downarrow$	63_1	28_3
28_0	4	1
63_2	6	9

$\rightarrow$	63_1	28_3
28_0	9	1
63_2	6	4

Point classes:
Set 0 has size 28 : { 3, 6, 7, 9, 11, 14, 15, 17, 19, 21, 24, 26, 29, 30, 33, 35, 37, 46, 56, 57, 60, 62, 64, 74, 75, 78, 80, 82 }
Set 1 has size 63 : { 0, 1, 2, 4, 5, 8, 10, 12, 13, 16, 18, 20, 22, 23, 25, 27, 28, 31, 32, 34, 36, 38, 39, 40, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 53, 54, 55, 58, 59, 61, 63, 65, 66, 67, 68, 69, 70, 71, 72, 73, 76, 77, 79, 81, 83, 84, 85, 86, 87, 88, 89, 90 }
Block classes:
Set 0 has size 63 : { 0, 1, 2, 5, 7, 9, 10, 13, 14, 16, 18, 19, 20, 23, 24, 26, 28, 29, 31, 32, 33, 34, 35, 36, 37, 38, 41, 42, 43, 44, 45, 46, 47, 48, 50, 53, 54, 56, 58, 59, 61, 62, 63, 64, 65, 66, 67, 68, 70, 73, 74, 76, 78, 79, 81, 82, 83, 84, 85, 86, 87, 88, 90 }
Set 1 has size 28 : { 3, 4, 6, 8, 11, 12, 15, 17, 21, 22, 25, 27, 30, 39, 40, 49, 51, 52, 55, 57, 60, 69, 71, 72, 75, 77, 80, 89 }
The TDA decomposition is

$\downarrow$	63_1	28_3
28_0	4	1
63_2	6	9

$\rightarrow$	63_1	28_3
28_0	9	1
63_2	6	4

Point classes:
Set 0 has size 28 : { 3, 6, 7, 9, 11, 14, 15, 17, 19, 21, 24, 26, 29, 30, 33, 35, 37, 46, 56, 57, 60, 62, 64, 74, 75, 78, 80, 82 }
Set 1 has size 63 : { 0, 1, 2, 4, 5, 8, 10, 12, 13, 16, 18, 20, 22, 23, 25, 27, 28, 31, 32, 34, 36, 38, 39, 40, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 53, 54, 55, 58, 59, 61, 63, 65, 66, 67, 68, 69, 70, 71, 72, 73, 76, 77, 79, 81, 83, 84, 85, 86, 87, 88, 89, 90 }
Block classes:
Set 0 has size 63 : { 0, 1, 2, 5, 7, 9, 10, 13, 14, 16, 18, 19, 20, 23, 24, 26, 28, 29, 31, 32, 33, 34, 35, 36, 37, 38, 41, 42, 43, 44, 45, 46, 47, 48, 50, 53, 54, 56, 58, 59, 61, 62, 63, 64, 65, 66, 67, 68, 70, 73, 74, 76, 78, 79, 81, 82, 83, 84, 85, 86, 87, 88, 90 }
Set 1 has size 28 : { 3, 4, 6, 8, 11, 12, 15, 17, 21, 22, 25, 27, 30, 39, 40, 49, 51, 52, 55, 57, 60, 69, 71, 72, 75, 77, 80, 89 }

Chapter 6

Group Theory

6.1 Generic Permutation Groups

Orbiter distinguishes two types of permutation groups: generic permutation groups and matrix groups. Both groups come with a permutation action. The difference is how a group element is stored. In a generic permutation group, a group element is stored as a permutation of the given degree. For a matrix group, a group element is stored as a matrix. Matrix group representations are often more efficient than the corresponding generic representation as a permutation group. However, not every group has a matrix group representation of reasonably small dimension.

In this section, we will talk about generic permutation groups in Orbiter. In Section 6.2, we will discuss matrix groups over finite fields.

In order to establish the permutation representation of a group, the technique of indexing is used. Indexing sets up a fixed bijection between the permutation domain (the set we act on) and the integer interval $[0, n-1]$ for some n. The integer associated to an element in the permutation domain is called the rank. Conversely, given an integer in $[0, n-1]$, the element in the permutation domain associated with it is obtained by the unrank function. The process of converting integers to elements of the permutation domain and vice-versa is indexing. We have seen indexing for projective points in Section 4.1.

The enumerators for projective points from Section 4.1 are used to realize the permutation domain. This enumerator relies on an enumerator for finite fields, as discussed in Sections 3.2 and 3.3. For extension fields, the enumerator for finite fields in turn depends on the choice of the irreducible polynomial which is used to create the field. For affine groups, a different enumerator is used to describe the permutation domain. This enumerator uses the base-q representation of integers, which associates a vector over $\mathbb{F}_q$ of length n with an integer in $[0, q^n - 1]$.

Permutation groups and matrix groups can be represented on a computer using the technique of stabilizer chains, or Sims chains (cf. [39, 65]). The stabilizer chain is define with respect to a sequence of points in the permutation domain called a base. A set of generators which allows to generate each group along the chain is called a strong generating set. Many algorithms for permutation groups rely on knowing a base and strong generating set. In Orbiter, permutation groups can be created from a base and strong generating set. Many types of groups come with their own built-in base and strong generating set. On the other hand, it is also possible to create groups from generating sets which are either not strong or for which a base is not known. For efficiency purposes, small basic orbits are desired.

Table 6.1 list Orbiter commands to create a generic permutation group.

Let us start with a cyclic group. The following command creates a cyclic group of order 6:

Creating a Generic Permutation Group

Command	Arguments	Purpose
-symmetric_group	n	Symmetric group of degree n and order $n!$.
-cyclic_group	n	Cyclic group of degree n and order n.
-identity_group	n	Identity group of degree n and order 1.
-dihedral_group	n	Dihedral group of degree n and order $2n$.
-bsgs	lab1 lab2 n order base k gens	Group with a given base and a given set of generators. Here, lab1 and lab2 are labels in ascii and latex, respectively, k is the number of generators, gens is the generators as an Orbiter vector of integers.
-subgroup_by_generators	lab order k gens	Subgroup given by generators. Here, k is the number of generators, gens is an Orbiter vector of integers.

Table 6.1: Creating a Generic Permutation Group

Example 200

```
Cyclic_6:
▷ $(ORBITER) -v 3 \
▷ ▷ -define G -permutation_group -cyclic_group 6 -end \
▷ ▷ -with G -do \
▷ ▷ ▷ -group_theoretic_activity \
▷ ▷ ▷ -report \
▷ ▷ -end
▷ pdflatex C_6_report.tex
▷ $(OPEN) C_6_report.pdf
```

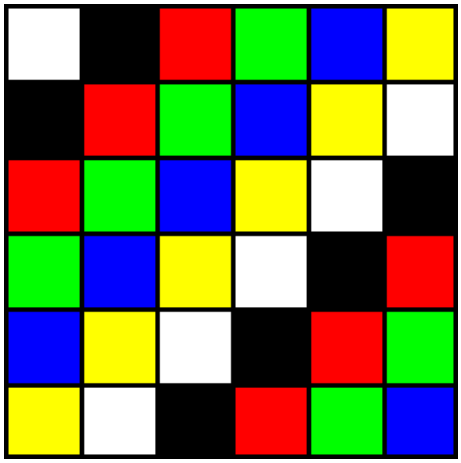
The following command

Example 201

```
Cyclic_6_group_table:
▷ $(ORBITER) -v 3 \
▷ ▷ -define G -permutation_group -cyclic_group 6 -end \
▷ ▷ -with G -do \
▷ ▷ -group_theoretic_activity \
▷ ▷ ▷ -export_group_table \
▷ ▷ -end
▷ $(ORBITER) -v 2 \
▷ ▷ -define all_one_r -vector -repeat 1 6 -end \
▷ ▷ -define all_one_c -vector -repeat 1 6 -end \
▷ ▷ -draw_matrix \
▷ ▷ ▷ -input_csv_file C_6_group_table.csv \
▷ ▷ ▷ -box_width 50 -bit_depth 24 \
▷ ▷ ▷ -partition 3 all_one_r all_one_c \
▷ ▷ -end
```

```
▷  $(OPEN) C_6_group_table_draw.bmp
```

produces a graphical representation of the group table of the cyclic group C_6, shown below



Next, let us consider the symmetric group $\mathrm{Sym}(n)$. The following command does so:

Example 202

```
Symmetric_3:
▷  $(ORBITER) -v 3 \
▷  ▷  -define G -permutation_group -symmetric_group 3 -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex Sym_3_report.tex
▷  $(OPEN) Sym_3_report.pdf
```

The report is shown below:

Stabilizer chain

Level	Base pt	Orbit length	Subgroup order
0	0	3	6
1	1	2	2

Basic Orbit 0

Basic orbit 0 has size 3
0, 1, 2

Basic Orbit 1

Basic orbit 1 has size 2
1, 2

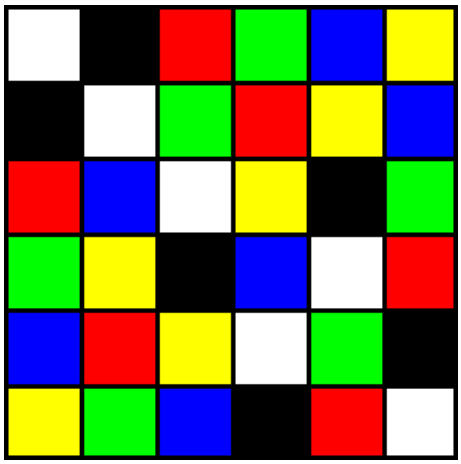
The following command

Example 203

```
Symmetric_3_group_table:
▷  $(ORBITER) -v 3 \
▷  ▷  -define G -permutation_group -symmetric_group 3 -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -export_group_table \
▷  ▷  -end
▷  $(ORBITER) -v 2 \
▷  ▷  -define all_one_r -vector -repeat 1 6 -end \
▷  ▷  -define all_one_c -vector -repeat 1 6 -end \
▷  ▷  -draw_matrix \
```

```
▷ ▷ ▷ -input_csv_file Sym_3_group_table.csv \
▷ ▷ ▷ -box_width 50 -bit_depth 24 \
▷ ▷ ▷ -partition 3 all_one_r all_one_c \
▷ ▷ -end
▷ $(OPEN) Sym_3_group_table_draw.bmp
```

produces a graphical representation of the group table of the symmetric group Sym(3), shown below

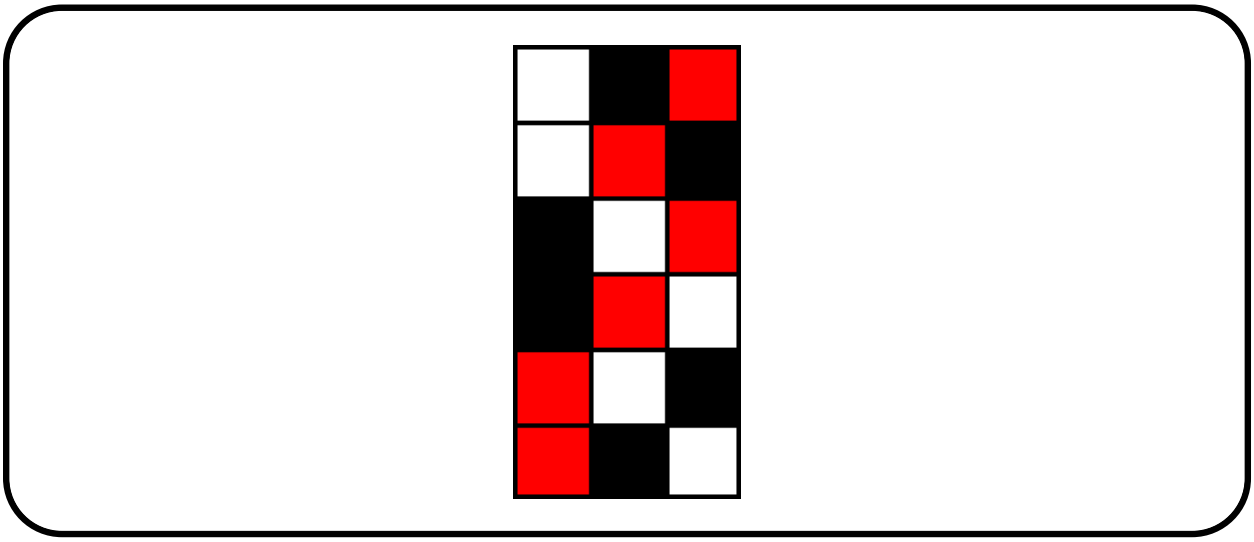


The next command

Example 204

```
Symmetric_3_elements:
▷ $(ORBITER) -v 3 \
▷ ▷ -define G -permutation_group -symmetric_group 3 -end \
▷ ▷ -with G -do \
▷ ▷ -group_theoretic_activity \
▷ ▷ ▷ -save_elements_csv "Symmetric3_elts.csv" \
▷ ▷ -end
▷ $(ORBITER) -v 2 \
▷ ▷ -define Sym3_elts -vector -load_csv_data_column \
▷ ▷ ▷ Symmetric3_elts.csv 1 -end \
▷ ▷ -save_matrix_csv Sym3_elts
▷ $(ORBITER) -v 2 \
▷ ▷ -define all_one_r -vector -repeat 1 6 -end \
▷ ▷ -define all_one_c -vector -repeat 1 3 -end \
▷ ▷ -draw_matrix \
▷ ▷ ▷ -input_csv_file Sym3_elts_matrix.csv \
▷ ▷ ▷ -box_width 50 -bit_depth 8 \
▷ ▷ ▷ -partition 3 \
▷ ▷ ▷ ▷ all_one_r all_one_c \
▷ ▷ -end
▷ $(OPEN) Sym3_elts_matrix_draw.bmp
```

produces a graphical representation of the elements of the symmetric group Sym(3), shown below



Let us create the groups of order 8. Up to isomorphism, there are 5 groups. Three are abelian (C_8, $C_4 \times C_2$ and $C_2 \times C_2 \times C_2$) and two are not (Dihedral group of order 8 and the quaternion group). The following commands can be used to create these five groups one-by-one:

Example 205

```
Group_8_1:
▷  $(ORBITER) -v 2 \
▷  ▷  -define gens -vector -dense \
▷  ▷  ▷  "1,2,3,4,5,6,7,0" \
▷  ▷  -end \
▷  ▷  -define G -permutation_group -symmetric_group 8 \
▷  ▷  ▷  -subgroup_by_generators G1 8 1 gens -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -export_group_table \
▷  ▷  -end
▷  $(ORBITER) -v 2 \
▷  ▷  -define all_one_r -vector -repeat 1 8 -end \
▷  ▷  -define all_one_c -vector -repeat 1 8 -end \
▷  ▷  -draw_matrix \
▷  ▷  ▷  -input_csv_file \
▷  ▷  ▷  ▷  Sym_8_Subgroup_G1_8_group_table.csv \
▷  ▷  ▷  -box_width 50 -bit_depth 24 \
▷  ▷  ▷  -partition 3 all_one_r all_one_c \
▷  ▷  -end
▷  convert Sym_8_Subgroup_G1_8_group_table_draw.bmp \
▷  ▷  Sym_8_Subgroup_G1_8_group_table_draw.png
▷  $(OPEN) Sym_8_Subgroup_G1_8_group_table_draw.png
```

Example 206

```
Group_8_2:
▷  $(ORBITER) -v 2 \
▷  ▷  -define gens -vector -dense \
▷  ▷  ▷  "1,2,3,0,4,5, 0,1,2,3,5,4" \
▷  ▷  -end \
▷  ▷  -define G -permutation_group -symmetric_group 6 \
▷  ▷  ▷  -subgroup_by_generators G2 8 2 gens -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -export_group_table \
▷  ▷  -end
▷  $(ORBITER) -v 2 \
▷  ▷  -define all_one_r -vector -repeat 1 8 -end \
▷  ▷  -define all_one_c -vector -repeat 1 8 -end \
▷  ▷  -draw_matrix \
▷  ▷  ▷  -input_csv_file \
▷  ▷  ▷  ▷  Sym_6_Subgroup_G2_8_group_table.csv \
▷  ▷  ▷  -box_width 50 -bit_depth 24 \
▷  ▷  ▷  -partition 3 all_one_r all_one_c \
▷  ▷  -end
▷  convert Sym_6_Subgroup_G2_8_group_table_draw.bmp \
▷  ▷  Sym_6_Subgroup_G2_8_group_table_draw.png
▷  $(OPEN) Sym_6_Subgroup_G2_8_group_table_draw.png
```

Example 207

```
Group_8_3:
▷  $(ORBITER) -v 2 \
▷  ▷  -define gens -vector -dense \
▷  ▷  ▷  "1,0,2,3,4,5, 0,1,3,2,4,5, 0,1,2,3,5,4" \
▷  ▷  -end \
▷  ▷  -define G -permutation_group -symmetric_group 6 \
▷  ▷  ▷  -subgroup_by_generators G3 8 3 gens -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -export_group_table \
▷  ▷  -end
▷  $(ORBITER) -v 2 \
▷  ▷  -define all_one_r -vector -repeat 1 8 -end \
▷  ▷  -define all_one_c -vector -repeat 1 8 -end \
▷  ▷  -draw_matrix \
▷  ▷  ▷  -input_csv_file \
▷  ▷  ▷  ▷  Sym_6_Subgroup_G3_8_group_table.csv \
▷  ▷  ▷  -box_width 50 -bit_depth 24 \
▷  ▷  ▷  -partition 3 all_one_r all_one_c \
▷  ▷  -end
▷  convert Sym_6_Subgroup_G3_8_group_table_draw.bmp \
▷  ▷  Sym_6_Subgroup_G3_8_group_table_draw.png
▷  $(OPEN) Sym_6_Subgroup_G3_8_group_table_draw.png
```

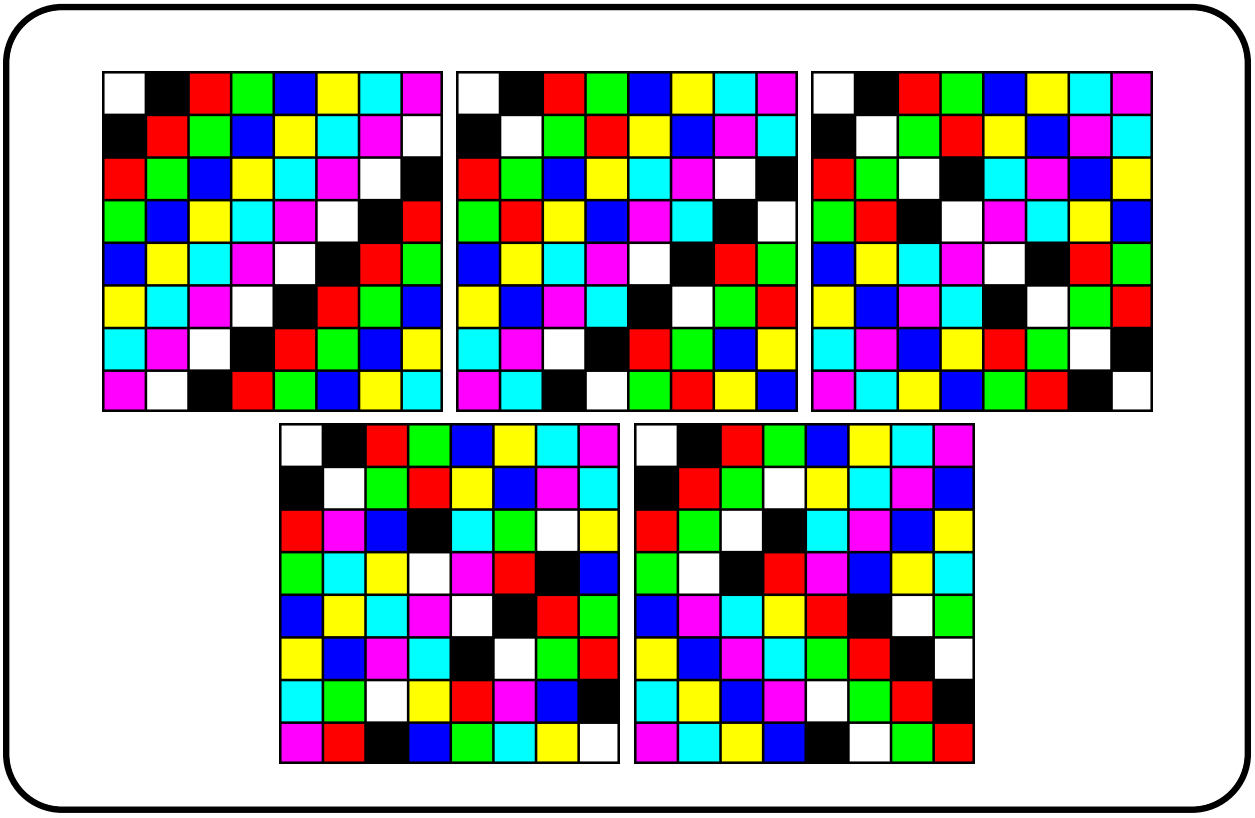
Example 208

```
Group_8_4:
▷  $(ORBITER) -v 2 \
▷  ▷  -define gens -vector -dense \
▷  ▷  ▷  "1,2,3,0, 0,3,2,1" \
▷  ▷  -end \
▷  ▷  -define G -permutation_group -symmetric_group 4 \
▷  ▷  ▷  -subgroup_by_generators G4 8 2 gens -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -export_group_table \
▷  ▷  -end
▷  $(ORBITER) -v 2 \
▷  ▷  -define all_one_r -vector -repeat 1 8 -end \
▷  ▷  -define all_one_c -vector -repeat 1 8 -end \
▷  ▷  -draw_matrix \
▷  ▷  ▷  -input_csv_file \
▷  ▷  ▷  ▷  Sym_4_Subgroup_G4_8_group_table.csv \
▷  ▷  ▷  -box_width 50 -bit_depth 24 \
▷  ▷  ▷  -partition 3 all_one_r all_one_c \
▷  ▷  -end
▷  convert Sym_4_Subgroup_G4_8_group_table_draw.bmp \
▷  ▷  Sym_4_Subgroup_G4_8_group_table_draw.png
▷  $(OPEN) Sym_4_Subgroup_G4_8_group_table_draw.png
```

Example 209

```
Group_8_5:
▷  $(ORBITER) -v 2 \
▷  ▷  -define gens -vector -dense \
▷  ▷  ▷  "2,3,1,0,6,7,5,4, 4,5,7,6,1,0,2,3" \
▷  ▷  -end \
▷  ▷  -define G -permutation_group -symmetric_group 8 \
▷  ▷  ▷  -subgroup_by_generators G5 8 2 gens -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -export_group_table \
▷  ▷  -end
▷  $(ORBITER) -v 2 \
▷  ▷  -define all_one_r -vector -repeat 1 8 -end \
▷  ▷  -define all_one_c -vector -repeat 1 8 -end \
▷  ▷  -draw_matrix \
▷  ▷  ▷  -input_csv_file \
▷  ▷  ▷  ▷  Sym_8_Subgroup_G5_8_group_table.csv \
▷  ▷  ▷  -box_width 50 -bit_depth 24 \
▷  ▷  ▷  -partition 3 all_one_r all_one_c \
▷  ▷  -end
▷  convert Sym_8_Subgroup_G5_8_group_table_draw.bmp \
▷  ▷  Sym_8_Subgroup_G5_8_group_table_draw.png
▷  $(OPEN) Sym_8_Subgroup_G5_8_group_table_draw.png
```

The Cayley tables are shown below



The Mathieu group M_{12} of order 95040 is generated by the permutations

$$(0,3)(2,9)(4,10)(5,11), \quad (0,7,8)(1,2,3)(4,11,10)(5,9,6),$$

taken from [73]. The generators can be written in list notation, which is the second row in the arrays

$$\begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ 3 & 1 & 9 & 0 & 10 & 11 & 6 & 7 & 8 & 2 & 4 & 5 \end{bmatrix} \text{ and } \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ 7 & 2 & 3 & 1 & 11 & 9 & 5 & 8 & 0 & 6 & 4 & 10 \end{bmatrix}.$$

We use a makefile variable to store these generators, written one after another:

Example 210

```
GENERATORS_M12="3,1,9,0,10,11,6,7,8,2,4,5, 7,2,3,1,11,9,5,8,0,6,4,10"
```

The following command creates the group and produces a report.

Example 211

```
Group_M12:
▷  $(ORBITER) -v 2 \
▷  ▷  -define gens -vector -dense $(GENERATORS_M12) -end \
▷  ▷  -define G -permutation_group -symmetric_group 12 \
▷  ▷  ▷  -subgroup_by_generators M12 95040 2 gens -end \
```

```
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex Sym_12_Subgroup_M12_95040_report.tex
▷  $(OPEN) Sym_12_Subgroup_M12_95040_report.pdf
```

Creating a Matrix Group (Part 1)		
Command	Arguments	Purpose
-GL	n F	$\mathrm{GL}(n,q)$. Here (and in all of below), F is a Orbiter object for a finite field $\mathbb{F}_q$.
-GGL	n F	$\Gamma\mathrm{L}(n,q)$.
-SL	n F	$\mathrm{SL}(n,q)$.
-SSL	n F	$\Sigma\mathrm{L}(n,q)$.
-PGL	n F	$\mathrm{PGL}(n,q)$.
-PGGL	n F	$\mathrm{P\Gamma L}(n,q)$.
-PSL	n F	$\mathrm{PSL}(n,q)$.
-PSSL	n F	$\mathrm{P\Sigma L}(n,q)$.
-AGL	n F	$\mathrm{AGL}(n,q)$.
-AGGL	n F	$\mathrm{A\Gamma L}(n,q)$.
-ASL	n F	$\mathrm{ASL}(n,q)$.
-ASSL	n F	$\mathrm{A\Sigma L}(n,q)$.
-GL_d_q_wr_Sym_n	d F n	$\mathrm{GL}(d,q)$ wreath product $\mathrm{Sym}(n)$.
-PGO	n F	$\mathrm{PGO}(n,q)$.
-PGOp	n F	$\mathrm{PGO}^+(n,q)$.
-PGOm	n F	$\mathrm{PGO}^-(n,q)$.
-PGGO	n F	$\mathrm{P\Gamma O}(n,q)$.
-PGGOp	n F	$\mathrm{P\Gamma O}^+(n,q)$.
-PGGOm	n F	$\mathrm{P\Gamma O}^-(n,q)$.

Table 6.2: Creating a Matrix Group (Part 1)

6.2 Matrix Groups

Tables 6.2-6.3 shows the Orbiter commands to create a matrix group. A matrix group always comes with a permutation group representation, mostly called the "natural" representation. In some commands, the permutation action is replaced by something else. For background information about the classical groups of matrices over finite fields, see cf. [70].

Let $V \simeq \mathbb{F}_q^n$ be a finite dimensional vector space over $\mathbb{F}_q$. The set of subspaces of V form the projective geometry $\mathrm{PG}(n-1,q)$.

Let π be a projective space. A collineation of a projective space π is a bijective mapping from the points of π to themselves which preserves collinearity. That is, a collineation φ maps any three collinear points P, Q, R to another collinear triple $\varphi(P), \varphi(Q), \varphi(R)$. The collineations form a group with respect to composition, the collineation group. If M is the matrix of an endomorphism, then Ψ_M is the induced map on projective space. By considering the homomorphism $M \mapsto \Psi_M$, the group $\mathrm{GL}(n+1,q)$ of invertible endomorphisms becomes a subgroup of the group of collineations of $\mathrm{PG}(n,q)$. This is the projectivity group $\mathrm{PGL}(n+1,q)$. It is isomorphic to $\mathrm{GL}(n+1,q)/\mathbb{F}_q^\times$. Another source of collineations is this: Let $\Phi \in \mathrm{Aut}(\mathbb{F}_q)$ be a field automorphism. Then Φ acts on projective space by sending $P(\mathbf{x})$ to $P(\mathbf{x}\Phi)$. This map is another type of collineation, called automorphic collineation. This way, $\mathrm{Aut}(\mathbb{F}_q)$ gives rise to a group of collineations. If $q = p^h$ for some prime p and some integer h then

$$\Phi_0 : \mathbb{F}_q \to \mathbb{F}_q,\ x \mapsto x^p$$

is a generator for the cyclic group $C_h \simeq \mathrm{Aut}(\mathbb{F}_q)$. The collineation group of $\mathrm{PG}(n,q)$ $(n \geq 2)$ is isomorphic to the semidirect product of the projectivity group and the automorphism group of the field. The collineation

Creating a Matrix Group (Part 2)

Command	Arguments	Purpose
-wedge		Create the group in the action on the wedge product.
-wedge_detached		Create the group in the action on the wedge product.
-PGL2OnConic		Create the action of $\mathrm{PGL}(2,q)$ on the conic.
-monomial		Create the group in the monomial action.
-diagonal		Create the group in the diagonal action.
-null_polarity_group		Create the subgroup corresponding to a null polarity.
-symplectic_group		Create the subgroup of type symplectic group.
-singer		Create the Singer subgroup.
-singer_and_frobenius	k	Create the Singer subgroup extended by a Frobenius automorphism. Here, k is the power of the Frobenius endomorphism.
-subfield_structure_action	s	Create the action on the field reduction w.r.t. the subfield $\mathbb{F}_r$, where $r^s = q$.
-subgroup_from_file	fname label	Create the subgroup from the given file and assign the given label.
-borel_upper		Create the upper Borel subgroup.
-borel_lower		Create the lower Borel subgroup.
-identity_group		Create the identity subgroup.
-orthogonal	epsilon	Create the orthogonal group of type epsilon as subgroup, where epsilon $\in \{0, 1, -1\}$.
-on_tensors		Create the tensor action.
-on_rank_one_tensors		Create the action on rank-one tensors.
-subgroup_by_generators	label order k gens	Create a subgroup with a known order from a set of generators. There are k generators. gens is a orbiter vector object.
-Janko1		Create the Janko group J_1 as a subgroup.
-export_magma		Export the group to Magma.
-import_group_of_plane	plane	Import the group of a translation plane. Here, plane must be an Orbiter object of type translation plane.
-lex_least_base		Create the lex-least base. This will make indexing of group elements deterministic.

Table 6.3: Creating a Matrix Group (Part 2)

group is $\mathrm{P\Gamma L}(n+1,q) = \mathrm{PGL}(n+1,q) \ltimes \mathrm{Aut}(\mathbb{F}_q)$. We use the following notation for elements of $\mathrm{P\Gamma L}(n+1,q)$. Let Φ_0 be a generator for $\mathrm{Aut}(\mathbb{F}_q)$ and let $M \in \mathrm{GL}(n+1,q)$. The map

$$(\Psi_M, \Phi_0^k) : \mathrm{PG}(n,q) \to \mathrm{PG}(n,q),\ P(\mathbf{x}) \mapsto P(\mathbf{y}), \quad \mathbf{y} = (\mathbf{x} \cdot M)^{\Phi_0^k}$$

is denoted as

$$M_k. \tag{6.1}$$

The identity element is I_0, where I is the identity matrix and 0 is the residue class modulo h. The rules for multiplication and inversion in the collineation group are given as

$$M_k \cdot N_l = \left(M \cdot N^{\Phi^{-k}}\right)_{k+l}, \tag{6.2}$$

$$\left(M_k\right)^{-1} = \left(\left(M^{-1}\right)^{\Phi^k}\right)_{-k}. \tag{6.3}$$

The affine group $\mathrm{AGL}(n,q)$ is the semidirect product of $\mathrm{GL}(n,q)$ with $\mathbb{F}_q^n$. The affine semilinear group $\mathrm{A\Gamma L}(n,q)$ is the semidirect product of $\mathrm{AGL}(n,q)$ with $\mathrm{Aut}(\mathbb{F}_q)$. The elements of $\mathrm{A\Gamma L}(n,q)$ are triples

$$M_{\mathbf{a},k} := (M, \mathbf{a}, k) \in \mathrm{GL}(n,q) \times \mathbb{F}_q^n \times \mathrm{Aut}(\mathbb{F}_q),$$

which act on $\mathbb{F}_q^n$:

$$\Big(\mathbf{x}, (M, \mathbf{a}, k)\Big) \mapsto (\mathbf{x} \cdot M + \mathbf{a})^{\Phi^k}.$$

The multiplication in $\mathrm{A\Gamma L}(n,q)$ is

$$M_{\mathbf{a},k} \cdot N_{\mathbf{b},l} = (MN)_{\mathbf{a}N^{\Phi^{-k}} + \mathbf{b}^{\Phi^{-k}}, k+l}.$$

The inverse of an element is

$$\left(M_{\mathbf{a},k}\right)^{-1} = \left(M^{-1}\right)_{\mathbf{a}^{\Phi^k} M^{-1}, -k}.$$

A correlation is a one-to-one mapping between the set of points and the set of hyperplanes which reverses incidence. So, if ρ is a correlation and P is a point and ℓ is a hyperplane then P^ρ is a hyperplane and ℓ^ρ is a point and

$$\ell^\rho \in P^\rho \iff P \in \ell.$$

A correlation of order two is called polarity. The standard polarity is the map

$$\rho : \mathcal{P} \leftrightarrow \mathcal{L},\ P(\mathbf{x}) \leftrightarrow [\mathbf{x}].$$

A group G can act on V in one of the types listed in Table 6.4.
One can create a matrix group over a finite field $\mathbb{F}_q$ is created as described in in two steps. In the first step, the field $\mathbb{F}_q$ is created as described in Sections 3.2 and 3.3. The field is stored in the symbol table. Then, the group is created using the symbolic label for the field.

For instance,

Example 212

```
PGL_4_2:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define G -linear_group -PGL 4 F -lex_least_base -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
```

Basic Actions

Type	Perm. Domain	Degree
General linear $\mathrm{GL}(n,q)$	all vectors of V	q^n
Affine $\mathrm{AGL}(n,q)$	all vectors of V	q^n
Projective $\mathrm{PGL}(n,q)$	$\mathfrak{G}r_1(V)$	$\frac{q^n-1}{q-1}$
Wreath product $\mathrm{GL}(d,q) \wr \mathrm{Sym}(n)$	$\mathfrak{G}r_1((\mathbb{F}_q^d)^{\otimes n})$ extended	$n + nq^d + \frac{q^{d^n}-1}{q-1}$
Orthogonal $\mathrm{PGO}(n,q)$	$Q(V)$	$\frac{q^{n-1}-1}{q-1}$
Orthogonal $\mathrm{PGO}^+(n,q)$	$Q^+(V)$	$\frac{(q^{n/2}-1)(q^{(n-2)/2}+1)}{q-1}$
Orthogonal $\mathrm{PGO}^-(n,q)$	$Q^-(V)$	$\frac{(q^{n/2}+1)(q^{(n-2)/2}-1)}{q-1}$

Table 6.4: Basic actions

```
▷ ▷ ▷ -report \
▷ ▷ -end
▷ pdflatex PGL_4_2_report.tex
▷ $(OPEN) PGL_4_2_report.pdf
```

creates the group $\mathrm{PGL}(4,2)$ acting on the 15 elements of $\mathfrak{G}r_1(\mathbb{F}_2^4)$. At first, the field $\mathbb{F}_2$ is created. Secondly, the group $G = \mathrm{PGL}(3,2)$ is created using the previously created field $\mathbb{F}_2$, and a report is generated. The report gives information about the permutation group action, including the underlying field and the projective geometry.

The Group $\mathrm{PGL}(4,2)$

The order of the group $\mathrm{PGL}(4,2)$ is 20160
The group acts on a set of size 15
Strong generators for a group of order 20160:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix},$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

1,0,0,0,0,1,0,0,0,0,1,0,1,0,0,1,
1,0,0,0,0,1,0,0,0,0,1,0,0,1,0,1,
1,0,0,0,0,1,0,0,0,0,1,0,0,0,1,1,
1,0,0,0,0,1,0,0,0,0,0,1,0,0,1,0,
1,0,0,0,0,0,1,0,0,1,0,0,0,0,0,1,
0,1,0,0,1,0,0,0,0,0,1,0,0,0,0,1,

The Action

Group action PGL(4, 2) of degree 15
We act on the following set:

0 = (1, 0, 0, 0)
1 = (0, 1, 0, 0)
2 = (0, 0, 1, 0)
3 = (0, 0, 0, 1)
4 = (1, 1, 1, 1)
5 = (1, 1, 0, 0)
6 = (1, 0, 1, 0)
7 = (0, 1, 1, 0)
8 = (1, 1, 1, 0)
9 = (1, 0, 0, 1)
10 = (0, 1, 0, 1)
11 = (1, 1, 0, 1)
12 = (0, 0, 1, 1)
13 = (1, 0, 1, 1)
14 = (0, 1, 1, 1)

The group is a matrix group.
The group acts on projective space PG(3, 2)
$q = 2$
$p = 2$
$e = 1$
$n = 3$
Number of points = 15
Number of lines = 35
Number of lines on a point = 7
Number of points on a line = 3

The finite field $\mathbb{F}_2$

$Z_i = \log_\alpha(1 + \alpha^i)$

i	γ_i	$-\gamma_i$	γ_i^{-1}	$\log_\alpha(\gamma_i)$	α^i	Z_i
0	$0 = 0$	0	DNE	DNE	1	DNE
1	$1 = 1$	1	1	0	1	DNE

+	0	1
0	0	1
1	1	0

·	1
1	1

$1^0 \equiv 1$
$1^1 \equiv 1$

Base and Stabilizer Chain

Group order 20160
tl=15, 14, 12, 8,
Base: (0, 1, 2, 3)
Strong generators for a group of order 20160:

$$\begin{bmatrix} 1&0&0&0\\0&1&0&0\\0&0&1&0\\1&0&0&1 \end{bmatrix}, \begin{bmatrix} 1&0&0&0\\0&1&0&0\\0&0&1&0\\0&1&0&1 \end{bmatrix}, \begin{bmatrix} 1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&1&1 \end{bmatrix},$$

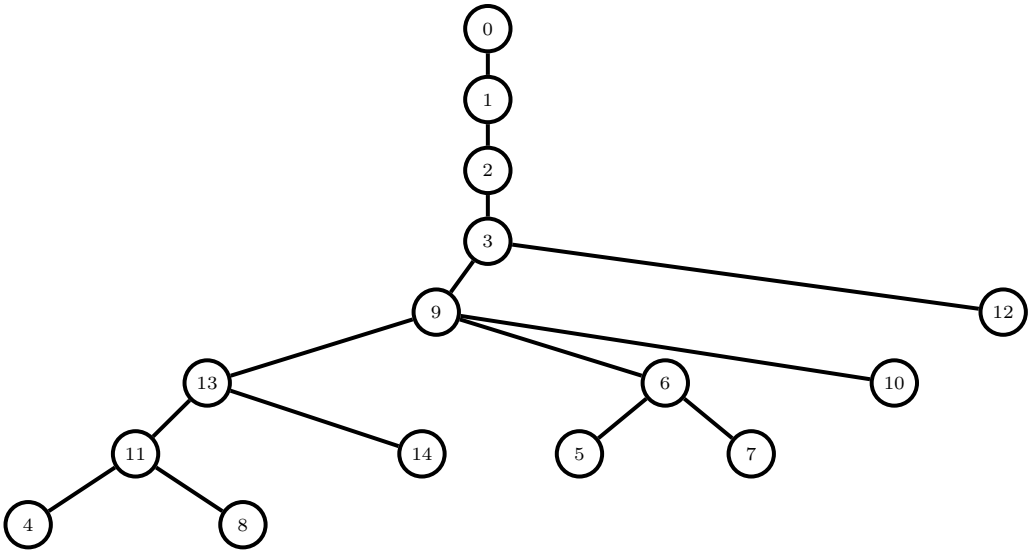
$$\begin{bmatrix} 1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0 \end{bmatrix}, \begin{bmatrix} 1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1 \end{bmatrix}, \begin{bmatrix} 0&1&0&0\\1&0&0&0\\0&0&1&0\\0&0&0&1 \end{bmatrix}$$

1,0,0,0,0,1,0,0,0,0,1,0,1,0,0,1,
1,0,0,0,0,1,0,0,0,0,1,0,0,1,0,1,
1,0,0,0,0,1,0,0,0,0,1,0,0,0,1,1,
1,0,0,0,0,1,0,0,0,0,0,1,0,0,1,0,
1,0,0,0,0,0,1,0,0,1,0,0,0,0,0,1,
0,1,0,0,1,0,0,0,0,0,1,0,0,0,0,1,

Stabilizer chain

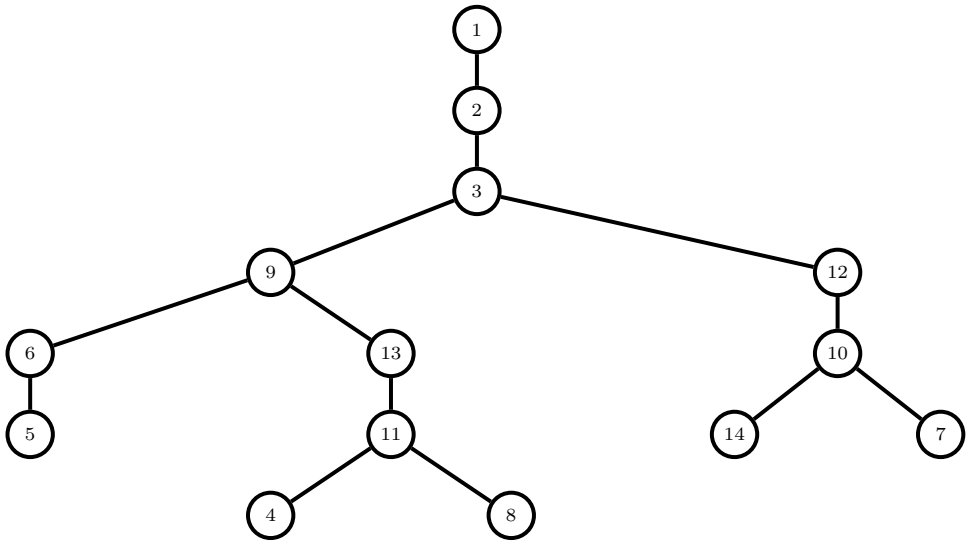
Level	Base pt	Orbit length	Subgroup order
0	0	15	20160
1	1	14	1344
2	2	12	96
3	3	8	8

Basic Orbit 0



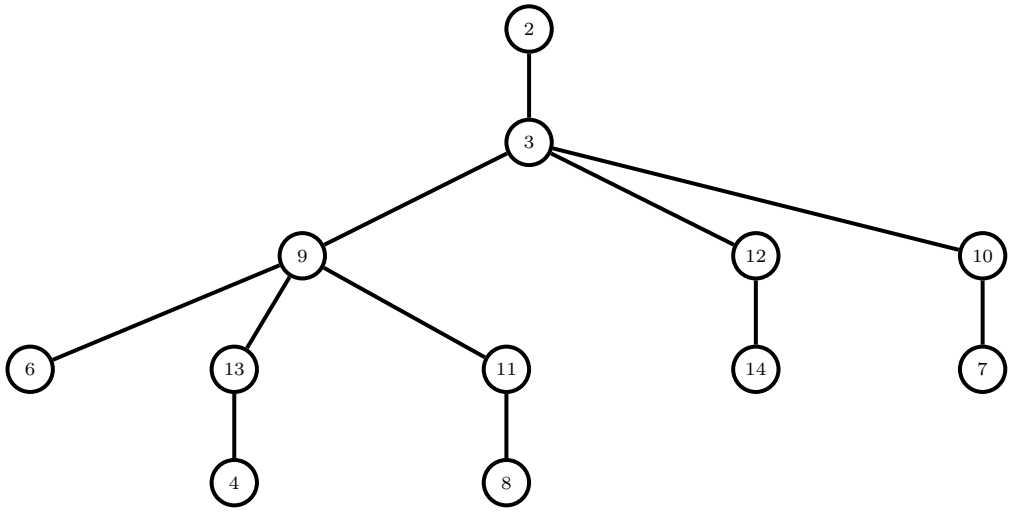
Basic orbit 0 has size 15
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14

Basic Orbit 1



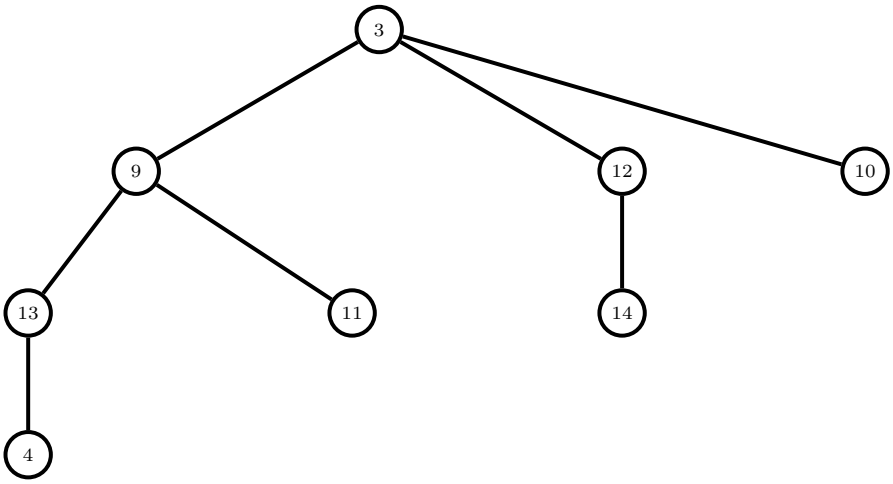
Basic orbit 1 has size 14
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14

Basic Orbit 2



Basic orbit 2 has size 12
2, 3, 4, 6, 7, 8, 9, 10, 11, 12, 13, 14

Basic Orbit 3



Basic orbit 3 has size 8
3, 4, 9, 10, 11, 12, 13, 14
GAP export:

```
Generators in GAP format are:
G := Group([(4, 10)(5, 15)(11, 12)(13, 14),
(4, 11)(5, 14)(10, 12)(13, 15),
(4, 13)(5, 12)(10, 14)(11, 15),
(3, 4)(7, 10)(8, 11)(9, 12),
(2, 3)(6, 7)(11, 13)(12, 14),
(1, 2)(7, 8)(10, 11)(14, 15)]);
```

Magma export:

```
G := GeneralLinearGroup(4, GF(2));
H := sub< G | [1,0,0,0, 0,1,0,0, 0,0,1,0, 1,0,0,1],
[1,0,0,0, 0,1,0,0, 0,0,1,0, 0,1,0,1],
[1,0,0,0, 0,1,0,0, 0,0,1,0, 0,0,1,1],
[1,0,0,0, 0,1,0,0, 0,0,0,1, 0,0,1,0],
[1,0,0,0, 0,0,1,0, 0,1,0,0, 0,0,0,1],
[0,1,0,0, 1,0,0,0, 0,0,1,0, 0,0,0,1] >;
```

Compact form:

```
Generators in compact permutation form are:
6 15
0 1 2 9 14 5 6 7 8 3 11 10 13 12 4
0 1 2 10 13 5 6 7 8 11 3 9 14 4 12
0 1 2 12 11 5 6 7 8 13 14 4 3 9 10
0 1 3 2 4 5 9 10 11 6 7 8 12 13 14
0 2 1 3 4 6 5 7 8 9 12 13 10 11 14
1 0 2 3 4 5 7 6 8 10 9 11 12 14 13
-1
```

The base has length 4
The basic orbits are:
Basic orbit 0 is orbit of 0 of length 15

```
Basic orbit 1 is orbit of 1 of length 14
Basic orbit 2 is orbit of 2 of length 12
Basic orbit 3 is orbit of 3 of length 8
```

We use the following Orbiter command creates PGL(4, 2) again. The command invokes two activities. The first creates a latex report for the group in the file PGL_4_2_report.tex. The second activity exports the permutation representation in Orbiter makefile format.

Example 213

```
PGL_4_2_export:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define G -linear_group -PGL 4 F -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -export_orbiter \
▷  ▷  -end
▷  pdflatex PGL_4_2_report.tex
▷  $(OPEN) PGL_4_2_report.pdf
```

The file PGL_4_2.makefile is created:

Example 214

```
PGL_4_2_generated:
▷  $(ORBITER) -v 2 \
▷  ▷  -define gens -vector -file PGL_4_2_gens.csv -end \
▷  ▷  -define G -permutation_group \
▷  ▷  -bsgs PGL_4_2 "{\rm PGL}(4,2)" 15 20160 "0,1,2,3" 6 gens -end \
```

This command can be used to recreate the group as permutation group directly. This group will be considered again in Section 6.2 below. The permutation representation itself is stored in the file PGL_4_2_gens.csv:

```
Row,C0,C1,C2,C3,C4,C5,C6,C7,C8,C9,C10,C11,C12,C13,C14
0,0,1,2,9,14,5,6,7,8,3,11,10,13,12,4
1,0,1,2,10,13,5,6,7,8,11,3,9,14,4,12
2,0,1,2,12,11,5,6,7,8,13,14,4,3,9,10
3,0,1,3,2,4,5,9,10,11,6,7,8,12,13,14
4,0,2,1,3,4,6,5,7,8,9,12,13,10,11,14
5,1,0,2,3,4,5,7,6,8,10,9,11,12,14,13
END
```

The command

Example 215

```
L_5_3:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 3 -end \
▷  ▷  -define G -linear_group -PSL 5 F -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex PSL_5_3_report.tex
▷  $(OPEN) PSL_5_3_report.pdf
```

creates PSL$(5, 3)$ of order 237783237120 in the action on the 121 points of PG$(4, 3)$.

The command

Example 216

```
L_4_5:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 5 -end \
▷  ▷  -define G -linear_group -PSL 4 F -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex PSL_4_5_report.tex
▷  $(OPEN) PSL_4_5_report.pdf
```

creates PSL$(4, 5)$ of order 7254000000 in the action on the 156 points of PG$(3, 5)$.

The command

Example 217

```
PGL_4_5:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 5 -end \
▷  ▷  -define G -linear_group -PGL 4 F -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex PGL_4_5_report.tex
▷  $(OPEN) PGL_4_5_report.pdf
```

creates PGL$(4, 5)$ of order 29016000000 in the action on the 156 points of PG$(3, 5)$. The group PSL$(4, 5)$ is a subgroup of PGL$(4, 5)$ of index 4.

The command

Example 218

```
PGGL_3_4:
▷  $(ORBITER) -v 2 \
▷  ▷  -define G -linear_group -PGGL 3 4 -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  ▷  -report_sylow \
▷  ▷  ▷  -report_classes \
▷  ▷  -end
▷  pdflatex PGGL_3_4_report.tex
▷  $(OPEN) PGGL_3_4_report.pdf
```

creates $\mathrm{P\Gamma L}(3,4)$ of order 120960 in the action on the 21 points of $\mathrm{PG}(2,4)$. The report contains one p-Sylow subgroup for each of the primes p dividing the order of the group.

The command

Example 219

```
PGGL_3_8:
▷  $(ORBITER) -v 2 \
▷  ▷  -define G -linear_group -PGGL 3 8 -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  ▷  -report_sylow \
▷  ▷  ▷  -report_classes \
▷  ▷  -end
▷  pdflatex PGGL_3_8_report.tex
▷  $(OPEN) PGGL_3_8_report.pdf
```

creates $\mathrm{P\Gamma L}(3,8)$ of order 49448448 in the action on the 73 points of $\mathrm{PG}(2,8)$. The report contains one p-Sylow subgroup for each of the primes p dividing the order of the group.

The command

Example 220

```
AGL_1_27:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 27 -end \
▷  ▷  -define G -linear_group -AGL 1 F -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex AGL_1_27_report.tex
▷  $(OPEN) AGL_1_27_report.pdf
```

creates AGL$(1, 27)$ of order 702 in the action on the 27 points of AG$(1, 27)$.

The command

Example 221

```
AGGL_2_27:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 27 -end \
▷  ▷  -define G -linear_group -AGGL 2 F -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex AGGL_2_27_report.tex
▷  $(OPEN) AGGL_2_27_report.pdf
```

creates AΓL$(2, 27)$ of order 1117679472 in the action on the 729 points of AG$(2, 27)$.

The command

Example 222

```
SP_4_2:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define G -linear_group -GL 4 F \
▷  ▷  ▷  -symplectic_group \
▷  ▷  -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex GL_4_2_Sp_4_2_report.tex
▷  $(OPEN) GL_4_2_Sp_4_2_report.pdf
```

creates the symplectic group Sp$(4, 2)$ of order 720 acting on the 16 points of AG$(4, 2)$.

The command

Example 223

```
PSP_4_4:
▷  $(ORBITER) -v 5 \
▷  ▷  -define F -finite_field -q 4 -end \
▷  ▷  -define G -linear_group -PGL 4 F \
▷  ▷  ▷  -symplectic_group \
▷  ▷  -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
```

```
▷  pdflatex PGL_4_4_Sp_4_4_report.tex
▷  $(OPEN) PGL_4_4_Sp_4_4_report.pdf
```

creates the symplectic group $\mathrm{PSp}(4,4)$ of order 979200 acting on the 85 points of $\mathrm{PG}(3,4)$.

The command

Example 224

```
PGO_5_2:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define G -linear_group -PGO 5 F -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex PGO_5_2_report.tex
▷  $(OPEN) PGO_5_2_report.pdf
```

creates the group $\mathrm{PGO}(5,2)$ acting on the 15 points of the $Q(4,2)$ quadric. The following latex report is produced:

The Group $\mathrm{PGO}(5,2)$

The order of the group $\mathrm{PGO}(5,2)$ is 720
The group acts on a set of size 15
Strong generators for a group of order 720:

$$\begin{bmatrix} 1&0&0&0&0\\ 0&1&0&0&0\\ 0&0&1&0&0\\ 0&0&0&1&0\\ 1&0&0&1&1 \end{bmatrix}, \begin{bmatrix} 1&0&0&0&0\\ 0&1&0&0&0\\ 0&0&1&0&0\\ 1&0&0&1&1\\ 0&0&0&0&1 \end{bmatrix}, \begin{bmatrix} 1&0&0&0&0\\ 0&1&0&0&0\\ 1&1&1&0&1\\ 1&1&0&1&1\\ 0&0&0&0&1 \end{bmatrix},$$

$$\begin{bmatrix} 1&0&0&0&0\\ 0&1&0&0&0\\ 0&0&1&0&1\\ 1&1&0&1&1\\ 0&1&0&1&0 \end{bmatrix}, \begin{bmatrix} 1&0&0&0&0\\ 1&1&1&0&0\\ 0&0&1&0&0\\ 0&0&0&1&0\\ 0&0&0&0&1 \end{bmatrix}, \begin{bmatrix} 1&0&0&0&0\\ 0&1&0&0&1\\ 0&0&1&0&0\\ 0&0&1&1&0\\ 0&0&0&0&1 \end{bmatrix}$$

1,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,1,0,1,0,0,1,1,
1,0,0,0,0,0,1,0,0,0,0,0,1,0,0,1,0,0,1,1,0,0,0,0,1,
1,0,0,0,0,0,1,0,0,0,1,1,1,0,1,1,1,0,1,1,0,0,0,0,1,
1,0,0,0,0,0,1,0,0,0,0,0,1,0,1,1,1,0,1,1,0,1,0,1,0,
1,0,0,0,0,1,1,1,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,1,

1,0,0,0,0,0,1,0,0,1,0,0,1,0,0,0,0,1,1,0,0,0,0,0,1,

The Action

Group action PGO$(5, 2)$ of degree 15
We act on the following set:

0 = (0, 1, 0, 0, 0)
1 = (0, 0, 1, 0, 0)
2 = (0, 0, 0, 1, 0)
3 = (0, 1, 0, 1, 0)
4 = (0, 0, 1, 1, 0)
5 = (0, 0, 0, 0, 1)
6 = (0, 1, 0, 0, 1)
7 = (0, 0, 1, 0, 1)
8 = (0, 1, 1, 1, 1)
9 = (1, 1, 1, 0, 0)
10 = (1, 1, 1, 1, 0)
11 = (1, 1, 1, 0, 1)
12 = (1, 0, 0, 1, 1)
13 = (1, 1, 0, 1, 1)
14 = (1, 0, 1, 1, 1)

The group is a matrix group.
The base action is on projective space PG$(4, 2)$
$q = 2$
$p = 2$
$e = 1$
$n = 4$
Number of points = 31
Number of lines = 155
Number of lines on a point = 15
Number of points on a line = 3

The finite field $\mathbb{F}_2$

$Z_i = \log_\alpha(1 + \alpha^i)$

i	γ_i	$-\gamma_i$	γ_i^{-1}	$\log_\alpha(\gamma_i)$	α^i	Z_i
0	$0 = 0$	0	DNE	DNE	1	DNE
1	$1 = 1$	1	1	0	1	DNE

+	0	1
0	0	1
1	1	0

·	1
1	1

$1^0 \equiv 1$
$1^1 \equiv 1$

Base and Stabilizer Chain

Group order 720
tl=15, 8, 3, 1, 1, 2,
Base: (0, 1, 2, 3, 4, 5)
Strong generators for a group of order 720:

$$\begin{bmatrix} 1&0&0&0&0\\ 0&1&0&0&0\\ 0&0&1&0&0\\ 0&0&0&1&0\\ 1&0&0&1&1 \end{bmatrix}, \begin{bmatrix} 1&0&0&0&0\\ 0&1&0&0&0\\ 0&0&1&0&0\\ 1&0&0&1&1\\ 0&0&0&0&1 \end{bmatrix}, \begin{bmatrix} 1&0&0&0&0\\ 0&1&0&0&0\\ 1&1&1&0&1\\ 1&1&0&1&1\\ 0&0&0&0&1 \end{bmatrix},$$

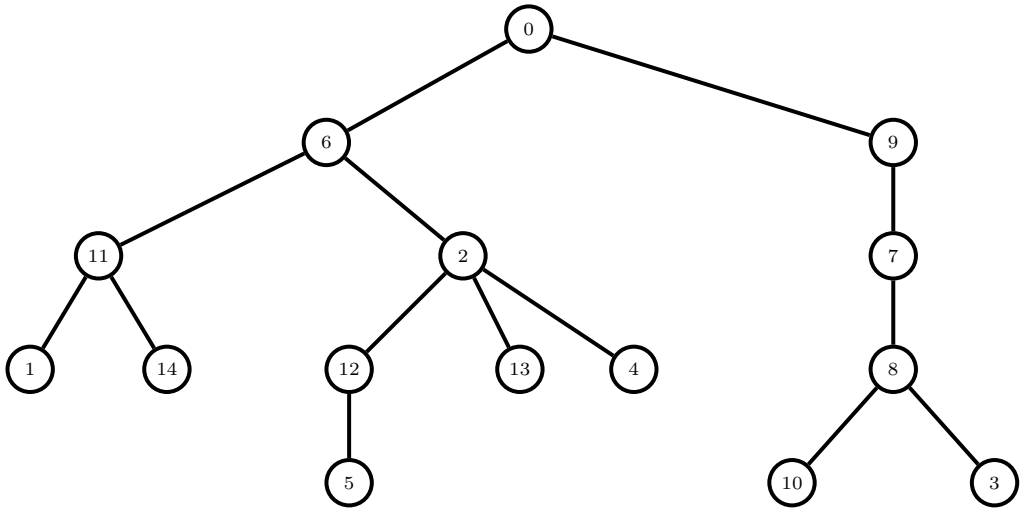
$$\begin{bmatrix} 1&0&0&0&0\\ 0&1&0&0&0\\ 0&0&1&0&1\\ 1&1&0&1&1\\ 0&1&0&1&0 \end{bmatrix}, \begin{bmatrix} 1&0&0&0&0\\ 1&1&1&0&0\\ 0&0&1&0&0\\ 0&0&0&1&0\\ 0&0&0&0&1 \end{bmatrix}, \begin{bmatrix} 1&0&0&0&0\\ 0&1&0&0&1\\ 0&0&1&0&0\\ 0&0&1&1&0\\ 0&0&0&0&1 \end{bmatrix}$$

1,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,1,0,1,0,0,1,1,
1,0,0,0,0,0,1,0,0,0,0,0,1,0,0,1,0,0,1,1,0,0,0,0,1,
1,0,0,0,0,0,1,0,0,0,1,1,1,0,1,1,1,0,1,1,0,0,0,0,1,
1,0,0,0,0,0,1,0,0,0,0,0,1,0,1,1,1,0,1,1,0,1,0,1,0,
1,0,0,0,0,1,1,1,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,1,
1,0,0,0,0,0,1,0,0,1,0,0,1,0,0,0,0,1,1,0,0,0,0,0,1,

Stabilizer chain

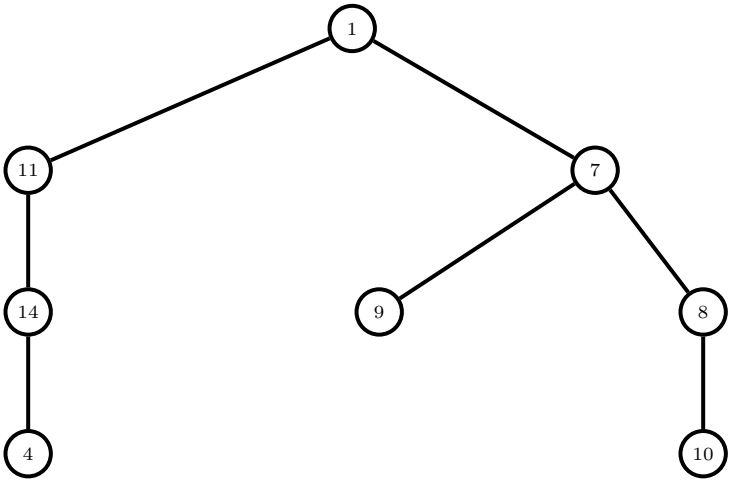
Level	Base pt	Orbit length	Subgroup order
0	0	15	720
1	1	8	48
2	2	3	6
3	3	1	2
4	4	1	2
5	5	2	2

Basic Orbit 0



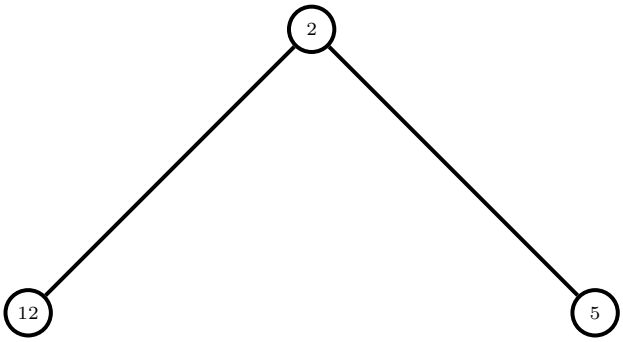
Basic orbit 0 has size 15
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14

Basic Orbit 1



Basic orbit 1 has size 8
1, 4, 7, 8, 9, 10, 11, 14

Basic Orbit 2



Basic orbit 2 has size 3
2, 5, 12

Basic Orbit 3

Basic orbit 3 has size 1
3

Basic Orbit 4

Basic orbit 4 has size 1
4

Basic Orbit 5

Basic orbit 5 has size 2
5, 12
GAP export:

```
Generators in GAP format are:
G := Group([(6, 13)(7, 14)(8, 15)(9, 12),
(3, 13)(4, 14)(5, 15)(9, 11),
(2, 12)(3, 14)(4, 13)(8, 10),
(2, 8, 9, 10, 12, 15)(3, 14, 7)(4, 13, 6)(5, 11),
(1, 10)(4, 11)(7, 12)(9, 14),
(1, 7)(3, 5)(4, 9)(10, 12)(11, 14)(13, 15)]);
```

Magma export:

Compact form:

```
Generators in compact permutation form are:
6 15
0 1 2 3 4 12 13 14 11 9 10 8 5 6 7
0 1 12 13 14 5 6 7 10 9 8 11 2 3 4
0 11 13 12 4 5 6 9 8 7 10 1 3 2 14
0 7 13 12 10 3 2 8 9 11 4 14 5 6 1
9 1 2 10 4 5 11 7 13 0 3 6 12 8 14
6 1 4 8 2 5 0 7 3 11 13 9 14 10 12
-1
```

The base has length 6
The basic orbits are:
Basic orbit 0 is orbit of 0 of length 15
Basic orbit 1 is orbit of 1 of length 8
Basic orbit 2 is orbit of 2 of length 3

```
Basic orbit 3 is orbit of 3 of length 1
Basic orbit 4 is orbit of 4 of length 1
Basic orbit 5 is orbit of 5 of length 2
```

The command

Example 225

```
PGGO_5_4:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F4 -finite_field -q 4 -end \
▷  ▷  -define G -linear_group -PGGO 5 F4 -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex PGGO_5_4_report.tex
▷  $(OPEN) PGGO_5_4_report.pdf
```

creates the group $\mathrm{P\Gamma O}(5,4)$ of order 1958400 acting on the 85 points of the $Q(4,4)$ quadric.

The command

Example 226

```
PGOp_6_2:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define G -linear_group -PGOp 6 F -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex PGOp_6_2_report.tex
▷  $(OPEN) PGOp_6_2_report.pdf
```

creates the group $\mathrm{P\Gamma O}^+(6,2)$ of order 40320 acting on the 35 points of the $Q^+(5,2)$ quadric.

The command

Example 227

```
PGOm_6_2:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define G -linear_group -PGOm 6 F -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
```

```
▷  pdflatex PGOm_6_2_report.tex
▷  $(OPEN) PGOm_6_2_report.pdf
```

creates the group $\mathrm{P\Gamma O}^-(6,2)$ of order 51840 acting on the 27 points of the $Q^-(5,2)$ quadric.

The symplectic group $\mathrm{PSp}(6,2)$ can be created using the following command:

Example 228

```
PSP_6_2:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define G -linear_group -PGL 6 F \
▷  ▷  ▷  -symplectic_group \
▷  ▷  -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex PGL_6_2_Sp_6_2_report.tex
▷  $(OPEN) PGL_6_2_Sp_6_2_report.pdf
```

The group $\mathrm{PGO}(7,2)$, isomorphic to $\mathrm{PSp}(6,2)$, can be created using the following command:

Example 229

```
PGO_7_2:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define G -linear_group -PGO 7 F -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex PGO_7_2_report.tex
▷  $(OPEN) PGO_7_2_report.pdf
```

The command

Example 230

```
NullPolarity_6_2:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define G -linear_group -PGL 6 F \
▷  ▷  ▷  -null_polarity_group \
▷  ▷  -end \
▷  ▷  -with G -do \
```

```
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex PGL_6_2_NullPolarity_6_2_report.tex
▷  $(OPEN) PGL_6_2_NullPolarity_6_2_report.pdf
```

creates the null polarity group of PG$(6, 2)$ acting on the 63 points of PG$(5, 2)$. This is the group preserving the dot product

$$\mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^{n} x_i y_i.$$

Creating Subgroups

Command	Arguments	Purpose
-Janko1		first Janko group, needs $\mathrm{PGL}(7,11)$
-monomial		subgroup of monomial matrices
-diagonal		subgroup of diagonal matrices
-null_polarity_group		null polarity group
-symplectic_group		symplectic group
-singer	k	subgroup of index k in the Singer cycle
-singer_and_frobenius	k	subgroup of index k in the Singer cycle, extended by the Frobenius automorphism of $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$
-borel_upper		Borel subgroup of upper triangular matrices
-borel_lower		Borel subgroup of lower triangular matrices
-identity_group		identity subgroup
-subgroup_from_file	f l	read subgroup from file f and give it the label l
-orthogonal	ϵ	orthogonal group $O^\epsilon(n,q)$, with $\epsilon \in \{\pm 1\}$ when n is even
-subgroup_by_generators	l o n $s_1 \ldots s_n$	Generate a subgroup from generators. The label "l" is used to denote the subgroup; o is the order of the subgroup; n is the number of generators and s_1, ..., s_n are the generators for the subgroup in vector form.

Table 6.5: Creating Subgroups

6.3 Subgroups

There are many ways to create subgroups of a group. Table 6.5 lists some commands to do so.

We start with an example of an explicit permutation group using a known base and strong generating set, using the bsgs command. Here is the cyclic group of order 13 acting on the permutation domain $[0, 12]$. The base is (0). When creating a group, we supply a label in ascii text and in tex. Then we specify the degree of the action, and the group order. After that, we specify the number of generators and the generators themselves. The labels will be used in reports about the group, for instance.

Example 231

```
GEN_C13="1,2,3,4,5,6,7,8,9,10,11,12,0"
# (0,1,2,3,4,5,6,7,8,9,10,11,12)
```

Example 232

```
C13:
▷  $(ORBITER) -v 2 \
▷  ▷  -define gens -vector -dense $(GEN_C13) -end \
▷  ▷  -define G -permutation_group \
▷  ▷  ▷  -bsgs C13 C_{13} 13 13 0 1 \
```

```
▷ ▷ ▷ ▷ gens \
▷ ▷ ▷ -end \
▷ ▷ -with G -do \
▷ ▷ -group_theoretic_activity \
▷ ▷ ▷ -export_orbiter \
▷ ▷ -end \
▷ ▷ -with G -do \
▷ ▷ -group_theoretic_activity \
▷ ▷ ▷ -export_group_table \
▷ ▷ -end \
▷ ▷ -with G -do \
▷ ▷ -group_theoretic_activity \
▷ ▷ ▷ -report \
▷ ▷ -end \
▷ ▷ -with G -do \
▷ ▷ -group_theoretic_activity \
▷ ▷ ▷ -save_elements_csv "C13_elts.csv" \
▷ ▷ -end
▷ pdflatex C13_report.tex
▷ $(OPEN) C13_report.pdf
```

The makefile variable `GEN_C13` is used to define the generator of the group, which is the cycle

$$(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12).$$

The command creates the group from the known base 0. After that, several activities are invoked. Specifically, these are group theoretic activities. They will be discussed in more detail in Section 6.5.

Let us take a closer look at the three activities performed in this example. The `-export_orbiter` command exports the group in Orbiter makefile format. The file `C13.makefile` is generated, which can be used to recreate the permutation group in an Orbiter makefile. Here is the content of the file:

Example 233

```
C13_generated:
▷ $(ORBITER) -v 2 \
▷ ▷ -define gens -vector -file C13_gens.csv -end \
▷ ▷ -define G -permutation_group \
▷ ▷ -bsgs C13 "C_{13}" 13 13 "0" 1 gens -end \
```

The activity `-report` produces a report for the cyclic group, shown below:

Stabilizer chain

Level	Base pt	Orbit length	Subgroup order
0	0	13	13

Basic Orbit 0

Basic orbit 0 has size 13
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12

The command `-save_elements_csv` creates a csv file containing all group elements. Each group element is listed one-by-one, using the list notation of permutations. The csv file `C13_elts.csv` has the following content:

```
Row,Element
0,"0,1,2,3,4,5,6,7,8,9,10,11,12"
1,"1,2,3,4,5,6,7,8,9,10,11,12,0"
2,"2,3,4,5,6,7,8,9,10,11,12,0,1"
3,"3,4,5,6,7,8,9,10,11,12,0,1,2"
4,"4,5,6,7,8,9,10,11,12,0,1,2,3"
5,"5,6,7,8,9,10,11,12,0,1,2,3,4"
6,"6,7,8,9,10,11,12,0,1,2,3,4,5"
7,"7,8,9,10,11,12,0,1,2,3,4,5,6"
8,"8,9,10,11,12,0,1,2,3,4,5,6,7"
9,"9,10,11,12,0,1,2,3,4,5,6,7,8"
10,"10,11,12,0,1,2,3,4,5,6,7,8,9"
11,"11,12,0,1,2,3,4,5,6,7,8,9,10"
12,"12,0,1,2,3,4,5,6,7,8,9,10,11"
END
```

It is possible to create a permutation group as a subgroup of the symmetric group, using the known base for the symmetric group. Because the base of the symmetric group is large, this way of creating the group is less efficient than creating the group with a known (small) base. Here is an example. We create C_{13} as a subgroup of Sym(13).

Example 234

```
C13_as_subgroup:
▷  $(ORBITER) -v 2 \
▷  ▷  -define G -permutation_group -symmetric_group 13 \
▷  ▷  ▷  -subgroup_by_generators C13 13 1 $(GEN_C13) -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -export_orbiter \
▷  ▷  -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -save_elements_csv "C13_elts.csv" \
▷  ▷  -end
▷  #pdflatex Perm13_Subgroup_C13_13_report.tex
▷  #$(OPEN) Perm13_Subgroup_C13_13_report.pdf
```

For instance, the command

Example 235

```
J1:
▷  $(ORBITER) -v 2 \
▷  ▷  -define G -linear_group -PGL 7 11 -Janko1 -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex Janko1_report.tex
▷  $(OPEN) Janko1_report.pdf
```

creates the first Janko group as a subgroup of $\mathrm{PGL}(7, 11)$.

The command

Example 236

```
PGL_3_11_singer:
▷  $(ORBITER) -v 2 \
▷  ▷  -define G -linear_group -PGL 3 11 -singer 19 -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex PGL_3_11_Singer_3_11_19_report.tex
▷  $(OPEN) PGL_3_11_Singer_3_11_19_report.pdf
```

creates a subgroup of the Singer cycle of order 7. The Singer cycle in $\mathrm{GL}(d,q)$ is a generator for a subgroup of order q^d-1. It induces an element of order $\frac{q^d-1}{q-1}$ on the associated projective geometry $\mathrm{PG}(d-1,q)$. The additional integer parameter k after the `-singer` command is used to create the subgroup of index k of the Singer cycle.

The command

Example 237

```
PGL_3_11_singer_and_frobenius:
▷ $(ORBITER) -v 2 \
▷ ▷ -define G -linear_group -PGL 3 11 -singer_and_frobenius 19 -end \
▷ ▷ -with G -do \
▷ ▷ -group_theoretic_activity \
▷ ▷ ▷ -report \
▷ ▷ -end
▷ ▷ pdflatex PGL_3_11_Singer_and_Frob3_11_19_report.tex
▷ ▷ $(OPEN) PGL_3_11_Singer_and_Frob3_11_19_report.pdf
```

creates a subgroup of index 19 of the Singer cycle of $\mathrm{PG}(2,11)$, extended by a group of order 3 that arises from the field extension $\mathbb{F}_{11}^3$ over $\mathbb{F}_{11}$. The group created by this command has order 21.

The quaternion group is a group of order 8 generated by the following matrices over $\mathbb{R}$:

$$i=\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix},\quad j=\begin{bmatrix}-1 & 1\\ 1 & 1\end{bmatrix},\quad k=\begin{bmatrix}0 & -1\\ 1 & 0\end{bmatrix}.$$

It is isomorphic to a subgroup of $\mathrm{SL}(2,3)$. The Orbiter command

Example 238

```
quaternion:
▷ $(ORBITER) -v 2 \
▷ ▷ -define G -linear_group -SL 2 3 \
▷ ▷ -subgroup_by_generators "quaternion" "8" 3 \
▷ ▷ ▷ "1,1,1,2, 2,1,1,1, 0,2,1,0" \
▷ ▷ -end \
▷ ▷ -with G -do \
▷ ▷ -group_theoretic_activity \
▷ ▷ ▷ -print_elements_tex \
▷ ▷ ▷ -report_group_table \
▷ ▷ ▷ -report \
▷ ▷ -end
▷ pdflatex SL_2_3_Subgroup_quaternion_8_report.tex
▷ $(OPEN) SL_2_3_Subgroup_quaternion_8_report.pdf
```

creates the group. The command produces the list of group elements shown below.

Element 0 / 8 of order 1:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$(0)(1)(2)(3)(4)(5)(6)(7)(8)$$

Element 1 / 8 of order 4:

$$\begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}$$

$$(0)(1,5,2,7)(3,4,6,8)$$

Element 2 / 8 of order 2:

$$\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

$$(0)(1,2)(3,6)(4,8)(5,7)$$

Element 3 / 8 of order 4:

$$\begin{bmatrix} 1 & 2 \\ 2 & 2 \end{bmatrix}$$

$$(0)(1,7,2,5)(3,8,6,4)$$

Element 4 / 8 of order 4:

$$\begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}$$

$$(0)(1,4,2,8)(3,7,6,5)$$

Element 5 / 8 of order 4:

$$\begin{bmatrix} 0 & 1 \\ 2 & 0 \end{bmatrix}$$

$$(0)(1,3,2,6)(4,5,8,7)$$

Element 6 / 8 of order 4:

$$\begin{bmatrix} 2 & 2 \\ 2 & 1 \end{bmatrix}$$

$$(0)(1,8,2,4)(3,5,6,7)$$

Element 7 / 8 of order 4:

$$\begin{bmatrix} 0 & 2 \\ 1 & 0 \end{bmatrix}$$

$$(0)(1,6,2,3)(4,7,8,5)$$

The group table is created as csv file:

```
Row,C0,C1,C2,C3,C4,C5,C6,C7
0,0,1,2,3,4,5,6,7
1,1,2,3,0,5,6,7,4
2,2,3,0,1,6,7,4,5
3,3,0,1,2,7,4,5,6
4,4,7,6,5,2,1,0,3
5,5,4,7,6,3,2,1,0
```

```
6,6,5,4,7,0,3,2,1
7,7,6,5,4,1,0,3,2
END
```

The group of the cube can be created over the field $\mathbb{F}_3$:

Example 239

```
cube_group:
▷  $(ORBITER) -v 2 \
▷  ▷  -define gens -vector -dense \
▷  ▷  ▷  "0,1,0,2,0,0,0,0,1, \
▷  ▷  ▷  0,0,1,0,1,0,2,0,0, \
▷  ▷  ▷  2,0,0,0,1,0,0,0,1" \
▷  ▷  -end \
▷  ▷  -define G -linear_group -GL 3 3 \
▷  ▷  -subgroup_by_generators "cube" "48" 3 \
▷  ▷  ▷  gens \
▷  ▷  -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -print_elements_tex \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex GL_3_3_Subgroup_cube_48_report.tex
▷  $(OPEN) GL_3_3_Subgroup_cube_48_report.pdf
```

The tetrahedral subgroup can be created as well:

Example 240

```
tetra_group:
▷  $(ORBITER) -v 3 \
▷  ▷  -define G -linear_group -GL 3 3 \
▷  ▷  -subgroup_by_generators "tetra" "12" 2 \
▷  ▷  ▷  "0,1,0,0,0,1,1,0,0, 0,0,1,2,0,0,0,2,0" \
▷  ▷  -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -print_elements_tex \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex GL_3_3_Subgroup_tetra_12_report.tex
▷  $(OPEN) GL_3_3_Subgroup_tetra_12_report.pdf
```

The Hesse group of order 216 extended by the automorphism group of the field can be created in PG$(3, 4)$

Example 241

```
GENERATORS_HESSE_GROUP="\
3000300030 \
2000201230 \
1000100111 \
1000220200 \
1002312010 \
0331003211 \
2200011331"
```

Example 242

```
Hesse_group:
▷  $(ORBITER) -v 3 \
▷  ▷  -define gens -vector -compact \
▷  ▷  ▷  $(GENERATORS_HESSE_GROUP) \
▷  ▷  -end \
▷  ▷  -define G -linear_group -PGGL 3 4 \
▷  ▷  -subgroup_by_generators "Hesse" "432" 7 gens \
▷  ▷  -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -print_elements_tex \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex PGGL_3_4_Subgroup_Hesse_432_report.tex
▷  $(OPEN) PGGL_3_4_Subgroup_Hesse_432_report.pdf
```

The group has order 432.

The Weyl group of type E_8 can be generated as a subgroup of $\mathrm{GL}(8,3)$ using the following command:

Example 243

```
GENERATORS_WEYL_GROUP_E8="\
-1,-1,-1,-1,0,0,0,0, \
0,0,0,1,0,0,0,0, \
1,0,0,0,0,0,0,0, \
0,0,1,0,0,0,0,0, \
0,1,0,1,1,0,0,0, \
0,0,0,0,0,1,0,0, \
0,0,0,0,0,0,1,0, \
0,0,0,0,0,0,0,1, \
-1,0,-1,-1,-1,-1,-1,-1, \
0,1,0,1,1,1,1,1, \
1,0,0,0,0,0,0,0, \
0,0,1,0,0,0,0,0, \
0,0,0,1,0,0,0,0, \
0,0,0,0,1,0,0,0, \
0,0,0,0,0,1,0,0, \
```

```
0,0,0,0,0,0,1,0"
```

Example 244

```
Weyl_E8:
▷  $(ORBITER) -v 3 \
▷  ▷  -define gens -vector -dense \
▷  ▷  ▷  $(GENERATORS_WEYL_GROUP_E8) \
▷  ▷  -end \
▷  ▷  -define G -linear_group -GL 8 3 \
▷  ▷  -subgroup_by_generators \
▷  ▷  ▷  "Weyl_E8" "696729600" 2 gens  \
▷  ▷  -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex GL_8_3_Subgroup_Weyl_E8_696729600_report.tex
▷  $(OPEN) GL_8_3_Subgroup_Weyl_E8_696729600_report.pdf
```

A latex report is generated in the file `GL_8_3_Subgroup_Weyl_E8_696729600_report.tex`. This command uses generators found by Gabi Nebe:

`http://www.math.rwth-aachen.de/~Gabriele.Nebe/LATTICES/E8.b.html`

We can test if a group is a subgroup of another. In the following example, we test whether $\mathrm{PGO}^+(6,2)$ is a subgroup of $\mathrm{PSp}(6,2)$. The fact that it is depends on the choice of forms associated with the groups and on the fact that the characteristic is two.

Example 245

```
test_subgroup:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define G1 -linear_group -PGOp 6 F -end \
▷  ▷  -define G2 -linear_group -PGL 6 F \
▷  ▷  ▷  -symplectic_group \
▷  ▷  -end \
▷  ▷  -with G1 -and G2 -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -is_subgroup_of \
▷  ▷  -end
```

Since the subgroup index is small (36), we create a set of coset representatives using the following command:

Example 246

```
coset_reps:
▷  $(ORBITER) -v 2 \
```

```
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define G1 -linear_group -PGOp 6 F -end \
▷  ▷  -define G2 -linear_group -PGL 6 F \
▷  ▷  ▷  -symplectic_group \
▷  ▷  -end \
▷  ▷  -with G1 -and G2 -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -coset_reps \
▷  ▷  -end
▷  pdflatex PGOp_6_2_coset_reps.tex
▷  $(OPEN) PGOp_6_2_coset_reps.pdf
```

The coset representatives are written to a csv file. The (shortened) list of coset representatives in latex is:

coset 0 / 36:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 100000 \\ 010000 \\ 001000 \\ 000100 \\ 000010 \\ 000001 \end{bmatrix}$$

$\vdots$

coset 35 / 36:

$$\begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 101110 \\ 101000 \\ 011101 \\ 011111 \\ 100010 \\ 110100 \end{bmatrix}$$

The following command reads the vector of coset representatives from the file just created.

Example 247

```
coset_reps_read:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define G1 -linear_group -PGOp 6 F -end \
▷  ▷  -define G2 -linear_group -PGL 6 F \
▷  ▷  ▷  -symplectic_group \
▷  ▷  -end \
▷  ▷  -define CR -vector_ge -action G2 \
▷  ▷  ▷  -read_csv \
▷  ▷  ▷  PGOp_6_2_coset_reps.csv Element \
▷  ▷  -end
```

The following command creates the group $\mathrm{PSp}(6,2)$ and computes a point stabilizer subgroup.

Example 248

```
SP_6_2_point_stab_subgroup:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define G -linear_group -PGL 6 F \
▷  ▷  ▷  -symplectic_group \
▷  ▷  -end \
▷  ▷  -define G0 -modified_group -from G \
▷  ▷  ▷  -point_stabilizer 0 \
▷  ▷  -end \
▷  ▷  -with G0 -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex PGL_6_2_Sp_6_2_Stab0_report.tex
▷  $(OPEN) PGL_6_2_Sp_6_2_Stab0_report.pdf
```

The next command creates the group $\mathrm{PGO}^+(6,2)$ and produces a report.

Example 249

```
PGOp_6_2_report:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define G -linear_group -PGOp 6 F -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex PGOp_6_2_report.tex
▷  $(OPEN) PGOp_6_2_report.pdf
```

The next command creates the group $\mathrm{PGO}^-(6,2)$ and produces a report.

Example 250

```
PGOm_6_2_report:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define G -linear_group -PGOm 6 F -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex PGOm_6_2_report.tex
▷  $(OPEN) PGOm_6_2_report.pdf
```

The next command creates the group $\mathrm{PGO}^-(6,2)$ and exports the group to Magma.

Example 251

```
PGOm_6_2_export_magma:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define G -linear_group -PGOm 6 F -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -export_magma \
▷  ▷  -end
```

The next command creates the group $\mathrm{PGO}^+(6,2)$ and computes a point stabilizer subgroup.

Example 252

```
PGOp_6_2_point_stab_subgroup:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define G -linear_group -PGOp 6 F -end \
▷  ▷  -define G0 -modified_group -from G \
▷  ▷  ▷  -point_stabilizer 0 \
▷  ▷  -end \
▷  ▷  -with G0 -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex PGOp_6_2_report.tex
▷  $(OPEN) PGOp_6_2_report.pdf
```

We can save the generators of $\mathrm{PGO}^+(6,2)$ from the report to a makefile variable:

Example 253

```
PGOp_6_2_GENS="\
1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,1,0, \
1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,1,1,1,1,0,0,1,0,0,1,0,0,1,0,1,0, \
1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,1,0,0,0,1,0,0,1,0,0,0,0,0,1,0, \
1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,0,0,1,0,0,1,0,0,0,1,0,0, \
1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,1,0,1,0,0,0,0,0,1,0,0,0,0,1,0,1, \
1,0,0,0,0,0,0,1,1,1,1,1,1,0,0,0,1,0,1,0,0,0,0,0,1,1,0,1,0,0,0,1,0,0,1,0,0, \
1,0,0,0,0,0,0,1,0,1,0,0,1,0,1,0,0,1,0,0,0,1,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1, \
0,1,0,1,1,0,0,0,1,0,0,0,0,1,0,0,0,0,0,1,0,1,0,1,0,0,0,0,0,0,1,0,0,1,1,0,0,1, \
0,0,1,1,1,1,1,0,1,0,0,0,0,0,1,1,1,1,1,1,1,1,1,0,0,0,1,0,1,0,1,0,1,0,1,1,0, \
0,0,0,0,1,0,1,0,1,1,1,1,1,0,1,0,1,0,1,0,0,1,1,0,1,0,0,0,0,0,0,1,1,1,1,1,0,"
```

Using this variable, we can use the following command to recreate the group as a subgroup of $\mathrm{PGL}(6,2)$ and write a report:

Example 254

```
PGOp_6_2_linear:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define G -linear_group -PGL 6 F \
▷  ▷  ▷  -subgroup_by_generators PGOp_6_2 \
▷  ▷  ▷  ▷  40320 10 $(PGOp_6_2_GENS) \
▷  ▷  -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex PGL_6_2_Subgroup_PGOp_6_2_40320_report.tex
▷  $(OPEN) PGL_6_2_Subgroup_PGOp_6_2_40320_report.pdf
```

In the following command, we compute a point stabilizer subgroup:

Example 255

```
PGOp_6_2_linear_stab_6:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define G -linear_group -PGL 6 F \
▷  ▷  ▷  -subgroup_by_generators PGOp_6_2 \
▷  ▷  ▷  ▷  40320 10 $(PGOp_6_2_GENS) \
▷  ▷  -end \
▷  ▷  -define G6 -modified_group -from G \
▷  ▷  ▷  -point_stabilizer 6 \
▷  ▷  -end \
▷  ▷  -with G6 -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex PGL_6_2_Subgroup_PGOp_6_2_40320_Stab6_report.tex
▷  $(OPEN) PGL_6_2_Subgroup_PGOp_6_2_40320_Stab6_report.pdf
```

Group Modifications

Command	Arguments	Purpose
-restricted_action	S label-tex	restricted action on the set S using the given tex-label.
-on_k_subspaces	k	induced action on k dimensional subspaces
-on_k_subsets	k	induced action on k subsets.
-on_wedge_product		action on the exterior square
-create_special_subgroup		Create the subgroup of special matrices.
-point_stabilizer	pt	Create the subgroup which stabilizes pt.
-projectivity_subgroup		Create the projectivity subgroup of a semilinear group.
-subfield_subgroup	k	Create the subfield subgroup w.r.t. $\mathbb{F}_s$. Here $k = \frac{q-1}{s-1}$.
-action_on_self_by_ right_multiplication		Create the action on the group itself by right multiplication.
-direct_product	list order gens	Create the direct product action of the two actions given in list (comma separated). Then create the subgroup of the given order from the given generators.
-polarity_extension	G0 P	Create the polarity extension of G0. P is the underlying projective space on which G0 acts.
-from	label	Specify the input group action (this option can be repeated).

Table 6.6: Group Modifications

6.4 New Actions from Old

It is possible to create new groups and group actions from old. This is done using the command -modified_group. Table 6.6 lists Orbiter commands that are available.
Let us create the action of Sym(4) on pairs:

Example 256

```
Symmetric_4_on_pairs:
▷  $(ORBITER) -v 3 \
▷  ▷  -define G -permutation_group -symmetric_group 4 -end \
▷  ▷  -define G_on_2subsets -modified_group -from G \
▷  ▷  ▷  -on_k_subsets 2 \
▷  ▷  -end \
▷  ▷  -with G_on_2subsets -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex Sym_4_OnSubsets_2_report.tex
▷  $(OPEN) Sym_4_OnSubsets_2_report.pdf
```

The next example creates a group in product action. Let C_n be the cyclic group of order n. The group we wish to create is C_{21}. Because of $C_{21} \simeq C_3 \times C_7$, there is a product action on a grid of type 3×7. To create

the cyclic group, we use the translation complement in the respective one-dimensional affine groups. This works because the numbers are prime. So, we create the group as a subgroup of $\mathrm{AGL}(1,3) \times \mathrm{AGL}(1,7)$. Recall the encoding of affine elements: The affine map

$$f(\mathbf{x}) = A\mathbf{x} + \mathbf{b}$$

with

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

is encoded in row-major ordering, with the entries of $\mathbf{b}$ following the entries of A

$$(a_{11}, a_{12}, \ldots, a_{nn}, b_1, \ldots, b_n).$$

The cyclic group is generated by the mapping

$$f : \mathbb{F}_3 \times \mathbb{F}_7 \to \mathbb{F}_3 \times \mathbb{F}_7,\ f(x,y) = (x+1, y+1)\,,$$

which encodes as

$$(1,1,1,1)$$

The command to create the product action and the subgroup of order 21 is:

Example 257

```
direct_product_AGL:
▷  $(ORBITER) -v 4 \
▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
▷  ▷  -define G1 -linear_group -AGL 1 3 \
▷  ▷  ▷  -end \
▷  ▷  -define G2 -linear_group -AGL 1 7 \
▷  ▷  ▷  -end \
▷  ▷  -define G -modified_group -direct_product "G1,G2" \
▷  ▷  ▷  "21" "1,1,1,1" \
▷  ▷  -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex product_AGL_1_3_AGL_1_7_report.tex
▷  $(OPEN) product_AGL_1_3_AGL_1_7_report.pdf
```

The polarity group $\mathrm{P\Gamma L}^*(n,q)$ arises from $\mathrm{P\Gamma L}(n,q)$ by means of extending by the polarity map. The degree of extension is two. The next command creates the group $\mathrm{P\Gamma L}^*(3,4)$ of order 241920:

Example 258

```
PGGLstar_3_4:
▷  $(ORBITER) -v 6 \
▷  ▷  -define F -finite_field -q 4 -end \
▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
```

```
▷  ▷  -define G0 -linear_group -PGGL 3 F -end \
▷  ▷  -define G -modified_group -polarity_extension \
▷  ▷  ▷  G0 P \
▷  ▷  -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex PGGL_3_4_polarity_ext_report.tex
▷  $(OPEN) PGGL_3_4_polarity_ext_report.pdf
```

Let us create a point-stabilizer subgroup. Continuing an example from Section 6.1, we consider the Mathieu group M_{12}, with its point stabilizer subgroup M_{11} of order 7920. To form the point stabilizer, we utilize a group modification. The following command also produces a report for the subgroup.

Example 259

```
Group_M11:
▷  $(ORBITER) -v 2 \
▷  ▷  -define gens -vector -dense $(GENERATORS_M12) -end \
▷  ▷  -define G -permutation_group -symmetric_group 12 \
▷  ▷  ▷  -subgroup_by_generators M12 95040 2 gens -end \
▷  ▷  -define G0 -modified_group -from G \
▷  ▷  ▷  -point_stabilizer 0 \
▷  ▷  -end \
▷  ▷  -with G0 -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex Sym_12_Subgroup_M12_95040_Stab0_report.tex
▷  $(OPEN) Sym_12_Subgroup_M12_95040_Stab0_report.pdf
```

In the following example, we will demonstrate two types of induced actions. One is the action induced on k-dimensional subspaces. The second is the restricted action on an invarant subset. The example we show is related to cubic surfaces. At first, we create the Eckardt surface in PG$(3, 13)$ from the arc

$$\{0, 1, 2, 3, 43, 113\}.$$

Then we export the set of 45 tritangent planes to file and we produce a report about the surface and its automorphism group. The next command creates the stabilizer of the surface from the generators given in the report, creates the induced action on planes, and restricts the action to the 45 tritangent planes stored in the file. The command sequence includes a makefile variable for the generators of the stabilizer of the surface:

Example 260

```
SURFACE_Q13_STAB_GENS="1,0,0,0,9,12,0,0,10,0,12,0,9,0,0,12, \
1,0,0,0,0,12,12,6,6,0,0,7,1,0,2,0, \
0,1,1,7,3,9,9,11,2,10,10,3,9,9,1,11"
```

Example 261

```
surface_q13_Eckardt_on_tritangent_planes:
▷  $(ORBITER) -v 2 \
▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
▷  ▷  -draw_options -embedded -end \
▷  ▷  -define F -finite_field -q 13 -end \
▷  ▷  -define gens -vector -field F \
▷  ▷  ▷  -dense $(SURFACE_Q13_STAB_GENS) \
▷  ▷  -end \
▷  ▷  -define TriP -set -file \
▷  ▷  ▷  surface_arc_lifting_trihedral_q13_arc_0_1_2_3_43_113_tritangent_planes.csv
\
▷  ▷  -end \
▷  ▷  -define G -linear_group -PGL 4 F \
▷  ▷  ▷  -subgroup_by_generators "stab" \
▷  ▷  ▷  ▷  24 3 gens \
▷  ▷  ▷  -end \
▷  ▷  -define G_on_planes -modified_group -from G \
▷  ▷  ▷  -on_k_subspaces 3 \
▷  ▷  -end \
▷  ▷  -define Gr -modified_group -from G_on_planes \
▷  ▷  ▷  -restricted_action TriP TriP \
▷  ▷  -end \
▷  ▷  -with Gr -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end \
▷  ▷  -define Orb -orbits -group Gr \
▷  ▷  ▷  -on_points \
▷  ▷  -end \
▷  ▷  -with Orb -do -orbits_activity \
▷  ▷  ▷  -report \
▷  ▷  -end \
▷  ▷  -with Orb -do -orbits_activity \
▷  ▷  ▷  -draw_tree 0 \
▷  ▷  -end \
▷  ▷  -with Orb -do -orbits_activity \
▷  ▷  ▷  -draw_tree 1 \
▷  ▷  -end \
▷  ▷  -with Orb -do -orbits_activity \
▷  ▷  ▷  -stabilizer 36 \
▷  ▷  -end
▷  pdflatex PGL_4_13_Gr_4_3_res_TriP_orbits_report.tex
▷  $(OPEN) PGL_4_13_Gr_4_3_res_TriP_orbits_report.pdf
```

Let us create the stabilizer of the Hirschfeld surface, isomorphic to the Weyl group of type E_6, of order 51840. The group is created as a subgroup of $\mathrm{P\Gamma L}(4,4)$. The following command computes the projectivity subgroup, of order 25920, and produces a report.

Example 262

```
Hirschfeld_stab_projectivity_group:
▷  $(ORBITER) -v 4 \
▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
▷  ▷  -define G -linear_group -PGGL 4 4 \
▷  ▷  ▷  -subgroup_by_generators "Hirschfeld_Stab" \
▷  ▷  ▷  51840 6 $(HIRSCHFELD_STAB_GENERATORS) \
▷  ▷  ▷  -end \
▷  ▷  -define G0 -modified_group -from G \
▷  ▷  ▷  -projectivity_subgroup \
▷  ▷  -end \
▷  ▷  -with G0 -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex PGGL_4_4_Subgroup_Hirschfeld_Stab_51840_ProjectivitySubgroup_report.te
x
▷  $(OPEN) PGGL_4_4_Subgroup_Hirschfeld_Stab_51840_ProjectivitySubgroup_report.pdf
```

In the next example, we consider the action of

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 2 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

in PGL$(4, 8)$ on the set of planes through a the fixed line 0. The Orbiter ranks of the set of planes through the line have been computed in Section 4.4. They are

$$0, 8, 1, 6, 4, 3, 7, 2, 5,$$

and are given as the first argument to the `apply` command. The second argument is the coding of the group element.

Example 263

```
on_planes:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 8 -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷  ▷  -define G -linear_group -PGL 4 F -end \
▷  ▷  -define G_on_planes -modified_group -from G \
▷  ▷  ▷  -on_k_subspaces 3 \
▷  ▷  -end \
▷  ▷  -with G_on_planes -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -apply "0,8,1,6,4,3,7,2,5" \
▷  ▷  ▷  "1,0,0,0, 0,1,0,0, 0,0,0,2, 0,0,1,1" \
▷  ▷  -end
▷  pdflatex PGL_4_8_Gr_4_3_apply.tex
```

```
▷  $(OPEN) PGL_4_8_Gr_4_3_apply.pdf
```

The apply command computes the permutation representation of the group element. In order to create the group acting on the planes, we use a restricted action. At first, we create the cyclic group of order 9 as a subgroup of PGL(4, 8). Then we create the action on planes and restrict the action to the 9 planes through the line that we fix. Here is the command:

Example 264

```
on_planes_restricted:
▷  $(ORBITER) -v 2 \
▷  ▷  -define planes -vector -dense "0,8,1,6,4,3,7,2,5" -end \
▷  ▷  -define F -finite_field -q 8 -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷  ▷  -define G -linear_group -PGL 4 F \
▷  ▷  ▷  -subgroup_by_generators "cyclic" \
▷  ▷  ▷  9 1 "1,0,0,0, 0,1,0,0, 0,0,0,2, 0,0,1,1" \
▷  ▷  ▷  -end \
▷  ▷  -define G_on_planes -modified_group -from G \
▷  ▷  ▷  -on_k_subspaces 3 \
▷  ▷  -end \
▷  ▷  -define Gr -modified_group -from G_on_planes \
▷  ▷  ▷  -restricted_action planes planes \
▷  ▷  -end \
▷  ▷  -with Gr -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex PGL_4_8_Gr_4_3_res_planes_report.tex
▷  $(OPEN) PGL_4_8_Gr_4_3_res_planes_report.pdf
```

Table 6.7 lists specific actions that can be requested when creating a linear group.
Let us consider an example of a tensor product action. For instance, the command

Example 265

```
T3_on_tensors:
▷  $(ORBITER) -v 2 \
▷  ▷  -define G \
▷  ▷  -linear_group -GL_d_q_wr_Sym_n 2 2 3 \
▷  ▷  ▷  -on_tensors -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex GL_2_2_wreath_Sym3_report.tex
▷  $(OPEN) GL_2_2_wreath_Sym3_report.pdf
▷
```

Creating Special Actions

Command	Arguments	Purpose
-wedge_detached		action on the exterior square. Unlike -wedge, this command does not establish the homomorphism to the original group. Instead, the group is created as subgroup of the larger general linear group.
-PGL2OnConic		induced action of $\mathrm{PGL}(2,q)$ on the conic in the plane $\mathrm{PG}(2,q)$
-subfield_structure_action	s	action by field reduction to the subfield of index s
-on_tensors		induced action of $\mathrm{GL}(d,q) \wr \mathrm{Sym}(n)$ on the tensor space
-on_rank_one_tensors		induced action of $\mathrm{GL}(d,q) \wr \mathrm{Sym}(n)$ on the tensor space

Table 6.7: Creating Special Actions

creates the group $\mathrm{GL}(2,2) \wr \mathrm{Sym}(3)$ acting on the 255 elements of $\mathrm{PG}(7,2)$ which are identified with the tensors of type $(2,2,2)$ over $\mathbb{F}_2$. Elements of this group are denoted in the notation of the semidirect product. A vector of elements in the linear group is followed by a permutation of the components.

The Group $\mathrm{GL}(2,2) \wr \mathrm{Sym}(3)$

The order of the group $\mathrm{GL}(2,2) \wr \mathrm{Sym}(3)$ is 1296
The group acts on a set of size 255

The Action

Group action $\mathrm{GL}(2,2) \wr \mathrm{Sym}(3)$res255 of degree 255

Base and Stabilizer Chain

Group order 1296
tl=3, 2, 1, 3, 2, 3, 2, 3, 2,
Strong generators for a group of order 1296.

$$\left(\begin{bmatrix}1&0\\0&1\end{bmatrix}\begin{bmatrix}1&0\\0&1\end{bmatrix}\begin{bmatrix}1&0\\1&1\end{bmatrix};\ id\right),\left(\begin{bmatrix}1&0\\0&1\end{bmatrix}\begin{bmatrix}1&0\\0&1\end{bmatrix}\begin{bmatrix}0&1\\1&0\end{bmatrix};\ id\right),\left(\begin{bmatrix}1&0\\0&1\end{bmatrix}\begin{bmatrix}1&0\\1&1\end{bmatrix}\begin{bmatrix}1&0\\0&1\end{bmatrix};\ id\right),$$
$$\left(\begin{bmatrix}1&0\\0&1\end{bmatrix}\begin{bmatrix}0&1\\1&0\end{bmatrix}\begin{bmatrix}1&0\\0&1\end{bmatrix};\ id\right),\left(\begin{bmatrix}1&0\\1&1\end{bmatrix}\begin{bmatrix}1&0\\0&1\end{bmatrix}\begin{bmatrix}1&0\\0&1\end{bmatrix};\ id\right),\left(\begin{bmatrix}0&1\\1&0\end{bmatrix}\begin{bmatrix}1&0\\0&1\end{bmatrix}\begin{bmatrix}1&0\\0&1\end{bmatrix};\ id\right),$$
$$\left(\begin{bmatrix}1&0\\0&1\end{bmatrix}\begin{bmatrix}1&0\\0&1\end{bmatrix}\begin{bmatrix}1&0\\0&1\end{bmatrix};\ (1,2)\right),\left(\begin{bmatrix}1&0\\0&1\end{bmatrix}\begin{bmatrix}1&0\\0&1\end{bmatrix}\begin{bmatrix}1&0\\0&1\end{bmatrix};\ (0,1)\right)$$

0,1,2,1,0,0,1,1,0,0,1,1,0,1,1,
0,1,2,1,0,0,1,1,0,0,1,0,1,1,0,

0,1,2,1,0,0,1,1,0,1,1,1,0,0,1,
0,1,2,1,0,0,1,0,1,1,0,1,0,0,1,
0,1,2,1,0,1,1,1,0,0,1,1,0,0,1,
0,1,2,0,1,1,0,1,0,0,1,1,0,0,1,
0,2,1,1,0,0,1,1,0,0,1,1,0,0,1,
1,0,2,1,0,0,1,1,0,0,1,1,0,0,1,

Stabilizer chain

Level	Base pt	Orbit length	Subgroup order
0	0	3	1296
1	1	2	432
2	2	1	216
3	4	3	216
4	5	2	72
5	8	3	36
6	9	2	12
7	12	3	6
8	13	2	2

It is also possible to restrict the action on all rank-one tensors, as the following example shows:

Example 266

```
T3r1:
▷  $(ORBITER) -v 4 \
▷  ▷  -define G \
▷  ▷  -linear_group -GL_d_q_wr_Sym_n 2 2 3 \
▷  ▷  ▷  -on_rank_one_tensors -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex GL_2_2_wreath_Sym3_report.tex
▷  $(OPEN) GL_2_2_wreath_Sym3_report.pdf
```

This creates an action of degree 27. The Orbiter report shows all points in the permutation domain.

The Group $\mathrm{GL}(2,2) \wr \mathrm{Sym}(3)$

The order of the group $\mathrm{GL}(2,2) \wr \mathrm{Sym}(3)$ is 1296
The group acts on a set of size 27

The Action

Group action $\mathrm{GL}(2,2) \wr \mathrm{Sym}(3)$res27 of degree 27
We act on the following set:

0 = (1, 0, 0, 0, 0, 0, 0, 0)
1 = (0, 1, 0, 0, 0, 0, 0, 0)
2 = (1, 1, 0, 0, 0, 0, 0, 0)
3 = (0, 0, 1, 0, 0, 0, 0, 0)
4 = (0, 0, 0, 1, 0, 0, 0, 0)
5 = (0, 0, 1, 1, 0, 0, 0, 0)
6 = (1, 0, 1, 0, 0, 0, 0, 0)
7 = (0, 1, 0, 1, 0, 0, 0, 0)
8 = (1, 1, 1, 1, 0, 0, 0, 0)
9 = (0, 0, 0, 0, 1, 0, 0, 0)
10 = (0, 0, 0, 0, 0, 1, 0, 0)
11 = (0, 0, 0, 0, 1, 1, 0, 0)
12 = (0, 0, 0, 0, 0, 0, 1, 0)
13 = (0, 0, 0, 0, 0, 0, 0, 1)
14 = (0, 0, 0, 0, 0, 0, 1, 1)
15 = (0, 0, 0, 0, 1, 0, 1, 0)
16 = (0, 0, 0, 0, 0, 1, 0, 1)
17 = (0, 0, 0, 0, 1, 1, 1, 1)
18 = (1, 0, 0, 0, 1, 0, 0, 0)
19 = (0, 1, 0, 0, 0, 1, 0, 0)
20 = (1, 1, 0, 0, 1, 1, 0, 0)
21 = (0, 0, 1, 0, 0, 0, 1, 0)
22 = (0, 0, 0, 1, 0, 0, 0, 1)
23 = (0, 0, 1, 1, 0, 0, 1, 1)
24 = (1, 0, 1, 0, 1, 0, 1, 0)
25 = (0, 1, 0, 1, 0, 1, 0, 1)
26 = (1, 1, 1, 1, 1, 1, 1, 1)

Base and Stabilizer Chain

Group order 1296
tl=3, 2, 1, 3, 2, 3, 2, 3, 2,
Strong generators for a group of order 1296:

$$\left(\begin{bmatrix}1&0\\0&1\end{bmatrix}\begin{bmatrix}1&0\\0&1\end{bmatrix}\begin{bmatrix}1&0\\1&1\end{bmatrix};\ id\right),\left(\begin{bmatrix}1&0\\0&1\end{bmatrix}\begin{bmatrix}1&0\\0&1\end{bmatrix}\begin{bmatrix}0&1\\1&0\end{bmatrix};\ id\right),\left(\begin{bmatrix}1&0\\0&1\end{bmatrix}\begin{bmatrix}1&0\\1&1\end{bmatrix}\begin{bmatrix}1&0\\0&1\end{bmatrix};\ id\right),$$
$$\left(\begin{bmatrix}1&0\\0&1\end{bmatrix}\begin{bmatrix}0&1\\1&0\end{bmatrix}\begin{bmatrix}1&0\\0&1\end{bmatrix};\ id\right),\left(\begin{bmatrix}1&0\\1&1\end{bmatrix}\begin{bmatrix}1&0\\0&1\end{bmatrix}\begin{bmatrix}1&0\\0&1\end{bmatrix};\ id\right),\left(\begin{bmatrix}0&1\\1&0\end{bmatrix}\begin{bmatrix}1&0\\0&1\end{bmatrix}\begin{bmatrix}1&0\\0&1\end{bmatrix};\ id\right),$$
$$\left(\begin{bmatrix}1&0\\0&1\end{bmatrix}\begin{bmatrix}1&0\\0&1\end{bmatrix}\begin{bmatrix}1&0\\0&1\end{bmatrix};\ (1,2)\right),\left(\begin{bmatrix}1&0\\0&1\end{bmatrix}\begin{bmatrix}1&0\\0&1\end{bmatrix}\begin{bmatrix}1&0\\0&1\end{bmatrix};\ (0,1)\right)$$

0,1,2,1,0,0,1,1,0,0,1,1,0,1,1,
0,1,2,1,0,0,1,1,0,0,1,0,1,1,0,
0,1,2,1,0,0,1,1,0,1,1,1,0,0,1,
0,1,2,1,0,0,1,0,1,1,0,1,0,0,1,
0,1,2,1,0,1,1,1,0,0,1,1,0,0,1,
0,1,2,0,1,1,0,1,0,0,1,1,0,0,1,
0,2,1,1,0,0,1,1,0,0,1,1,0,0,1,
1,0,2,1,0,0,1,1,0,0,1,1,0,0,1,

Orbiter supports the induced action on the wedge product by means of a modified group command. The following command shows how the induced action of $\mathrm{PGL}(4,2)$ on the wedge product is created:

Example 267

```
PGL_4_2_wedge:
▷  $(ORBITER) -v 3 \
```

```
▷  ▷  -define G -linear_group -PGL 4 2 -end \
▷  ▷  -define G2 -modified_group -from G \
▷  ▷  ▷  -on_wedge_product \
▷  ▷  -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex PGL_4_2_report.tex
▷  $(OPEN) PGL_4_2_report.pdf
```

This creates a matrix group of order 20160 and dimension 4. The representation is only evaluated when needed, for instance when acting on elements of the wedge product.

A different way of creating wedge actions is by applying an attribute during the creation of the group. This has the effect that the group is created in the wedge representation right away. The following example createds the representation of PGL$(4, 2)$ in the wedge product action.

Example 268

```
PGL_4_2_wd:
▷  $(ORBITER) -v 3 \
▷  ▷  -define G -linear_group -PGL 4 2 -wedge_detached -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex PGL_6_2_Wedge_4_2_detached_report.tex
▷  $(OPEN) PGL_6_2_Wedge_4_2_detached_report.pdf
```

This creates a matrix group of order 20160 and dimension 10.

6.5 Group Theoretic Activities

Once a group has been created as in Sections 6.1 and 6.2, a group theoretic activity can be performed. For this purpose, Orbiter provides the `-group_theoretic_activity` option. Tables 6.8- 6.10 lists the commands that are available. Table 6.11 lists commands for element processing.

The command

Example 269

```
PGL_3_2_elements:
▷  $(ORBITER) -v 5 \
▷  ▷  -define G -linear_group -PGL 3 2 -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -save_elements_csv "PGL_3_2_elements.csv" \
▷  ▷  -end
```

creates all elements of $\mathrm{PGL}(3,2)$ and writes them into the file `PGL_3_2_elements.csv`.

The command

Example 270

```
Sym_3_elements:
▷  $(ORBITER) -v 3 \
▷  ▷  -define G -permutation_group -symmetric_group 3 -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -print_elements_tex \
▷  ▷  -end
▷  $(ORBITER) -v 2 \
▷  ▷  -draw_options \
▷  ▷  ▷  -nodes \
▷  ▷  ▷  -embedded -radius 250 \
▷  ▷  ▷  -xin 10000 -yin 10000 \
▷  ▷  ▷  -xout 1000000 -yout 600000 \
▷  ▷  ▷  -scale 0.3 -line_width 1.0 \
▷  ▷  -end \
▷  ▷  -tree_draw -file Sym_3_elements_tree.txt -end
▷  pdflatex Sym_3_elements_tree_draw.tex
▷  $(OPEN) Sym_3_elements_tree_draw.pdf
```

creates a tree of the elements of $\mathrm{Sym}(3)$:

Group theoretic activities (Part 1)		
Command	Arguments	Purpose
-apply	a s	Applies the group element given by the coded vector s to the element a.
-element_processing	options	Perform an activity for a sequence of group elements. For options, see Table 6.11.
-multiply	s_1 s_2	Multiplies group elements s_1 and s_2, assuming the elements are given in coded form. Produces a latex report.
-inverse	s	Computes the inverse of s, which is given in coded form. Produces a latex report.
-consecutive_powers	s n	Computes all powers s^i for $i = 1, \ldots, n$. s is given in coded form. Produces a latex report.
-raise_to_the_power	s n	Computes the n-th power of s, which is given in coded form. Produces a latex report.
-export_orbiter		Exports the group to Orbiter.
-export_gap		Exports the group to GAP [31].
-export_magma		Exports the group to Magma [16].
-canonical_image_GAP	S	Compute the canonical image of the set S using GAP.
-canonical_image	S	Compute the canonical image of the set S using Orbiter.
-search_element_of_order	i	Finds all elements of order i in the group ($i \in \mathbb{N}$).
-find_standard_generators	a b c	Finds all pairs of elements x, y in G such that $\|x\| = a$, $\|y\| = b$, and $\|xy\| = c$. Such pairs are called standard generators.
-random_element	label	Creates a random element. Writes the element to file in coded form.
-element_rank	s	Determines the rank of the group element s in the given group. s is given in coded form.
-element_unrank	r	Produces the group element whose rank is r.
-find_singer_cycle		Finds all Singer cycles whose matrix is a companion matrix.
-classes_based_on_normal_form		
-normalizer		See Section 6.6.
-centralizer_of_element		See Section 6.6.

Table 6.8: Group theoretic activities (Part 1)

Group theoretic activities (Part 2)

Command	Arguments	Purpose
-permutation_representation_ of_element	g	Compute the permutation representation of g
-orbits_on_group_elements_under_ conjugation		
-normalizer_of_cyclic_subgroup		See Section 6.6.
-classes		See Section 6.6.
-report		Produce a latex report for the group.
-report_sylow		Include Sylow subgroups in the report (requires -report).
-report_group_table		Include the group table in the report (requires -report).
-report_classes		Include the conjugacy classes in the report (requires -report).
-export_group_table		Exports the group table as csv-file.
-conjugacy_class_of	g	Compute conjugacy class of g.
-isomorphism_Klein_quadric	fname	
-print_elements		Produces a printout of all group elements.
-print_elements_tex		Produces a latex report of all group elements.
-print_given_elements_tex	label file	Produces a latex report of all group elements in file. The latex file will be named label.tex. The group elements are given in coded (make-element) form.
-save_elements_csv	fname	Writes the elements of the group to the given csv file.
-export_inversion_graphs	fname	Exports the inversion graphs associated to the group elements to the given file.

Table 6.9: Group theoretic activities (Part 2)

Group theoretic activities (Part 3)

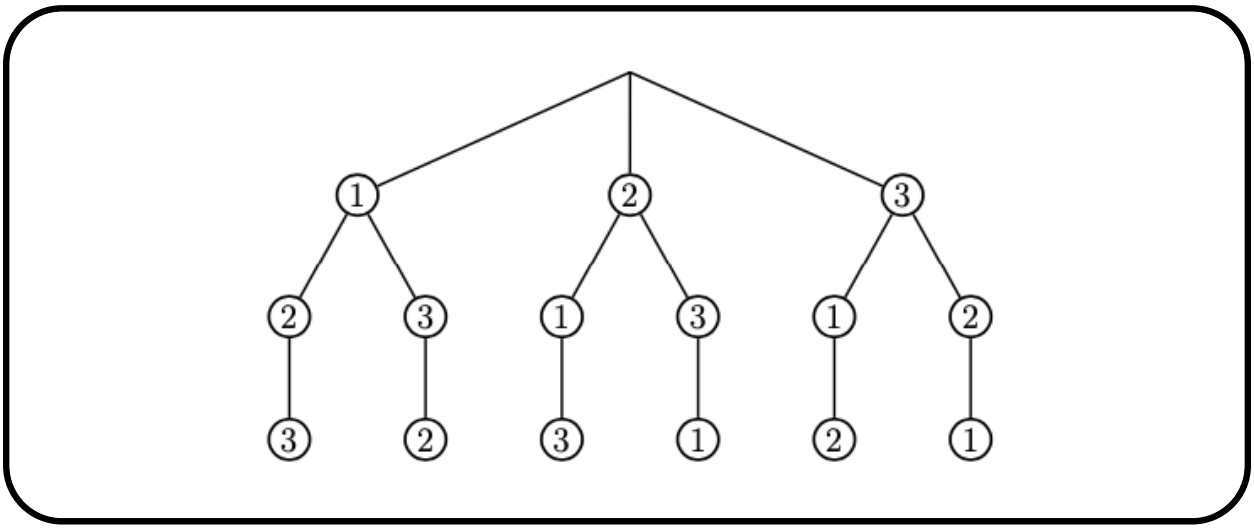
Command	Arguments	Purpose
-multiply_elements_csv_column_major_ordering	f1 f2 f3	
-multiply_elements_csv_row_major_ordering	f1 f2 f3	
-apply_elements_csv_to_set	f1 f2 S	
-order_of_products	$g_1 \dots g_n$	Creates a table of the orders of all products $g_i g_j$, $1 \le i, j \le n$.
-reverse_isomorphism_exterior_square		Given a set of generators of a subgroup of $\mathrm{PGO}^+(6,q)$ as 6×6 matrixes, compute the inverse image of the generators in $\mathrm{PGL}(4,q)$ (if possible).
-is_subgroup_of		
-coset_reps		
-evaluate_word	word gens	
-multiply_all_elements_in_lex_order		Multiplies all group elements together. The main purpose is for speed-testing.
-stats	prefix	Report statistics on group operations.
-move_a_b	a b	Determine a group element mapping a to b. Computes the stabilizer of b in the given group G (the group for which the activity is applied).
-orbit_of	pt	
-orbits_on_set_system_from_file	f c n	
-orbit_of_set_from_file	fname	
-linear_codes	control d $n_{\max}$	Classify linear codes with prescribed minimum distance d. Assumes that the group is $\mathrm{PGL}(r,q)$ or $\mathrm{P\Gamma L}(r,q)$. For each $n \le n_{\max}$, the $[n,k,\ge d]$ codes are classified with $n-k=r$. See Section 10.
-tensor_permutations		Computes the permutation representation of generators of the wreath product.
-classify_ovoids	descr	
-representation_on_polynomials	R	Compute the representation in the action on homogeneous polynomials as defined in the ring R.

Table 6.10: Group theoretic activities (Part 3)

Element Processing Options

Command	Arguments	Purpose
-input	label	Specify the input source for a sequence of group elements.
-print		Print each group element.
-apply_isomorphism_wedge_product_4to6		Applies a specific isomorphism from $\mathrm{GL}(4,q)$ to $\mathrm{GL}(6,q)$
-with_permutation		Print with permutation representation.
-with_fix_structure		Print with fix structure on the underlying projective space.
-order_of_products_of_pairs		Make a table of the order of all pairwise products.
-conjugate	data	Conjugate each group element by the element in data.
-print_action_on_surface	surface	Print the action on a cubic surface object.

Table 6.11: Element Processing Options



It is possible to compute all powers of a fixed element, as in the following command:

Example 271

```
Cycle_12_power:
▷  $(ORBITER) -v 5 \
▷  ▷  -define G -permutation_group -symmetric_group 12 -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -consecutive_powers \
▷  ▷  ▷  "1,2,3,4,5,6,7,8,9,10,11,0" 12 \
▷  ▷  -end
▷  pdflatex Sym_12_all_powers.tex
▷  $(OPEN) Sym_12_all_powers.pdf
▷
```

We create the 12 powers of the cycle

$$(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11).$$

The output is

i	$(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11)^i$
1	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11)
2	(0, 2, 4, 6, 8, 10)(1, 3, 5, 7, 9, 11)
3	(0, 3, 6, 9)(1, 4, 7, 10)(2, 5, 8, 11)
4	(0, 4, 8)(1, 5, 9)(2, 6, 10)(3, 7, 11)
5	(0, 5, 10, 3, 8, 1, 6, 11, 4, 9, 2, 7)
6	(0, 6)(1, 7)(2, 8)(3, 9)(4, 10)(5, 11)
7	(0, 7, 2, 9, 4, 11, 6, 1, 8, 3, 10, 5)
8	(0, 8, 4)(1, 9, 5)(2, 10, 6)(3, 11, 7)
9	(0, 9, 6, 3)(1, 10, 7, 4)(2, 11, 8, 5)
10	(0, 10, 8, 6, 4, 2)(1, 11, 9, 7, 5, 3)
11	(0, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1)
12	id

The command

Example 272

```
PGL_3_4_singer:
▷  $(ORBITER) -v 5 \
▷  ▷  -define G -linear_group -PGL 3 4 -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -find_singer_cycle \
▷  ▷  -end
```

finds all Singer cycles in PGL$(3, 4)$ whose matrix is the companion matrix of a polynomial. The first one found is

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 2 & 3 & 2 \end{bmatrix}$$

whose projective order is 21. Here, we are using the numeric form of field elements, so 2 is ω and 3 is $\omega + 1$.

Suppose we want to multiply two elements in a group. The following command shows an example in GL$(2, 8)$. We multiply the elements coded by 0,1,2,3 and 4,5,6,7:

Example 273

```
GL_2_8_multiply:
▷  $(ORBITER) -v 5 \
▷  ▷  -define G -linear_group -GL 2 8 -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -multiply "0,1,2,3" "4,5,6,7" \
▷  ▷  -end
▷  pdflatex GL_2_8_mult.tex
▷  $(OPEN) GL_2_8_mult.pdf
```

The output is

$$\begin{bmatrix} 0 & 1 \\ \gamma & \gamma^5 \end{bmatrix} \cdot \begin{bmatrix} \gamma^2 & \gamma^3 \\ \gamma^6 & \gamma^4 \end{bmatrix} = \begin{bmatrix} \gamma^6 & \gamma^4 \\ \gamma & \gamma^5 \end{bmatrix}$$

0,1,2,3,
4,5,6,7,
6,7,2,3,

Note that the output shows the codings of the three group elements. This way, the result of this computation can be processed further easily. The same example over $\mathbb{F}_7$, noting that $7 \equiv 0 \mod 7$ is:

Example 274

```
GL_2_7_multiply:
▷  $(ORBITER) -v 5 \
▷  ▷  -define G -linear_group -GL 2 7 -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -multiply "0,1,2,3" "4,5,6,0" \
▷  ▷  -end
▷  pdflatex GL_2_7_mult.tex
▷  $(OPEN) GL_2_7_mult.pdf
```

The output is

$$\begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} \cdot \begin{bmatrix} 4 & 5 \\ 6 & 0 \end{bmatrix} = \begin{bmatrix} 6 & 0 \\ 5 & 3 \end{bmatrix}$$

0,1,2,3,
4,5,6,0,
6,0,5,3,

We can compute the inverse of a group element:

Example 275

```
GL_2_7_inv:
▷  $(ORBITER) -v 5 \
▷  ▷  -define G -linear_group -GL 2 7 -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -inverse "0,1,2,3" \
▷  ▷  -end
▷  pdflatex GL_2_7_inv.tex
▷  $(OPEN) GL_2_7_inv.pdf
```

The output is

$$\begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix}^{-1} = \begin{bmatrix} 2 & 4 \\ 1 & 0 \end{bmatrix}$$

0,1,2,3,
2,4,1,0,

We can raise a group element to a power:

Example 276

```
GL_2_7_power:
▷  $(ORBITER) -v 5 \
▷  ▷  -define G -linear_group -GL 2 7 -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -raise_to_the_power "0,1,2,3" 2 \
▷  ▷  -end
▷  pdflatex GL_2_7_power.tex
▷  $(OPEN) GL_2_7_power.pdf
```

The output is

$$\begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix}^{2} = \begin{bmatrix} 2 & 3 \\ 6 & 4 \end{bmatrix}$$

0,1,2,3,
2,3,6,4,

The next example computes the action of a specific group element on the set of planes through a line. The planes have been computed in Section 4.4.

Example 277

```
on_planes:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 8 -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷  ▷  -define G -linear_group -PGL 4 F -end \
▷  ▷  -define G_on_planes -modified_group -from G \
▷  ▷  ▷  -on_k_subspaces 3 \
▷  ▷  -end \
▷  ▷  -with G_on_planes -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -apply "0,8,1,6,4,3,7,2,5" \
▷  ▷  ▷  "1,0,0,0, 0,1,0,0, 0,0,0,2, 0,0,1,1" \
▷  ▷  -end
▷  pdflatex PGL_4_8_Gr_4_3_apply.tex
▷  $(OPEN) PGL_4_8_Gr_4_3_apply.pdf
```

The output is

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \gamma \\ 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1000 \\ 0100 \\ 0002 \\ 0011 \end{bmatrix}$$

1,0,0,0,0,1,0,0,0,0,0,2,0,0,1,1,
maps:
$0 \mapsto 8$
$8 \mapsto 1$
$1 \mapsto 3$
$6 \mapsto 5$
$4 \mapsto 7$
$3 \mapsto 4$
$7 \mapsto 6$
$2 \mapsto 0$
$5 \mapsto 2$

The command `-random_element` can be used to create a random element in a group. Here is an example. We create the group $\mathrm{PGL}(4,2)$ and then ask for a random element:

Example 278

```
PGL_4_2_random_element:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define G -linear_group -PGL 4 F -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -random_element r1 \
```

```
▷  ▷  -end \
▷  -print_symbols
```

Group Theoretic Activities Based on Magma

Command	Arguments	Purpose
-classes		Compute a report of the conjugacy classes of elements.
-centralizer_of_element	label coding	Compute the centralizer of the group element, using label to create file names.
-normalizer_of_cyclic_subgroup	label s	Compute the normalizer of the cyclic subgroup generated by the element s.
-normalizer		Compute the normalizer of a subgroup in the larger group.

Table 6.12: Group Theoretic Activities Based on Magma

6.6 Group Theoretic Activities Based on Magma

Orbiter has an interface to Magma [16]. This allows Orbiter commands to utilize Magma for specialized (and often difficult) computations in Group theory. Applications are computing conjugacy classes in groups, normalizers and other things. By using these commands through Orbiter, the tasks can be performed for and kind of Orbiter group, including matrix groups. This adds functionality that does not exist in Magma. We will see examples below.

Table 6.12 lists the group theoretic commands in Orbiter that rely on Magma. The actual computation of conjugacy classes happens inside Magma, using a permutation representation of the group. The class representatives and centralizer groups are presented in Orbiter as matrix groups, using the latex interface to display the result. The communication between Magma and Orbiter is based on files. Orbiter writes a file that is used as Magma input. Magma produces a file that is then read by Orbiter.

For instance, the three-step command sequence

Example 279

```
PGGL_2_4_classes:
▷  $(ORBITER) -v 3 \
▷  ▷  -define G \
▷  ▷  -linear_group -PGGL 2 4 \
▷  ▷  -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -classes \
▷  ▷  -end
▷  $(MAGMA_PATH)magma PGGL_2_4_classes.magma
▷  $(ORBITER) -v 3 \
▷  ▷  -define G \
▷  ▷  -linear_group -PGGL 2 4 \
▷  ▷  -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -classes \
▷  ▷  -end
▷  pdflatex PGGL_2_4_classes_out.tex
▷  $(OPEN) PGGL_2_4_classes_out.pdf
```

```
▷ $(OPEN) PGGL_2_4_classes_out.csv
```

computes the classes of elements in $\mathrm{P\Gamma L}(2,4)$. The first Orbiter command produces the file `PGGL_2_4_classes.magma`. The magma command reads this file and produces the file `PGGL_2_4_classes_out.txt`. The second Orbiter command reads the file `PGGL_2_4_classes_out.txt` and produces the latex report `PGGL_2_4_classes_out.tex`.

The report produced by Orbiter is too long to be reproduced here fully. Let us look at just one conjugacy class. Here is the output for class 1 / 7 (numbering starts from 0, so this is the second class):

Order of element = 2
Class size = 10
Centralizer order = 12
Normalizer order = 12
Representing element is

$$c_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}_1$$

of order 2 and with 3 fixed points. $0, 1, 1, 0, 1,$
The normalizer is generated by:
Strong generators for a group of order 12:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}_1, \begin{bmatrix} \omega^2 & 0 \\ 0 & 1 \end{bmatrix}_1, \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}_1$$

1,0,0,1,1,
1,0,0,2,1,
0,1,1,0,1,

The command sequence

Example 280

```
PGGL_2_4_cent_2A:
▷ $(ORBITER) -v 3 \
▷ ▷ -define G \
▷ ▷ -linear_group -PGGL 2 4 -end \
▷ ▷ -with G -do \
▷ ▷ -group_theoretic_activity \
▷ ▷ ▷ -centralizer_of_element "2A" "1,0, 0,1, 1" \
▷ ▷ ▷ -report \
▷ ▷ -end
▷ $(MAGMA_PATH)magma element_2A_centralizer.magma
▷ $(ORBITER) -v 6 \
▷ ▷ -define G \
▷ ▷ -linear_group -PGGL 2 4 -end \
▷ ▷ -with G -do \
▷ ▷ -group_theoretic_activity \
```

```
▷ ▷ ▷ -centralizer_of_element "2A" "1,0, 0,1, 1" \
▷ ▷ ▷ -report \
▷ ▷ -end
▷ pdflatex PGGL_2_4_elt_2A_centralizer.tex
▷ $(OPEN) PGGL_2_4_elt_2A_centralizer.pdf
▷
```

computes the centralizer of the Baer involution

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_1 .$$

The centralizer is a group of order 40320, isomorphic to $\mathrm{PGL}(4,2).Z_2$. Orbiter produces a list of strong generators, shown below:

Strong generators for a group of order 40320:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_1 , \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}_1 , \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}_0 ,$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}_0 , \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}_0 , \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}_0 ,$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_0 , \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}_0 , \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}_0$$

1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1,
1,0,0,0,0,1,0,0,0,0,1,0,1,0,0,1,1,
1,0,0,0,0,1,0,0,0,0,1,0,1,1,1,1,0,
1,0,0,0,0,1,0,0,0,0,1,0,1,1,0,1,0,
1,0,0,0,0,1,0,0,1,1,0,1,1,1,1,0,0,
1,0,0,0,0,1,0,0,0,0,1,1,1,1,1,0,0,
1,0,0,0,0,1,0,0,1,1,1,0,0,0,0,1,0,
1,0,0,0,1,1,1,1,1,0,1,0,0,1,0,0,0,
0,1,0,1,1,1,1,1,1,0,0,1,0,1,1,1,0,

The end of the report has a list of generators in coded form. This list can be used to create the centralizer in Orbiter.

The command

Example 281

```
Hirschfeld_stab_classes:
▷  $(ORBITER) -v 9 \
▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
▷  ▷  -define G -linear_group -PGGL 4 4 \
▷  ▷  ▷  -subgroup_by_generators "Hirschfeld_Stab" \
▷  ▷  ▷  51840 6 $(HIRSCHFELD_STAB_GENERATORS) \
▷  ▷  ▷  -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -classes \
▷  ▷  -end
▷  $(ORBITER) -v 9 \
▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
▷  ▷  -define G -linear_group -PGGL 4 4 \
▷  ▷  ▷  -subgroup_by_generators "Hirschfeld_Stab" \
▷  ▷  ▷  51840 6 $(HIRSCHFELD_STAB_GENERATORS) \
▷  ▷  ▷  -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -classes \
▷  ▷  -end
▷  pdflatex PGGL_4_4_Subgroup_Hirschfeld_Stab_51840_classes_out.tex
▷  $(OPEN) PGGL_4_4_Subgroup_Hirschfeld_Stab_51840_classes_out.pdf
```

computes the conjugacy classes of the Weyl group of type E_6. The group is the stabilizer of the Hirschfeld surface (see Section 8.2). The generators have been stored in a makefile variable. They have been computed with the commands described in Section 16.2.

Orbiter can compute the normalizer of a subgroup. The group must be constructed as a subgroup H of a larger group G containing H. Typically, the group G is the group in which H is generated as a subgroup (either the full linear group of the full symmetric group). Then, the normalizer of H in G is computed. Here is an example in a symmetric group. We first greate a subgroup of order 5, using a makefile variable. The generator is

$$(0,1,2,3,4)(5,6,7,8,9)(10)(11)(12).$$

We store it in a makefile variable:

Example 282

```
GENERATORS_H5="1,2,3,4,0,6,7,8,9,5,10,11,12"
# (0, 1, 2, 3, 4)(5, 6, 7, 8, 9)
```

The command

Example 283

```
Normalizer_of_H5:
▷  $(ORBITER) -v 2 \
▷  ▷  -define G -permutation_group -symmetric_group 13 \
▷  ▷  ▷  -subgroup_by_generators H5 5 1 \
```

```
▷ ▷ ▷ ▷ $(GENERATORS_H5) -end \
▷ ▷ -with G -do \
▷ ▷ -group_theoretic_activity \
▷ ▷ ▷ -normalizer \
▷ ▷ -end
▷ pdflatex Sym_13_Subgroup_H5_5_normalizer.tex
▷ $(OPEN) Sym_13_Subgroup_H5_5_normalizer.pdf
```

computes the normalizer of H insize Sym(13). The normalizer is a group of order 1200. Because of the way in which Orbiter and Magma collaborate, the command has to be executed twice. After the first execution, a magma session is started. The magma session has to be terminated by typing

```
quit;
```

The Orbiter command has to be run one more time after that. The following report is produced:

The group $Perm$13SubgroupH5order5 of order 5 is:
Strong generators for a group of order 5:

$$(0,1,2,3,4)(5,6,7,8,9)$$

1, 2, 3, 4, 0, 6, 7, 8, 9, 5, 10, 11, 12,
Inside the group of order 6227020800, the normalizer has order 1200:

Strong generators for a group of order 1200:

$$(11,12),$$

$$(10,11),$$

$$(5,9,8,7,6),$$

$$(1,2,4,3)(6,7,9,8),$$

$$(0,5)(1,9)(2,8)(3,7)(4,6)$$

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 11,
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 10, 12,
0, 1, 2, 3, 4, 9, 5, 6, 7, 8, 10, 11, 12,
0, 2, 4, 1, 3, 5, 7, 9, 6, 8, 10, 11, 12,
5, 9, 8, 7, 6, 0, 4, 3, 2, 1, 10, 11, 12,

Consider this example of a subgroup which is not cyclic: The group

$$H = \langle \begin{bmatrix} \alpha^4 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \rangle \simeq C_2 \times C_2$$

is a subgroup of $G = \mathrm{PGL}(2,9)$. To compute the normalizer of H in G, the following command sequence can be used:

Example 284

```
Normalizer_of_Z22_in_PGL_2_9:
▷  $(ORBITER) -v 2 \
▷  ▷  -define G -linear_group -PGL 2 9 \
▷  ▷  -subgroup_by_generators Z22 4 2 \
▷  ▷  ▷  "2,0,0,1, 0,1,1,0" -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -normalizer \
▷  ▷  -end
▷  pdflatex PGL_2_9_Subgroup_Z22_4_normalizer.tex
▷  $(OPEN) PGL_2_9_Subgroup_Z22_4_normalizer.pdf
```

It produces a report showing that the normalizer is a group of order 24 (it is isomorphic to Sym(4), though the report does not tell us this fact directly):

The group PGL(2, 9)SubgroupZ22order4 of order 4 is:
Strong generators for a group of order 4:

$$\begin{bmatrix} \alpha^4 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

1,0,0,2,
0,1,1,0,
Inside the group of order 720, the normalizer has order 24:

Strong generators for a group of order 24:

$$\begin{bmatrix} \alpha^4 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} \alpha^2 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} \alpha^4 & \alpha^4 \\ \alpha^4 & 1 \end{bmatrix},$$

$$\begin{bmatrix} \alpha^4 & \alpha^6 \\ \alpha^2 & 1 \end{bmatrix}$$

1,0,0,2,
1,0,0,5,
1,1,1,2,
1,7,5,2,

The command

Example 285

```
PGOm_6_2_classes:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define G -linear_group -PGOm 6 F -end \
▷  ▷  -with G -do \
```

```
▷ ▷ -group_theoretic_activity \
▷ ▷ ▷ -classes \
▷ ▷ -end
▷ pdflatex PGOm_6_2_classes_out.tex
▷ $(OPEN) PGOm_6_2_classes_out.pdf
```

computes the conjugacy classes of $\mathrm{PGO}^-(6,2)$. The command has to be exectuted twice. Let us look at the class 2A of size 36. We store the representative as a makefile variable:

Example 286

```
PGOM_6_2_CLASS_2A_REP="1,0,0,0,0,0,0,1,0,0,\
0,0,0,0,1,0,0,0,0,0,1,1,1,1,0,0,1,0,0,1,0,0,1,0,1,0"
```

To create the whole conjugacy class, we use the following command:

Example 287

```
PGOm_6_2_Class_2A:
▷ $(ORBITER) -v 9 \
▷ ▷ -orbiter_path $(ORBITER_EXE_PATH) \
▷ ▷ -define F -finite_field -q 2 -end \
▷ ▷ -define G -linear_group -PGOm 6 F -end \
▷ ▷ -with G -do \
▷ ▷ -group_theoretic_activity \
▷ ▷ ▷ -conjugacy_class_of "2A" $(PGOM_6_2_CLASS_2A_REP) \
▷ ▷ -end
```

The elements are stored in a csv file. We use a makefile variable to store the file name:

Example 288

```
FILE_NAME_PGOM_6_2_CLASS_2A="PGOm_6_2_class_of_2A.csv"
```

The following command prints all group elements in the class:

Example 289

```
PGOm_6_2_Class_2A_print:
▷ $(ORBITER) -v 9 \
▷ ▷ -orbiter_path $(ORBITER_EXE_PATH) \
▷ ▷ -define PGOm_6_2_Class2A -vector \
▷ ▷ ▷ -file $(FILE_NAME_PGOM_6_2_CLASS_2A) -end \
▷ ▷ -define F -finite_field -q 2 -end \
▷ ▷ -define G -linear_group -PGOm 6 F -end \
▷ ▷ -with G -do \
▷ ▷ -group_theoretic_activity \
▷ ▷ ▷ -element_processing \
```

```
▷ ▷ ▷ ▷ -input PGOm_6_2_Class2A \
▷ ▷ ▷ ▷ -print \
▷ ▷ ▷ -end \
▷ ▷ -end
▷ pdflatex PGOm_6_2_Class2A_elements.tex
▷ $(OPEN) PGOm_6_2_Class2A_elements.pdf
```

We decide to pick group element 8 as our favorite class representative. We store the element in a makefile variable and compute the centralizer:

Example 290

```
PGOM_6_2_CLASS_2A_REP_CLEAN="1,0,0,0,0,0,0,1,0,0,\
0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1"
```

Example 291

```
PGOm_6_2_Class_2A_centralizer:
▷ $(ORBITER) -v 9 \
▷ ▷ -orbiter_path $(ORBITER_EXE_PATH) \
▷ ▷ -define F -finite_field -q 2 -end \
▷ ▷ -define G -linear_group -PGOm 6 F -end \
▷ ▷ -with G -do \
▷ ▷ -group_theoretic_activity \
▷ ▷ ▷ -centralizer_of_element \
▷ ▷ ▷ ▷ "2Arep" $(PGOM_6_2_CLASS_2A_REP_CLEAN) \
▷ ▷ -end
▷ pdflatex PGOm_6_2_elt_2Arep_centralizer.tex
▷ $(OPEN) PGOm_6_2_elt_2Arep_centralizer.pdf
```

We store the generators for the centralizer in a makefile variable.

Example 292

```
PGOM_6_2_CLASS_2A_CENTRALIZER_GENS="\
1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,1,1, \
1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,1,0, \
1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1, \
1,0,0,0,0,0,1,1,0,0,0,1,0,0,1,0,0,0,0,0,0,1,0,0,1,0,0,0,1,1,0,0,0,0,0,1, \
1,0,0,0,0,0,1,1,0,0,1,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,0,0,0,1,1, \
1,0,0,0,0,0,1,1,1,1,0,0,1,0,0,1,0,0,1,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1, \
0,1,1,1,1,0,1,0,1,1,1,0,1,1,0,1,1,0,1,1,1,0,1,0,0,0,0,0,1,0,1,1,1,1,1,1, \
1,1,0,0,0,1,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,1,0,0,1,1,0,0,0,0,0,1, \
0,1,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1"
```

We recompute the centralizer as a subgroup of $\mathrm{PGL}(6,2)$:

Example 293

```
PGOm_6_2_Class_2A_centralizer_recover:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define Class2A_cent_gens -vector \
▷  ▷  ▷  -dense $(PGOM_6_2_CLASS_2A_CENTRALIZER_GENS) -end \
▷  ▷  -define G -linear_group -PGL 6 2 \
▷  ▷  ▷  -subgroup_by_generators "2ACent" \
▷  ▷  ▷  1440 9 Class2A_cent_gens \
▷  ▷  ▷  -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex PGL_6_2_Subgroup_2ACent_1440_report.tex
▷  $(OPEN) PGL_6_2_Subgroup_2ACent_1440_report.pdf
```

Group Theoretic Activities Based on GAP		
Command	Arguments	Purpose
-canonical_image	set	Compute the canonical image of the given set under the group G.
-export_gap		Export the group G to GAP [31] / Fining [5].

Table 6.13: Group Theoretic Activities Based on GAP

6.7 Group Theoretic Activities Based on GAP

Orbiter has an interface to GAP [31]. This allows Orbiter commands to utilize GAP specific functionality. One application is the computation of canonical images of sets in the context of permutation groups. This is facilitated by means of the GAP package images [40].

Table 6.13 lists the group theoretic commands in Orbiter that rely on GAP.

How can we export a group from Orbiter to GAP? For semilinear matrix groups, we need to export to the GAP package Fining [5]. This is because semilinear groups are not available in GAP. Here is an example. We create the stabilizer of the Hirschfeld surface in PG$(3,4)$ and export the group to GAP/Fining as a group of collineations.

Example 294

```
Hirschfeld_stab_export_gap:
▷  $(ORBITER) -v 9 \
▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
▷  ▷  -define G -linear_group -PGGL 4 4 \
▷  ▷  ▷  -subgroup_by_generators "Hirschfeld_Stab" \
▷  ▷  ▷  51840 6 $(HIRSCHFELD_STAB_GENERATORS) \
▷  ▷  ▷  -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -export_gap \
▷  ▷  -end
```

The command creates a GAP/Fining file shown below:

```
LoadPackage("fining");
mat1 := [[Z(4)^0,0*Z(4),0*Z(4),0*Z(4)],
[0*Z(4),Z(4)^0,0*Z(4),0*Z(4)],
[0*Z(4),0*Z(4),Z(4)^0,0*Z(4)],
[0*Z(4),0*Z(4),0*Z(4),Z(4)^0]];
frob1 := FrobeniusAutomorphism(GF(4))^1;
psi1 := ProjectiveSemilinearMap(mat1, frob1,GF(4));
mat2 := [[Z(4)^0,0*Z(4),0*Z(4),0*Z(4)],
[0*Z(4),Z(4)^1,0*Z(4),0*Z(4)],
[0*Z(4),0*Z(4),Z(4)^1,0*Z(4)],
[0*Z(4),0*Z(4),0*Z(4),Z(4)^0]];
frob2 := FrobeniusAutomorphism(GF(4))^0;
psi2 := ProjectiveSemilinearMap(mat2, frob2,GF(4));
```

```
mat3 := [[Z(4)^0,0*Z(4),0*Z(4),0*Z(4)],
[0*Z(4),Z(4)^2,0*Z(4),0*Z(4)],
[0*Z(4),0*Z(4),Z(4)^2,0*Z(4)],
[Z(4)^0,0*Z(4),0*Z(4),Z(4)^0]];
frob3 := FrobeniusAutomorphism(GF(4))^0;
psi3 := ProjectiveSemilinearMap(mat3, frob3,GF(4));
mat4 := [[Z(4)^0,0*Z(4),0*Z(4),0*Z(4)],
[0*Z(4),Z(4)^0,0*Z(4),0*Z(4)],
[Z(4)^0,Z(4)^0,Z(4)^0,0*Z(4)],
[0*Z(4),Z(4)^0,0*Z(4),Z(4)^0]];
frob4 := FrobeniusAutomorphism(GF(4))^1;
psi4 := ProjectiveSemilinearMap(mat4, frob4,GF(4));
mat5 := [[Z(4)^0,0*Z(4),0*Z(4),0*Z(4)],
[0*Z(4),0*Z(4),Z(4)^0,0*Z(4)],
[0*Z(4),Z(4)^0,0*Z(4),0*Z(4)],
[0*Z(4),0*Z(4),0*Z(4),Z(4)^0]];
frob5 := FrobeniusAutomorphism(GF(4))^0;
psi5 := ProjectiveSemilinearMap(mat5, frob5,GF(4));
mat6 := [[0*Z(4),Z(4)^0,0*Z(4),0*Z(4)],
[Z(4)^0,0*Z(4),0*Z(4),0*Z(4)],
[0*Z(4),0*Z(4),0*Z(4),Z(4)^0],
[0*Z(4),0*Z(4),Z(4)^0,0*Z(4)]];
frob6 := FrobeniusAutomorphism(GF(4))^0;
psi6 := ProjectiveSemilinearMap(mat6, frob6,GF(4));
gens := [psi1, psi2, psi3, psi4, psi5, psi6];
G := Group(gens);
Size(G);
```

When we load this file into GAP, we see the following output (for printing purposes, it has been shortened and reformatted slightly):

```
gap> LoadPackage("fining");
true
gap> mat1 := [[Z(4)^0,0*Z(4),0*Z(4),0*Z(4)],
> [0*Z(4),Z(4)^0,0*Z(4),0*Z(4)],
> [0*Z(4),0*Z(4),Z(4)^0,0*Z(4)],
> [0*Z(4),0*Z(4),0*Z(4),Z(4)^0]];
[ [ Z(2)^0, 0*Z(2), 0*Z(2), 0*Z(2) ],
[ 0*Z(2), Z(2)^0, 0*Z(2), 0*Z(2) ],
[ 0*Z(2), 0*Z(2), Z(2)^0, 0*Z(2) ],
[ 0*Z(2), 0*Z(2), 0*Z(2), Z(2)^0 ] ]
gap> frob1 := FrobeniusAutomorphism(GF(4))^1;
FrobeniusAutomorphism( GF(2^2) )
gap> psi1 := ProjectiveSemilinearMap(mat1, frob1,GF(4));
< a collineation: <cmat 4x4 over GF(2,2)>, F^2>
...
gap> gens := [psi1, psi2, psi3, psi4, psi5, psi6];
[ < a collineation: <cmat 4x4 over GF(2,2)>, F^2>,
< a collineation: <cmat 4x4 over GF(2,2)>, F^0>,
< a collineation: <cmat 4x4 over GF(2,2)>, F^0>,
< a collineation: <cmat 4x4 over GF(2,2)>, F^2>,
< a collineation: <cmat 4x4 over GF(2,2)>, F^0>,
```

```
< a collineation: <cmat 4x4 over GF(2,2)>, F^0> ]
gap> G := Group(gens);
<projective collineation group with 6 generators>
gap> Size(G);
51840
gap>
```

Suppose we want to compute the least image of the set $\{3, 5\}$ in the permutation group C_6. The following command sets up the group C_6 and invokes the Orbiter `-canonical_image` command.

Example 295

```
Cyclic_6_canonical_image:
▷  $(ORBITER) -v 3 \
▷  ▷  -define G -permutation_group -cyclic_group 6 -end \
▷  ▷  -with G -do -group_theoretic_activity \
▷  ▷  ▷  -canonical_image_GAP 3,5 \
▷  ▷  -end
```

The command creates a GAP file shown below:

```
G := Group([(1, 2, 3, 4, 5, 6)]);
LoadPackage("images");
MinimalImage(G, [4,6], OnSets);
```

Running this command in GAP yields:

```
gap> MinimalImage(G, [4,6], OnSets);
[ 1, 3 ]
```

Note that the permutation representations in GAP are 1-based, while Orbiter uses 0-based representations. So, the least image of $\{1, 3\}$ in GAP really corresponds to the set $\{0, 2\}$ in Orbiter.

Let us try a larger example. We want to compute the canonical form of the Edge curve by means of their rational points over $\mathbb{F}_{17}$. We use the following makefile variable to specify the set of rational points:

Example 296

```
EDGE_CURVE_Q17_AS_POINTS="4, 7, 16, 19, 20, 23, 32, 35, 89, 100, 244, 251"
```

The next command creates $\mathrm{PGL}(3, 17)$ and invokes the `-canonical_image` image command.

Example 297

```
Edge_qurve_q17_canonical_image:
▷  $(ORBITER) -v 3 \
▷  ▷  -define G -linear_group -PGL 3 17 -end \
```

```
▷ ▷ -with G -do -group_theoretic_activity \
▷ ▷ ▷ -canonical_image_GAP $(EDGE_CURVE_Q17_AS_POINTS) \
▷ ▷ -end
```

The command creates a GAP input file called `PGL_3_17_canonical_image.gap`.

```
G := Group([(4, 39, 45, 46, 49, 41, 51, 47, 52, 50, 44, 43, 40, 48, 38, 42)(5,
7, 13, 14, 17, 9, 19, 15, 20, 18, 12, 11, 8, 16, 6, 10)(21, 23, 29, 30, 33,
25, 35, 31, 36, 34, 28, 27, 24, 32, 22, 26)(54, 56, 62, 63, 66, 58, 68, 64,
69, 67, 61, 60, 57, 65, 55, 59)(71, 73, 79, 80, 83, 75, 85, 81, 86, 84, 78,
77, 74, 82, 72, 76)(88, 90, 96, 97, 100, 92, 102, 98, 103, 101, 95, 94, 91,
99, 89, 93)(105, 107, 113, 114, 117, 109, 119, 115, 120, 118, 112, 111, 108,
116, 106, 110)(122, 124, 130, 131, 134, 126, 136, 132, 137, 135, 129, 128,
125, 133, 123, 127)(139, 141, 147, 148, 151, 143, 153, 149, 154, 152, 146,
145, 142, 150, 140, 144)(156, 158, 164, 165, 168, 160, 170, 166, 171, 169,
163, 162, 159, 167, 157, 161)(173, 175, 181, 182, 185, 177, 187, 183, 188,
186, 180, 179, 176, 184, 174, 178)(190, 192, 198, 199, 202, 194, 204, 200,
205, 203, 197, 196, 193, 201, 191, 195)(207, 209, 215, 216, 219, 211, 221,
217, 222, 220, 214, 213, 210, 218, 208, 212)(224, 226, 232, 233, 236, 228,
238, 234, 239, 237, 231, 230, 227, 235, 225, 229)(241, 243, 249, 250, 253,
245, 255, 251, 256, 254, 248, 247, 244, 252, 242, 246)(258, 260, 266, 267,
270, 262, 272, 268, 273, 271, 265, 264, 261, 269, 259, 263)(275, 277, 283,
284, 287, 279, 289, 285, 290, 288, 282, 281, 278, 286, 276, 280)(292, 294,
300, 301, 304, 296, 306, 302, 307, 305, 299, 298, 295, 303, 293, 297),
(4, 71, 173, 190, 241, 105, 275, 207, 292, 258, 156, 139, 88, 224, 54, 122)
(5, 10, 6, 16, 8, 11, 12, 18, 20, 15, 19, 9, 17, 14, 13, 7)(37, 70, 172,
189, 240, 104, 274, 206, 291, 257, 155, 138, 87, 223, 53, 121)(38, 72, 174,
191, 242, 106, 276, 208, 293, 259, 157, 140, 89, 225, 55, 123)(39, 73, 175,
192, 243, 107, 277, 209, 294, 260, 158, 141, 90, 226, 56, 124)(40, 74, 176,
193, 244, 108, 278, 210, 295, 261, 159, 142, 91, 227, 57, 125)(41, 75, 177,
194, 245, 109, 279, 211, 296, 262, 160, 143, 92, 228, 58, 126)(42, 76, 178,
195, 246, 110, 280, 212, 297, 263, 161, 144, 93, 229, 59, 127)(43, 77, 179,
196, 247, 111, 281, 213, 298, 264, 162, 145, 94, 230, 60, 128)(44, 78, 180,
197, 248, 112, 282, 214, 299, 265, 163, 146, 95, 231, 61, 129)(45, 79, 181,
198, 249, 113, 283, 215, 300, 266, 164, 147, 96, 232, 62, 130)(46, 80, 182,
199, 250, 114, 284, 216, 301, 267, 165, 148, 97, 233, 63, 131)(47, 81, 183,
200, 251, 115, 285, 217, 302, 268, 166, 149, 98, 234, 64, 132)(48, 82, 184,
201, 252, 116, 286, 218, 303, 269, 167, 150, 99, 235, 65, 133)(49, 83, 185,
202, 253, 117, 287, 219, 304, 270, 168, 151, 100, 236, 66, 134)(50, 84, 186,
203, 254, 118, 288, 220, 305, 271, 169, 152, 101, 237, 67, 135)(51, 85, 187,
204, 255, 119, 289, 221, 306, 272, 170, 153, 102, 238, 68, 136)(52, 86, 188,
205, 256, 120, 290, 222, 307, 273, 171, 154, 103, 239, 69, 137),
(3, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36)(4, 38,
39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 37)(53, 54, 55, 56,
57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69)(70, 71, 72, 73, 74, 75,
76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86)(87, 88, 89, 90, 91, 92, 93, 94,
95, 96, 97, 98, 99, 100, 101, 102, 103)(104, 105, 106, 107, 108, 109, 110,
111, 112, 113, 114, 115, 116, 117, 118, 119, 120)(121, 122, 123, 124, 125,
126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137)(138, 139, 140,
141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154)(155,
156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170,
```

```
171)(172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185,
186, 187, 188)(189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200,
201, 202, 203, 204, 205)(206, 207, 208, 209, 210, 211, 212, 213, 214, 215,
216, 217, 218, 219, 220, 221, 222)(223, 224, 225, 226, 227, 228, 229, 230,
231, 232, 233, 234, 235, 236, 237, 238, 239)(240, 241, 242, 243, 244, 245,
246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256)(257, 258, 259, 260,
261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273)(274, 275,
276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290)
(291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305,
306, 307),
(3, 37, 53, 70, 87, 104, 121, 138, 155, 172, 189, 206, 223, 240, 257, 274,
291)(4, 54, 71, 88, 105, 122, 139, 156, 173, 190, 207, 224, 241, 258, 275,
292, 21)(22, 38, 55, 72, 89, 106, 123, 140, 157, 174, 191, 208, 225, 242,
259, 276, 293)(23, 39, 56, 73, 90, 107, 124, 141, 158, 175, 192, 209, 226,
243, 260, 277, 294)(24, 40, 57, 74, 91, 108, 125, 142, 159, 176, 193, 210,
227, 244, 261, 278, 295)(25, 41, 58, 75, 92, 109, 126, 143, 160, 177, 194,
211, 228, 245, 262, 279, 296)(26, 42, 59, 76, 93, 110, 127, 144, 161, 178,
195, 212, 229, 246, 263, 280, 297)(27, 43, 60, 77, 94, 111, 128, 145, 162,
179, 196, 213, 230, 247, 264, 281, 298)(28, 44, 61, 78, 95, 112, 129, 146,
163, 180, 197, 214, 231, 248, 265, 282, 299)(29, 45, 62, 79, 96, 113, 130,
147, 164, 181, 198, 215, 232, 249, 266, 283, 300)(30, 46, 63, 80, 97, 114,
131, 148, 165, 182, 199, 216, 233, 250, 267, 284, 301)(31, 47, 64, 81, 98,
115, 132, 149, 166, 183, 200, 217, 234, 251, 268, 285, 302)(32, 48, 65, 82,
99, 116, 133, 150, 167, 184, 201, 218, 235, 252, 269, 286, 303)(33, 49, 66,
83, 100, 117, 134, 151, 168, 185, 202, 219, 236, 253, 270, 287, 304)(34,
50, 67, 84, 101, 118, 135, 152, 169, 186, 203, 220, 237, 254, 271, 288,
305)(35, 51, 68, 85, 102, 119, 136, 153, 170, 187, 204, 221, 238, 255,
272, 289, 306)(36, 52, 69, 86, 103, 120, 137, 154, 171, 188, 205, 222,
239, 256, 273, 290, 307),
(2, 3)(5, 21)(6, 22)(7, 23)(8, 24)(9, 25)(10, 26)(11, 27)(12, 28)(13, 29)
(14, 30)(15, 31)(16, 32)(17, 33)(18, 34)(19, 35)(20, 36)(53, 172)(54, 181)
(55, 173)(56, 182)(57, 174)(58, 183)(59, 175)(60, 184)(61, 176)(62, 185)
(63, 177)(64, 186)(65, 178)(66, 187)(67, 179)(68, 188)(69, 180)(70, 121)
(71, 127)(72, 133)(73, 122)(74, 128)(75, 134)(76, 123)(77, 129)(78, 135)
(79, 124)(80, 130)(81, 136)(82, 125)(83, 131)(84, 137)(85, 126)(86, 132)
(87, 240)(88, 253)(89, 249)(90, 245)(91, 241)(92, 254)(93, 250)(94, 246)
(95, 242)(96, 255)(97, 251)(98, 247)(99, 243)(100, 256)(101, 252)(102,
248)(103, 244)(104, 138)(105, 145)(106, 152)(107, 142)(108, 149)(109,
139)(110, 146)(111, 153)(112, 143)(113, 150)(114, 140)(115, 147)(116,
154)(117, 144)(118, 151)(119, 141)(120, 148)(155, 274)(156, 289)(157,
287)(158, 285)(159, 283)(160, 281)(161, 279)(162, 277)(163, 275)(164,
290)(165, 288)(166, 286)(167, 284)(168, 282)(169, 280)(170, 278)(171,
276)(189, 223)(190, 235)(191, 230)(192, 225)(193, 237)(194, 232)(195,
227)(196, 239)(197, 234)(198, 229)(199, 224)(200, 236)(201, 231)(202,
226)(203, 238)(204, 233)(205, 228)(206, 257)(207, 271)(208, 268)(209,
265)(210, 262)(211, 259)(212, 273)(213, 270)(214, 267)(215, 264)(216,
261)(217, 258)(218, 272)(219, 269)(220, 266)(221, 263)(222, 260)(292,
307)(293, 306)(294, 305)(295, 304)(296, 303)(297, 302)(298, 301)(299,
300),
(1, 2)(6, 13)(7, 10)(8, 17)(9, 11)(12, 19)(14, 16)(15, 18)(21, 37)(22,
53)(23, 70)(24, 87)(25, 104)(26, 121)(27, 138)(28, 155)(29, 172)(30,
189)(31, 206)(32, 223)(33, 240)(34, 257)(35, 274)(36, 291)(38, 54)(39,
71)(40, 88)(41, 105)(42, 122)(43, 139)(44, 156)(45, 173)(46, 190)(47,
```

```
207)(48, 224)(49, 241)(50, 258)(51, 275)(52, 292)(56, 72)(57, 89)(58,
106)(59, 123)(60, 140)(61, 157)(62, 174)(63, 191)(64, 208)(65, 225)(66,
242)(67, 259)(68, 276)(69, 293)(74, 90)(75, 107)(76, 124)(77, 141)(78,
158)(79, 175)(80, 192)(81, 209)(82, 226)(83, 243)(84, 260)(85, 277)(86,
294)(92, 108)(93, 125)(94, 142)(95, 159)(96, 176)(97, 193)(98, 210)(99,
227)(100, 244)(101, 261)(102, 278)(103, 295)(110, 126)(111, 143)(112,
160)(113, 177)(114, 194)(115, 211)(116, 228)(117, 245)(118, 262)(119,
279)(120, 296)(128, 144)(129, 161)(130, 178)(131, 195)(132, 212)(133,
229)(134, 246)(135, 263)(136, 280)(137, 297)(146, 162)(147, 179)(148,
196)(149, 213)(150, 230)(151, 247)(152, 264)(153, 281)(154, 298)(164,
180)(165, 197)(166, 214)(167, 231)(168, 248)(169, 265)(170, 282)(171,
299)(182, 198)(183, 215)(184, 232)(185, 249)(186, 266)(187, 283)(188,
300)(200, 216)(201, 233)(202, 250)(203, 267)(204, 284)(205, 301)(218,
234)(219, 251)(220, 268)(221, 285)(222, 302)(236, 252)(237, 269)(238,
286)(239, 303)(254, 270)(255, 287)(256, 304)(272, 288)(273, 305)(290,
306)]);
LoadPackage("images");
MinimalImage(G, [5,8,17,20,21,24,33,36,90,101,245,252], OnSets);
```

Running this file through GAP yields the canonical form:

```
gap> MinimalImage(G, [5,8,17,20,21,24,33,36,90,101,245,252], OnSets);
[ 1, 2, 3, 4, 5, 6, 55, 104, 108, 112, 206, 228 ]
```

Keep in mind that the permutation representation in Gap is 1-based, so the numbers have been incremented by one.

6.8 Linear Groups, Advanced Topics

It is sometimes necessary to control the finite field that is used in the construction of a matrix group. For prime fields, this is not an issue. For extension fields, the choice of polynomial does matter, as the generators depend on specific choices made for the finite field. Magma and GAP use Conway polynomials, which are difficult to compute. Orbiter has a built-in table of primitive polynomials. As explained in Section 3.3, Orbiter allows to specify the polynomial that should be used to create the finite field. The next example shows an instance where choosing the polynomial is important. We are recreating a group from the electronic Atlas on finite simple groups [73].

The electronic Atlas of finite simple groups [73] lists generators for $U_3(3)$ as 3×3 matrices over the field $\mathbb{F}_9$ using the following short Magma [16] program:

```
F<w>:=GF(9);
x:=CambridgeMatrix(1,F,3,[
"164",
"506",
"851"]);
y:=CambridgeMatrix(1,F,3,[
"621",
"784",
"066"]);
G<x,y>:=MatrixGroup<3,F|x,y>;
```

The generators are given using the Magma command `CambridgeMatrix`, which allows for more efficient coding of field elements. The field elements are coded as base-3 integers (like in Orbiter) with respect to the Magma version of $\mathbb{F}_9$. The polynomial for $\mathbb{F}_9$ can be determined using the following Magma command, which can be typed into Magma (or the free Magma online calculator at [68]):

```
F<w>:=GF(9);
print DefiningPolynomial(F);
```

It results in

```
$.1^2 + 2*$.1 + 2
```

which is the Magma way of printing the polynomial $X^2 + 2X + 2$. If α is a root of the polynomial over $\mathbb{F}_3$, then

$$\alpha^2 = \alpha + 1.$$

The coefficient vector of the polynomial is $(1, 2, 2)$. As an integer written in base-3, we obtain

$$1 \cdot 3^2 + 2 \cdot 3 + 2 = 17.$$

The desired subgroup can now be created using the command

Example 298

```
U_3_3:
▷  $(ORBITER) -v 3 \
```

```
▷  ▷  -define F -finite_field -q 9 \
▷  ▷  ▷  -override_polynomial "17" \
▷  ▷  -end \
▷  ▷  -define G -linear_group -PGL 3 F \
▷  ▷  ▷  -subgroup_by_generators \
▷  ▷  ▷  "U_3_3" "6048" 2 \
▷  ▷  ▷  "1,6,4, 5,0,6, 8,5,1, \
▷  ▷  ▷  6,2,1, 7,8,4, 0,6,6" \
▷  ▷  ▷  -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex PGL_3_9_Subgroup_U_3_3_6048_report.tex
▷  #$(OPEN) PGL_3_9_Subgroup_U_3_3_6048_report.pdf
```

Group theoretic activities will be discussed in Section 6.5.

As an example of a large group, consider the Conway group Co_3. Following [67], the group can be generated using two matrices of dimension 22 over $\mathbb{F}_2$. We use the makefile variables to give each generator in compact form. Then we define vectors for each of the generators. We concatenate the two generators to form one long vector, which is passed to the `-subgroup_by_generators` command. Finally, we create a report for the group.

Example 299

```
CONWAY_GEN1="\
1101110001000001010000\
1111010111110100001011\
0000001000000100010101\
1111100110110001001110\
0101010000000010011101\
0000010000000100010101\
0010000000000100010101\
0001000011000000111111\
1110100100110100010011\
0000000000000110010101\
0000000000100100010101\
0110111111010011101111\
0000000000001100010101\
0000000000000100000101\
0000000001000100010101\
0000000000000100011101\
0001000110000010011010\
0000000000000000010101\
0000000000000101010101\
0000000000000100010100\
0000000000000100010111\
0000000000000100010001"
```

Example 300

```
CONWAY_GEN2="\
0101000010111010111111\
0110010100011110110000\
0011010000111111010111\
0001101110001011010011\
1010010000100001011110\
1101000000001010100011\
1100101010001111010101\
1000110100110101010101\
0100110001010000000111\
1100000010100101010010\
0101110110011100000101\
0101111101010011111001\
1000010101010101010001\
0001010000111100100111\
0011010010111011001111\
0100110010110011111010\
1101011001111101100011\
0100101001001000100001\
1100101100001001110011\
0101110110010100000001\
0000001101111000101110\
1101101010101110000101"
```

Example 301

```
Co3:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define g1 -vector -field F -format 22 -compact $(CONWAY_GEN1) -end \
▷  ▷  -define g2 -vector -field F -format 22 -compact $(CONWAY_GEN2) -end \
▷  ▷  -define gens -vector -concatenate g1,g2 -end \
▷  ▷  -define G -linear_group -PGL 22 2 \
▷  ▷  ▷  -subgroup_by_generators "Co3" "495766656000" 2 gens \
▷  ▷  ▷  -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex PGL_22_2_Subgroup_Co3_495766656000_report.tex
▷  #$(OPEN) PGL_22_2_Subgroup_Co3_495766656000_report.pdf
```

The next example creates the Ree group in 7 dimensions over the field $\mathbb{F}_{27}$. Again, we use makefile variables to specify the two generators as 7×7 matrices over $\mathbb{F}_{27}$ and concatenate them, before passing them to the `-subgroup_by_generators` command.

Example 302

```
Ree_gen1="21,5,1,6,17,1,1, 3,13,5,21,6,6,18, 21,3,21,21,22,6,14, \
14,18,1,5,13,6,7, 3,3,2,1,24,16,3, 17,3,22,10,16,24,26, \
21,21,6,18,20,2,5"
```

Example 303

```
Ree_gen2="16,3,11,5,16,22,20, 24,6,18,24,7,1,26, 9,23,17,18,23,20,13, \
9,7,2,15,17,5,11, 3,3,6,21,4,24,16, 25,8,6,24,21,12,7, \
24,15,2,13,11,14,24"
```

Example 304

```
Ree_27:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 27 \
▷  ▷  ▷  -override_polynomial "34" \
▷  ▷  -end \
▷  ▷  -define g1 -vector -field F -format 7 -dense $(Ree_gen1) -end \
▷  ▷  -define g2 -vector -field F -format 7 -dense $(Ree_gen2) -end \
▷  ▷  -define gens -vector -concatenate g1,g2 -end \
▷  ▷  -define G -linear_group -PGL 7 F \
▷  ▷  ▷  -subgroup_by_generators "Ree_27" "10073444472" 2 gens \
▷  ▷  ▷  -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
```

Chapter 7

Orbit Algorithms

7.1 Schreier Trees

Orbiter provides three different orbit algorithms: Schreier trees, poset classification and orderly generation using canonical forms. All algorithms are invoked by means of an orbit data structure. For background on Schreier trees, see [19, 39, 65]. Poset based algorithms will be discussed in Section 7.2. Orderly generation using canonical forms is discussed in Section 16.2.

Table 7.1 lists the Orbiter commands that can be used to compute orbits. An object of type orbit is created. Table 7.2 lists activities for an object of type orbit. Additional options for reporting are listed in Table 7.6.

Consider the group $\mathrm{PGL}(4,2)$ in the natural action on the set of points of $\mathrm{PG}(3,2)$. The degree of the action is 15. The action is transitive. The following example computes the Schreier tree for the action:

Example 305

```
orbits_PGL_4_2_on_points_draw_tree:
▷  $(ORBITER) -v 4 \
▷  ▷  -draw_options -embedded -end \
▷  ▷  -define G -linear_group -PGL 4 2 -end \
▷  ▷  -define Orb -orbits -group G \
▷  ▷  ▷  -on_points \
▷  ▷  -end \
▷  ▷  -with Orb -do -orbits_activity \
▷  ▷  ▷  -report \
▷  ▷  -end \
▷  ▷  -with Orb -do -orbits_activity \
▷  ▷  ▷  -export_something "orbit" 0 \
▷  ▷  -end
▷  ▷  -with Orb -do -orbits_activity \
▷  ▷  ▷  -draw_tree 0 \
▷  ▷  -end
▷  pdflatex PGL_4_2_orbits_report.tex
▷  $(OPEN) PGL_4_2_orbits_report.pdf
```

To draw the tree, the following command is used:

Computing Orbits

Command	Arguments	Purpose
-group	G	Specify the group that acts.
-on_points		Perform Schreier algorithm to compute the orbits of G in the natural action and under the group included in the action.
-on_points_with_generators	gens	Perform Schreier algorithm to compute the orbits of G in the natural action and under the group given by a set of generators.
-on_subsets	k control	Consider the induced action on subsets of size k. This option will invoke the poset algorithm.
-of_one_subset	G set-label	Compute the orbit of the given set under the group G. This algorithm will build up the Schreier tree. The orbit will be saved to a csv file.
-on_subspaces	k control	Consider the induced action on subspaces of size k. We assume that the group action is linear. This option will invoke the poset algorithm.
-on_tensors	d control	Consider the orbits on the d-fold tensor product.
-on_partition	k control	Consider the action on partitions of type $k+k+k$. In this case, $3k$ must equal the degree of the defining action of G.
-on_polynomials	R	Consider the action of G on homogenous polynomials as defined in ring R. The action must be linear and the number of variables in R must match the degree of the matrix group.
-on_one_polynomial	R equation	Consider the action of G on polynomials of degree d in n variables. G must be a matrix group. Here, the action must be linear on an n-dimensional space.
-on_cubic_curves	arc-generator	Classify cubic curves by means of $(9,3)$-arcs. Options are of arc-generator type, see Tab. 7.7.
-on_cubic_surfaces	P control	Classify cubic surfaces in the projective space P, using the given poset classification control options.

Table 7.1: Computing Orbits

Orbit Based Activities		
Command	Arguments	Purpose
-report		Create a report on the orbits.
-export_something	type data	Export orbit data of the given type.
-export_trees		Create drawings of the Schreier trees.
-draw_tree	i	Create a drawing of the i-th orbit as a tree.
-stabilizer	i	Compute the stabilizer of the given point i.
-stabilizer_of_orbit_rep	i	Compute the stabilizer of the orbit representative of orbit i.
-Kramer_Mesner_matrix	t k	Compute the Kramer Mesner matrix of orbits on t-subsets and k-subsets. This option requires a poset type problem.
-recognize	S	Recognize S in the orbit data structure.
-report_options	descr	Set additional parameters for the report, see Table 7.6.

Table 7.2: Orbit Based Activities

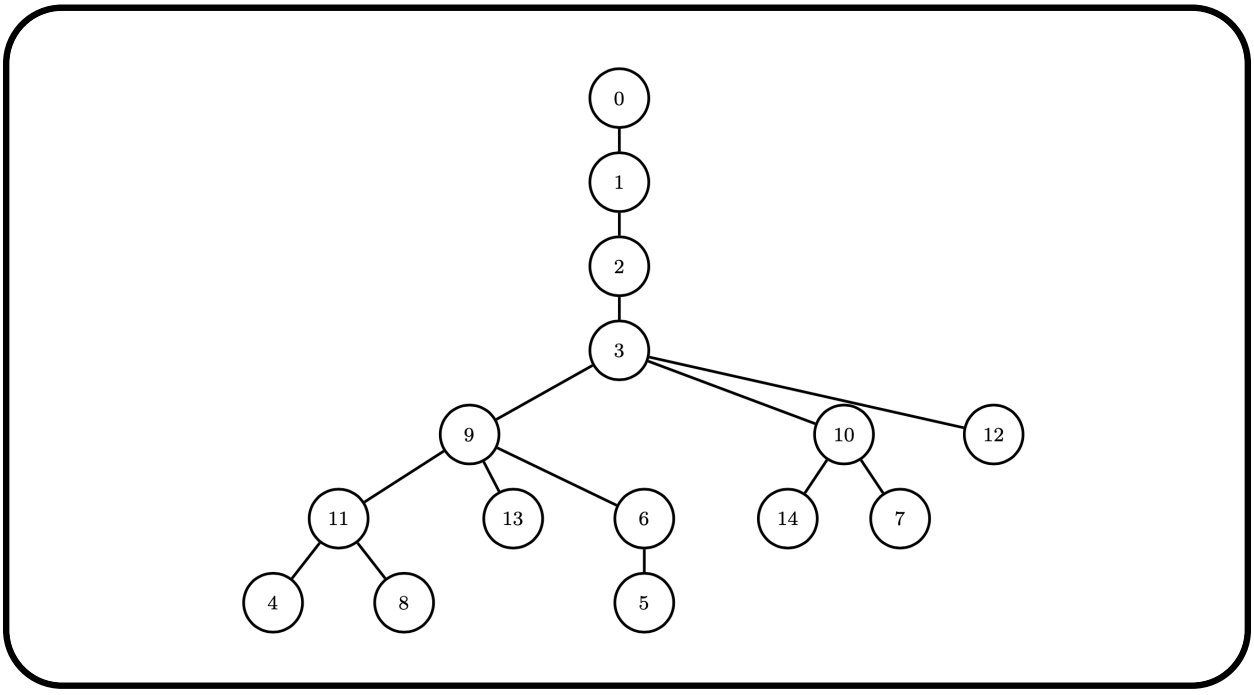
Poset Classification Report Options		
Command	Arguments	Purpose
-select_orbits_by_level	level	Select the levels which should be drawn.
-select_orbits_by_stabilizer_order	m	Select the orbits whose stabilizer order is equal to m.
-select_orbits_by_stabilizer_order_multiple_of	m	Select the orbits whose stabilizer order is a multiple of m.
-include_projective_stabilizer		Include the projective stabilizer in the report. This option assumes that the acting group is a collineation group. The projective stabilizer is a subgroup of the collineation stabilizer.
-draw_poset		Include a drawing of the poset of orbits in the report. This option requires one of the following four options to set the type of drawing.
-type_aux		Poset drawing of auxiliary type. The auxiliary poset includes the flag orbits between layers.
-type_ordinary		Poset drawing of ordinary type.
-type_tree		Poset drawing of type spanning tree.
-type_detailed		Poset drawing of type detailed. The detailed drawing includes the flag orbits between layers.
-fname	fname	Set the file name.

Table 7.3: Poset Classification Report Options

Example 306

```
orbits_PGL_4_2_on_points_export_trees:
▷  $(ORBITER) -v 4 \
▷  ▷  -draw_options -embedded -end \
▷  ▷  -define G -linear_group -PGL 4 2 -end \
▷  ▷  -define Orb -orbits -group G \
▷  ▷  ▷  -on_points \
▷  ▷  -end \
▷  ▷  -with Orb -do -orbits_activity \
▷  ▷  ▷  -report \
▷  ▷  -end \
▷  ▷  -with Orb -do -orbits_activity \
▷  ▷  ▷  -export_trees \
▷  ▷  -end
▷  $(ORBITER) -v 3 \
▷  ▷  -draw_layered_graph \
▷  ▷  ▷  orbit_PGL_4_2_0.layered_graph \
▷  ▷  -radius 500 -spanning_tree -embedded \
▷  ▷  ▷  -line_width 1.1 -x_stretch 1.4 -scale 0.25 \
▷  ▷  -end
▷  pdflatex orbit_PGL_4_2_0_draw.tex
▷  $(OPEN) orbit_PGL_4_2_0_draw.pdf
```

The Schreier tree is shown below

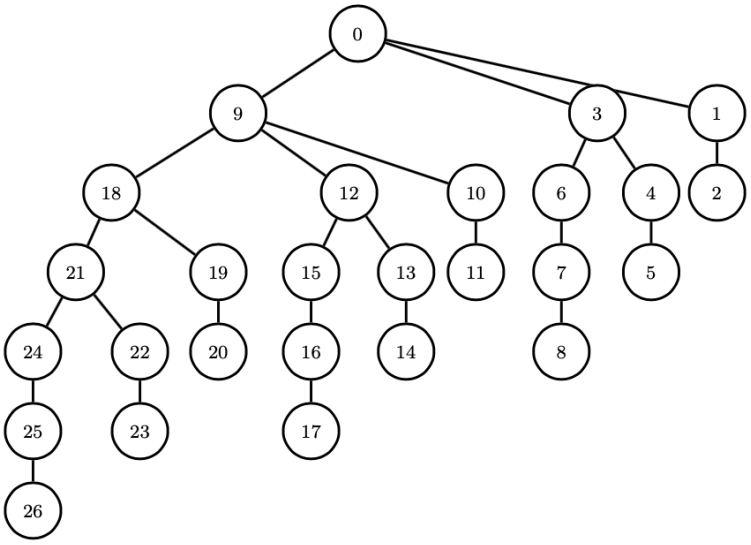


The next command creates the projective plane of order 4 using a line transitive group. The design is the orbit of a certain set of size 5. The group is created in product action. The set results from a search for a special class of linear spaces as described in Section 13.4.

Example 307

```
DD_PP4_orbit:
▷  $(ORBITER) -v 4 \
▷  ▷  -define G1 -linear_group -AGL 1 3 \
▷  ▷  ▷  -end \
▷  ▷  -define G2 -linear_group -AGL 1 7 \
▷  ▷  ▷  -end \
▷  ▷  -define G -modified_group -direct_product "G1,G2" \
▷  ▷  ▷  "21" "1,1,1,1" \
▷  ▷  -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end \
▷  ▷  -define S -vector -dense "0,1,3,13,20" -end \
▷  ▷  -define Orb -orbits -group G \
▷  ▷  ▷  -of_one_subset S \
▷  ▷  -end
```

Consider the wreath product acting on rank-one tensors from Section 6.4. The following command sequence computes the orbits, exports the Schreier tree, and produces the drawing



Example 308

```
T3r1_orbits:
▷  $(ORBITER) -v 4 \
▷  ▷  -draw_options -embedded -end \
▷  ▷  -define G \
▷  ▷  -linear_group -GL_d_q_wr_Sym_n 2 2 3 \
```

```
▷ ▷ ▷ -on_rank_one_tensors -end \
▷ ▷ -define Orb -orbits -group G \
▷ ▷ ▷ -on_points \
▷ ▷ -end \
▷ ▷ -with Orb -do -orbits_activity \
▷ ▷ ▷ -report \
▷ ▷ -end \
▷ ▷ -with Orb -do -orbits_activity \
▷ ▷ ▷ -draw_tree 0 \
▷ ▷ -end
▷ pdflatex GL_2_2_wreath_Sym3_orbit_0_tree.tex
▷ $(OPEN) GL_2_2_wreath_Sym3_orbit_0_tree.pdf
```

In the next example, we compute the orbits of the linear group $\mathrm{PGL}(3,2)$ on plane cubics:

Example 309

```
orbits_cubic_curves_q2:
▷ $(ORBITER) -v 4 \
▷ ▷ -define F -finite_field -q 2 -end \
▷ ▷ -define G -linear_group -PGL 3 F -end \
▷ ▷ -define R -polynomial_ring \
▷ ▷ ▷ -field F \
▷ ▷ ▷ -number_of_variables 3 \
▷ ▷ ▷ -homogeneous_of_degree 3 \
▷ ▷ ▷ -monomial_ordering_partition \
▷ ▷ ▷ -variables "X,Y,Z" "X,Y,Z" \
▷ ▷ -end \
▷ ▷ -define Orb -orbits -group G \
▷ ▷ ▷ -on_polynomials R \
▷ ▷ -end
▷ #pdflatex poly_orbits_d3_n3_q2.tex
▷ #$(OPEN) poly_orbits_d3_n3_q2.pdf
```

In the next example, we compute the orbits of the linear group $\mathrm{PGL}(4,2)$ on homogeneous polynomials of degree 3 in 4 variables:

Example 310

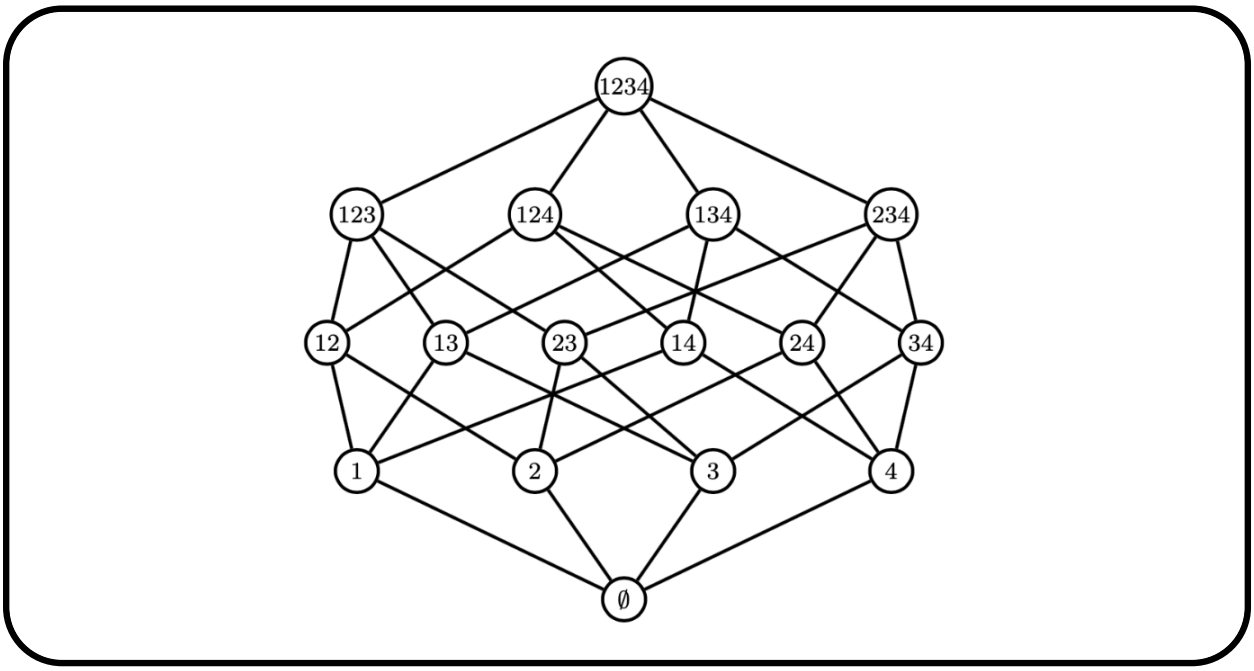
```
poly_orbits_d3_n3_q2.csv:
▷ $(ORBITER) -v 4 \
▷ ▷ -draw_options -yout 500000 -radius 15 -nodes_empty \
▷ ▷ ▷ -line_width 0.5 -y_stretch 0.25 -embedded -end \
▷ ▷ -define F -finite_field -q 2 -end \
▷ ▷ -define G -linear_group -PGL 4 F -end \
▷ ▷ -define R -polynomial_ring \
▷ ▷ ▷ -field F \
▷ ▷ ▷ -number_of_variables 4 \
▷ ▷ ▷ -homogeneous_of_degree 3 \
```

```
▷ ▷ ▷ -monomial_ordering_partition \
▷ ▷ ▷ -variables "X,Y,Z,W" "X,Y,Z,W" \
▷ ▷ -end \
▷ ▷ -define Orb -orbits -group G \
▷ ▷ ▷ -on_polynomials R \
▷ ▷ -end \
▷ ▷ -with Orb -do -orbits_activity \
▷ ▷ ▷ -report \
▷ ▷ -end \
▷ ▷ -with Orb -do -orbits_activity \
▷ ▷ ▷ -draw_tree 6 \
▷ ▷ -end
▷ pdflatex PGL_4_2_orbit_6_tree.tex
▷ $(OPEN) PGL_4_2_orbit_6_tree.pdf
```

This command computes the orbits of all cubic forms in 4 variables over the field $\mathbb{F}_2$, confirming the work of Dickson [26] and an enumerative result of Cooley [22].

7.2 Poset Classification

A partially ordered set (poset) is a set together with a partial order. For instance, the set of subsets of a fixed set form an order structure with respect to set-inclusion. The Hasse diagram is a diagram whose nodes represent the element. Nodes are arranged from top to bottom, and relations are indicated by lines. Consider for example set of subsets of a 4-element set, say $\{1, 2, 3, 4\}$:



Transitivity is implied by following paths in the same direction. For instance, $\{1\}$ is a subset of $\{1, 2\}$, which is a subset of $\{1, 2, 3\}$. There is no line from $\{1\}$ to $\{1, 2, 3\}$ as the relation can be inferred.

Partial orders can be useful in the problem of classifying combinatorial objects by means of substructures. A poset action is group action on a poset. Let G be a group and let $\mathcal{P}$ be a poset. For a poset action, we require that

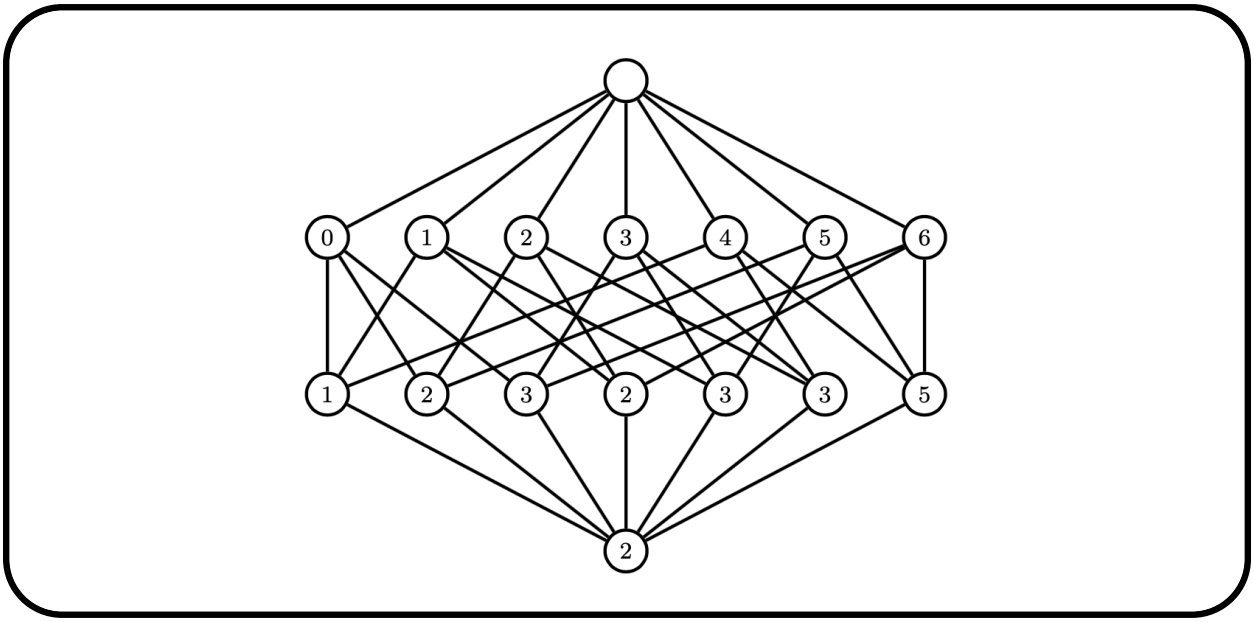
$$x \leq y \Rightarrow xg \leq yg \quad \text{for all } x, y \in \mathcal{P},\ g \in G.$$

For background on poset actions, see Plesken [61]. The orbits of G on $\mathcal{P}$ form another poset, the poset of orbits. A ranked poset can be decomposed into layers according to the rank function. It is then possible to build the poset by considering adjacent levels inductively. In each step, the next layer is constructed from the previous layer and all the layers below. This allows one to reduce a classification problem to a sequence of smaller, and in many cases easier steps.

Orbiter uses the algorithm of Schmalz [64] to perform poset classification. Two versions are available: one for subset-type posets and one for subspace-type posets. The subspace lattice of $V(3, 2) = \mathbb{F}_2^3$ is shown below:

Options for Computing Posets		
Command	Arguments	Purpose
-problem_label	str	Use str as a prefix for files that are created.
-path	p	Use path p for files that are created.
-depth	d	Set search depth to d.
-v	v	Set verbosity to v. Larger numbers mean more output.
-gv	v	Set verbosity for group theoretic operations to v. Larger numbers mean more output.
-recover	fname	Recover from the given file.
-lex		Use the lexicographic ordering to speed up the search.
-w		Save orbits at level d only.
-W		Save orbits at all levels.
-write_data_files		Save data to files.
-t		Write a file containing the search tree at level d.
-T		Write a file containing the search tree at all levels.
-draw_options	options	Drawing options according to Table 17.2.
-preferred_choice	$n\ a\ b$	At node n, choose b instead of a as orbit representative. This option can be repeated.
-clique_test	graph	Classify cliques in the given graph.

Table 7.4: Options for Computing Posets



The basis elements are listed, using the enumerator for elements of the projective geometry PG$(2, 2)$ explained in Section 4.1.

The commands shown in Table 7.4 can be used to control the poset classification algorithm. By default,

Orbiter will choose the lexiographically least orbit representatives. It is possible to direct Orbiter to choose different orbit representatives. To this end, the nodes in the orbit tree are labeled. The node number is the zero-based number of a given node in the tree, using the breadth first ordering.

For posets whose orbits have been computed, there are activities as shown in Table 7.5.

For reporting purposes, the options in Table 7.6 can be used.

Suppose that orbiter chooses element a at node n. Suppose we are interested in choosing element b instead. The command

`-preferred_choice` n a b

can be used to force Orbiter to choose b instead of a at node n.

Activities for Posets of Orbits		
Command	Arguments	Purpose
-report	options	Produce a latex report. For options, see Table 7.6. Requires -orbiter_path option from Section 2.2.
-export_level_to_cpp	i	Export level i to cpp file.
-export_history_to_cpp	i	Export history up to level i to cpp file.
-write_tree		Create the depth first tree from the poset of orbits. A tree file is created.
-find_node_by_stabilizer_order	k	Search for all nodes whose stabilizer has order k.
-draw_poset		Draw the poset of orbits.
-draw_full_poset		Draw the poset of orbits in full.
-plesken		Compute the Plesken matrix of the poset of orbits. See [61].
-print_data_structure		Print the data structure of the poset of orbits.
-list		List the nodes in the poset of orbits.
-list_all		List all nodes in the poset of orbits.
-table_of_nodes		Produce a table of all nodes in the poset of orbits.
-make_relations_with_flag_orbits		Produce a bitmap drawing of the neighboring relations in the poset with flag orbits.
-level_summary_csv		Write a summary of number of orbits at each level to a csv file.
-orbit_reps_csv		Write orbit representatives to a csv file.
-node_label_is_group_order		When drawing the poset of orbits, display the group order in the orbit nodes.
-node_label_is_element		When drawing the poset of orbits, display the element rank in the orbit nodes.
-show_orbit_decomposition		Show the orbits of the stabilizers on the orbit representatives (in the induced action on the set).
-show_stab		Show the stabilizers.
-save_stab		Save the stabilizer generators.
-show_whole_orbits		Show the whole orbits.
-export_schreier_trees		Export all Schreier trees.
-draw_schreier_trees		Draw all Schreier trees.
-test_multi_edge_in_ decomposition_matrix		
-recognize	set	Recognizes the given set in the poset of orbits. This option can be repeated.

Table 7.5: Activities for Posets of Orbits

Poset Classification Report Options

Command	Arguments	Purpose
-select_orbits_by_level	level	Select the levels which should be drawn.
-select_orbits_by_stabilizer_order	m	Select the orbits whose stabilizer order is equal to m.
-select_orbits_by_stabilizer_order_multiple_of	m	Select the orbits whose stabilizer order is a multiple of m.
-include_projective_stabilizer		Include the projective stabilizer in the report. This option assumes that the acting group is a collineation group. The projective stabilizer is a subgroup of the collineation stabilizer.
-draw_poset		Include a drawing of the poset of orbits in the report. This option requires one of the following four options to set the type of drawing.
-type_aux		Poset drawing of auxiliary type. The auxiliary poset includes the flag orbits between layers.
-type_ordinary		Poset drawing of ordinary type.
-type_tree		Poset drawing of type spanning tree.
-type_detailed		Poset drawing of type detailed. The detailed drawing includes the flag orbits between layers.
-fname	fname	Set the file name.

Table 7.6: Poset Classification Report Options

7.3 Orbits on Subsets

The lattice of subsets of a set X is $\mathfrak{P}(X)$, the set of all subsets of X, ordered with respect to inclusion. Assume that a group G acts on X, and hence on the lattice by means of the induced action on subsets. The orbits of G on subsets form a new poset, the poset of orbits. Poset classification is the process of computing the poset of orbits. Orbiter has an algorithm to perform poset classification. In many cases, we are not interested in the full lattice of subsets $\mathfrak{P}(X)$ but rather in a subposet of it. We require that the subposet is closed under the group action and that the following property holds:

$$x, y \in \mathfrak{P}(X) \text{ and } x \leq y \;\Rightarrow\; \Big(y \in \mathcal{P} \rightarrow x \in \mathcal{P}\Big).$$

The join of two subsets in the poset may or may not belong to the poset. Let us consider the action of the Singer cycle on PG$(3,2)$. The following command computes the orbits of the group G generated by a Singer cycle in PG$(3,2)$:

Example 311

```
PGL_3_2_singer:
▷  $(ORBITER) -v 3 \
▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
▷  ▷  -define Control -poset_classification_control \
▷  ▷  ▷  -problem_label PGL_3_2_singer_1 \
▷  ▷  ▷  -W -depth 7 \
▷  ▷  ▷  -draw_options \
▷  ▷  ▷  ▷  -radius 200 -embedded \
▷  ▷  ▷  -end \
▷  ▷  -end \
▷  ▷  -define G -linear_group -PGL 3 2 -singer 1 -end \
▷  ▷  -define Orb -orbits -group G \
▷  ▷  ▷  -on_subsets 7 Control \
▷  ▷  -end \
▷  ▷  -with Orb -do -orbits_activity \
▷  ▷  ▷  -report \
▷  ▷  ▷  -report_options \
▷  ▷  ▷  ▷  -draw_poset -type_ordinary \
▷  ▷  ▷  -end \
▷  ▷  -end
▷  pdflatex PGL_3_2_singer_1_poset.tex
▷  $(OPEN) PGL_3_2_singer_1_poset.pdf
```

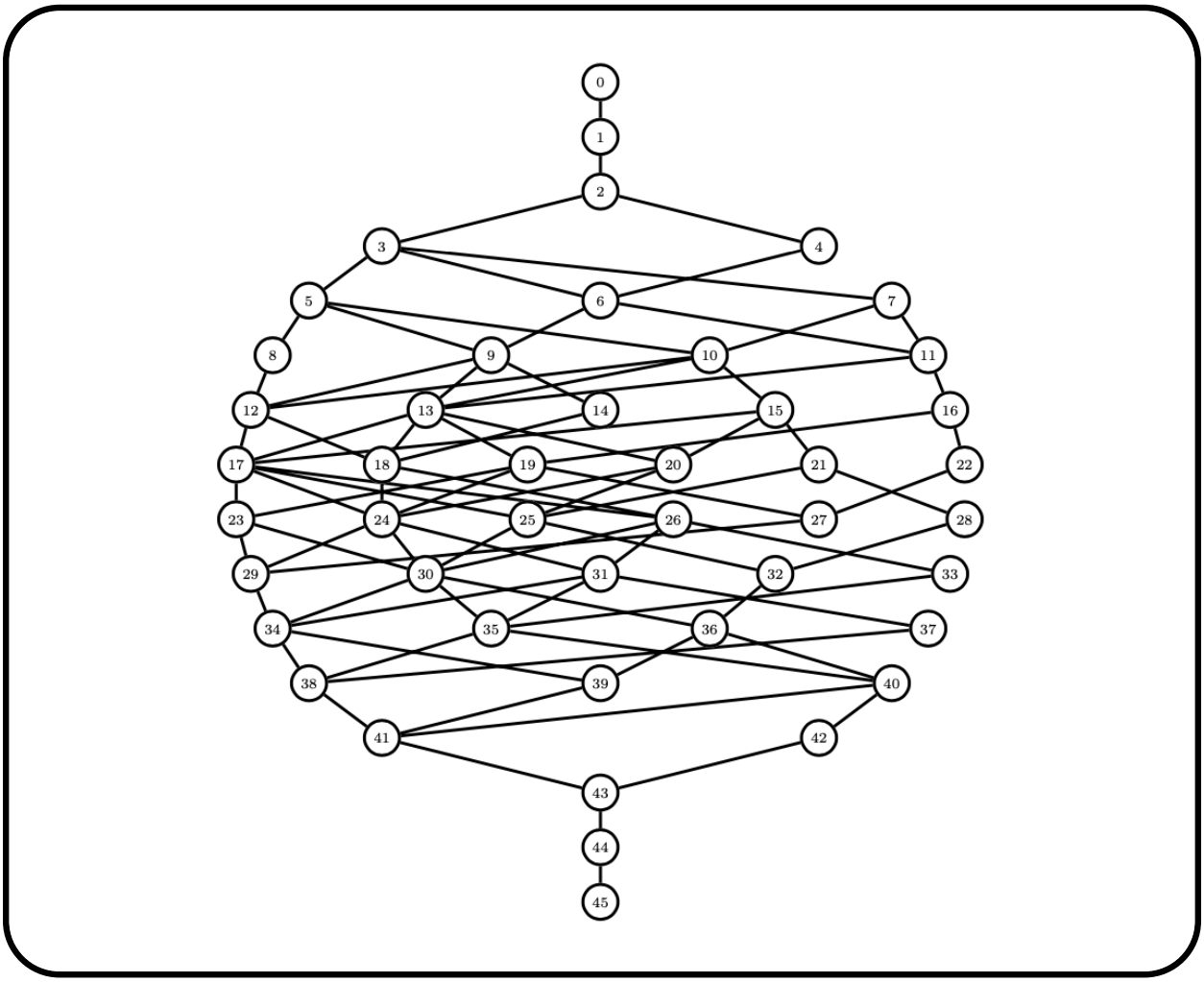
The next command computes the orbits of the projective group PGL$(4,2)$ acting on all subsets of PG$(3,2)$:

Example 312

```
PG_3_2_subsets:
▷  $(ORBITER) -v 3 \
▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
▷  ▷  -define Control -poset_classification_control \
▷  ▷  ▷  -problem_label PGL_4_2 \
▷  ▷  ▷  -depth 15 \
▷  ▷  ▷  -draw_options \
▷  ▷  ▷  ▷  -radius 200 -embedded \
```

```
▷  ▷  ▷  -end \
▷  ▷  -end \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define G -linear_group -PGL 4 F -end \
▷  ▷  -define Orb -orbits -group G \
▷  ▷  ▷  -on_subsets 15 Control \
▷  ▷  -end \
▷  ▷  -with Orb -do -orbits_activity \
▷  ▷  ▷  -report \
▷  ▷  ▷  -report_options \
▷  ▷  ▷  ▷  -draw_poset -type_ordinary \
▷  ▷  ▷  -end \
▷  ▷  -end
▷  pdflatex PGL_4_2_poset.tex
▷  $(OPEN) PGL_4_2_poset.pdf
```

A diagram showing the poset of orbits is produced:



Orbiter can compute orbits of groups acting in various different actions. The following example computes the orbits of $\mathrm{PGL}(3,2)$ on the subsets of lines of $\mathrm{PG}(2,2)$.

Example 313

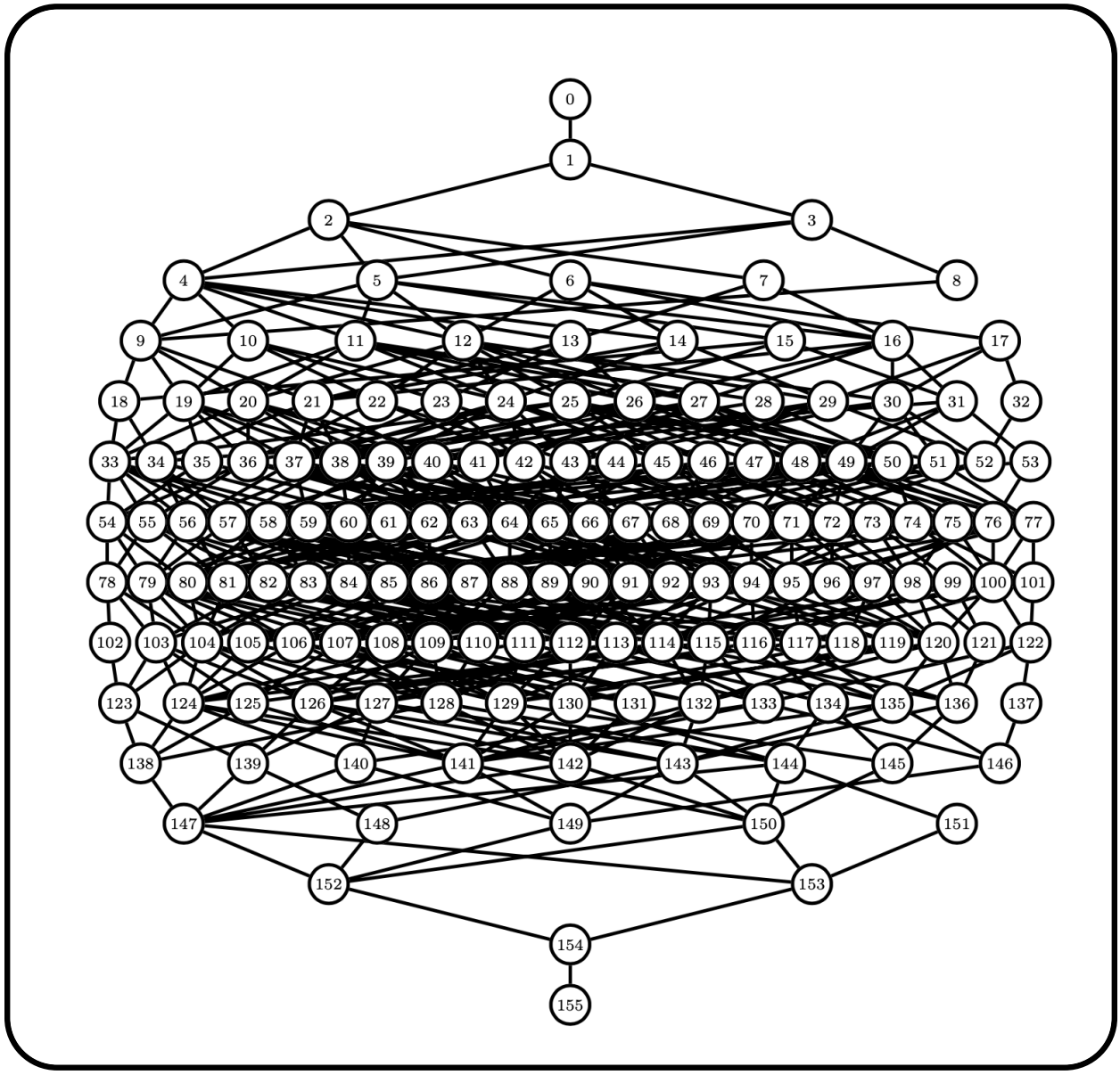
```
PGL_3_2_on_lines:
▷  $(ORBITER) -v 3 \
▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
▷  ▷  -define Control -poset_classification_control \
▷  ▷  ▷  -problem_label PGL_3_2_lines \
▷  ▷  ▷  -W -depth 7 \
▷  ▷  -end \
▷  ▷  -define G -linear_group -PGL 3 2 -end \
▷  ▷  -define G_on_lines -modified_group -from G \
▷  ▷  ▷  -on_k_subspaces 2 \
▷  ▷  -end \
▷  ▷  -define Orb -orbits -group G_on_lines \
▷  ▷  ▷  -on_subsets 7 Control \
▷  ▷  -end \
▷  ▷  -with Orb -do -orbits_activity \
▷  ▷  ▷  -report \
▷  ▷  ▷  -report_options \
▷  ▷  ▷  ▷  -draw_poset -type_ordinary \
▷  ▷  ▷  -end \
▷  ▷  -end
▷  pdflatex PGL_3_2_lines_poset.tex
▷  $(OPEN) PGL_3_2_lines_poset.pdf
```

The following example computes the orbits of $\mathrm{PGO}(5,2)$ on the power set lattice of points of $Q(4,2)$:

Example 314

```
PGO_5_2_on_subsets:
▷  $(ORBITER) -v 3 \
▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
▷  ▷  -define Control -poset_classification_control \
▷  ▷  ▷  -problem_label PGO_5_2 \
▷  ▷  ▷  -depth 15 \
▷  ▷  ▷  -w \
▷  ▷  -end \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define G -linear_group -PGO 5 F -end \
▷  ▷  -define Orb -orbits -group G \
▷  ▷  ▷  -on_subsets 15 Control \
▷  ▷  -end \
▷  ▷  -with Orb -do -orbits_activity \
▷  ▷  ▷  -report \
▷  ▷  ▷  -report_options \
▷  ▷  ▷  ▷  -draw_poset -type_ordinary \
▷  ▷  ▷  -end \
▷  ▷  -end
▷  pdflatex PGO_5_2_poset.tex
▷  $(OPEN) PGO_5_2_poset.pdf
```

The poset of orbits is shown below:



7.4 Orbits on Subspaces

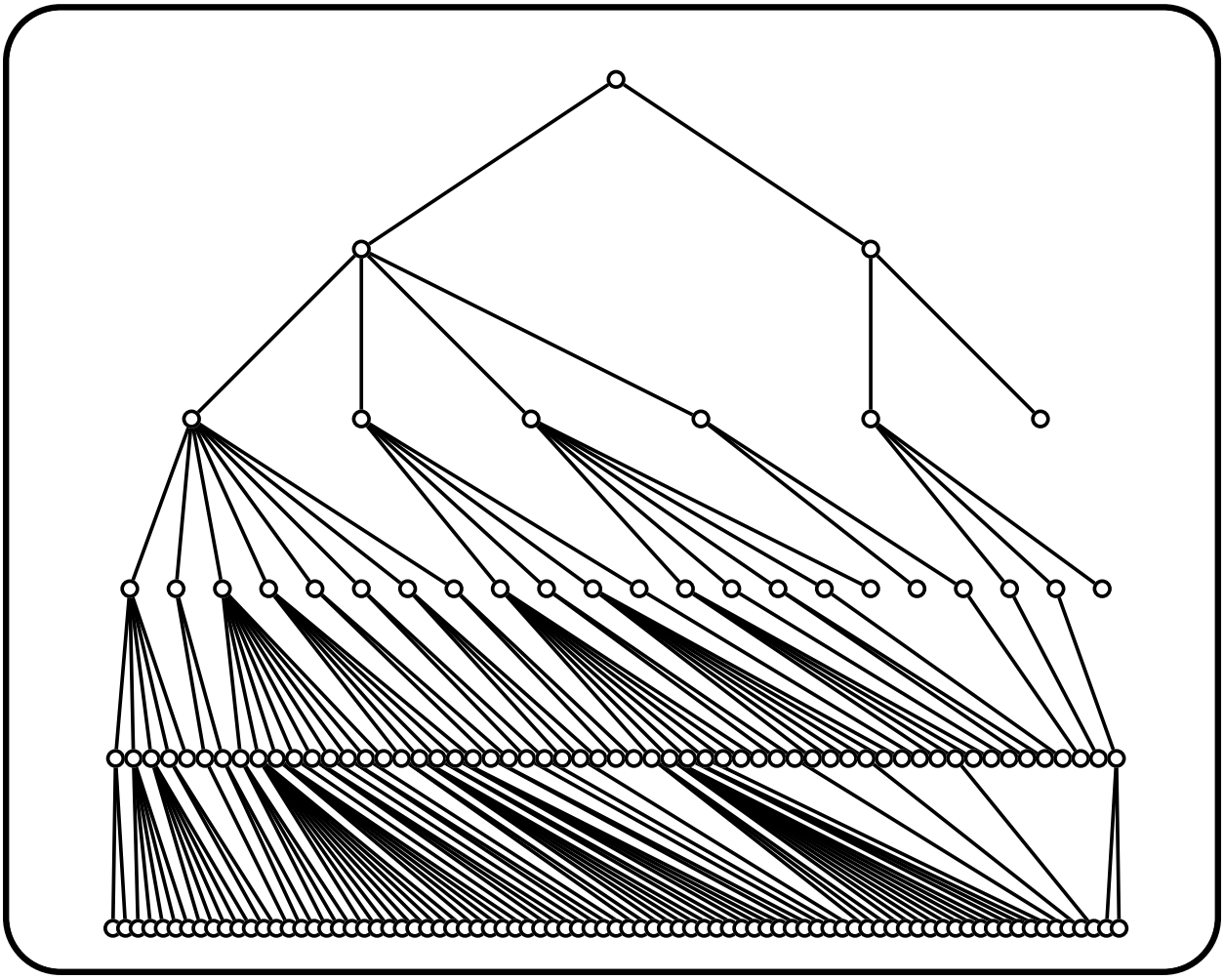
Orbiter can compute the orbits of a group on the lattice of subspaces of a finite vector space.

Suppose we want to compute the orbits of the projective group on the subspaces of the exterior algebra. The following command creates PGL$(5,3)$ in the wedge action as in Section 6.4. It then computes the orbits of subspaces of the exterior algebra of dimension 5. Because of the nature of the poset classification algorithm, all orbits of dimension i with $0 \leq i \leq 5$ are computed.

Example 315

```
PGL_5_3_wedge_subspace_orbits:
▷  $(ORBITER) -v 3 \
▷  ▷  -draw_options -radius 75 -embedded -nodes_empty -end \
▷  ▷  -define G -linear_group -PGL 5 3 -end \
▷  ▷  -define G2 -modified_group -from G \
▷  ▷  ▷  -on_wedge_product \
▷  ▷  -end \
▷  ▷  -define Control -poset_classification_control \
▷  ▷  ▷  -draw_options -radius 200 -embedded -end \
▷  ▷  ▷  -problem_label PGL_5_3_wedge -W -depth 5 \
▷  ▷  -end \
▷  ▷  -define Orb -orbits -group G2 \
▷  ▷  ▷  -on_subspaces 5 Control \
▷  ▷  -end \
▷  ▷  -with Orb -do -orbits_activity \
▷  ▷  ▷  -draw_tree 0 \
▷  ▷  -end
▷  pdflatex PGL_5_3_OnWedge_tree_lvl_5.tex
▷  $(OPEN) PGL_5_3_OnWedge_tree_lvl_5.pdf
```

The computation finds 81 orbits at level 5. The lex-least spanning tree for the orbits is shown below:



The orthogonal group is the stabilizer of a non-degenerate quadric. Suppose we want to classify the subspaces in PG$(3,2)$ under the action of the orthogonal group. In PG$(3,2)$ there are two distinct nondegenerate quadrics, $Q^+(3,2)$ and $Q^-(3,2)$. The $Q^+(3,2)$ quadric is a given by the equation

$$x_0x_1 + x_2x_3 = 0.$$

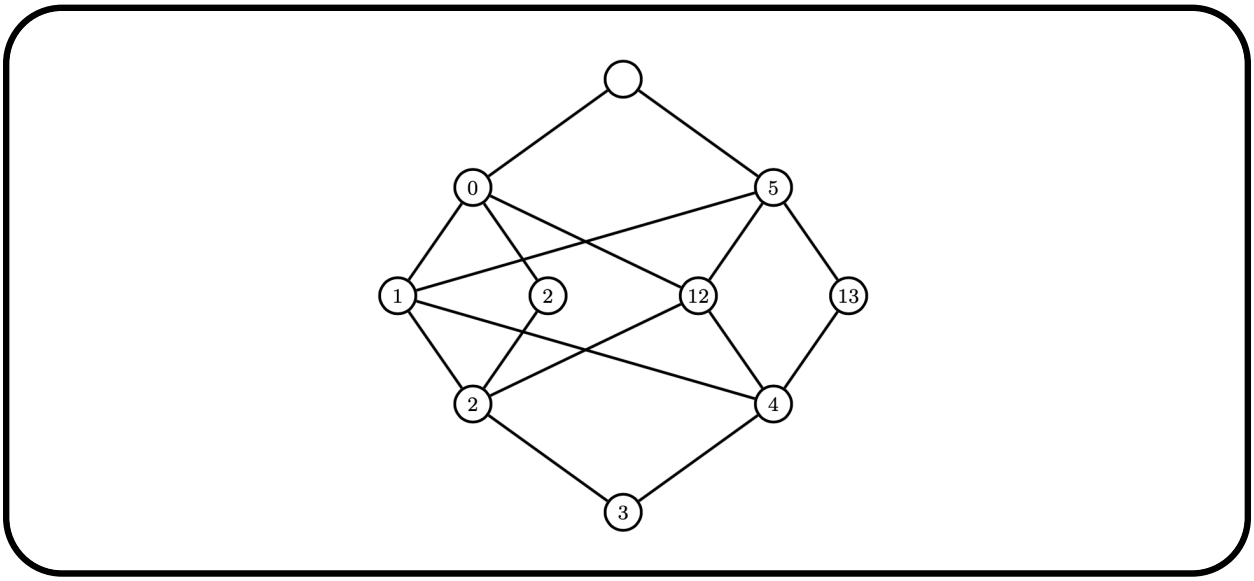
The command

Example 316

```
subspaces_Op_4_2:
▷  $(ORBITER) -v 5 \
▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
▷  ▷  -define Control -poset_classification_control \
▷  ▷  ▷  -draw_options -radius 200 -end \
▷  ▷  ▷  -problem_label Op_4_2 -W -depth 4 \
▷  ▷  -end \
▷  ▷  -define G -linear_group -PGL 4 2 -orthogonal 1 -end \
▷  ▷  -define Orb -orbits -group G \
▷  ▷  ▷  -on_subspaces 4 Control \
▷  ▷  -end \
```

```
▷  ▷  -with Orb -do -orbits_activity \
▷  ▷  ▷  -report \
▷  ▷  ▷  -report_options \
▷  ▷  ▷  ▷  -draw_poset -type_ordinary \
▷  ▷  ▷  -end \
▷  ▷  -end
▷  pdflatex PGL_4_2_Orthogonal_plus_4_2_poset.tex
▷  $(OPEN) PGL_4_2_Orthogonal_plus_4_2_poset.pdf
```

produces a classification of all subspaces of $\mathrm{PG}(3,2)$ under $\mathrm{PGO}^+(4,2)$, shown below:



The nodes show the ranks of points in $\mathrm{PG}(3,2)$ as described in Section 4.1.

7.5 Orbits on Set-Partitions

Orbiter can compute the orbits of a group on set-partitions. The set-partition needs to have three parts of equal size.

The command

Example 317

```
C6_on_partition:
▷ $(ORBITER) -v 5 \
▷ ▷ -orbiter_path $(ORBITER_EXE_PATH) \
▷ ▷ -define Control -poset_classification_control \
▷ ▷ ▷ -problem_label C6 \
▷ ▷ ▷ -depth 2 \
▷ ▷ ▷ -W \
▷ ▷ ▷ -draw_options \
▷ ▷ ▷ ▷ -radius 200 -embedded \
▷ ▷ ▷ -end \
▷ ▷ -end \
▷ ▷ -define G -permutation_group -cyclic_group 6 -end \
▷ ▷ -define Orb -orbits -group G \
▷ ▷ ▷ -on_partition 2 Control \
▷ ▷ -end
```

computes the orbits of the cyclic group C_6 on set-partitions of type $2+2+2$. There are 15 set-partitions, and they fall into 5 orbits, with stabilizer orders

$$3, 1, 2, 2, 6.$$

The orbit count gives

$$6\Big(\frac{1}{3}+\frac{1}{1}+\frac{1}{2}+\frac{1}{2}+\frac{1}{6}\Big) = 15.$$

The command

Example 318

```
PGL_2_17_on_partition:
▷ $(ORBITER) -v 5 \
▷ ▷ -define Control -poset_classification_control \
▷ ▷ ▷ -problem_label PGL_2_17 \
▷ ▷ ▷ -depth 6 \
▷ ▷ ▷ -W \
▷ ▷ -end \
▷ ▷ -define G -linear_group -PGL 2 17 -end \
▷ ▷ -define Orb -orbits -group G \
▷ ▷ ▷ -on_partition 6 Control \
▷ ▷ -end
```

computes the orbits of the group $\mathrm{PGL}(2,17)$ on set-partitions of type $6+6+6$. The number of set-partitions is

$$\frac{\binom{18}{6}\cdot\binom{12}{6}}{3!} = 2858856$$

There are 720 orbits. The orbit stabilizer statistic is

$$(1^{480}, 2^{184}, 3^{11}, 4^{20}, 6^{15}, 8, 12^6, 18, 24, 36).$$

The orbit-stabilizer count confirms that

$$4896\Big(\frac{480}{1} + \frac{184}{2} + \frac{11}{3} + \frac{20}{4} + \frac{15}{6} + \frac{1}{8} + \frac{6}{12} + \frac{1}{18} + \frac{1}{24} + \frac{1}{36} + \Big) = 2858856.$$

Classifying Arcs

Command	Arguments	Purpose
-control	label	Set the label of the poset classification control object.
-projective_space	P	Set the projective space to P.
-d	d	Require that no more than d points lie on a line.
-target_size	t	Specify the size of the arc to be t.
-conic_test		Require that no 6 points of the arc lie on a conic.
-test_nb_Eckardt_points	n	Require exactly n Eckardt points.
-affine		Classify arcs in the affine geometry, assuming that $x_0 = 0$ is the hyperplane at infinity. The condition that no more that d point lie on a line applies to affine lines only.
-no_arc_testing		Do not test the "at most d points per line" condition.
-forbidden_point_set	set	The arc must not contain any of the given points.
-override_group	label	Override the group under which we classify.

Table 7.7: Classifying Arcs

7.6 Arcs and Caps in Projective Spaces

In $\mathrm{PG}(n,q)$, an arc is a set of points, no $n+1$ in a hyperplane. A cap is set of points, no three collinear. Here, we restrict our attention to arcs in $\mathrm{PG}(2,q)$. Arcs in higher dimensional projective spaces are equivalent to MDS codes and will be treated in Section 10. Our main examples will be the construction of the Lunelli-Sce hyperoval in $\mathrm{PG}(2,16)$ (cf. [52]) and the Pellegrino cap in $\mathrm{AG}(4,3)$. The uniqueness of this cap was proven by Hill [34].

A (k,d)-arc in a projective plane π is a set S of k points such that very line intersects S in at most d points. Arcs are related to linear codes and other structures. Two arcs S_1 and S_2 are equivalent if there is a projectivity Φ such that $\Phi(A) = B$. The problem of classifying arcs is the problem of determining the orbits of the projectivity group on arcs. At times, we consider the larger group of collineations. In that case, the problem of classifying arcs is the problem of determining the orbits of the collineation group on arcs. Orbiter can solve such classification problems, at least for small parameter cases. Table 7.7 list the commands available to classify arcs. Here is an example. A hyperoval in a plane $\mathrm{PG}(2,2^e)$ is a $(2^e+2,2)$-arc. It is interesting to classify the hyperovals up to collineation equivalence under the group $\mathrm{P\Gamma L}(3,2^e)$. The command

Example 319

```
hyperoval_16_classify:
▷  $(ORBITER) -v 4 \
▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
▷  ▷  -define F -finite_field -q 16 -end \
▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
▷  ▷  -define Control -poset_classification_control \
▷  ▷  ▷  -problem_label hyperoval_q16 \
```

```
▷ ▷ ▷ -W -depth 18 \
▷ ▷ ▷ -draw_options \
▷ ▷ ▷ ▷ -radius 200 \
▷ ▷ ▷ -end \
▷ ▷ -end \
▷ ▷ -with P -do \
▷ ▷ -projective_space_activity \
▷ ▷ ▷ -classify_arcs \
▷ ▷ ▷ ▷ -control Control \
▷ ▷ ▷ ▷ -target_size 18 \
▷ ▷ ▷ ▷ -d 2 \
▷ ▷ ▷ -end \
▷ ▷ -end
```

performs the classification of hyperovals in PG(2, 16). There are exactly two hyperovals in this plane. Orbiter also finds the stabilizers of these arcs. They have orders 16320 and 144, respectively. The two hyperovals are the regular hyperoval and the Lunelli-Sce hyperoval. Here is the relevant output from the Orbiter report (in the output, the Lunelli-Sce hyperoval is orbit 0, and the regular hyperoval is orbit 1):

Orbits at Level 18

There are 2 orbits at level 18.

Orbit 0 / 2 at Level 18

Node number: 4212

$$\{0, 1, 2, 3, 52, 67, 89, 106, 126, 141, 159, 176, 184, 199, 220, 235, 245, 262\}_{144}$$

0 : 0 = (1, 0, 0)
1 : 1 = (0, 1, 0)
2 : 2 = (0, 0, 1)
3 : 3 = (1, 1, 1)
4 : 52 = (3, 2, 1)
5 : 67 = (2, 3, 1)
6 : 89 = (8, 4, 1)
7 : 106 = (9, 5, 1)
8 : 126 = (13, 6, 1)
9 : 141 = (12, 7, 1)
10 : 159 = (14, 8, 1)
11 : 176 = (15, 9, 1)
12 : 184 = (7, 10, 1)
13 : 199 = (6, 11, 1)
14 : 220 = (11, 12, 1)
15 : 235 = (10, 13, 1)
16 : 245 = (4, 14, 1)
17 : 262 = (5, 15, 1)

Strong generators for a group of order 144:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \delta^4 & \delta^9 & 1 \end{bmatrix}_1 , \begin{bmatrix} 1 & \delta^7 & \delta^{13} \\ \delta^8 & \delta^9 & \delta^{10} \\ \delta & \delta^6 & 1 \end{bmatrix}_3 , \begin{bmatrix} \delta^5 & \delta^5 & \delta^5 \\ \delta^5 & \delta^2 & \delta^{11} \\ \delta^5 & \delta^{14} & 1 \end{bmatrix}_0$$

1,0,0,0,1,0,9,5,1,1,
1,7,6,14,5,10,2,15,1,3,

1,1,1,1,3,15,1,5,10,0,
There are 0 extensions
Number of generators 3

Orbit 1 / 2 at Level 18

Node number: 4213

$$\{0, 1, 2, 3, 52, 70, 83, 109, 127, 139, 156, 174, 186, 199, 217, 229, 256, 264\}_{16320}$$

0 : 0 = (1, 0, 0)
1 : 1 = (0, 1, 0)
2 : 2 = (0, 0, 1)
3 : 3 = (1, 1, 1)
4 : 52 = (3, 2, 1)
5 : 70 = (5, 3, 1)
6 : 83 = (2, 4, 1)
7 : 109 = (12, 5, 1)
8 : 127 = (14, 6, 1)
9 : 139 = (10, 7, 1)
10 : 156 = (11, 8, 1)
11 : 174 = (13, 9, 1)
12 : 186 = (9, 10, 1)
13 : 199 = (6, 11, 1)
14 : 217 = (8, 12, 1)
15 : 229 = (4, 13, 1)
16 : 256 = (15, 14, 1)
17 : 264 = (7, 15, 1)

Strong generators for a group of order 16320:

$$\begin{bmatrix} \delta^{6} & 0 & 0 \\ 0 & \delta^{3} & 0 \\ 0 & 0 & 1 \end{bmatrix}_{2}, \begin{bmatrix} \delta^{9} & 0 & 0 \\ 0 & \delta^{7} & 0 \\ 0 & 0 & 1 \end{bmatrix}_{1}, \begin{bmatrix} \delta^{2} & 0 & 0 \\ 0 & \delta^{11} & 0 \\ \delta^{4} & \delta^{7} & 1 \end{bmatrix}_{3},$$

$$\begin{bmatrix} \delta^{10} & 0 & 0 \\ 0 & \delta^{3} & 0 \\ \delta & \delta^{11} & 1 \end{bmatrix}_{3}, \begin{bmatrix} \delta & 0 & 0 \\ \delta^{12} & \delta^{2} & \delta^{5} \\ \delta^{14} & \delta^{10} & 1 \end{bmatrix}_{1}, \begin{bmatrix} \delta^{5} & 0 & 0 \\ \delta & \delta & \delta \\ \delta^{6} & \delta^{8} & 1 \end{bmatrix}_{0},$$

$$\begin{bmatrix} \delta^{12} & 1 & \delta^{2} \\ \delta^{4} & \delta^{3} & \delta^{7} \\ \delta^{6} & \delta^{3} & 1 \end{bmatrix}_{2}, \begin{bmatrix} \delta^{5} & \delta^{3} & \delta^{6} \\ \delta^{11} & \delta^{6} & \delta^{10} \\ \delta^{10} & \delta^{6} & 1 \end{bmatrix}_{3}$$

1,0,0,0,3,0,0,0,5,2,
1,0,0,0,6,0,0,0,15,1,
1,0,0,0,5,0,4,11,6,3,
1,0,0,0,14,0,15,2,11,3,
1,0,0,13,2,9,6,5,12,1,
1,0,0,13,13,13,2,8,10,0,
1,8,11,7,15,10,5,15,8,2,
1,6,2,15,2,11,11,2,10,3,
There are 0 extensions
Number of generators 8

In the theory of cubic surfaces, we are interested in non-conical arcs of size 6. Here, non-conical means that the set of points does not lie on a conic. Cubic surfaces are associated with non-conical arcs of size 6 (in a many-to-one relationship when considering isomorphism classes). The following example demonstrates how non-conical 6-arecs can be classified in Orbiter:

Example 320

```
nc_arcs_16:
▷  $(ORBITER) -v 4 \
▷  ▷  -define F -finite_field -q 16 -end \
▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
▷  ▷  -define Control -poset_classification_control \
▷  ▷  ▷  -problem_label nc_arcs_q16_d2 \
▷  ▷  ▷  -W -depth 6 \
▷  ▷  -end \
▷  ▷  -with P -do \
▷  ▷  -projective_space_activity \
▷  ▷  ▷  -classify_arcs \
▷  ▷  ▷  ▷  -control Control \
▷  ▷  ▷  ▷  -target_size 6 \
▷  ▷  ▷  ▷  -d 2 \
▷  ▷  ▷  ▷  -conic_test \
▷  ▷  ▷  -end \
▷  ▷  -end
▷  #pdflatex nc_arcs_q16_d2_poset.tex
▷  #$(OPEN) nc_arcs_q16_d2_poset.pdf
```

The number of Eckardt points of the surface can be recovered from properties of the arc. For this reason, it is interesting to classify arcs so that the associated cubic surface has a fixed number of Eckardt points.

7.7 Cubic Curves

Orbiter can classify cubic curves in $\mathrm{PG}(2,q)$. To this end, the $(9,3)$-arcs in $\mathrm{PG}(2,q)$ are classified first. From this classification, the classification of curves is computed. This classification only works for arcs which contain a $(9,3)$ arc. For very small fields, this is not always the case. Here is an example. The command

Example 321

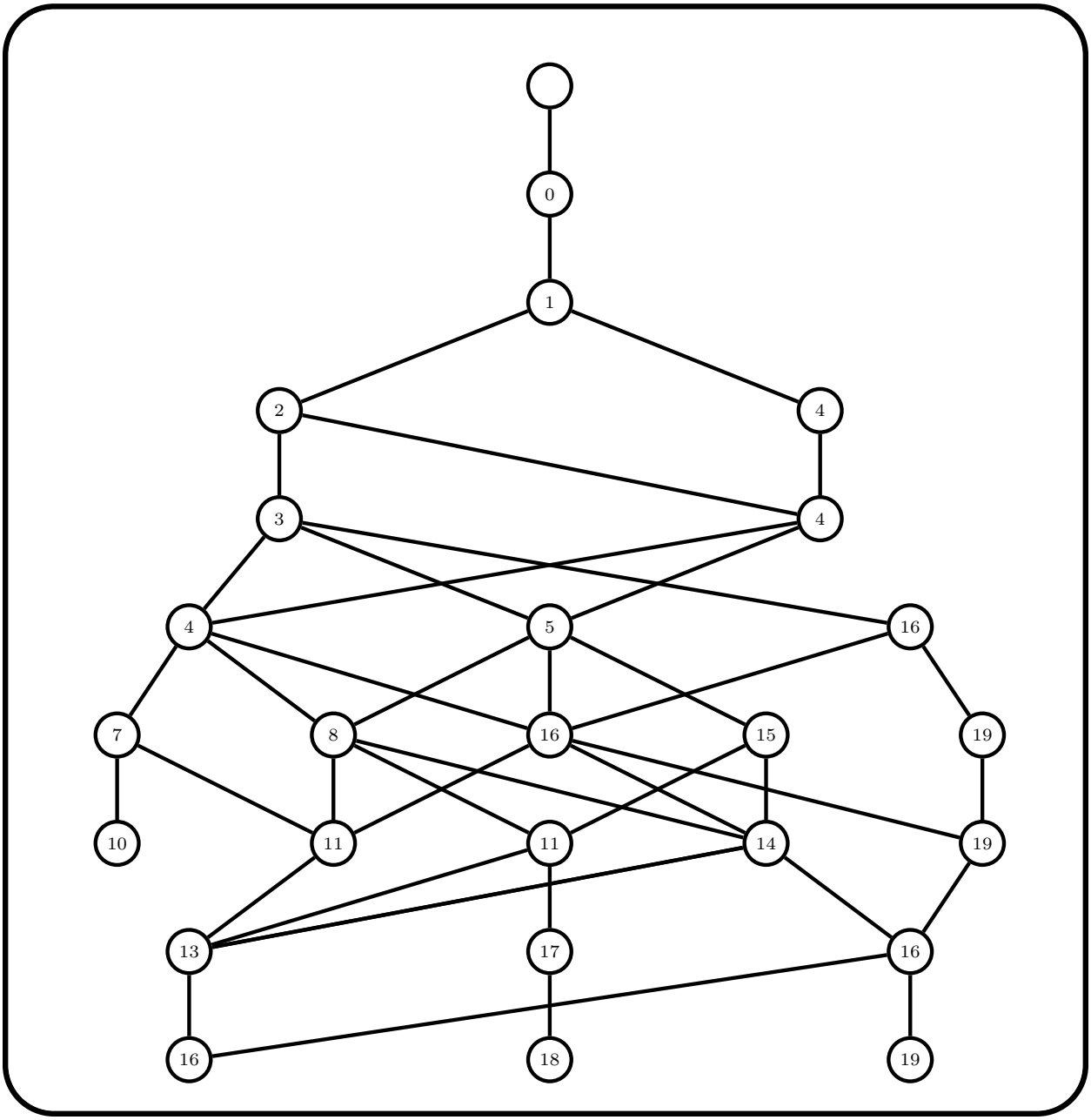
```
cubic_curves_PG_2_4:
▷  $(ORBITER) -v 3 \
▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
▷  ▷  -draw_options -radius 200 -embedded -end \
▷  ▷  -define F -finite_field -q 4 -end \
▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
▷  ▷  -define Control -poset_classification_control \
▷  ▷  ▷  -problem_label cc_4 -W -depth 9 \
▷  ▷  ▷  -draw_options -radius 200 -embedded -end \
▷  ▷  -end \
▷  ▷  -define Arc_control -arc_generator_control \
▷  ▷  ▷  -projective_space P \
▷  ▷  ▷  -control Control \
▷  ▷  ▷  -target_size 9 \
▷  ▷  ▷  -d 3 \
▷  ▷  -end \
▷  ▷  -define Orb -orbits \
▷  ▷  ▷  -on_cubic_curves Arc_control \
▷  ▷  -end \
▷  ▷  -with Orb -do -orbits_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  ▷  -with Orb -do -orbits_activity \
▷  ▷  ▷  -draw_tree 0 \
▷  ▷  -end
▷  pdflatex cc_4_poset_lvl_9.tex
▷  $(OPEN) cc_4_poset_lvl_9.pdf
▷  pdflatex Cubic_curves_q4.tex
▷  $(OPEN) Cubic_curves_q4.pdf
```

classifies the cubic curves in $\mathrm{PG}(2,4)$ containing at least 9 points. There are none. The command

Example 322

```
cubic_curves_PG_2_4_draw:
▷  $(ORBITER) -v 3 \
▷  ▷  -draw_layered_graph cc_4_poset_lvl_9.layered_graph \
▷  ▷  ▷  -radius 300 -embedded -line_width 1.1 \
▷  ▷  ▷  -y_stretch 0.9 -scale 0.25 \
▷  ▷  -end
▷  pdflatex cc_4_poset_lvl_9_draw.tex
▷  $(OPEN) cc_4_poset_lvl_9_draw.pdf
```

draws the poset of orbits of the classification of 9-arcs, shown below:



The command

Example 323

```
poly_orbits_q4:
▷  $(ORBITER) -v 4 \
▷  ▷  -draw_options -yout 500000 -radius 15 -nodes_empty \
▷  ▷  ▷  -line_width 0.5 -y_stretch 0.25 -embedded -end \
▷  ▷  -define F -finite_field -q 4 -print_numerically -end \
▷  ▷  -define R -polynomial_ring \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -number_of_variables 3 \
```

```
▷ ▷ ▷ -homogeneous_of_degree 3 \
▷ ▷ ▷ -monomial_ordering_partition \
▷ ▷ ▷ -variables "X0,X1,X2" "X_0,X_1,X_2" \
▷ ▷ ▷ -end \
▷ ▷ -define G -linear_group -PGGL 3 F -end \
▷ ▷ -define Orb -orbits -group G \
▷ ▷ ▷ -on_polynomials R \
▷ ▷ -end \
▷ ▷ -with Orb -do -orbits_activity \
▷ ▷ ▷ -report \
▷ ▷ -end \
▷ ▷ -with Orb -do -orbits_activity \
▷ ▷ ▷ -export_something "orbit" 0 \
▷ ▷ -end
▷ pdflatex poly_orbits_d3_n2_q4.tex
▷ $(OPEN) poly_orbits_d3_n2_q4.pdf
```

uses the basic Schreier algorithm to classify all cubic curves in PG$(2, 4)$. This is done using the action of the group PΓL$(3, 4)$ on all cubic forms over $\mathbb{F}_4$. The output is shown below:

The Varieties of degree 3 in $PG(2, 4)$, summary

Objects with 13 Points

There are 1 objects with 13 Points:
orbit : rep : go : poly : Pts
4 : (11) : 576 : 11=$X_0^3 + X_1^3$: 2,3,4,5,6,11,12,14,15,16,18,19,20

Objects with 12 Points

There are 1 objects with 12 Points:
orbit : rep : go : poly : Pts
2 : (9) : 108 : 9=$X_0X_1X_2$: 0,1,2,4,5,6,7,8,9,10,13,17

Objects with 10 Points

There are 1 objects with 10 Points:
orbit : rep : go : poly : Pts
25 : (87743) : 20 : 87743=$X_0^3+2X_1^3+2X_2^3+X_0^2X_1+X_0^2X_2+X_0X_1X_2$: 3,5,8,10,13,14,15,17,18,20

Objects with 9 Points

There are 4 objects with 9 Points:
orbit : rep : go : poly : Pts
1 : (3) : 288 : 3=$X_0^2X_1$: 0,1,2,7,8,9,10,13,17
6 : (18) : 432 : 18=$X_0^3 + X_1^3 + X_2^3$: 4,5,6,7,8,9,10,13,17
8 : (24) : 54 : 24=$3X_0^3 + 2X_1^3 + X_2^3$: 3,11,12,14,15,16,18,19,20

13 : (175) : 24 : 175=$X_1^3 + X_2^3 + X_0^2X_1 + X_0^2X_2$: 0,3,4,7,10,11,12,13,17

Objects with 8 Points

There are 2 objects with 8 Points:
orbit : rep : go : poly : Pts
16 : (87383) : 12 : 87383=$X_0^3 + X_0X_1X_2$: 1,2,3,10,13,16,17,19
20 : (87466) : 4 : 87466=$X_1^3 + X_2^3 + X_0^2X_1 + X_0X_1X_2$: 0,3,4,10,13,14,17,18

Objects with 7 Points

There are 1 objects with 7 Points:
orbit : rep : go : poly : Pts
11 : (50) : 6 : 50=$2X_0^3 + X_1^3 + X_2^3 + X_0^2X_1$: 5,10,12,13,14,17,19

Objects with 6 Points

There are 4 objects with 6 Points:
orbit : rep : go : poly : Pts
18 : (87404) : 6 : 87404=$2X_0^3 + X_1^3 + X_2^3 + X_0X_1X_2$: 10,11,13,14,17,20
21 : (87468) : 3 : 87468=$2X_0^3 + X_1^3 + X_2^3 + X_0^2X_1 + X_0X_1X_2$: 5,10,13,15,16,17
23 : (87471) : 4 : 87471=$X_0^3 + 2X_1^3 + X_2^3 + X_0^2X_1 + X_0X_1X_2$: 5,7,8,9,15,16
26 : (91990) : 60 : 91990=$2X_0^2X_2 + X_1^2X_2 + X_0X_1X_2$: 0,1,2,4,5,6

Objects with 5 Points

There are 4 objects with 5 Points:
orbit : rep : go : poly : Pts
0 : (0) : 5760 : 0=X_0^3 : 1,2,10,13,17
9 : (34) : 192 : 34=$2X_0^3 + X_1^3 + X_0^2X_1$: 2,5,11,16,18
10 : (44) : 6 : 44=$X_2^3 + X_0^2X_1$: 0,1,3,15,20
14 : (176) : 8 : 176=$X_0^3 + X_1^3 + X_2^3 + X_0^2X_1 + X_0^2X_2$: 10,13,16,17,19

Objects with 4 Points

There are 3 objects with 4 Points:
orbit : rep : go : poly : Pts
3 : (10) : 12 : 10=$X_0^3 + X_1^3 + X_2^3 + X_0^2X_1 + X_0^2X_2 + X_0X_1^2 + X_1^2X_2 + X_0X_2^2 + X_1X_2^2 + X_0X_1X_2$: 3,4,7,10
17 : (87388) : 6 : 87388=$2X_0^3 + X_1^3 + X_0X_1X_2$: 2,12,15,18
22 : (87470) : 2 : 87470=$2X_1^3 + X_2^3 + X_0^2X_1 + X_0X_1X_2$: 0,6,19,20

Objects with 3 Points

There are 2 objects with 3 Points:
orbit : rep : go : poly : Pts
7 : (19) : 18 : 19=$2X_0^3 + X_1^3 + X_2^3$: 10,13,17

15 : (180) : 9 : 180=$X_0^3 + 2X_1^3 + X_2^3 + X_0^2X_1 + X_0^2X_2$: 5,15,18

Objects with 2 Points

There are 1 objects with 2 Points:
orbit : rep : go : poly : Pts
24 : (87472) : 4 : 87472=$2X_0^3 + 2X_1^3 + X_2^3 + X_0^2X_1 + X_0X_1X_2$: 11,12

Objects with 1 Points

There are 2 objects with 1 Points:
orbit : rep : go : poly : Pts
5 : (12) : 288 : 12=$2X_0^3 + X_1^3$: 2
12 : (55) : 48 : 55=$3X_0^3 + 2X_1^3 + X_2^3 + X_0^2X_1$: 4

Objects with 0 Points

There are 1 objects with 0 Points:
orbit : rep : go : poly : Pts
19 : (87409) : 126 : 87409=$3X_0^3 + 2X_1^3 + X_2^3 + X_0X_1X_2$:

This classification involves curves that are degenerate or singular, or contain lines.

Chapter 8

Cubic Surfaces and Quartic Curves

8.1 Generalities

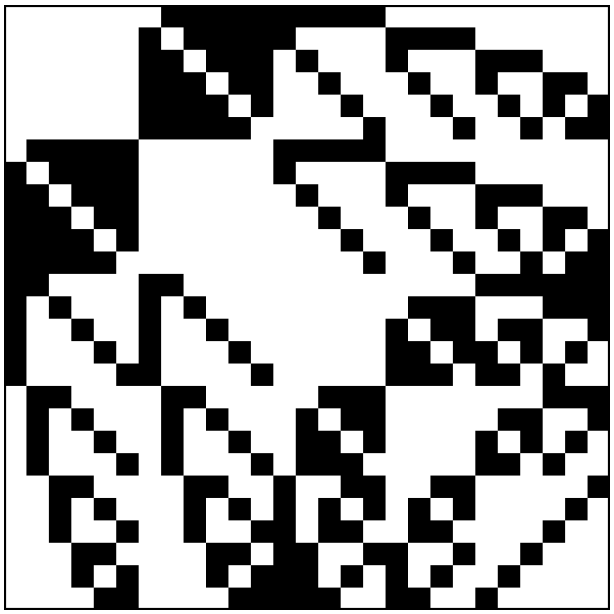
Orbiter can create, classify and investigate cubic surfaces over small finite fields. In this section, some global commands associated with cubic surfaces will be shown. Section 8.2 demonstrates how cubic surfaces can be created in Orbiter. Section 8.5 shows Orbiter commands related to quartic curves. Section 8.4 discusses how cubic surfaces can be classified. Section 8.7 covers the GAP interface for cubic surfaces.

The command

Example 324

```
surface_4_0_lines_vs_lines:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 4 -end \
▷  ▷  -define P -projective_space -n 3 -field F -end \
▷  ▷  -with P -do \
▷  ▷  -projective_space_activity \
▷  ▷  ▷  -export_cubic_surface_line_vs_line_incidence_matrix \
▷  ▷  -end
▷  $(ORBITER) -v 2 -draw_matrix \
▷  ▷  -input_csv_file PG_3_4_lines_vs_lines_incma.csv \
▷  ▷  -box_width 32 -bit_depth 24 \
▷  ▷  -partition 3 27 27 \
▷  ▷  -end
▷  convert PG_3_4_lines_vs_lines_incma_draw.bmp PG_3_4_lines_vs_lines_incma_draw.p
ng
```

exports the adjacency matrix of the graph of incident lines of a cubic surface. The lines are ordered w.r.t. the Schlaefli labeling. The command invokes a projective space activity, and hence creates a projective space first. The order of the field is irrelevant. The command also produces a drawing of the adjacency matrix, shown below:



Using Schlaefli labels, the lines are ordered

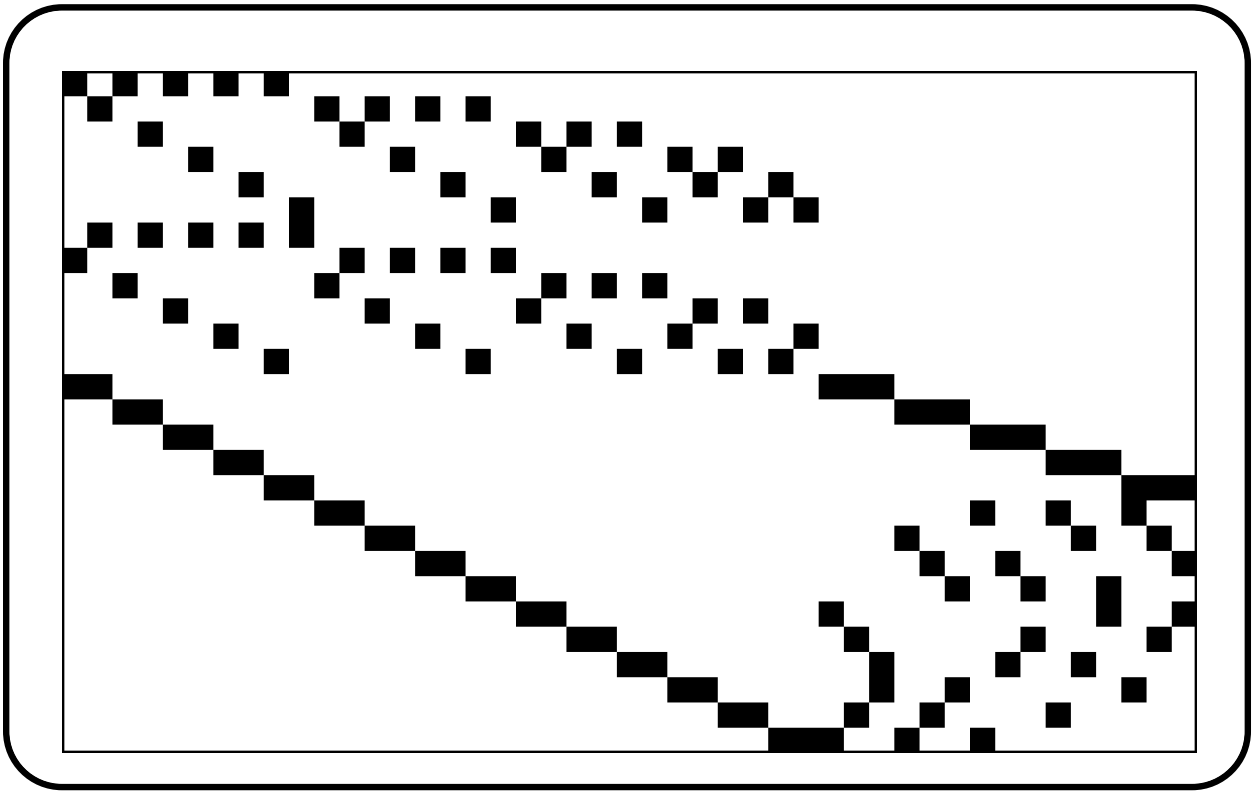
$$a_1, \ldots, a_6, b_1, \ldots, b_6, c_{12}, c_{13}, \ldots, c_{56}.$$

The command

Example 325

```
surface_4_0_lines_vs_tritangent_planes:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 4 -end \
▷  ▷  -define P -projective_space -n 3 -field F -end \
▷  ▷  -with P -do \
▷  ▷  -projective_space_activity \
▷  ▷  ▷  -export_cubic_surface_line_tritangent_plane_incidence_matrix \
▷  ▷  -end
▷  $(ORBITER) -v 2 -draw_matrix \
▷  ▷  -input_csv_file PG_3_4_lines_tritplanes_incma.csv \
▷  ▷  -box_width 32 -bit_depth 24 \
▷  ▷  -partition 3 27 45 \
▷  ▷  -end
▷  convert PG_3_4_lines_tritplanes_incma_draw.bmp PG_3_4_lines_tritplanes_incma_dr
aw.png
```

exports the incidence matrix between lines and tritangent planes of a cubic surface. The lines are ordered w.r.t. the Schlaefli labeling. The tritangent planes are ordered w.r.t. the Schlaefli labeling. The command invokes a projective space activity, and hence creates a projective space first. The order of the field is irrelevant. The command also produces a drawing of the incidence matrix, shown below:



Creating a Cubic Surface (Part 1)

Command	Arguments	Purpose
-space	P	Specify the 3-dimensional projective space P $= \mathrm{PG}(3,q)$.
-label_txt	label	Override the ascii label of the curve.
-label_tex	label	Override the latex label of the curve.
-label_for_summary	label	Override the ascii label of the curve, to be used in summary commands.
-catalogue	i	Create the i-th surface in the Orbiter catalogue. Here, i is an index variable used to index all surfaces in $\mathrm{PG}(3,q)$. The index i is zero-based. The automorphism group is created as well.
-by_coefficients	list-of-coeff-pairs	Create a surface from a list of coefficient-monomial pairs. The automorphism group is not created.
-by_rank	rank q_0	Create a surface from the numerical rank of the equation over the subfield $\mathbb{F}_{q_0}$. Here, we think of the equation as a point in $\mathrm{PG}(19,q_0)$ and use the Orbiter point rank.
-family_Eckardt	a b	Create the Eckardt surface with parameters (a,b) as in see [13] (where it is called the Hilbert, Cohn-Vossen surface). The equation is $X_3^3 - b^2(X_0^2+X_1^2+X_2^2)X_3 + \frac{b^3}{a}(a^2+1)X_0X_1X_2 = 0$. The automorphism group is created as well.

Table 8.1: Creating a Cubic Surface (Part 1)

8.2 Creating Cubic Surfaces

In this section, we describe ways in which surfaces can be created in Orbiter.

Orbiter contains a built-in catalogue of cubic surfaces with 27 lines for small finite fields $\mathbb{F}_q$. The surfaces in the catalogue all come with their automorphism group. It is also possible to create surfaces from known families, or to create surfaces from associated objects like 6-arcs. Some of these constructions only create the surface, not the automorphism group.

Tables 8.1-8.3 summarize the Orbiter commands that can be used to create cubic surfaces. The commands require a projective space object. Not all of the surfaces created may have 27 lines, and some of the constructions may yield degenerate surfaces.
Let us look at some examples. The next command creates the unique surface with 27 lines over the field $\mathbb{F}_4$. It is the Hirschfeld surface. In this command, the surface is pulled from Orbiter's built-in catalogue of cubic surfaces. The surface has Orbiter Catalogue Number (OCN) equal to 0.

Example 326

```
surface_4_0:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 4 -end \
▷  ▷  -define P -projective_space -n 3 -field F -end \
```

Creating a Cubic Surface (Part 2)

Command	Arguments	Purpose
`-family_G13`	a	Create a member of the G_{13} family with parameter a. The surface has 13 Eckardt points.
`-family_F13`	a	Create a member of the F_{13} family with parameter a. The surface has 13 Eckardt points.
`-family_bes`	$a\ c$	Create a member of the "bes"-family with parameter a. The surface has 5 Eckardt points.
`-family_general_abcd`	$a\ b\ c\ d$	Create a member of the general family with parameters a, b, c, d.
`-arc_lifting`	A	Create the surface associated with the arc $A = a_1, \ldots, a_6$ in $\mathrm{PG}(2, q)$ by means of the Clebsch map. Each of the a_i is the rank of a point in $\mathrm{PG}(2, q)$. Use the trihedral pair algorithm. Here, A is a comma-separated string containing the numerical ranks of the P_i in $\mathrm{PG}(2, q)$.
`-arc_lifting_with_two_lines`	$A\ L$	Create the surface associated with the arc $A = a_1, \ldots, a_6$ in $\mathrm{PG}(2, q)$ by means of the Clebsch map, defined by the lines ℓ_1 and ℓ_2 whose ranks are given in L. If both of the lines are given as 0, the program will pick a suitable set of lines automatically.

Table 8.2: Creating a Cubic Surface (Part 2)

Creating a Cubic Surface (Part 3)

Command	Arguments	Purpose
-Cayley_form	$k\ l\ m\ n$	Create the surface from Cayley's normal form, using the parameters k, l, m, n.
-by_equation	R label label_tex vars eqn para para_tex para_values	Create the surface from an equation.
-by_symbolic_object	R formula	Create the surface from an equation. Here, R is a ring and formula is the equation of the surface.
-by_double_six	label label_tex D	Create the surface from a double six D.
-by_skew_hexagon	label label	Create the surface from a skew hexagon.
-select_double_six	L	Relabel the lines by choosing the 12 lines in L as new double six. The entries in L are line indices with respect to the old double six. They are integers in the interval $[0, 26]$. This command can be repeated. In each application, the double six refers to the previous labeling.
-override_group	descr	Override the automorphism group of the surface by the given group.
-random		Create a random cubic surface with 27 lines over the given field.
-transform	elt	Apply the transformation given by the group element.
-transform_inverse	elt	Apply the inverse transformation given by the group element.

Table 8.3: Creating a Cubic Surface (Part 3)

```
▷  ▷  -define S -cubic_surface -space P -catalogue 0 -end \
```

There is a 4-parameter normal form $\mathcal{F}_{abcd}$ for cubic surfaces with 27 lines. Here, a, b, c, d are field elements in $\mathbb{F}_q$. The following command creates the surface $\mathcal{F}_{abcd}$ with $(a, b, c, d) = (2, 3, 3, 4)$ over the field $\mathbb{F}_7$:

Example 327

```
Family_general_F7:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 7 -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷  ▷  -define S7_abcd_2_3_3_4 -cubic_surface \
▷  ▷  ▷  -space P -family_general_abcd 2 3 3 4 \
▷  ▷  -end
```

This command uses a built-in command to realize the surface of type $\mathcal{F}_{abcd}$. We will later see that it is possible to create surfaces symbolically from an equation.

Another way of creating a cubic surface is as member of a known infinite family. However, different choices of the parameters may lead to isomorphic surfaces. For instance,

Example 328

```
surface_eckardt_13_4_12:
▷  $(ORBITER) -v 6 \
▷  ▷  -define F -finite_field -q 13 -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷  ▷  -define Eckardt_4_12 -cubic_surface \
▷  ▷  ▷  -space P -family_Eckardt 4 12 \
▷  ▷  -end
```

creates the member of the Eckardt family described in [13] with parameters $(a, b) = (4, 12)$ over the field $\mathbb{F}_{13}$. On the other hand, the command

Example 329

```
Eckardt_13:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 13 -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷  ▷  -define Eckardt_3_1 -cubic_surface \
▷  ▷  ▷  -space P -family_Eckardt 3 1 \
▷  ▷  -end
```

creates the Eckardt surface over the field $\mathbb{F}_{13}$ from the parameter set $(a, b) = (3, 1)$. The two surfaces are isomorphic.

It is possible to create a random cubic surface of order q. The following example creates a random cubic surface over the field $\mathbb{F}_{11}$:

Example 330

```
surface_11_random:
▷  $(ORBITER) -v 10 \
▷  ▷  -define F -finite_field -q 11 -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 6 -end \
▷  ▷  -define S -cubic_surface -space P -random -end
```

The only restriction is that q is within the range for which the classification of cubic surfaces is available in Orbiter.

A surface can be created symbolically, starting from an equation. For example, consider the surface of type $\mathcal{F}_{abcd}$ from above. We encode the formula in algebraic notation as a makefile variable. Here it is important to note that we use notation such as $1+1+\cdots+1$ (k times) for the integer k. This makes the formula valid over very small fields such as $\mathbb{F}_3$ and $\mathbb{F}_4$ where the Orbiter conventions for field elements prohibit using 3 or 4. There is no automatic reduction modulo the characteristic when specifying field elements.

Example 331

```
F_abcd_eqn="-(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*(b - d)*X0^2*X2 \
+ (a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*(a + b - c - d)*X0*X1*X2 \
+ (a^2*c - a^2*d - a*c^2 + b*c^2 + a*d - b*c)*(b - d)*X0*X1*X3 \
- (a*d - b*c)*(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*X0*X2^2 \
- (a^2*c*d - a*b*c^2 - a^2*d + a*b*d + b*c^2 - b*c*d)*(b - d)*X0*X2*X3 \
- (a - c)*(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*X1^2*X2 \
- (a - c)*(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*X1^2*X3 \
+ (a*d - b*c)*(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*X1*X2^2 \
+ ((1+1)*a^2*b*c*d - a^2*b*d^2 - (1+1)*a^2*c*d^2 \
- (1+1)*a*b^2*c^2 + a*b^2*c*d + (1+1)*a*b*c^2*d + a*b*c*d^2 \
- b^2*c^2*d - a^2*b*c + a^2*c*d + a^2*d^2 + a*b^2*c + a*b*c^2 \
- (1+1+1+1)*a*b*c*d - a*c^2*d + a*c*d^2 + b^2*c^2)*X1*X2*X3 \
+ c*a*(a*d - b*c - a + b + c - d)*(b - d)*X1*X3^2"
```

The following command chooses $(a, b, c, d) = (2, 30, 30, 2)$ and creates the surface over $\mathbb{F}_{31}$.

Example 332

```
surface_F_abcd_Eckardt_q31_by_equation:
▷  $(ORBITER) -v 10 \
▷  ▷  -define F -finite_field -q 31 -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷  ▷  -define R -polynomial_ring \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -number_of_variables 4 \
▷  ▷  ▷  -homogeneous_of_degree 3 \
▷  ▷  ▷  -monomial_ordering_partition \
▷  ▷  ▷  -variables "X0,X1,X2,X3" "X_0,X_1,X_2,X_3" \
▷  ▷  ▷  -end \
▷  ▷  -define F_abcd -cubic_surface -space P \
▷  ▷  ▷  -by_equation \
▷  ▷  ▷  R \
```

```
▷ ▷ ▷ "F_abcd" \
▷ ▷ ▷ "\DF_{a,b,c,d}\D" "X0,X1,X2,X3" \
▷ ▷ ▷ $(F_abcd_eqn) \
▷ ▷ ▷ "a,b,c,d" \
▷ ▷ ▷ "\D(a,b,c,d)=(2,30,30,2)\D" \
▷ ▷ ▷ "2,30,30,2" \
▷ ▷ -end
```

The surface is of Eckardt type.

A cubic surface is determined uniquely up to isomorphism by a non-conical 6-arc in a plane. The points of the arc are given in Orbiter ranks. The following command chooses the arc $0, 1, 2, 3, 43, 113$ and creates the associated cubic surface over $\mathbb{F}_{13}$.

Example 333

```
surface_q13_Eckardt_by_arc_lifting:
▷ $(ORBITER) -v 3 \
▷ ▷ -define F -finite_field -q 13 -end \
▷ ▷ -define P -projective_space -n 3 -field F -v 0 -end \
▷ ▷ -define S -cubic_surface -space P -arc_lifting "0,1,2,3,43,113" -end \
▷ ▷ -with S -do \
▷ ▷ -cubic_surface_activity \
▷ ▷ ▷ -report \
▷ ▷ -end
```

The surface is of Eckardt type.

Cubic Surface Activities

Command	Arguments	Purpose
-report		Produce a latex report about the cubic surface.
-report_group_elements	csv-file heading	Report properties of elements of the automorphism group. The group elements must be given in make-element-form in a column in a csv-file.
-export_something	type	Exports data about the surface to a file. The type of data to be exported must be specified according to Table 8.5.
-export_gap		Export a cubic surface to GAP. See Section 8.7.
-all_quartic_curves		Creates all quartic curves with 28 bitangents from the surface by projecting along the tangent cone of a point not on any line.
-export_all_quartic_curves		Export all quartic curves arising from the orbits of points not on lines of the surface. The curves are written to a csv file.
-export_something_with_group_element	what label	Exports data about the surface to a file. The type of data to be exported must be specified according to Table 8.5.
-action_on_module	type basis generators	
-Clebsch_map_up	line1 line2	Define the birational Clebsch map from a plane to the surface. The skew lines line1 and line2 are used to define the map.
-Clebsch_map_up_single_point	pt line1 line2	Map a single point to the surface under the Clebsch map defined by the pair of lines line1 and line2.

Table 8.4: Cubic Surface Activities

8.3 Cubic Surface Activities

Table 8.4 lists activities for a cubic surface object.
Table 8.5 lists options for exporting data.

The next command creates the Hirschfeld surface over $\mathbb{F}_4$ and produces a report for it:

Example 334

```
surface_4_0_report:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 4 -end \
▷  ▷  -define P -projective_space -n 3 -field F -end \
▷  ▷  -define S -cubic_surface -space P -catalogue 0 -end \
▷  ▷  -with S -do \
▷  ▷  -cubic_surface_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
```

Options for Exporting Data: Cubic Surfaces

Command	Purpose
points	The points on the surface.
points_off	The points off the surface.
Eckardt_points	The Eckardt points of the surface.
double_points	The double points of the surface.
single_points	The single points of the surface.
zero_points	The zero points of the surface. A point is a zero point if it does not lie on any line.
singular_points	The singular points of the surface.
lines	The set of lines of the surface.
tritangent_planes	The set of tritangent planes of the surface.
Hesse_planes	The set of Hesse planes of the surface.

Table 8.5: Options for Exporting Data: Cubic Surfaces

```
▷ pdflatex surface_catalogue_q4_iso0_report.tex
▷ $(OPEN) surface_catalogue_q4_iso0_report.pdf
```

The following command creates the Hirschfeld surface over $\mathbb{F}_4$ and exports some data:

Example 335

```
surface_4_0_export:
▷ $(ORBITER) -v 3 \
▷ ▷ -define F -finite_field -q 4 -end \
▷ ▷ -define P -projective_space -n 3 -field F -end \
▷ ▷ -define S -cubic_surface -space P -catalogue 0 -end \
▷ ▷ -with S -do \
▷ ▷ -cubic_surface_activity \
▷ ▷ ▷ -export_something "points" \
▷ ▷ -end \
▷ ▷ -with S -do \
▷ ▷ -cubic_surface_activity \
▷ ▷ ▷ -export_something "points_off" \
▷ ▷ -end \
▷ ▷ -with S -do \
▷ ▷ -cubic_surface_activity \
▷ ▷ ▷ -export_something "lines" \
▷ ▷ -end \
▷ ▷ -with S -do \
▷ ▷ -cubic_surface_activity \
▷ ▷ ▷ -export_something "Hesse_planes" \
▷ ▷ -end
```

The following data is exported: the set of points on and off the surface, the set of lines, and the set of Hesse planes.

The following command creates the cubic surface over $\mathbb{F}_7$ with catalogue number 0. It also writes a report and exports the set of lines:

Example 336

```
surface_7_0:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 7 -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷  ▷  -define S -cubic_surface -space P -catalogue 0 -end \
▷  ▷  -with S -do \
▷  ▷  -cubic_surface_activity \
▷  ▷  ▷  -report \
▷  ▷  -end \
▷  ▷  -with S -do \
▷  ▷  -cubic_surface_activity \
▷  ▷  ▷  -export_something "lines" \
▷  ▷  -end
▷  pdflatex surface_catalogue_q7_iso0_report.tex
▷  $(OPEN) surface_catalogue_q7_iso0_report.pdf
```

It is often desired to produce a nice equation of a cubic surface. This can be done by applying a change of coordinates. In the following example, a "nice" equation for the cubic surface over $\mathbb{F}_{16}$ with catalogue number 0 is produced:

Example 337

```
surface_16_0:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 16 -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷  ▷  -define S16_0 -cubic_surface -space P -catalogue 0 \
▷  ▷  ▷  -transform "1,0,0,0,0,1,0,12,0,0,1,12,0,0,0,1,0" \
▷  ▷  ▷  -transform "15,11,4,0,0,0,12,0,0,12,0,0,0,0,0,1,3" \
▷  ▷  -end \
▷  ▷  -with S16_0 -do \
▷  ▷  -cubic_surface_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
```

The following command parses the formula and creates the surface with $(a, b, c, d) = (2, 30, 30, 2)$ over $\mathbb{F}_{31}$:

Example 338

```
surface_F_abcd_Eckardt_q31:
▷  $(ORBITER) -v 10 \
▷  ▷  -define F -finite_field -q 31 -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷  ▷  -define R -polynomial_ring \
▷  ▷  ▷  -field F \
```

```
▷ ▷ ▷ -number_of_variables 4 \
▷ ▷ ▷ -homogeneous_of_degree 3 \
▷ ▷ ▷ -monomial_ordering_partition \
▷ ▷ ▷ -variables "X0,X1,X2,X3" "X_0,X_1,X_2,X_3" \
▷ ▷ ▷ -end \
▷ ▷ -define F_abcd -symbolic_object \
▷ ▷ ▷ -field F \
▷ ▷ ▷ -text $(F_abcd_eqn) \
▷ ▷ ▷ -end \
▷ ▷ -define abcd_values -symbolic_object \
▷ ▷ ▷ -field F \
▷ ▷ ▷ -text "2,30,30,2" \
▷ ▷ ▷ -end \
▷ ▷ -define F_abcd_sub -symbolic_object \
▷ ▷ ▷ -field F \
▷ ▷ ▷ -managed_variables "X0,X1,X2,X3" \
▷ ▷ ▷ -substitute "a,b,c,d" F_abcd abcd_values \
▷ ▷ -end \
▷ ▷ -define S -cubic_surface -space P \
▷ ▷ ▷ -by_symbolic_object \
▷ ▷ ▷ R \
▷ ▷ ▷ F_abcd_sub \
▷ ▷ -end \
▷ ▷ -with S -do \
▷ ▷ -cubic_surface_activity \
▷ ▷ ▷ -report \
▷ ▷ -end
▷ pdflatex surface_equation_F_abcd_sub_q31_report.tex
▷ $(OPEN) surface_equation_F_abcd_sub_q31_report.pdf
```

The command creates a report which shows information about the surface. The surface has 6 Eckardt points.

In this case, the following surface is created:

The equation of the surface is :

$$X_0^3 + 2X_1^3 + 4X_2^3 + 7X_3^3 + 7X_0^2X_1 + 2X_0^2X_2 + 2X_0^2X_3 + 9X_1^2X_2 + 9X_1^2X_3 + 6X_0X_2^2$$
$$+9X_1X_2^2 + 4X_2^2X_3 + 10X_0X_3^2 + 9X_1X_3^2 + 3X_2X_3^2 + 8X_0X_1X_2 + 4X_0X_1X_3 = 0$$

(1, 2, 4, 7, 7, 2, 2, 0, 9, 9, 6, 9, 4, 10, 9, 3, 8, 4, 0, 0)

Summary

Object	Number	Orbit type
Lines	27	$12, 15$
Points on surface	199	$10, 15, 24, 30, 60^2$
Singular points	0	
Eckardt points	10	10
Double points	105	$15, 30, 60$
Single points	84	$24, 60$
Points off lines	0	
Hesse planes	0	
Axes	10	10
Single sixes	72	$2, 20^2, 30$
Double sixes	36	$1, 10^2, 15$
Tritangent planes	45	$5, 10, 30$
Trihedral pairs	120	$10, 15^2, 20, 60$
Type of points on lines	12^{27}	
Type of lines on points	3^{10}, 2^{105}, 1^{84}	

Activity for Cubic Surfaces After Classification

Command	Arguments	Purpose
-report	options	Create a latex report.
-stats	prefix	Report statistics on group operations.
-identify_Eckardt		
-identify_F13		
-identify_Bes		
-identify_general_abcd		
-isomorphism_testing	surf1 surf2	Compute an isomorphism from surface 1 to surface 2. surf1 and surf2 must be surface objects.
-recognize	surf	Recognizes the isomorphism type of the given surface object in the Orbiter Catalogue of cubic surfaces.
-create_source_code		Create C++ source code for the cubic surfaces of the given order.
-sweep_Cayley		Identify the isomorphism type of all surfaces of Cayley type.

Table 8.6: Activity for Cubic Surfaces After Classification

8.4 Classification of Cubic Surfaces

There are several different approaches to classify cubic surfaces with 27 lines over finite fields $\mathbb{F}_q$ in Orbiter. Classification means to determine the non-equivalent surfaces under the action of the collineation group $\mathrm{P\Gamma L}(4,q)$ of $\mathrm{PG}(3,q)$. The approach described in [13] relies on Schlaefli's notion of a double six as a substructure [63]. The approach described in [41] utilizes the relation to non-conical six-arcs in a plane. A third approach is described in [42]. All three approaches are available in Orbiter.

Table 8.6 lists the Orbiter commands that can be performed once the classification is completed.

In $\mathrm{PG}(3,4)$, there is only one type of cubic surfaces with 27 lines. It is a member of the Hirschfeld family, described in [36]. The following Orbiter command can be used to construct this surface and to prove its uniqueness for $\mathbb{F}_4$. The following command utilizes the algorithm of [13] to do so:

Example 339

```
surface_classify_q4:
▷  $(ORBITER) -v 5 \
▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
▷  ▷  -define F -finite_field -q 4 -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷  ▷  -define Control -poset_classification_control -W \
▷  ▷  ▷  -problem_label "surf_q4" \
▷  ▷  -end \
▷  ▷  -define report_options -poset_classification_report_options \
▷  ▷  -end \
▷  ▷  -define Orb -orbits \
▷  ▷  ▷  -on_cubic_surfaces P Control \
▷  ▷  -end \
▷  ▷  -with Orb -do -orbits_activity \
```

```
▷ ▷ ▷ -report \
▷ ▷ -end
▷ pdflatex Surfaces_q4.tex
▷ $(OPEN) Surfaces_q4.pdf
```

The `-report` option creates a latex report. After some redactions, the report contains the following elements.

The semilinear group

The Action

Group action $\mathrm{P\Gamma L}(4,4)$ of degree 85
The group is a matrix group.
⋮
The base action is on projective space $\mathrm{PG}(3,4)$
$q = 4$
$p = 2$
$e = 2$
$n = 3$
Number of points = 85
Number of lines = 357
Number of lines on a point = 21
Number of points on a line = 5
⋮

The orthogonal group

The Action

Group action $\mathrm{P\Gamma L}(4,4)$OnWedge of degree 1365
The group is a matrix group.
The base action is on projective space $\mathrm{PG}(3,4)$
$q = 4$
$p = 2$
$e = 2$
$n = 3$
Number of points = 85
Number of lines = 357
Number of lines on a point = 21
Number of points on a line = 5
⋮

The group stabilizing the fixed line

The Action

Group action PΓL(4, 4)OnWedgeres100 of degree 100

⋮

Strong generators for a group of order 5529600: ⋮

The classification of five-plus-ones

Poset classification up to depth 5

The Orbits

Number of Orbits By Level

Depth	Nb of orbits
0	1
1	1
2	1
3	1
4	3
5	4

Summary of Orbit Representatives

N = node
D = depth or level
O = orbit with a level
Rep = orbit representative
(S,O) = (order of stabilizer, orbit length)
L = number of live points
F = number of flags
Gen = number of generators for the stabilizer of the orbit rep.

Table 8.7: Orbit Representatives

N	D	O	Rep	(S,O)	L	F	Gen
0	0	0	{ }	(5529600, 1)	100	1	12
1	1	0	{ 0 }	(55296, 100)	64	1	9
2	2	0	{ 0, 3 }	(1728, 3200)	36	1	7
3	3	0	{ 0, 3, 56 }	(144, 38400)	16	3	6
4	4	0	{ 0, 3, 56, 76 }	(288, 19200)	4	2	7
5	4	1	{ 0, 3, 56, 77 }	(96, 57600)	4	1	6

6	4	2	{ 0, 3, 56, 80 }	(72, 76800)	4	2	6
7	5	0	{ 0, 3, 56, 76, 96 }	(1440, 3840)			10
8	5	1	{ 0, 3, 56, 76, 97 }	(96, 57600)			5
9	5	2	{ 0, 3, 56, 80, 92 }	(360, 15360)			7
10	5	3	{ 0, 3, 56, 80, 93 }	(120, 46080)			7

Poset of Orbits in Detail

⋮

Classification of $5+1$ Configurations in $\mathrm{PG}(3,4)$

The order of the group is 1974067200
The group has 4 orbits on five plus one configurations in $\mathrm{PG}(3,4)$.

Of these, 1 impose 19 conditions.
Of these, 1 are associated with double sixes. They are:
0/1 is orbit 3/4 $\{0,3,56,80,93\}_{120}$ orbit length 46080
The overall number of five plus one configurations associated with double sixes in $\mathrm{PG}(3,4)$ is: 46080

Flag orbits for double sixes

The number of primary orbits below is 4
The number of primary orbits above is 1
The number of flag orbits is 1
The flag orbits are:

(1) Flag orbit 0 / 1 down=(3,0) up=(0,-1) is (0, 3, 56, 80, 93, 16, 340, 38, 61, 156, 0, 16, 340, 38, 61, 156, 165, 155, 72, 54, 25, 356, 0) with a stabilizer of order 120
Strong generators for a group of order 120:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \omega^2 & 0 & 0 \\ 0 & 0 & \omega^2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_1, \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \omega & 0 & 0 \\ \omega^2 & 0 & \omega & 0 \\ 0 & \omega^2 & 0 & 1 \end{bmatrix}_1, \begin{bmatrix} 1 & 0 & 0 & 0 \\ \omega^2 & \omega & 0 & 0 \\ 0 & 0 & \omega & 0 \\ 0 & 0 & \omega^2 & 1 \end{bmatrix}_1,$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ \omega^2 & \omega^2 & 0 & 0 \\ 0 & 0 & \omega^2 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}_0, \begin{bmatrix} 1 & 0 & 0 & 0 \\ \omega^2 & \omega & 0 & 0 \\ \omega & 0 & \omega & 0 \\ \omega^2 & 1 & \omega^2 & 1 \end{bmatrix}_0, \begin{bmatrix} 1 & 1 & 0 & 0 \\ \omega^2 & 0 & 0 & 0 \\ 0 & 0 & \omega^2 & \omega^2 \\ 0 & 0 & 1 & 0 \end{bmatrix}_1,$$

$$\begin{bmatrix} 1 & \omega & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \omega & \omega & 1 & \omega^2 \\ 0 & \omega^2 & 0 & 1 \end{bmatrix}_1$$

1,0,0,0,0,3,0,0,0,0,3,0,0,0,0,1,1,
1,0,0,0,0,2,0,0,3,0,2,0,0,3,0,1,1,
1,0,0,0,3,2,0,0,0,0,2,0,0,0,3,1,1,
1,0,0,0,3,3,0,0,0,0,3,0,0,0,1,1,0,
1,0,0,0,3,2,0,0,2,0,2,0,3,1,3,1,0,
1,1,0,0,3,0,0,0,0,0,3,3,0,0,1,0,1,

1,2,0,0,0,1,0,0,2,2,1,3,0,3,0,1,1,
nb received = 0

Double Sixes

The order of the group is 1974067200
The group has 1 orbits:

The orbits are:

(1) 0/1 $\{16, 340, 38, 61, 156, 165, 155, 72, 54, 25, 356, 0\}_{1440}$ orbit length 1370880
Strong generators for a group of order 1440:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \omega^2 & 0 & 0 \\ 0 & 0 & \omega^2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_1, \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \omega^2 & 0 & 0 \\ \omega^2 & 0 & \omega^2 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}_0, \begin{bmatrix} 1 & 0 & 0 & 0 \\ \omega^2 & 0 & \omega & 0 \\ \omega & \omega & 0 & 0 \\ \omega^2 & \omega^2 & 1 & 1 \end{bmatrix}_0,$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ \omega & 0 & \omega & 0 \\ 1 & \omega & 0 & 0 \\ \omega & 1 & \omega & 1 \end{bmatrix}_1, \begin{bmatrix} 0 & 0 & \omega^2 & 0 \\ 0 & 0 & 1 & \omega^2 \\ \omega^2 & 0 & \omega & 0 \\ \omega & \omega^2 & \omega & 1 \end{bmatrix}_0, \begin{bmatrix} 1 & 1 & 0 & 0 \\ \omega^2 & 0 & 0 & 0 \\ 0 & 0 & \omega^2 & \omega^2 \\ 0 & 0 & 1 & 0 \end{bmatrix}_1$$

1,0,0,0,0,3,0,0,0,0,3,0,0,0,0,1,1,
1,0,0,0,0,3,0,0,3,0,3,0,0,1,0,1,0,
1,0,0,0,3,0,2,0,2,2,0,0,3,3,1,1,0,
1,0,0,0,2,0,2,0,1,2,0,0,2,1,2,1,1,
0,0,1,0,0,0,2,1,1,0,3,0,3,1,3,2,0,
1,1,0,0,3,0,0,0,0,0,3,3,0,0,1,0,1,

The overall number of objects is: 1370880

Flag orbits for surfaces

The number of primary orbits below is 1
The number of primary orbits above is 1
The number of flag orbits is 1
The flag orbits are:

(1) Flag orbit 0 / 1 down=(0,0) up=(0,-1) is (16, 340, 38, 61, 156, 165, 155, 72, 54, 25, 356, 0, 71, 55, 26, 100, 1, 138, 109, 345, 84, 85, 122, 110, 145, 139, 81) with a stabilizer of order 1440
Strong generators for a group of order 1440:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \omega^2 & 0 & 0 \\ 0 & 0 & \omega^2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_1, \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \omega^2 & 0 & 0 \\ \omega^2 & 0 & \omega^2 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}_0, \begin{bmatrix} 1 & 0 & 0 & 0 \\ \omega^2 & 0 & \omega & 0 \\ \omega & \omega & 0 & 0 \\ \omega^2 & \omega^2 & 1 & 1 \end{bmatrix}_0,$$

$$\left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ \omega & 0 & \omega & 0 \\ 1 & \omega & 0 & 0 \\ \omega & 1 & \omega & 1 \end{array}\right]_1, \left[\begin{array}{cccc} 0 & 0 & \omega^2 & 0 \\ 0 & 0 & 1 & \omega^2 \\ \omega^2 & 0 & \omega & 0 \\ \omega & \omega^2 & \omega & 1 \end{array}\right]_0, \left[\begin{array}{cccc} 1 & 1 & 0 & 0 \\ \omega^2 & 0 & 0 & 0 \\ 0 & 0 & \omega^2 & \omega^2 \\ 0 & 0 & 1 & 0 \end{array}\right]_1$$

1,0,0,0,0,3,0,0,0,0,3,0,0,0,0,1,1,
1,0,0,0,0,3,0,0,3,0,3,0,0,1,0,1,0,
1,0,0,0,3,0,2,0,2,2,0,0,3,3,1,1,0,
1,0,0,0,2,0,2,0,1,2,0,0,2,1,2,1,1,
0,0,1,0,0,0,2,1,1,0,3,0,3,1,3,2,0,
1,1,0,0,3,0,0,0,0,0,3,3,0,0,1,0,1,
nb received = 0

Surfaces

The order of the group is 1974067200
The group has 1 orbits:

The orbits are:

(1) 0/1 $\{16, 340, 38, 61, 156, 165, 155, 72, 54, 25, 356, 0, 71, 55, 26, 100, 1, 138, 109, 345, 84, 85, 122, 110, 145, 139, 81\}_{51840}$ orbit length 38080
Strong generators for a group of order 51840:

$$\left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right]_1, \left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & \omega & 0 & 0 \\ 0 & 0 & \omega & 0 \\ 0 & 0 & 0 & 1 \end{array}\right]_0, \left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & \omega^2 & 0 & 0 \\ 0 & 0 & \omega^2 & 0 \\ 1 & 0 & 0 & 1 \end{array}\right]_0,$$

$$\left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{array}\right]_0, \left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ \omega^2 & \omega & \omega & 0 \\ 0 & 0 & \omega & 0 \\ 1 & 0 & \omega^2 & 1 \end{array}\right]_0, \left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 1 & 0 & \omega & 0 \\ \omega & \omega & 0 & 0 \\ \omega & \omega & 1 & 1 \end{array}\right]_0,$$

$$\left[\begin{array}{cccc} \omega^2 & \omega & \omega^2 & 1 \\ \omega^2 & 0 & 1 & 0 \\ \omega & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{array}\right]_0, \left[\begin{array}{cccc} \omega & \omega & 1 & 1 \\ 0 & 1 & 0 & \omega \\ \omega & \omega^2 & 0 & \omega \\ 0 & 1 & 0 & 0 \end{array}\right]_0$$

1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1,
1,0,0,0,0,2,0,0,0,0,2,0,0,0,0,1,0,
1,0,0,0,0,3,0,0,0,0,3,0,1,0,0,1,0,
1,0,0,0,0,1,0,0,1,1,1,0,1,1,0,1,0,
1,0,0,0,3,2,2,0,0,0,2,0,1,0,3,1,0,
1,0,0,0,1,0,2,0,2,2,0,0,2,2,1,1,0,
1,3,1,2,1,0,2,0,3,2,0,0,2,0,0,0,0,
1,1,3,3,0,3,0,1,1,2,0,1,0,3,0,0,0,

The overall number of objects is: 38080

The Group $\mathrm{P\Gamma L}(4,4)$

The order of the group is 1974067200

Cubic Surfaces with 27 Lines in $\mathrm{PG}(3,4)$

The order of the group is 1974067200
The group has 1 orbits:

The orbits are:

(1) 0/1 $\{16, 340, 38, 61, 156, 165, 155, 72, 54, 25, 356, 0, 71, 55, 26, 100, 1, 138, 109, 345, 84, 85, 122, 110, 145, 139, 81\}_{51840}$ orbit length 38080
Strong generators for a group of order 51840:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_1, \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \omega & 0 & 0 \\ 0 & 0 & \omega & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_0, \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \omega^2 & 0 & 0 \\ 0 & 0 & \omega^2 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}_0,$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}_0, \begin{bmatrix} 1 & 0 & 0 & 0 \\ \omega^2 & \omega & \omega & 0 \\ 0 & 0 & \omega & 0 \\ 1 & 0 & \omega^2 & 1 \end{bmatrix}_0, \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & \omega & 0 \\ \omega & \omega & 0 & 0 \\ \omega & \omega & 1 & 1 \end{bmatrix}_0,$$

$$\begin{bmatrix} \omega^2 & \omega & \omega^2 & 1 \\ \omega^2 & 0 & 1 & 0 \\ \omega & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}_0, \begin{bmatrix} \omega & \omega & 1 & 1 \\ 0 & 1 & 0 & \omega \\ \omega & \omega^2 & 0 & \omega \\ 0 & 1 & 0 & 0 \end{bmatrix}_0$$

1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1,
1,0,0,0,0,2,0,0,0,0,2,0,0,0,0,1,0,
1,0,0,0,0,3,0,0,0,0,3,0,1,0,0,1,0,
1,0,0,0,0,1,0,0,1,1,1,0,1,1,0,1,0,
1,0,0,0,3,2,2,0,0,0,2,0,1,0,3,1,0,
1,0,0,0,1,0,2,0,2,2,0,0,2,2,1,1,0,
1,3,1,2,1,0,2,0,3,2,0,0,2,0,0,0,0,
1,1,3,3,0,3,0,1,1,2,0,1,0,3,0,0,0,

The overall number of objects is: 38080

Surface 4#0

The equation

The equation of the surface is :

$$X_0^2X_3 + X_1^2X_2 + X_1X_2^2 + X_0X_3^2 = 0$$

(0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0)
Number of points on the surface 45
The automorphism group of the surface has order 51840
The automorphism group is the following group
Strong generators for a group of order 51840:

$$\left[\begin{array}{cccc} 1&0&0&0\\ 0&1&0&0\\ 0&0&1&0\\ 0&0&0&1 \end{array}\right]_1, \left[\begin{array}{cccc} 1&0&0&0\\ 0&\omega&0&0\\ 0&0&\omega&0\\ 0&0&0&1 \end{array}\right]_0, \left[\begin{array}{cccc} 1&0&0&0\\ 0&\omega^2&0&0\\ 0&0&\omega^2&0\\ 1&0&0&1 \end{array}\right]_0,$$

$$\left[\begin{array}{cccc} 1&0&0&0\\ 0&1&0&0\\ 1&1&1&0\\ 1&1&0&1 \end{array}\right]_0, \left[\begin{array}{cccc} 1&0&0&0\\ \omega^2&\omega&\omega&0\\ 0&0&\omega&0\\ 1&0&\omega^2&1 \end{array}\right]_0, \left[\begin{array}{cccc} 1&0&0&0\\ 1&0&\omega&0\\ \omega&\omega&0&0\\ \omega&\omega&1&1 \end{array}\right]_0,$$

$$\left[\begin{array}{cccc} \omega^2&\omega&\omega^2&1\\ \omega^2&0&1&0\\ \omega&1&0&0\\ 1&0&0&0 \end{array}\right]_0, \left[\begin{array}{cccc} \omega&\omega&1&1\\ 0&1&0&\omega\\ \omega&\omega^2&0&\omega\\ 0&1&0&0 \end{array}\right]_0$$

1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1,
1,0,0,0,0,2,0,0,0,0,2,0,0,0,0,1,0,
1,0,0,0,0,3,0,0,0,0,3,0,1,0,0,1,0,
1,0,0,0,0,1,0,0,1,1,1,0,1,1,0,1,0,
1,0,0,0,3,2,2,0,0,0,2,0,1,0,3,1,0,
1,0,0,0,1,0,2,0,2,2,0,0,2,2,1,1,0,
1,3,1,2,1,0,2,0,3,2,0,0,2,0,0,0,0,
1,1,3,3,0,3,0,1,1,2,0,1,0,3,0,0,0,

General information

Points on lines:

$$5^{27}$$

Lines on points:

$$3^{45}$$

The 27 Lines

$$\ell_0 = a_1 = \left[\begin{array}{cccc} 1&0&\omega^2&0\\ 0&1&1&\omega \end{array}\right]_{72} = \left[\begin{array}{cccc} 1&0&3&0\\ 0&1&1&2 \end{array}\right]_{72} = \mathbf{Pl}(3,2,3,0,3,1)_{308}$$

$$\ell_1 = a_2 = \left[\begin{array}{cccc} 1&0&\omega&0\\ 0&1&0&\omega^2 \end{array}\right]_{54} = \left[\begin{array}{cccc} 1&0&2&0\\ 0&1&0&3 \end{array}\right]_{54} = \mathbf{Pl}(2,3,0,0,2,1)_{238}$$

$$\ell_2 = a_3 = \left[\begin{array}{cccc} 1&0&1&0\\ 0&1&0&1 \end{array}\right]_{25} = \left[\begin{array}{cccc} 1&0&1&0\\ 0&1&0&1 \end{array}\right]_{25} = \mathbf{Pl}(1,1,0,0,1,1)_{177}$$

$$\ell_3 = a_4 = \left[\begin{array}{cccc} 0&0&1&0\\ 0&0&0&1 \end{array}\right]_{356} = \left[\begin{array}{cccc} 0&0&1&0\\ 0&0&0&1 \end{array}\right]_{356} = \mathbf{Pl}(0,1,0,0,0,0)_{1}$$

$$\ell_4 = a_5 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}_0 = \mathbf{Pl}(1,0,0,0,0,0)_0$$

$$\ell_5 = a_6 = \begin{bmatrix} 1 & 0 & \omega^2 & 1 \\ 0 & 1 & 0 & \omega \end{bmatrix}_{155} = \begin{bmatrix} 1 & 0 & 3 & 1 \\ 0 & 1 & 0 & 2 \end{bmatrix}_{155} = \mathbf{Pl}(3,2,0,2,3,1)_{314}$$

$$\ell_6 = b_1 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_{340} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_{340} = \mathbf{Pl}(0,0,0,1,0,0)_9$$

$$\ell_7 = b_2 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}_{38} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}_{38} = \mathbf{Pl}(0,0,1,1,1,1)_{198}$$

$$\ell_8 = b_3 = \begin{bmatrix} 1 & \omega & 0 & 0 \\ 0 & 0 & 1 & \omega^2 \end{bmatrix}_{61} = \begin{bmatrix} 1 & 2 & 0 & 0 \\ 0 & 0 & 1 & 3 \end{bmatrix}_{61} = \mathbf{Pl}(0,0,2,3,2,1)_{265}$$

$$\ell_9 = b_4 = \begin{bmatrix} 1 & 0 & \omega^2 & 1 \\ 0 & 1 & 1 & \omega \end{bmatrix}_{156} = \begin{bmatrix} 1 & 0 & 3 & 1 \\ 0 & 1 & 1 & 2 \end{bmatrix}_{156} = \mathbf{Pl}(3,0,3,2,3,1)_{335}$$

$$\ell_{10} = b_5 = \begin{bmatrix} 1 & \omega^2 & 0 & 1 \\ 0 & 0 & 1 & \omega \end{bmatrix}_{165} = \begin{bmatrix} 1 & 3 & 0 & 1 \\ 0 & 0 & 1 & 2 \end{bmatrix}_{165} = \mathbf{Pl}(0,2,3,2,3,1)_{337}$$

$$\ell_{11} = b_6 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}_{16} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}_{16} = \mathbf{Pl}(0,0,1,0,0,0)_2$$

$$\ell_{12} = c_{12} = \begin{bmatrix} 1 & 0 & \omega & 1 \\ 0 & 1 & 0 & \omega^2 \end{bmatrix}_{138} = \begin{bmatrix} 1 & 0 & 2 & 1 \\ 0 & 1 & 0 & 3 \end{bmatrix}_{138} = \mathbf{Pl}(2,3,0,3,2,1)_{256}$$

$$\ell_{13} = c_{13} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}_{109} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}_{109} = \mathbf{Pl}(1,1,0,1,1,1)_{189}$$

$$\ell_{14} = c_{14} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_{345} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_{345} = \mathbf{Pl}(0,1,0,1,0,0)_{13}$$

$$\ell_{15} = c_{15} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}_{84} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}_{84} = \mathbf{Pl}(1,0,0,1,0,0)_{10}$$

$$\ell_{16} = c_{16} = \begin{bmatrix} 1 & 0 & \omega^2 & 0 \\ 0 & 1 & 0 & \omega \end{bmatrix}_{71} = \begin{bmatrix} 1 & 0 & 3 & 0 \\ 0 & 1 & 0 & 2 \end{bmatrix}_{71} = \mathbf{Pl}(3,2,0,0,3,1)_{299}$$

$$\ell_{17} = c_{23} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}_{85} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}_{85} = \mathbf{Pl}(1,1,1,1,0,0)_{16}$$

$$\ell_{18} = c_{24} = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}_{122} = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}_{122} = \mathbf{Pl}(0,1,1,1,1,1)_{202}$$

$$\ell_{19} = c_{25} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}_{110} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}_{110} = \mathbf{Pl}(1,0,1,1,1,1)_{199}$$

$$\ell_{20} = c_{26} = \begin{bmatrix} 1 & 0 & \omega & 0 \\ 0 & 1 & 1 & \omega^2 \end{bmatrix}_{55} = \begin{bmatrix} 1 & 0 & 2 & 0 \\ 0 & 1 & 1 & 3 \end{bmatrix}_{55} = \mathbf{Pl}(2,3,2,0,2,1)_{244}$$

$$\ell_{21} = c_{34} = \left[\begin{array}{cccc} 1 & \omega & 0 & 1 \\ 0 & 0 & 1 & \omega^2 \end{array}\right]_{145} = \left[\begin{array}{cccc} 1 & 2 & 0 & 1 \\ 0 & 0 & 1 & 3 \end{array}\right]_{145} = \mathbf{Pl}(0,3,2,3,2,1)_{271}$$

$$\ell_{22} = c_{35} = \left[\begin{array}{cccc} 1 & 0 & \omega & 1 \\ 0 & 1 & 1 & \omega^2 \end{array}\right]_{139} = \left[\begin{array}{cccc} 1 & 0 & 2 & 1 \\ 0 & 1 & 1 & 3 \end{array}\right]_{139} = \mathbf{Pl}(2,0,2,3,2,1)_{267}$$

$$\ell_{23} = c_{36} = \left[\begin{array}{cccc} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{array}\right]_{26} = \left[\begin{array}{cccc} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{array}\right]_{26} = \mathbf{Pl}(1,1,1,0,1,1)_{180}$$

$$\ell_{24} = c_{45} = \left[\begin{array}{cccc} 1 & \omega^2 & 0 & 0 \\ 0 & 0 & 1 & \omega \end{array}\right]_{81} = \left[\begin{array}{cccc} 1 & 3 & 0 & 0 \\ 0 & 0 & 1 & 2 \end{array}\right]_{81} = \mathbf{Pl}(0,0,3,2,3,1)_{332}$$

$$\ell_{25} = c_{46} = \left[\begin{array}{cccc} 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{array}\right]_{100} = \left[\begin{array}{cccc} 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{array}\right]_{100} = \mathbf{Pl}(0,1,1,0,0,0)_{6}$$

$$\ell_{26} = c_{56} = \left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{array}\right]_{1} = \left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{array}\right]_{1} = \mathbf{Pl}(1,0,1,0,0,0)_{3}$$

Rank of lines: (72, 54, 25, 356, 0, 155, 340, 38, 61, 156, 165, 16, 138, 109, 345, 84, 71, 85, 122, 110, 55, 145, 139, 26, 81, 100, 1)
Rank of points on Klein quadric: (308, 238, 177, 1, 0, 314, 9, 198, 265, 335, 337, 2, 256, 189, 13, 10, 299, 16, 202, 199, 244, 271, 267, 180, 332, 6, 3)

All Points on surface

The surface has 45 points

Eckardt Points

The surface has 45 Eckardt points:
$0 : E_{56} = a_5 \cap b_6 \cap c_{56} = P_0 = P_0 = \mathbf{P}(1,0,0,0) = \mathbf{P}(1,0,0,0),\ T = 0$
$1 : E_{51} = a_5 \cap b_1 \cap c_{15} = P_1 = P_1 = \mathbf{P}(0,1,0,0) = \mathbf{P}(0,1,0,0),\ T = 4$
$2 : E_{46} = a_4 \cap b_6 \cap c_{46} = P_2 = P_2 = \mathbf{P}(0,0,1,0) = \mathbf{P}(0,0,1,0),\ T = 20$
$3 : E_{41} = a_4 \cap b_1 \cap c_{14} = P_3 = P_3 = \mathbf{P}(0,0,0,1) = \mathbf{P}(0,0,0,1),\ T = 84$
$4 : E_{32} = a_3 \cap b_2 \cap c_{23} = P_4 = P_4 = \mathbf{P}(1,1,1,1) = \mathbf{P}(1,1,1,1),\ T = 27$
$5 : E_{52} = a_5 \cap b_2 \cap c_{25} = P_5 = P_5 = \mathbf{P}(1,1,0,0) = \mathbf{P}(1,1,0,0),\ T = 1$
$6 : E_{54} = a_5 \cap b_4 \cap c_{45} = P_6 = P_6 = \mathbf{P}(\omega,1,0,0) = \mathbf{P}(2,1,0,0),\ T = 2$
$7 : E_{53} = a_5 \cap b_3 \cap c_{35} = P_7 = P_7 = \mathbf{P}(\omega^2,1,0,0) = \mathbf{P}(3,1,0,0),\ T = 3$
$8 : E_{36} = a_3 \cap b_6 \cap c_{36} = P_8 = P_8 = \mathbf{P}(1,0,1,0) = \mathbf{P}(1,0,1,0),\ T = 5$
$9 : E_{16} = a_1 \cap b_6 \cap c_{16} = P_9 = P_9 = \mathbf{P}(\omega,0,1,0) = \mathbf{P}(2,0,1,0),\ T = 10$
$10 : E_{26} = a_2 \cap b_6 \cap c_{26} = P_{10} = P_{10} = \mathbf{P}(\omega^2,0,1,0) = \mathbf{P}(3,0,1,0),\ T = 15$
$11 : E_{14,23,56} = c_{14} \cap c_{23} \cap c_{56} = P_{11} = P_{11} = \mathbf{P}(0,1,1,0) = \mathbf{P}(0,1,1,0),\ T = 9$
$12 : E_{13,24,56} = c_{13} \cap c_{24} \cap c_{56} = P_{12} = P_{12} = \mathbf{P}(1,1,1,0) = \mathbf{P}(1,1,1,0),\ T = 6$
$13 : E_{65} = a_6 \cap b_5 \cap c_{56} = P_{13} = P_{13} = \mathbf{P}(\omega,1,1,0) = \mathbf{P}(2,1,1,0),\ T = 12$
$14 : E_{12,34,56} = c_{12} \cap c_{34} \cap c_{56} = P_{14} = P_{14} = \mathbf{P}(\omega^2,1,1,0) = \mathbf{P}(3,1,1,0),\ T = 18$
$15 : E_{15,23,46} = c_{15} \cap c_{23} \cap c_{46} = P_{15} = P_{23} = \mathbf{P}(1,0,0,1) = \mathbf{P}(1,0,0,1),\ T = 21$
$16 : E_{31} = a_3 \cap b_1 \cap c_{13} = P_{16} = P_{26} = \mathbf{P}(0,1,0,1) = \mathbf{P}(0,1,0,1),\ T = 25$
$17 : E_{15,24,36} = c_{15} \cap c_{24} \cap c_{36} = P_{17} = P_{27} = \mathbf{P}(1,1,0,1) = \mathbf{P}(1,1,0,1),\ T = 22$
$18 : E_{21} = a_2 \cap b_1 \cap c_{12} = P_{18} = P_{30} = \mathbf{P}(0,\omega,0,1) = \mathbf{P}(0,2,0,1),\ T = 46$
$19 : E_{15,26,34} = c_{15} \cap c_{26} \cap c_{34} = P_{19} = P_{31} = \mathbf{P}(1,\omega,0,1) = \mathbf{P}(1,2,0,1),\ T = 24$
$20 : E_{61} = a_6 \cap b_1 \cap c_{16} = P_{20} = P_{34} = \mathbf{P}(0,\omega^2,0,1) = \mathbf{P}(0,3,0,1),\ T = 67$

$21 : E_{15} = a_1 \cap b_5 \cap c_{15} = P_{21} = P_{35} = \mathbf{P}(1, \omega^2, 0, 1) = \mathbf{P}(1, 3, 0, 1),\ T = 23$
$22 : E_{42} = a_4 \cap b_2 \cap c_{24} = P_{22} = P_{38} = \mathbf{P}(0, 0, 1, 1) = \mathbf{P}(0, 0, 1, 1),\ T = 41$
$23 : E_{13,25,46} = c_{13} \cap c_{25} \cap c_{46} = P_{23} = P_{39} = \mathbf{P}(1, 0, 1, 1) = \mathbf{P}(1, 0, 1, 1),\ T = 26$
$24 : E_{14,25,36} = c_{14} \cap c_{25} \cap c_{36} = P_{24} = P_{42} = \mathbf{P}(0, 1, 1, 1) = \mathbf{P}(0, 1, 1, 1),\ T = 30$
$25 : E_{62} = a_6 \cap b_2 \cap c_{26} = P_{25} = P_{47} = \mathbf{P}(\omega, \omega, 1, 1) = \mathbf{P}(2, 2, 1, 1),\ T = 53$
$26 : E_{25} = a_2 \cap b_5 \cap c_{25} = P_{26} = P_{48} = \mathbf{P}(\omega^2, \omega, 1, 1) = \mathbf{P}(3, 2, 1, 1),\ T = 80$
$27 : E_{16,25,34} = c_{16} \cap c_{25} \cap c_{34} = P_{27} = P_{51} = \mathbf{P}(\omega, \omega^2, 1, 1) = \mathbf{P}(2, 3, 1, 1),\ T = 55$
$28 : E_{12} = a_1 \cap b_2 \cap c_{12} = P_{28} = P_{52} = \mathbf{P}(\omega^2, \omega^2, 1, 1) = \mathbf{P}(3, 3, 1, 1),\ T = 79$
$29 : E_{43} = a_4 \cap b_3 \cap c_{34} = P_{29} = P_{53} = \mathbf{P}(0, 0, \omega, 1) = \mathbf{P}(0, 0, 2, 1),\ T = 62$
$30 : E_{12,35,46} = c_{12} \cap c_{35} \cap c_{46} = P_{30} = P_{54} = \mathbf{P}(1, 0, \omega, 1) = \mathbf{P}(1, 0, 2, 1),\ T = 36$
$31 : E_{35} = a_3 \cap b_5 \cap c_{35} = P_{31} = P_{59} = \mathbf{P}(\omega, 1, \omega, 1) = \mathbf{P}(2, 1, 2, 1),\ T = 49$
$32 : E_{63} = a_6 \cap b_3 \cap c_{36} = P_{32} = P_{60} = \mathbf{P}(\omega^2, 1, \omega, 1) = \mathbf{P}(3, 1, 2, 1),\ T = 76$
$33 : E_{14,26,35} = c_{14} \cap c_{26} \cap c_{35} = P_{33} = P_{61} = \mathbf{P}(0, \omega, \omega, 1) = \mathbf{P}(0, 2, 2, 1),\ T = 51$
$34 : E_{23} = a_2 \cap b_3 \cap c_{23} = P_{34} = P_{62} = \mathbf{P}(1, \omega, \omega, 1) = \mathbf{P}(1, 2, 2, 1),\ T = 39$
$35 : E_{13} = a_1 \cap b_3 \cap c_{13} = P_{35} = P_{67} = \mathbf{P}(\omega, \omega^2, \omega, 1) = \mathbf{P}(2, 3, 2, 1),\ T = 50$
$36 : E_{16,24,35} = c_{16} \cap c_{24} \cap c_{35} = P_{36} = P_{68} = \mathbf{P}(\omega^2, \omega^2, \omega, 1) = \mathbf{P}(3, 3, 2, 1),\ T = 74$
$37 : E_{45} = a_4 \cap b_5 \cap c_{45} = P_{37} = P_{69} = \mathbf{P}(0, 0, \omega^2, 1) = \mathbf{P}(0, 0, 3, 1),\ T = 83$
$38 : E_{64} = a_6 \cap b_4 \cap c_{46} = P_{38} = P_{70} = \mathbf{P}(1, 0, \omega^2, 1) = \mathbf{P}(1, 0, 3, 1),\ T = 31$
$39 : E_{12,36,45} = c_{12} \cap c_{36} \cap c_{45} = P_{39} = P_{75} = \mathbf{P}(\omega, 1, \omega^2, 1) = \mathbf{P}(2, 1, 3, 1),\ T = 59$
$40 : E_{34} = a_3 \cap b_4 \cap c_{34} = P_{40} = P_{76} = \mathbf{P}(\omega^2, 1, \omega^2, 1) = \mathbf{P}(3, 1, 3, 1),\ T = 71$
$41 : E_{24} = a_2 \cap b_4 \cap c_{24} = P_{41} = P_{79} = \mathbf{P}(\omega, \omega, \omega^2, 1) = \mathbf{P}(2, 2, 3, 1),\ T = 58$
$42 : E_{13,26,45} = c_{13} \cap c_{26} \cap c_{45} = P_{42} = P_{80} = \mathbf{P}(\omega^2, \omega, \omega^2, 1) = \mathbf{P}(3, 2, 3, 1),\ T = 70$
$43 : E_{14} = a_1 \cap b_4 \cap c_{14} = P_{43} = P_{81} = \mathbf{P}(0, \omega^2, \omega^2, 1) = \mathbf{P}(0, 3, 3, 1),\ T = 72$
$44 : E_{16,23,45} = c_{16} \cap c_{23} \cap c_{45} = P_{44} = P_{82} = \mathbf{P}(1, \omega^2, \omega^2, 1) = \mathbf{P}(1, 3, 3, 1).\ T = 33$
Set of tangent planes: (0, 4, 20, 84, 27, 1, 2, 3, 5, 10, 15, 9, 6, 12, 18, 21, 25, 22, 46, 24, 67, 23, 41, 26, 30, 53, 80, 55, 79, 62, 36, 49, 76, 51, 39, 50, 74, 83, 31, 59, 71, 58, 70, 72, 33)
Line type of Eckardt points: 5^{27}, 3^{240}, 1^{90}
Plane type of Eckardt points: 13^{45}, 9^{40}

Hesse planes

Number of Hesse planes: 40
Set of Hesse planes: (7, 8, 11, 13, 14, 16, 17, 19, 28, 29, 32, 34, 35, 37, 38, 40, 42, 43, 44, 45, 47, 48, 52, 54, 56, 57, 60, 61, 63, 64, 65, 66, 68, 69, 73, 75, 77, 78, 81, 82)
subspace 0 / 40 is 7:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & \omega \end{bmatrix}$$

$\vdots$
subspace 39 / 40 is 82:

$$\begin{bmatrix} 1 & 0 & \omega^2 & 0 \\ 0 & 1 & \omega^2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$0 : 7 : E_{56}, E_{31}, E_{15,24,36}, E_{16,25,34}, E_{12}, E_{14,26,35}, E_{23}, E_{45}, E_{64}$
$\vdots$
$39 : 82 : E_{41}, E_{52}, E_{16}, E_{12,34,56}, E_{15,24,36}, E_{35}, E_{23}, E_{64}, E_{13,26,45}$

Axes

Number of axes: 240
Axes:
$0 : 0 = 0{,}0 = E_{23}, E_{31}, E_{12}$
$\vdots$
$239 : 239 = 119{,}1 = E_{12,36,45}, E_{14,26,35}, E_{13,25,46}$
$\vdots$

Tritangent planes

The 45 tritangent planes are:

$$\pi_{12} = \pi_0 = 79 = \begin{bmatrix} 1 & 0 & 0 & \omega^2 \\ 0 & 1 & 0 & \omega^2 \\ 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

$$= V(\omega^2 X_0 + \omega^2 X_1 + X_2 + X_3) = V(3X_0 + 3X_1 + X_2 + X_3)$$

dual pt rank $= 52 = (3, 3, 1, 1)$.
$\vdots$

$$\pi_{16,25,34} = \pi_{44} = 55 = \begin{bmatrix} 1 & 0 & 0 & \omega \\ 0 & 1 & 0 & \omega \\ 0 & 0 & 1 & \omega^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 3 \end{bmatrix}$$

$$= V(\omega X_0 + \omega X_1 + \omega^2 X_2 + X_3) = V(2X_0 + 2X_1 + 3X_2 + X_3)$$

dual pt rank $= 79 = (2, 2, 3, 1)$.

Karaoglu [41] describes a different algorithm, based on non-conical six-arcs and Steiner trihedral pairs. The command

Example 340

```
surface_classify_q4_arc_lifting_two_lines:
▷  $(ORBITER) -v 10 \
▷  ▷  -define F -finite_field -q 4 -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷  ▷  -define Control -poset_classification_control \
▷  ▷  ▷  -problem_label sixarcs_q4 -W \
▷  ▷  -end \
▷  ▷  -with P -do \
▷  ▷  -projective_space_activity \
▷  ▷  ▷  -control_six_arcs Control \
▷  ▷  ▷  -classify_surfaces_through_arcs_and_two_lines \
▷  ▷  -end
▷  pdflatex surfaces_arc_lifting_4.tex
▷  $(OPEN) surfaces_arc_lifting_4.pdf
```

classifies all cubic surfaces with 27 lines over the field $\mathbb{F}_4$ using the algorithm of Karaoglu. The result agrees with the previous algorithm. In PG$(3, 4)$, the only surface with 27 lines is the Hirschfeld surface.

The classification of cubic surfaces with double sixes can be used to perform isomorphism testing and recognition of surfaces. Suppose we want to create an isomorphism between the surface $F_{a,b,c,d}$ with $(a, b, c, d) = (25, 5, 5, 25)$ over $\mathbb{F}_{31}$ and the appropriate surface chosen to be part of the transversal of orbit representatives by the classification algorithm. We can use the following command to create the surface, to classify all surfaces with 27 lines over $\mathbb{F}_{31}$, and to establish an isomorphism between the given surface and the standard model in the classified list:

Example 341

```
Family_general_F31_25_5_5_25_recognize:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 31 -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷  ▷  -define Surf -cubic_surface \
▷  ▷  ▷  -space P -family_general_abcd 25 5 5 25 \
▷  ▷  -end \
▷  ▷  -with Surf -do \
▷  ▷  -cubic_surface_activity \
▷  ▷  ▷  -report \
▷  ▷  -end \
▷  ▷  -define Control -poset_classification_control -W \
▷  ▷  ▷  -problem_label "surf_q31" \
▷  ▷  -end \
▷  ▷  -with P -do \
▷  ▷  -projective_space_activity \
▷  ▷  ▷  -classify_surfaces_with_double_sixes AllSurf Control \
▷  ▷  -end \
▷  ▷  -with AllSurf -do \
▷  ▷  -classification_of_cubic_surfaces_with_double_sixes_activity \
▷  ▷  ▷  -recognize Surf \
▷  ▷  -end
```

The following command creates all surfaces in the form F_{abcd} over $\mathbb{F}_{11}$ and identifies the isomorphism type of each surface. The result is written to a csv file.

Example 342

```
surface_F_abcd_identify_q11:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 11 -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷  ▷  -define Control -poset_classification_control -W \
▷  ▷  ▷  -problem_label "surf_q11" \
▷  ▷  -end \
▷  ▷  -with P -do \
▷  ▷  -projective_space_activity \
▷  ▷  ▷  -classify_surfaces_with_double_sixes AllSurf Control \
▷  ▷  -end \
▷  ▷  -with AllSurf -do \
```

```
▷  ▷  -classification_of_cubic_surfaces_with_double_sixes_activity \
▷  ▷  ▷  -identify_general_abcd \
▷  ▷  -end
```

The next example illustrates isomorphism testing for smooth cubic surfaces with 27 lines. In order to perform isomorphism testing, the classification of cubic surfaces of that order is required. Let us consider an example over the field $\mathbb{F}_{11}$. From the previous command, we know that the two surfaces of type F_{abcd} with $(a, b, c, d) = (2, 4, 4, 2)$ and $(a, b, c, d) = (5, 10, 7, 5)$, over $\mathbb{F}_{11}$ are isomorphic. Both surfaces have 10 Eckardt points. The following command sequence creates both surfaces, classifies all surfaces over $\mathbb{F}_{11}$, and then performs an isomorphism test for the two given surfaces. The surfaces are found to be isomorphic. A transformation mapping one surface to the other is computed.

Example 343

```
Family_general_F11_iso_test:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 11 -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷  ▷  -define Surf1 -cubic_surface \
▷  ▷  ▷  -space P -family_general_abcd 2 4 4 2 \
▷  ▷  -end \
▷  ▷  -define Surf2 -cubic_surface \
▷  ▷  ▷  -space P -family_general_abcd 5 10 7 5 \
▷  ▷  -end \
▷  ▷  -define Control -poset_classification_control -W \
▷  ▷  ▷  -problem_label "surf_q11" \
▷  ▷  -end \
▷  ▷  -with P -do \
▷  ▷  -projective_space_activity \
▷  ▷  ▷  -classify_surfaces_with_double_sixes AllSurf Control \
▷  ▷  -end \
▷  ▷  -with AllSurf -do \
▷  ▷  -classification_of_cubic_surfaces_with_double_sixes_activity \
▷  ▷  ▷  -isomorphism_testing Surf1 Surf2 \
▷  ▷  -end
```

Creating Quartic Curves

Command	Arguments	Purpose
-space	P	Specify the 2-dimensional projective space P $= \mathrm{PG}(2,q)$.
-label_txt	label	Override the ascii label of the curve.
-label_tex	label	Override the latex label of the curve.
-label_for_summary	label	Override the ascii label of the curve, to be used in summary commands.
-catalogue	OCN	Create the quartic curve from the Orbiter catalogue with the given Orbiter catalogue number.
-by_coefficients	coeffs	Create a quartic curve given the coefficients of the equation.
-by_equation	eqn	Create a quartic curve from an equation.
-by_symbolic_object	R equation	Create a quartic curve from an equation using the polynomial ring R. The equation is given as a symbolic object.
-from_cubic_surface	S i	Create the quartic curve from ith orbit of the automorphism group of the surface S on points not on lines (i is zero based).
-override_group	descr	Override the automorphism group of the curve by the given group.
-transform	elt	Apply the transformation given by the group element.
-transform_inverse	elt	Apply the inverse transformation given by the group element.

Table 8.8: Creating Quartic Curves

8.5 Quartic Curves

Cubic surfaces with 27 lines are associated with quartic curves with 28 bitangents (see [35]), which in turn are associated with del Pezzo surfaces. Orbiter can classify quartic curves based on a known classification of cubic surfaces. Orbiter also has a catalogue of quartic curves for small field sizes.

Table 8.8 lists options to create a quartic curve object.

Table 8.9 lists activities for a quartic curve object.
Table 8.10 lists options for exporting data.

Suppose we want to create a quartic curve. We can utilize the Orbiter catalogue of isomorphism types of quartic curves. The follwing command creates the unique curve over the field $\mathbb{F}_9$.

Example 344

```
quartic_curve_9_0:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 9 -end \
▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
▷  ▷  -define C -quartic_curve -space P -catalogue 0 -end
```

Quartic Curve Activities		
Command	Arguments	Purpose
-report		Produce a latex report about the curve.
-export_something	type	Exports data of the quartic curve to a file. The type of data to be exported must be specified according to Table 8.10.
-create_surface		Create a cubic surface from the curve.
-extract_orbit_on_bitangents_by_length	l	Extract the bitangents in the unique orbit of length l. If there is no orbit of length l, or if there are multiple orbits of length l, an error is raised.
-extract_specific_orbit_on_bitangents_by_length	$l\ i$	Extract the ith orbit on bitangents of length l.
-extract_specific_orbit_on_kovalevski_points_by_length	$l\ i$	Extract the ith orbit on Kovalevski points of length l.

Table 8.9: Quartic Curve Activities

Options for Exporting Data: Cubic Surfaces	
Command	Purpose
points	The points on the quartic curve.
equation	The equation of the quartic curves as a coefficient vector of length 15.
bitangents	The 28 bitangents.
Kovalevski_points	The Kovalevski points (points off the curve where 4 bitangents meet).
singular_points	The singular points of the curve.

Table 8.10: Options for Exporting Data: Quartic Curves

The Edge curve over $\mathbb{F}_q$ has an equation of the form

$$X^4 - Y^4 - Z^4 + 2f^2Y^2Z^2 + 4fX^2YZ = 0,$$

with f a primitive element of $\mathbb{F}_q$ Let us create the Edge curve over $\mathbb{F}_9$. Since we are in characteristic 3, we write 4 as 1. The default field polynomial chosen by Orbiter is always primitive, so we can pick $\alpha = f$, which encodes to 3. The following command creates the Edge curve:

Example 345

```
quartic_curve_edge_q9:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 9 -end \
▷  ▷  -define P2 -projective_space -n 2 -field F -end \
▷  ▷  -define R -polynomial_ring \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -number_of_variables 3 \
▷  ▷  ▷  -homogeneous_of_degree 4 \
▷  ▷  ▷  -monomial_ordering_partition \
▷  ▷  ▷  -variables "X,Y,Z" "X,Y,Z" \
▷  ▷  -end \
▷  ▷  -define Edge -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -managed_variables "X,Y,Z" \
▷  ▷  ▷  -text "X^4-Y^4-Z^4+2*f^2*Y^2*Z^2+1*f*X^2*Y*Z" \
▷  ▷  -end \
▷  ▷  -define f -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -text "3" \
▷  ▷  -end \
▷  ▷  -define Edge_f3 -symbolic_object \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -managed_variables "X,Y,Z" \
▷  ▷  ▷  -substitute "f" Edge f \
▷  ▷  -end \
▷  ▷  -define C -quartic_curve \
▷  ▷  ▷  -space P2 \
▷  ▷  ▷  -by_symbolic_object R Edge_f3 -end \
▷  ▷  -with C -do \
▷  ▷  ▷  -quartic_curve_activity \
▷  ▷  ▷  ▷  -report \
▷  ▷  ▷  -end
▷  pdflatex quartic_curve_equation_Edge_f3_q9_report.tex
▷  $(OPEN) quartic_curve_equation_Edge_f3_q9_report.pdf
```

The report shows that the curve is non-singular and has 8 rational points.

We can create quartic curves from cubic surfaces. The following command creates all quartic curves arising from points on no line of the cubic surface over $\mathbb{F}_7$:

Example 346

```
surface_7_0_make_quartic_curves:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 7 -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷  ▷  -define S -cubic_surface -space P -catalogue 0 -end \
▷  ▷  -with S -do \
▷  ▷  -cubic_surface_activity \
▷  ▷  ▷  -report \
▷  ▷  ▷  -all_quartic_curves \
▷  ▷  -end \
▷  #pdflatex surface_catalogue_q7_iso0_report.tex
▷  #$(OPEN) surface_catalogue_q7_iso0_report.pdf
```

The next command creates all cubic surfaces arising from the second isomorphism class of cubic surfaces over the field $\mathbb{F}_9$:

Example 347

```
surface_9_1_make_quartic_curves:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 9 -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷  ▷  -define S -cubic_surface -space P -catalogue 1 -end \
▷  ▷  -with S -do \
▷  ▷  -cubic_surface_activity \
▷  ▷  ▷  -report \
▷  ▷  ▷  -all_quartic_curves \
▷  ▷  -end \
▷  #pdflatex surface_catalogue_q9_iso1_report.tex
▷  #$(OPEN) surface_catalogue_q9_iso1_report.pdf
```

Suppose we want to study the (unique) quartic curve for $q = 9$. The follwing command creates the curve whose catalogue number is 0, and produces a report:

Example 348

```
quartic_curve_9_0_report:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 9 -end \
▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
▷  ▷  -define C -quartic_curve -space P -catalogue 0 -end \
▷  ▷  -with C -do \
▷  ▷  ▷  -quartic_curve_activity \
▷  ▷  ▷  ▷  -report \
▷  ▷  ▷  -end
▷  pdflatex quartic_curve_catalogue_q9_iso0_report.tex
▷  $(OPEN) quartic_curve_catalogue_q9_iso0_report.pdf
```

The report contains the following information:

The equation

The equation of the quartic curve is :

$$\alpha^3 X_0^3 X_1 + \alpha^4 X_0^3 X_2 + \alpha^7 X_0 X_1^3 + \alpha^6 X_1^3 X_2 + \alpha^2 X_0 X_2^3 + X_1 X_2^3$$

$$(0,0,0,8,2,4,5,7,1,0,0,0,0,0,0)$$

The gradient

The gradient of the quartic curve is :

$$\alpha^7 X_1^3 + \alpha^2 X_2^3$$

$$(0,4,7,0,0,0,0,0,0,0)$$

$$\alpha^3 X_0^3 + X_2^3$$

$$(8,0,1,0,0,0,0,0,0,0)$$

$$\alpha^4 X_0^3 + \alpha^6 X_1^3$$

$$(2,5,0,0,0,0,0,0,0,0)$$

General information

Number of bitangents	28
Number of points	28
Fullness	is full
Number of Kovalevski points	63
Bitangent line type (a_0, a_1, a_2)	$(0, 28, 0)$
Number of singular points	0

All points on the curve

The surface has 28 points:
The points on the quartic curve are:

$0: P_0 = (1,0,0)$
$1: P_1 = (0,1,0)$
$2: P_2 = (0,0,1)$
$3: P_3 = (1,1,1)$
$4: P_4 = (1,1,0)$
$5: P_5 = (2,1,0)$
$6: P_{14} = (3,0,1)$
$7: P_{17} = (6,0,1)$
$8: P_{24} = (5,1,1)$
$9: P_{25} = (6,1,1)$
$10: P_{30} = (2,2,1)$
$11: P_{32} = (4,2,1)$
$12: P_{34} = (6,2,1)$
$13: P_{38} = (1,3,1)$
$14: P_{41} = (4,3,1)$
$15: P_{44} = (7,3,1)$
$16: P_{46} = (0,4,1)$
$17: P_{51} = (5,4,1)$
$18: P_{53} = (7,4,1)$
$19: P_{57} = (2,5,1)$
$20: P_{58} = (3,5,1)$
$21: P_{62} = (7,5,1)$
$22: P_{76} = (3,7,1)$
$23: P_{77} = (4,7,1)$
$24: P_{78} = (5,7,1)$
$25: P_{82} = (0,8,1)$
$26: P_{83} = (1,8,1)$
$27: P_{84} = (2,8,1)$

The points by rank are: (0, 1, 2, 3, 4, 5, 14, 17, 24, 25, 30, 32, 34, 38, 41, 44, 46, 51, 53, 57, 58, 62, 76, 77, 78, 82, 83, 84)
The Kovalevski points are:
$0: P_7 = (4,1,0) = c_{13} \cap c_{14} \cap c_{36} \cap c_{46}$
$1: P_8 = (5,1,0) = a_2 \cap a_4 \cap c_{25} \cap c_{45}$
$2: P_9 = (6,1,0) = b_1 \cap b_6 \cap c_{12} \cap c_{26}$
$3: P_{10} = (7,1,0) = a_3 \cap b_5 \cap c_{35} \cap d$
$4: P_{11} = (8,1,0) = b_2 \cap b_3 \cap c_{24} \cap c_{34}$
$5: P_{12} = (1,0,1) = a_3 \cap a_4 \cap c_{23} \cap c_{24}$
$6: P_{13} = (2,0,1) = c_{34} \cap c_{36} \cap c_{45} \cap c_{56}$
$7: P_{15} = (4,0,1) = b_3 \cap b_6 \cap c_{13} \cap c_{16}$
$8: P_{16} = (5,0,1) = a_5 \cap a_6 \cap c_{25} \cap c_{26}$
$9: P_{18} = (7,0,1) = a_2 \cap b_1 \cap c_{35} \cap c_{46}$
$10: P_{19} = (8,0,1) = b_4 \cap b_5 \cap c_{14} \cap c_{15}$
$11: P_{20} = (0,1,1) = a_2 \cap b_3 \cap c_{14} \cap c_{56}$
$12: P_{21} = (2,1,1) = b_2 \cap b_4 \cap c_{26} \cap c_{46}$
$13: P_{22} = (3,1,1) = a_4 \cap b_5 \cap c_{12} \cap c_{36}$
$14: P_{23} = (4,1,1) = a_6 \cap b_1 \cap c_{23} \cap c_{45}$
$15: P_{26} = (7,1,1) = c_{16} \cap c_{25} \cap c_{34} \cap d$
$16: P_{27} = (8,1,1) = a_3 \cap a_5 \cap c_{13} \cap c_{15}$
$17: P_{28} = (0,2,1) = c_{12} \cap c_{13} \cap c_{25} \cap c_{35}$
$18: P_{29} = (1,2,1) = b_1 \cap b_5 \cap c_{16} \cap c_{56}$
$19: P_{31} = (3,2,1) = a_3 \cap a_6 \cap c_{34} \cap c_{46}$
$20: P_{33} = (5,2,1) = a_2 \cap b_4 \cap c_{24} \cap d$
$21: P_{35} = (7,2,1) = b_2 \cap b_6 \cap c_{23} \cap c_{36}$
$22: P_{36} = (8,2,1) = a_4 \cap b_3 \cap c_{15} \cap c_{26}$
$23: P_{37} = (0,3,1) = a_5 \cap b_1 \cap c_{24} \cap c_{36}$
$24: P_{39} = (2,3,1) = a_2 \cap a_6 \cap c_{12} \cap c_{16}$
$25: P_{40} = (3,3,1) = b_3 \cap b_4 \cap c_{35} \cap c_{45}$
$26: P_{42} = (5,3,1) = a_4 \cap b_2 \cap c_{13} \cap c_{56}$
$27: P_{43} = (6,3,1) = a_3 \cap b_6 \cap c_{14} \cap c_{25}$
$28: P_{45} = (8,3,1) = c_{15} \cap c_{23} \cap c_{46} \cap d$
$29: P_{47} = (1,4,1) = a_6 \cap b_2 \cap c_{14} \cap c_{35}$
$30: P_{48} = (2,4,1) = b_3 \cap b_5 \cap c_{23} \cap c_{25}$
$31: P_{49} = (3,4,1) = a_5 \cap b_6 \cap c_{56} \cap d$
$32: P_{50} = (4,4,1) = a_2 \cap a_3 \cap c_{26} \cap c_{36}$
$33: P_{52} = (6,4,1) = b_1 \cap b_4 \cap c_{13} \cap c_{34}$
$34: P_{54} = (8,4,1) = c_{12} \cap c_{15} \cap c_{24} \cap c_{45}$
$35: P_{55} = (0,5,1) = a_4 \cap a_6 \cap b_4 \cap b_6$
$36: P_{56} = (1,5,1) = c_{13} \cap c_{26} \cap c_{45} \cap d$
$37: P_{59} = (4,5,1) = c_{24} \cap c_{25} \cap c_{46} \cap c_{56}$
$38: P_{60} = (5,5,1) = c_{12} \cap c_{14} \cap c_{23} \cap c_{34}$
$39: P_{61} = (6,5,1) = a_2 \cap a_5 \cap b_2 \cap b_5$
$40: P_{63} = (8,5,1) = c_{15} \cap c_{16} \cap c_{35} \cap c_{36}$
$41: P_{64} = (0,6,1) = a_1 \cap b_5 \cap c_{26} \cap c_{34}$
$42: P_{65} = (1,6,1) = a_1 \cap b_4 \cap c_{25} \cap c_{36}$
$43: P_{66} = (2,6,1) = a_1 \cap b_6 \cap c_{24} \cap c_{35}$
$44: P_{67} = (3,6,1) = a_1 \cap a_2 \cap c_{13} \cap c_{23}$
$45: P_{68} = (4,6,1) = a_1 \cap b_2 \cap c_{12} \cap d$
$46: P_{69} = (5,6,1) = a_1 \cap a_3 \cap b_1 \cap b_3$
$47: P_{70} = (6,6,1) = a_1 \cap a_4 \cap c_{16} \cap c_{46}$
$48: P_{71} = (7,6,1) = a_1 \cap a_5 \cap c_{14} \cap c_{45}$
$49: P_{72} = (8,6,1) = a_1 \cap a_6 \cap c_{15} \cap c_{56}$

$50: P_{73} = (0,7,1) = a_3 \cap b_2 \cap c_{16} \cap c_{45}$
$51: P_{74} = (1,7,1) = a_5 \cap b_3 \cap c_{12} \cap c_{46}$
$52: P_{75} = (2,7,1) = a_4 \cap b_1 \cap c_{14} \cap d$
$53: P_{79} = (6,7,1) = c_{23} \cap c_{26} \cap c_{35} \cap c_{56}$
$54: P_{80} = (7,7,1) = a_6 \cap b_5 \cap c_{13} \cap c_{24}$
$55: P_{81} = (8,7,1) = a_2 \cap b_6 \cap c_{15} \cap c_{34}$
$56: P_{85} = (3,8,1) = c_{14} \cap c_{16} \cap c_{24} \cap c_{26}$
$57: P_{86} = (4,8,1) = a_4 \cap a_5 \cap c_{34} \cap c_{35}$
$58: P_{87} = (5,8,1) = b_5 \cap b_6 \cap c_{45} \cap c_{46}$
$59: P_{88} = (6,8,1) = a_6 \cap b_3 \cap c_{36} \cap d$
$60: P_{89} = (7,8,1) = a_3 \cap b_4 \cap c_{12} \cap c_{56}$
$61: P_{90} = (8,8,1) = b_1 \cap b_2 \cap c_{15} \cap c_{25}$
$62: P_6 = (3,1,0) = a_5 \cap b_4 \cap c_{16} \cap c_{23}$
The Kovalevski points by rank are: (7, 8, 9, 10, 11, 12, 13, 15, 16, 18, 19, 20, 21, 22, 23, 26, 27, 28, 29, 31, 33, 35, 36, 37, 39, 40, 42, 43, 45, 47, 48, 49, 50, 52, 54, 55, 56, 59, 60, 61, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 79, 80, 81, 85, 86, 87, 88, 89, 90, 6)
The points off the curve are:

$0: P_6 = (3,1,0)$
$1: P_7 = (4,1,0)$
$2: P_8 = (5,1,0)$
$3: P_9 = (6,1,0)$
$4: P_{10} = (7,1,0)$
$5: P_{11} = (8,1,0)$
$6: P_{12} = (1,0,1)$
$7: P_{13} = (2,0,1)$
$8: P_{15} = (4,0,1)$
$9: P_{16} = (5,0,1)$
$10: P_{18} = (7,0,1)$
$11: P_{19} = (8,0,1)$
$12: P_{20} = (0,1,1)$
$13: P_{21} = (2,1,1)$
$14: P_{22} = (3,1,1)$
$15: P_{23} = (4,1,1)$
$16: P_{26} = (7,1,1)$
$17: P_{27} = (8,1,1)$
$18: P_{28} = (0,2,1)$
$19: P_{29} = (1,2,1)$
$20: P_{31} = (3,2,1)$
$21: P_{33} = (5,2,1)$
$22: P_{35} = (7,2,1)$
$23: P_{36} = (8,2,1)$
$24: P_{37} = (0,3,1)$
$25: P_{39} = (2,3,1)$
$26: P_{40} = (3,3,1)$
$27: P_{42} = (5,3,1)$
$28: P_{43} = (6,3,1)$
$29: P_{45} = (8,3,1)$
$30: P_{47} = (1,4,1)$
$31: P_{48} = (2,4,1)$
$32: P_{49} = (3,4,1)$
$33: P_{50} = (4,4,1)$
$34: P_{52} = (6,4,1)$
$35: P_{54} = (8,4,1)$
$36: P_{55} = (0,5,1)$
$37: P_{56} = (1,5,1)$
$38: P_{59} = (4,5,1)$
$39: P_{60} = (5,5,1)$
$40: P_{61} = (6,5,1)$
$41: P_{63} = (8,5,1)$
$42: P_{64} = (0,6,1)$
$43: P_{65} = (1,6,1)$
$44: P_{66} = (2,6,1)$
$45: P_{67} = (3,6,1)$
$46: P_{68} = (4,6,1)$
$47: P_{69} = (5,6,1)$
$48: P_{70} = (6,6,1)$
$49: P_{71} = (7,6,1)$
$50: P_{72} = (8,6,1)$
$51: P_{73} = (0,7,1)$
$52: P_{74} = (1,7,1)$
$53: P_{75} = (2,7,1)$
$54: P_{79} = (6,7,1)$
$55: P_{80} = (7,7,1)$
$56: P_{81} = (8,7,1)$
$57: P_{85} = (3,8,1)$
$58: P_{86} = (4,8,1)$
$59: P_{87} = (5,8,1)$
$60: P_{88} = (6,8,1)$
$61: P_{89} = (7,8,1)$
$62: P_{90} = (8,8,1)$

(6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 18, 19, 20, 21, 22, 23, 26, 27, 28, 29, 31, 33, 35, 36, 37, 39, 40, 42, 43, 45, 47, 48, 49, 50, 52, 54, 55, 56, 59, 60, 61, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 79, 80, 81, 85, 86, 87, 88, 89, 90)
The lines and their points of contact are:

$$a_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & \alpha^3 \end{bmatrix}_8 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 8 \end{bmatrix}_8 \quad P_0 = \mathbf{P}(1,0,0)\ 4\times$$

$$a_2 = \begin{bmatrix} 1 & 0 & \alpha^6 \\ 0 & 1 & 1 \end{bmatrix}_{51} = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 1 \end{bmatrix}_{51} \quad P_{83} = \mathbf{P}(1,8,1)\ 4\times$$

$$a_3 = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & \alpha^6 \end{bmatrix}_{15} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 5 \end{bmatrix}_{15} \quad P_{57} = \mathbf{P}(2,5,1)\ 4\times$$

$$a_4 = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & \alpha^2 \end{bmatrix}_{17} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 7 \end{bmatrix}_{17} \quad P_{53} = \mathbf{P}(7,4,1)\ 4\times$$

$$a_5 = \begin{bmatrix} 1 & 0 & \alpha^2 \\ 0 & 1 & \alpha^7 \end{bmatrix}_{74} = \begin{bmatrix} 1 & 0 & 7 \\ 0 & 1 & 4 \end{bmatrix}_{74} \quad P_{30} = \mathbf{P}(2,2,1)\ 4\times$$

$$a_6 = \begin{bmatrix} 1 & 0 & \alpha^2 \\ 0 & 1 & \alpha^2 \end{bmatrix}_{77} = \begin{bmatrix} 1 & 0 & 7 \\ 0 & 1 & 7 \end{bmatrix}_{77} \quad P_{5} = \mathbf{P}(2,1,0)\ 4\times$$

$$b_1 = \begin{bmatrix} 1 & 0 & \alpha^6 \\ 0 & 1 & \alpha^7 \end{bmatrix}_{54} = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 4 \end{bmatrix}_{54} \quad P_{58} = \mathbf{P}(3,5,1)\ 4\times$$

$$b_2 = \begin{bmatrix} 1 & 0 & \alpha^7 \\ 0 & 1 & \alpha^6 \end{bmatrix}_{45} = \begin{bmatrix} 1 & 0 & 4 \\ 0 & 1 & 5 \end{bmatrix}_{45} \quad P_{14} = \mathbf{P}(3,0,1)\ 4\times$$

$$b_3 = \begin{bmatrix} 1 & 0 & \alpha \\ 0 & 1 & 1 \end{bmatrix}_{31} = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 1 \end{bmatrix}_{31} \quad P_{62} = \mathbf{P}(7,5,1)\ 4\times$$

$$b_4 = \begin{bmatrix} 1 & 0 & \alpha^5 \\ 0 & 1 & \alpha^2 \end{bmatrix}_{67} = \begin{bmatrix} 1 & 0 & 6 \\ 0 & 1 & 7 \end{bmatrix}_{67} \quad P_{77} = \mathbf{P}(4,7,1)\ 4\times$$

$$b_5 = \begin{bmatrix} 1 & 0 & \alpha^5 \\ 0 & 1 & \alpha^3 \end{bmatrix}_{68} = \begin{bmatrix} 1 & 0 & 6 \\ 0 & 1 & 8 \end{bmatrix}_{68} \quad P_{41} = \mathbf{P}(4,3,1)\ 4\times$$

$$b_6 = \begin{bmatrix} 1 & 0 & \alpha \\ 0 & 1 & \alpha^2 \end{bmatrix}_{37} = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 7 \end{bmatrix}_{37} \quad P_{3} = \mathbf{P}(1,1,1)\ 4\times$$

$$c_{12} = \begin{bmatrix} 1 & 0 & \alpha^3 \\ 0 & 1 & \alpha^4 \end{bmatrix}_{82} = \begin{bmatrix} 1 & 0 & 8 \\ 0 & 1 & 2 \end{bmatrix}_{82} \quad P_{17} = \mathbf{P}(6,0,1)\ 4\times$$

$$c_{13} = \begin{bmatrix} 1 & 0 & \alpha \\ 0 & 1 & \alpha^4 \end{bmatrix}_{32} = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 2 \end{bmatrix}_{32} \quad P_{84} = \mathbf{P}(2,8,1)\ 4\times$$

$$c_{14} = \begin{bmatrix} 1 & 0 & \alpha^5 \\ 0 & 1 & 1 \end{bmatrix}_{61} = \begin{bmatrix} 1 & 0 & 6 \\ 0 & 1 & 1 \end{bmatrix}_{61} \quad P_{32} = \mathbf{P}(4,2,1)\ 4\times$$

$$c_{15} = \begin{bmatrix} 1 & 0 & \alpha^5 \\ 0 & 1 & 0 \end{bmatrix}_{60} = \begin{bmatrix} 1 & 0 & 6 \\ 0 & 1 & 0 \end{bmatrix}_{60} \quad P_{1} = \mathbf{P}(0,1,0)\ 4\times$$

$$c_{16} = \begin{bmatrix} 1 & 0 & \alpha \\ 0 & 1 & \alpha^6 \end{bmatrix}_{35} = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 5 \end{bmatrix}_{35} \quad P_{51} = \mathbf{P}(5,4,1)\ 4\times$$

$$c_{23} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & \alpha^5 \end{bmatrix}_{16} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 6 \end{bmatrix}_{16} \quad P_{82} = \mathbf{P}(0,8,1)\ 4\times$$

$$c_{24} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & \alpha^7 \end{bmatrix}_{14} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 4 \end{bmatrix}_{14} \quad P_{25} = \mathbf{P}(6,1,1)\ 4\times$$

$$c_{25} = \begin{bmatrix} 1 & 0 & \alpha^2 \\ 0 & 1 & \alpha^4 \end{bmatrix}_{72} = \begin{bmatrix} 1 & 0 & 7 \\ 0 & 1 & 2 \end{bmatrix}_{72} \quad P_{76} = \mathbf{P}(3,7,1)\ 4\times$$

$$c_{26} = \begin{bmatrix} 1 & 0 & \alpha^2 \\ 0 & 1 & \alpha^3 \end{bmatrix}_{78} = \begin{bmatrix} 1 & 0 & 7 \\ 0 & 1 & 8 \end{bmatrix}_{78} \quad P_{44} = \mathbf{P}(7,3,1)\ 4\times$$

$$c_{34} = \begin{bmatrix} 1 & 0 & \alpha^4 \\ 0 & 1 & \alpha^3 \end{bmatrix}_{28} = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 8 \end{bmatrix}_{28} \quad P_{38} = \mathbf{P}(1,3,1)\ 4\times$$

$$c_{35} = \begin{bmatrix} 1 & 0 & \alpha^6 \\ 0 & 1 & \alpha^4 \end{bmatrix}_{52} = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 2 \end{bmatrix}_{52} \quad P_{24} = \mathbf{P}(5,1,1)\ 4\times$$
$$c_{36} = \begin{bmatrix} 1 & 0 & \alpha^4 \\ 0 & 1 & \alpha^7 \end{bmatrix}_{24} = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 4 \end{bmatrix}_{24} \quad P_{78} = \mathbf{P}(5,7,1)\ 4\times$$
$$c_{45} = \begin{bmatrix} 1 & 0 & \alpha^4 \\ 0 & 1 & \alpha^6 \end{bmatrix}_{25} = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 5 \end{bmatrix}_{25} \quad P_{34} = \mathbf{P}(6,2,1)\ 4\times$$
$$c_{46} = \begin{bmatrix} 1 & 0 & \alpha^6 \\ 0 & 1 & \alpha \end{bmatrix}_{53} = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 3 \end{bmatrix}_{53} \quad P_{46} = \mathbf{P}(0,4,1)\ 4\times$$
$$c_{56} = \begin{bmatrix} 1 & 0 & \alpha^4 \\ 0 & 1 & 1 \end{bmatrix}_{21} = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 1 \end{bmatrix}_{21} \quad P_4 = \mathbf{P}(1,1,0)\ 4\times$$
$$d = \begin{bmatrix} 1 & \alpha^6 & 0 \\ 0 & 0 & 1 \end{bmatrix}_{59} = \begin{bmatrix} 1 & 5 & 0 \\ 0 & 0 & 1 \end{bmatrix}_{59} \quad P_2 = \mathbf{P}(0,0,1)\ 4\times$$

Rank of lines: (8, 51, 15, 17, 74, 77, 54, 45, 31, 67, 68, 37, 82, 32, 61, 60, 35, 16, 14, 72, 78, 28, 52, 24, 25, 53, 21, 59)
Line type: 1^{28}

28	1	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 0

point types: 1^{28}

28	1	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 0

point types for points off the curve: 4^{63}

63	4	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 0

Lines on points off the curve:
Off point 0 = $P_6 = (3,1,0)$ lies on 4 bisecants : { 4, 9, 16, 17 }
Off point 1 = $P_7 = (4,1,0)$ lies on 4 bisecants : { 13, 14, 23, 25 }
Off point 2 = $P_8 = (5,1,0)$ lies on 4 bisecants : { 1, 3, 19, 24 }
Off point 3 = $P_9 = (6,1,0)$ lies on 4 bisecants : { 6, 11, 12, 20 }
Off point 4 = $P_{10} = (7,1,0)$ lies on 4 bisecants : { 2, 10, 22, 27 }
Off point 5 = $P_{11} = (8,1,0)$ lies on 4 bisecants : { 7, 8, 18, 21 }
Off point 6 = $P_{12} = (1,0,1)$ lies on 4 bisecants : { 2, 3, 17, 18 }
Off point 7 = $P_{13} = (2,0,1)$ lies on 4 bisecants : { 21, 23, 24, 26 }
Off point 8 = $P_{15} = (4,0,1)$ lies on 4 bisecants : { 8, 11, 13, 16 }
Off point 9 = $P_{16} = (5,0,1)$ lies on 4 bisecants : { 4, 5, 19, 20 }
Off point 10 = $P_{18} = (7,0,1)$ lies on 4 bisecants : { 1, 6, 22, 25 }
Off point 11 = $P_{19} = (8,0,1)$ lies on 4 bisecants : { 9, 10, 14, 15 }

Off point 12 = $P_{20} = (0, 1, 1)$ lies on 4 bisecants : { 1, 8, 14, 26 }
Off point 13 = $P_{21} = (2, 1, 1)$ lies on 4 bisecants : { 7, 9, 20, 25 }
Off point 14 = $P_{22} = (3, 1, 1)$ lies on 4 bisecants : { 3, 10, 12, 23 }
Off point 15 = $P_{23} = (4, 1, 1)$ lies on 4 bisecants : { 5, 6, 17, 24 }
Off point 16 = $P_{26} = (7, 1, 1)$ lies on 4 bisecants : { 16, 19, 21, 27 }
Off point 17 = $P_{27} = (8, 1, 1)$ lies on 4 bisecants : { 2, 4, 13, 15 }
Off point 18 = $P_{28} = (0, 2, 1)$ lies on 4 bisecants : { 12, 13, 19, 22 }
Off point 19 = $P_{29} = (1, 2, 1)$ lies on 4 bisecants : { 6, 10, 16, 26 }
Off point 20 = $P_{31} = (3, 2, 1)$ lies on 4 bisecants : { 2, 5, 21, 25 }
Off point 21 = $P_{33} = (5, 2, 1)$ lies on 4 bisecants : { 1, 9, 18, 27 }
Off point 22 = $P_{35} = (7, 2, 1)$ lies on 4 bisecants : { 7, 11, 17, 23 }
Off point 23 = $P_{36} = (8, 2, 1)$ lies on 4 bisecants : { 3, 8, 15, 20 }
Off point 24 = $P_{37} = (0, 3, 1)$ lies on 4 bisecants : { 4, 6, 18, 23 }
Off point 25 = $P_{39} = (2, 3, 1)$ lies on 4 bisecants : { 1, 5, 12, 16 }
Off point 26 = $P_{40} = (3, 3, 1)$ lies on 4 bisecants : { 8, 9, 22, 24 }
Off point 27 = $P_{42} = (5, 3, 1)$ lies on 4 bisecants : { 3, 7, 13, 26 }
Off point 28 = $P_{43} = (6, 3, 1)$ lies on 4 bisecants : { 2, 11, 14, 19 }
Off point 29 = $P_{45} = (8, 3, 1)$ lies on 4 bisecants : { 15, 17, 25, 27 }
Off point 30 = $P_{47} = (1, 4, 1)$ lies on 4 bisecants : { 5, 7, 14, 22 }
Off point 31 = $P_{48} = (2, 4, 1)$ lies on 4 bisecants : { 8, 10, 17, 19 }
Off point 32 = $P_{49} = (3, 4, 1)$ lies on 4 bisecants : { 4, 11, 26, 27 }
Off point 33 = $P_{50} = (4, 4, 1)$ lies on 4 bisecants : { 1, 2, 20, 23 }
Off point 34 = $P_{52} = (6, 4, 1)$ lies on 4 bisecants : { 6, 9, 13, 21 }
Off point 35 = $P_{54} = (8, 4, 1)$ lies on 4 bisecants : { 12, 15, 18, 24 }
Off point 36 = $P_{55} = (0, 5, 1)$ lies on 4 bisecants : { 3, 5, 9, 11 }
Off point 37 = $P_{56} = (1, 5, 1)$ lies on 4 bisecants : { 13, 20, 24, 27 }
Off point 38 = $P_{59} = (4, 5, 1)$ lies on 4 bisecants : { 18, 19, 25, 26 }
Off point 39 = $P_{60} = (5, 5, 1)$ lies on 4 bisecants : { 12, 14, 17, 21 }
Off point 40 = $P_{61} = (6, 5, 1)$ lies on 4 bisecants : { 1, 4, 7, 10 }
Off point 41 = $P_{63} = (8, 5, 1)$ lies on 4 bisecants : { 15, 16, 22, 23 }
Off point 42 = $P_{64} = (0, 6, 1)$ lies on 4 bisecants : { 0, 10, 20, 21 }
Off point 43 = $P_{65} = (1, 6, 1)$ lies on 4 bisecants : { 0, 9, 19, 23 }
Off point 44 = $P_{66} = (2, 6, 1)$ lies on 4 bisecants : { 0, 11, 18, 22 }
Off point 45 = $P_{67} = (3, 6, 1)$ lies on 4 bisecants : { 0, 1, 13, 17 }
Off point 46 = $P_{68} = (4, 6, 1)$ lies on 4 bisecants : { 0, 7, 12, 27 }
Off point 47 = $P_{69} = (5, 6, 1)$ lies on 4 bisecants : { 0, 2, 6, 8 }
Off point 48 = $P_{70} = (6, 6, 1)$ lies on 4 bisecants : { 0, 3, 16, 25 }
Off point 49 = $P_{71} = (7, 6, 1)$ lies on 4 bisecants : { 0, 4, 14, 24 }
Off point 50 = $P_{72} = (8, 6, 1)$ lies on 4 bisecants : { 0, 5, 15, 26 }
Off point 51 = $P_{73} = (0, 7, 1)$ lies on 4 bisecants : { 2, 7, 16, 24 }
Off point 52 = $P_{74} = (1, 7, 1)$ lies on 4 bisecants : { 4, 8, 12, 25 }
Off point 53 = $P_{75} = (2, 7, 1)$ lies on 4 bisecants : { 3, 6, 14, 27 }
Off point 54 = $P_{79} = (6, 7, 1)$ lies on 4 bisecants : { 17, 20, 22, 26 }
Off point 55 = $P_{80} = (7, 7, 1)$ lies on 4 bisecants : { 5, 10, 13, 18 }
Off point 56 = $P_{81} = (8, 7, 1)$ lies on 4 bisecants : { 1, 11, 15, 21 }
Off point 57 = $P_{85} = (3, 8, 1)$ lies on 4 bisecants : { 14, 16, 18, 20 }
Off point 58 = $P_{86} = (4, 8, 1)$ lies on 4 bisecants : { 3, 4, 21, 22 }
Off point 59 = $P_{87} = (5, 8, 1)$ lies on 4 bisecants : { 10, 11, 24, 25 }
Off point 60 = $P_{88} = (6, 8, 1)$ lies on 4 bisecants : { 5, 8, 23, 27 }
Off point 61 = $P_{89} = (7, 8, 1)$ lies on 4 bisecants : { 2, 9, 12, 26 }
Off point 62 = $P_{90} = (8, 8, 1)$ lies on 4 bisecants : { 6, 7, 15, 19 }

The follwing command creates the curve over $\mathbb{F}_{13}$ with catalogue number 0 and produces a report. The comand also exports the orbit of bitangents of length 4:

Example 349

```
quartic_curve_13_0_report:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 13 -end \
▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
▷  ▷  -define C -quartic_curve -space P -catalogue 0 \
▷  ▷  ▷  -transform "10,4,1,11,5,11,4,1,1" \
▷  ▷  ▷  -transform_inverse "9,1,0,12,9,0,2,10,11" \
▷  ▷  -end \
▷  ▷  -with C -do \
▷  ▷  ▷  -quartic_curve_activity \
▷  ▷  ▷  ▷  -report \
▷  ▷  ▷  -end \
▷  ▷  -with C -do \
▷  ▷  ▷  -quartic_curve_activity \
▷  ▷  ▷  ▷  -extract_orbit_on_bitangents_by_length 4 \
▷  ▷  ▷  -end
▷  #pdflatex quartic_curve_catalogue_q13_iso0_report.tex
▷  #$(OPEN) quartic_curve_catalogue_q13_iso0_report.pdf
```

Suppose we want to create reports on all quartic curves for a given value of q, utilizing the Orbiter knowledge base of quartic curves. Here is an example, considering the quartic curves over $\mathbb{F}_{19}$. We use a makefile variable to set the number of curves:

Example 350

```
NB_QUARTIC_CURVES_Q19=14
```

Now, we create each curve one-by-one and produce a report. To do so, we utilize the loop command. Once done, we translate the latex files:

Example 351

```
quartic_curves_19_report:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 19 -end \
▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
▷  ▷  -loop L 0 $(NB_QUARTIC_CURVES_Q19) 1 \
▷  ▷  -define C -quartic_curve -space P -catalogue %L -end \
▷  ▷  -with C -do \
▷  ▷  ▷  -quartic_curve_activity \
▷  ▷  ▷  ▷  -report \
▷  ▷  ▷  -end \
▷  ▷  -end_loop L
▷  ▷  pdflatex quartic_curve_catalogue_q19_iso0_report.tex
▷  ▷  pdflatex quartic_curve_catalogue_q19_iso1_report.tex
```

```
▷  ▷  pdflatex quartic_curve_catalogue_q19_iso2_report.tex
▷  ▷  pdflatex quartic_curve_catalogue_q19_iso3_report.tex
▷  ▷  pdflatex quartic_curve_catalogue_q19_iso4_report.tex
▷  ▷  pdflatex quartic_curve_catalogue_q19_iso5_report.tex
▷  ▷  pdflatex quartic_curve_catalogue_q19_iso6_report.tex
▷  ▷  pdflatex quartic_curve_catalogue_q19_iso7_report.tex
▷  ▷  pdflatex quartic_curve_catalogue_q19_iso8_report.tex
▷  ▷  pdflatex quartic_curve_catalogue_q19_iso9_report.tex
▷  ▷  pdflatex quartic_curve_catalogue_q19_iso10_report.tex
▷  ▷  pdflatex quartic_curve_catalogue_q19_iso11_report.tex
▷  ▷  pdflatex quartic_curve_catalogue_q19_iso12_report.tex
▷  ▷  pdflatex quartic_curve_catalogue_q19_iso13_report.tex
```

In the next example, we will study the generation of a quartic curve from a cubic surface. The example is the quartic curve over $\mathbb{F}_9$. It is interesting, as it contains the maximum number of Kovalevski points, which is 63 (a Kovalevski point is a point where 4 bitangents meet). There are two cubic surfaces with 27 lines over $\mathbb{F}_9$. The first one has 10 Eckardt points, whereas the second one has 9. The quartic curve arises from the surface with 9 Eckardt points (OCN=1). The surface has a unique point not on any line. For this reason, we set the orbit number on points not on lines to zero. Here is the command:

Example 352

```
quartic_curve_q9_1:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 9 -end \
▷  ▷  -define P2 -projective_space -n 2 -field F -end \
▷  ▷  -define P3 -projective_space -n 3 -field F -end \
▷  ▷  -define S9_1 -cubic_surface -space P3 -catalogue 1 -end \
▷  ▷  -define C -quartic_curve -space P2 -from_cubic_surface S9_1 0 -end \
▷  ▷  -with C -do \
▷  ▷  ▷  -quartic_curve_activity \
▷  ▷  ▷  ▷  -report \
▷  ▷  ▷  -end
▷  pdflatex quartic_curve_surface_surface_pt_orb_0_report.tex
▷  $(OPEN) quartic_curve_surface_surface_pt_orb_0_report.pdf
```

8.6 Classification of Quartic Curves

Let us now address the problem of classification of quartic curves. For a fixed field $\mathbb{F}_q$, q odd, we can exploit the previously classified list of cubic surfaces as a start. For each surface, and for each orbit on points not on lines, we perform the projection along the tangent cone to create one quartic curve. This guarantees that each isomorphism type of quartic curve with 28 bitangents will be created. Afterwards, we apply isomorphism testing based on canonical forms. This is based on subtructures and using canonical froms for graphs using the built-in package Nauty.

Let us look at some examples of this algorithm. We try $q = 7$ and then $q = 13$. We will set a makefile variable to the number of (isomorphism types of) cubic surfaces with 27 lines over $\mathbb{F}_q$. For $q = 7$, there is exactly one isomorphism type, so we put

Example 353

```
NB_CUBIC_SURFACES_Q7=1
```

The next command creates a list of quartic curves using a projection construction. For each isomorphism type of cubic surface, and for each point not on any line, we consider the intersection of the tangent cone with the surface and project onto a plane not containing the point. Because of symmetry, it suffices to perform this projection only for a set of representatives of the orbits on points not on lines. Here is the Orbiter command for $q = 7$:

Example 354

```
quartic_curves_q7:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 7 -end \
▷  ▷  -define P -projective_space -n 3 -field F -end \
▷  ▷  -loop L 0 $(NB_CUBIC_SURFACES_Q7) 1 \
▷  ▷  ▷  -define S_%L -cubic_surface -space P -catalogue %L -end \
▷  ▷  -end_loop L \
▷  ▷  -print_symbols \
▷  ▷  -loop L 0 $(NB_CUBIC_SURFACES_Q7) 1 \
▷  ▷  ▷  -with S_%L -do \
▷  ▷  ▷  -cubic_surface_activity \
▷  ▷  ▷  ▷  -export_all_quartic_curves \
▷  ▷  ▷  -end \
▷  ▷  -end_loop L \
▷  ▷  -print_symbols
```

The resulting curves are written to file. Unfortunately, in this example, there is no point which does not lie on any line of the surface. This means that no quartic curve with 28 lines exists over $\mathbb{F}_7$.

We move on to the next example, which is $q = 13$. Again, we use a makefile variable to set the number of isomorphism types of cubic surfaces with 27 lines over $\mathbb{F}_{13}$. There are exactly 4 types:

Example 355

```
NB_CUBIC_SURFACES_Q13=4
```

Just like before, we create all quartic curves arising from projections:

Example 356

```
quartic_curves_q13:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 13 -end \
▷  ▷  -define P -projective_space -n 3 -field F -end \
▷  ▷  -loop L 0 $(NB_CUBIC_SURFACES_Q13) 1 \
▷  ▷  ▷  -define S_%L -cubic_surface -space P -catalogue %L -end \
▷  ▷  -end_loop L \
▷  ▷  -print_symbols \
▷  ▷  -loop L 0 $(NB_CUBIC_SURFACES_Q13) 1 \
▷  ▷  ▷  -with S_%L -do \
▷  ▷  ▷  -cubic_surface_activity \
▷  ▷  ▷  ▷  -export_all_quartic_curves \
▷  ▷  ▷  -end \
▷  ▷  -end_loop L \
▷  ▷  -print_symbols
```

Every cubic surface creates a number of quartic curves. Namely, each orbit of the automorphism group on points not on lines of the cubic surface creates one quartic curve. The curves arising this way are written to file. There is one file for each surface. To prepare for the classification, we first combine these output files into one:

Example 357

```
quartic_curves_q13_combine:
▷  $(ORBITER) -v 3 \
▷  ▷  -csv_file_concatenate_from_mask $(NB_CUBIC_SURFACES_Q13) \
▷  ▷  ▷  surface_catalogue_q13_iso%ld_quartics.csv \
▷  ▷  ▷  quartics_q13.csv
```

The following command performs the classification of quartic curves that are in the combined file. The result of this computation is the classification of quartic curves with 28 bitangents over the given field (here, $\mathbb{F}_{13}$):

Example 358

```
quartic_curves_q13_classify:
▷  $(ORBITER) -v 30 \
▷  ▷  -define F -finite_field -q 13 -end \
▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
▷  ▷  -define R -polynomial_ring \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -number_of_variables 3 \
▷  ▷  ▷  -homogeneous_of_degree 4 \
▷  ▷  ▷  -monomial_ordering_partition \
▷  ▷  ▷  -variables "y0,y1,y2" "y_0,y_1,y_2" \
▷  ▷  -end \
▷  ▷  -define O -orbits -classification_by_canonical_form \
▷  ▷  ▷  -space P \
```

```
▷ ▷ ▷ -ring R \
▷ ▷ ▷ -input_fname_mask quartics_q13.csv \
▷ ▷ ▷ -nb_files 1 \
▷ ▷ ▷ -output_fname quartic_curves_q13_classified \
▷ ▷ ▷ -label_po_go "PO_GO" \
▷ ▷ ▷ -label_po_index "PO_INDEX" \
▷ ▷ ▷ -label_po "PO" \
▷ ▷ ▷ -label_so "orbit" \
▷ ▷ ▷ -label_equation "curve" \
▷ ▷ ▷ -label_points "pts_on_curve" \
▷ ▷ ▷ -label_lines "bitangents" \
▷ ▷ ▷ -carry_through "NB_E" \
▷ ▷ ▷ -carry_through "NB_DOUBLE" \
▷ ▷ ▷ -carry_through "NB_SINGLE" \
▷ ▷ ▷ -carry_through "NB_ZERO" \
▷ ▷ ▷ -algorithm_substructure \
▷ ▷ ▷ -substructure_size 4 \
▷ ▷ ▷ -end \
▷ ▷ -end \
▷ -with 0 -do -orbits_activity \
▷ ▷ -report \
▷ ▷ -report_options \
▷ ▷ ▷ -fname quartics_q13 \
▷ ▷ -end \
▷ -end
▷ pdflatex quartics_q13_orbits.tex
▷ $(OPEN) quartics_q13_orbits.pdf
```

Over the given field, we find exactly two isomorphism types. The output is shown below, showing the canonical equation of each curve, the automorphism group by means of generators and group order, and the number of rational points of the curve over the given field.

Classification
$q = 13$
Number of isomorphism classes: 2
Automorphism group order statistic: 48, 24

The isomorphism classes are:
Isomorphism class 0 / 2:
Number of rational points over $\mathbb{F}_{13}$: 8
Canonical equation:

$$5X_0^3X_1 + 4X_0^3X_2 + 7X_0X_1^3 + 11X_1^3X_2 + 10X_0X_2^3 + 6X_1X_2^3 + 8X_0^2X_1^2 \\ +7X_0^2X_2^2 + 7X_1^2X_2^2 + 5X_0^2X_1X_2 + 7X_0X_1^2X_2 + X_0X_1X_2^2$$

Stabilizer order: 24
The stabilizer:
Strong generators for a group of order 24:

$$\begin{bmatrix} 10 & 0 & 0 \\ 4 & 2 & 6 \\ 1 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 11 & 2 & 4 \\ 0 & 0 & 4 \\ 0 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 12 & 12 & 12 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

1,0,0,3,8,11,4,4,4,
1,12,11,0,0,11,0,6,0,
1,1,1,0,12,0,12,0,0,
Isomorphism class 1 / 2:
Number of rational points over $\mathbb{F}_{13}$: 4
Canonical equation:

$$5X_2^4 + 5X_0^3X_1 + 7X_0X_1^3 + X_0^2X_1^2$$

Stabilizer order: 48
The stabilizer:
Strong generators for a group of order 48:

$$\begin{bmatrix} 12 & 0 & 0 \\ 0 & 12 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 8 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 4 & 0 & 0 \\ 9 & 12 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 10 & 0 \\ 3 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

1,0,0,0,1,0,0,0,12,
1,0,0,0,1,0,0,0,5,
1,0,0,12,3,0,0,0,10,
0,1,0,12,12,0,0,0,4,

The classification also produces a cpp file. This is C++ code for the classification. The file contains the list of curves from the classification, together with extra information about the automorphism group. It can be used to build the knowledge about these quartic curves into Orbiter, by means of Orbiter's knowledge base of combinatorial objects. This way, the data that is computed by Orbiter can feed back into a new version of Orbiter.

8.7 Interface to GAP

We wish to export a cubic surface to GAP. The following command creates the Hirschfeld surface and exports it to GAP

Example 359

```
surface_4_0_export_gap:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 4 -end \
▷  ▷  -define P -projective_space -n 3 -field F -end \
▷  ▷  -define S -cubic_surface -space P -catalogue 0 -end \
▷  ▷  -with S -do \
▷  ▷  -cubic_surface_activity \
▷  ▷  ▷  -export_gap \
▷  ▷  -end
```

The following GAP file is written:

```
LoadPackage("fining");
# Cubic surface catalogue_q4_iso0
# Group:
mat1 := [[Z(4)^1,0*Z(4),0*Z(4),0*Z(4)],
[0*Z(4),Z(4)^1,0*Z(4),0*Z(4)],
[0*Z(4),0*Z(4),Z(4)^1,0*Z(4)],
[0*Z(4),0*Z(4),0*Z(4),Z(4)^1]];
frob1 := FrobeniusAutomorphism(GF(4))^1;
psi1 := ProjectiveSemilinearMap(mat1, frob1,GF(4));
mat2 := [[Z(4)^2,0*Z(4),0*Z(4),0*Z(4)],
[0*Z(4),Z(4)^1,0*Z(4),0*Z(4)],
[0*Z(4),0*Z(4),Z(4)^1,0*Z(4)],
[0*Z(4),0*Z(4),0*Z(4),Z(4)^2]];
frob2 := FrobeniusAutomorphism(GF(4))^1;
psi2 := ProjectiveSemilinearMap(mat2, frob2,GF(4));
mat3 := [[Z(4)^0,0*Z(4),0*Z(4),0*Z(4)],
[0*Z(4),Z(4)^2,0*Z(4),0*Z(4)],
[0*Z(4),0*Z(4),Z(4)^2,0*Z(4)],
[0*Z(4),0*Z(4),0*Z(4),Z(4)^0]];
frob3 := FrobeniusAutomorphism(GF(4))^0;
psi3 := ProjectiveSemilinearMap(mat3, frob3,GF(4));
mat4 := [[Z(4)^2,0*Z(4),0*Z(4),0*Z(4)],
[0*Z(4),Z(4)^2,0*Z(4),0*Z(4)],
[0*Z(4),0*Z(4),Z(4)^2,0*Z(4)],
[Z(4)^2,0*Z(4),0*Z(4),Z(4)^2]];
frob4 := FrobeniusAutomorphism(GF(4))^0;
psi4 := ProjectiveSemilinearMap(mat4, frob4,GF(4));
mat5 := [[Z(4)^0,0*Z(4),0*Z(4),0*Z(4)],
[0*Z(4),Z(4)^2,0*Z(4),0*Z(4)],
[Z(4)^1,0*Z(4),Z(4)^2,0*Z(4)],
[0*Z(4),Z(4)^1,0*Z(4),Z(4)^0]];
frob5 := FrobeniusAutomorphism(GF(4))^1;
psi5 := ProjectiveSemilinearMap(mat5, frob5,GF(4));
```

```
mat6 := [[Z(4)^0,0*Z(4),0*Z(4),0*Z(4)],
[0*Z(4),Z(4)^2,0*Z(4),0*Z(4)],
[Z(4)^1,Z(4)^2,Z(4)^2,0*Z(4)],
[Z(4)^0,Z(4)^1,0*Z(4),Z(4)^0]];
frob6 := FrobeniusAutomorphism(GF(4))^1;
psi6 := ProjectiveSemilinearMap(mat6, frob6,GF(4));
mat7 := [[Z(4)^1,0*Z(4),0*Z(4),0*Z(4)],
[Z(4)^1,Z(4)^1,0*Z(4),0*Z(4)],
[0*Z(4),0*Z(4),Z(4)^1,0*Z(4)],
[0*Z(4),0*Z(4),Z(4)^1,Z(4)^1]];
frob7 := FrobeniusAutomorphism(GF(4))^1;
psi7 := ProjectiveSemilinearMap(mat7, frob7,GF(4));
mat8 := [[Z(4)^2,0*Z(4),0*Z(4),0*Z(4)],
[Z(4)^2,Z(4)^0,Z(4)^0,0*Z(4)],
[0*Z(4),Z(4)^0,0*Z(4),0*Z(4)],
[Z(4)^2,Z(4)^0,0*Z(4),Z(4)^2]];
frob8 := FrobeniusAutomorphism(GF(4))^1;
psi8 := ProjectiveSemilinearMap(mat8, frob8,GF(4));
mat9 := [[Z(4)^0,Z(4)^2,Z(4)^0,Z(4)^2],
[0*Z(4),Z(4)^2,Z(4)^2,Z(4)^1],
[Z(4)^2,Z(4)^0,Z(4)^2,Z(4)^0],
[Z(4)^0,Z(4)^1,0*Z(4),Z(4)^0]];
frob9 := FrobeniusAutomorphism(GF(4))^0;
psi9 := ProjectiveSemilinearMap(mat9, frob9,GF(4));
gens := [psi1, psi2, psi3, psi4, psi5, psi6, psi7, psi8, psi9];
G := Group(gens);
Size(G);
r := PolynomialRing(GF(4),["X0","X1","X2","X3"]);
Eqn := (r.1)^2*(r.4)+(r.2)^2*(r.3)+(r.2)*(r.3)^2+(r.1)*(r.4)^2;
```

Chapter 9

Applications

9.1 Number Theory

Representation Theory Activities

Command	Arguments	Purpose
-representation_on_polynomials	R	Computes the representation of the given group on homogeneous polynomials from the ring R. This is a group theoretic activity as described in Section 6.5.

Table 9.1: Representation Theory Activities

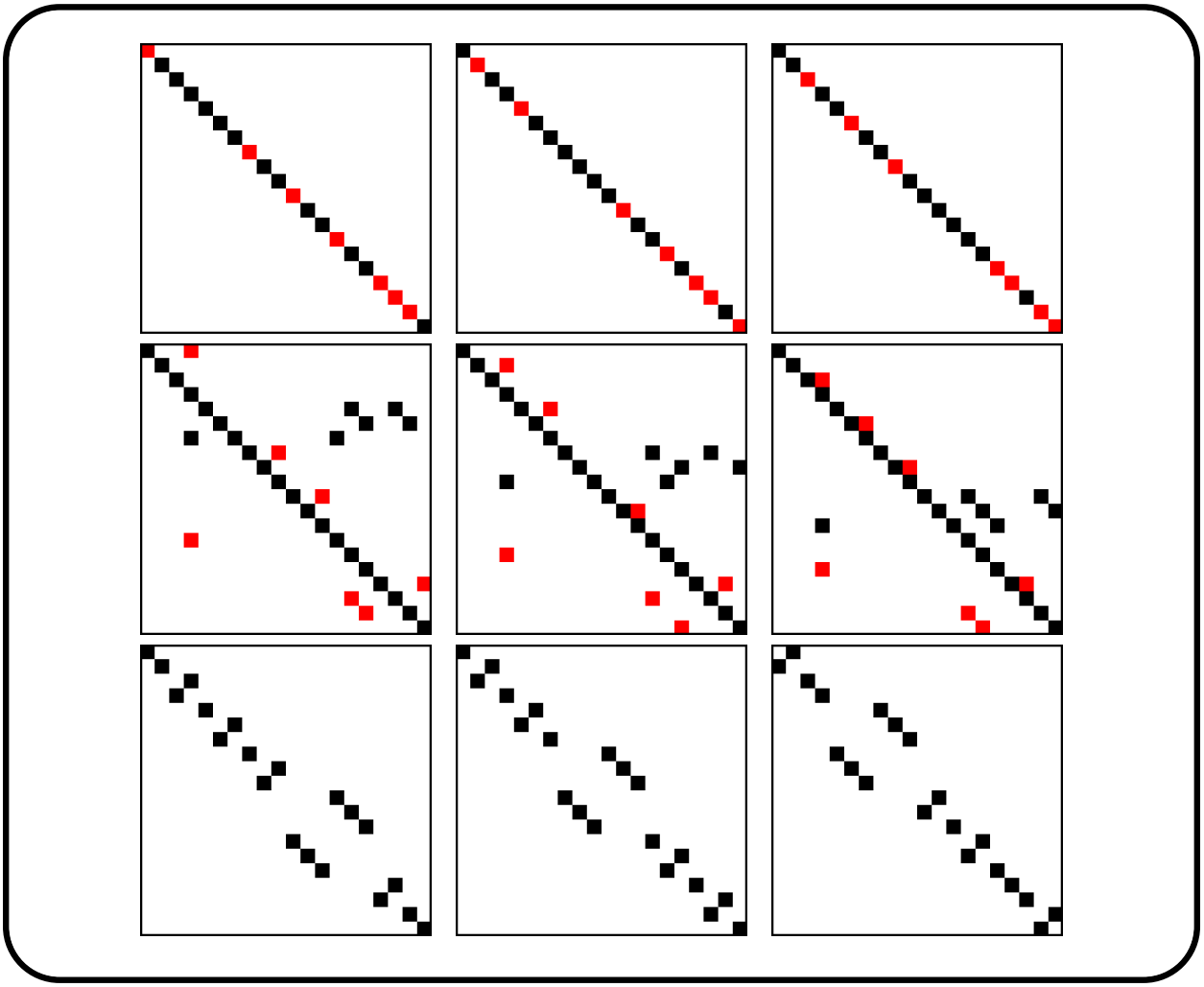
9.2 Representation Theory

Table 9.1 list the Orbiter commands for group representations. The command

Example 360

```
representation_on_polynomials_of_degree_3:
▷  $(ORBITER) -v 4 \
▷  ▷  -define F -finite_field -q 3 -end \
▷  ▷  -define G -linear_group -PGL 4 F -end \
▷  ▷  -define R -polynomial_ring \
▷  ▷  ▷  -field F \
▷  ▷  ▷  -number_of_variables 4 \
▷  ▷  ▷  -homogeneous_of_degree 3 \
▷  ▷  ▷  -monomial_ordering_partition \
▷  ▷  ▷  -variables "X,Y,Z,W" "X,Y,Z,W" \
▷  ▷  -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -representation_on_polynomials R \
▷  ▷  -end
▷  $(ORBITER) -v 2 \
▷  ▷  -loop L 0 9 1 -draw_matrix \
▷  ▷  ▷  -input_csv_file PGL_4_3_rep_3_%L.csv \
▷  ▷  ▷  -box_width 40 -bit_depth 24 -partition 3 20 20 -end \
▷  ▷  -end_loop L
```

creates $G = \mathrm{PGL}(4,3)$ and computes the representation on polynomials of degree 3 in 4 variables. The representation has degree 20. The second command produces bitmap drawings for the representing matrices associated with a generating set of the group.



Cryptographic Commands

Command	Arguments	Purpose
-solovay_strassen	a n	Performs n Solovay / Strassen tests on the number a
-miller_rabin	a n	Performs n Miller / Rabin tests on the number a
-fermat	a n	Performs n Fermat tests on the number a
-find_pseudoprime	a n_1 n_2 n_3	Computes a pseudoprime which survives n_1 Fermat tests, n_2 Miller Rabin tests, n_3 Solovay Strassen tests
-find_strong_pseudoprime	a n_1 n_2	Computes a pseudoprime which survives n_1 Fermat tests and n_2 Miller Rabin tests
-RSA_encrypt_text	d n b text	Using blocks of b letters at a time, encrypt "text" using RSA with exponent d modulo n
-RSA	d n list-of-integers	encrypt the given sequence of integers using RSA with exponent d modulo n

Table 9.2: Cryptographic Commands

9.3 Cryptography

In Table 9.2, some global cryptographic commands are shown. Some of these commands require a finite field and appear as a finite field activity, see Table 9.3. For instance,

Example 361

```
EC_add:
▷ $(ORBITER) -v 2 \
▷ ▷ -define F -finite_field -q 11 -end \
▷ ▷ -with F -do \
▷ ▷ -finite_field_activity \
▷ ▷ -EC_add 1 3 "1,4" "1,4" -end
```

adds the point $(1, 4)$ on the curve $y^2 = x^3 + x + 3 \mod 11$ to itself. The command

Example 362

```
EC_cyclic_subgroup:
▷ $(ORBITER) -v 2 \
▷ ▷ -define F -finite_field -q 11 -end \
▷ ▷ -with F -do \
▷ ▷ -finite_field_activity \
▷ ▷ -EC_cyclic_subgroup 1 3 "1,4" -end
```

computes the cyclic subgroup generated by the point $(1, 4)$ on the curve $y^2 = x^3 + x + 3 \mod 11$. The command

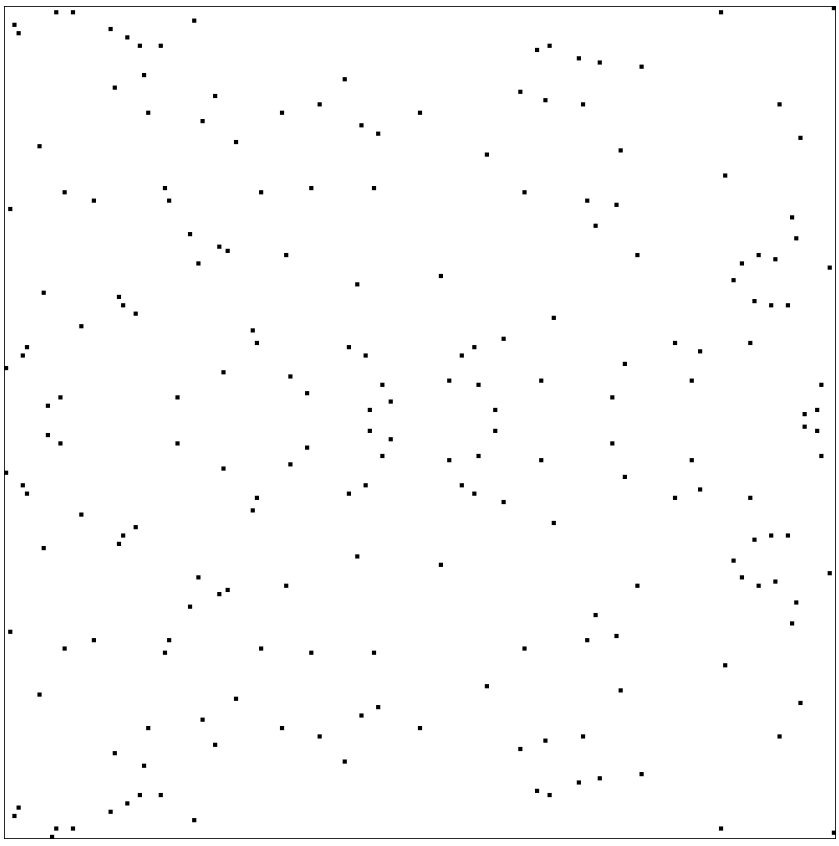
Finite Field Activities for Cryptography		
Command	Arguments	Purpose
-EC_add	$a\ b\ i_1\ i_2$	On the elliptic curve $y^2 \equiv x^3 + ax + b$ in $\mathbb{F}_q$, add the points with indices i_1 and i_2, each given as a pair x, y
-EC_points	$a\ b$	Computes all points of the elliptic curve $y^2 \equiv x^3 + ax + b$ over $\mathbb{F}_q$
-EC_multiple_of	$a\ b$ pt n	Computes the n fold multiple of the given point pt on the elliptic curve $y^2 \equiv x^3+ax+b$ over $\mathbb{F}_q$
-EC_cyclic_subgroup	$a\ b$ pt	Computes the cyclic subgroup generated by the given point pt on the elliptic curve $y^2 \equiv x^3 + ax + b$ over $\mathbb{F}_q$
-EC_Koblitz_encoding	$a\ b\ s$ pt plain	Computes the Koblitz encoding of "plain" (all caps) on the elliptic curve $y^2 \equiv x^3+ax+b$ over $\mathbb{F}_q$ using the base point pt and the secret exponent s
-EC_bsgs	$a\ b$ pt n cipher	Prepare the baby-step giant-step tables for the ciphertext "cipher" on the elliptic curve $y^2 \equiv x^3+ax+b$ over $\mathbb{F}_q$ using the base point pt of order n
-EC_bsgs_decode	$a\ b$ pt n cipher round-keys	Decodes the ciphertext "cipher" on the elliptic curve $y^2 \equiv x^3 + ax + b$ over $\mathbb{F}_q$ using the base point pt of order n and the round keys "keys"
-EC_discrete_log	$a\ b$ pt base-pt	Computes the elliptic curve discrete log analogue of pt with respect to base-pt on the elliptic curve $y^2 \equiv x^3 + ax + b$ over $\mathbb{F}_q$
-NTRU_encrypt	$N\ p\ H\ R\ M$	NTRU encryption for the message $M(X)$ using the public key $H(X)$ and one-time-key $R(X)$.
-polynomial_center_lift	$A(X)$	Compute the center lift mod q for the coefficients of A
-polynomial_reduce_mod_p	$p\ A(X)$	Reduce the coefficients of the polynomial A modulo p

Table 9.3: Finite Field Activities for Cryptography

Example 363

```
EC_points_199:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 199 -end \
▷  ▷  -with F -do \
▷  ▷  -finite_field_activity \
▷  ▷  -EC_points "EC_5_7_q199" 5 7 -end
▷  $(ORBITER) -v 2 \
▷  ▷  -draw_matrix -input_csv_file EC_5_7_q199_points_xy.csv \
▷  ▷  -box_width 10 -bit_depth 24 \
▷  ▷  -partition 2 199 199 -end
```

computes all points on the curve $y^2 = x^3 + 5x + 7 \mod 199$ and produces a bitmap drawing of the points in the affine plane:



Both the x-axis and the y-axis are indexed by the field elements from 0 to 198.

The command

Example 364

```
EC_Koblitz_encoding:
▷  $(ORBITER) -v 6 -seed 17 \
```

```
▷  ▷  -define F -finite_field -q 199 -end \
▷  ▷  -with F -do \
▷  ▷  -finite_field_activity \
▷  ▷  -EC_Koblitz_encoding 5 7 67 "147,164" "DEADBEEF" \
▷  ▷  -end
```

encode the message "DEADBEEF" on the curve $y^2 = x^3 + 5x + 7 \mod 199$ using the base point $(147, 164)$ and the secret key 67. The ith input character is encoded as two points (R_i, T_i) on the curve using the Elgamal scheme. A random round key is generated for each plaintext symbol. As seen in this example, the `-seed` command can be used to seed the random number generator with an arbitrary integer (here 17).

The command

Example 365

```
EC_bsgs:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 199 -end \
▷  ▷  -with F -do \
▷  ▷  -finite_field_activity \
▷  ▷  -EC_bsgs 5 7 "147,164" 212 \
▷  ▷  "172,158,45,195,50,22,10,103,55,33,50,22,\
▷  ▷  145,105,31,74,73,155,67,60,25,6" \
▷  ▷  -end
```

performs a baby-step-giant-step brute force attack on the ciphertext sequence

$$R_i = (172, 158), (45, 195), (50, 22), (10, 103), (55, 33), \\ (50, 22), (145, 105), (31, 74), (73, 155), (67, 60), (25, 6),$$

using the base point $(147, 164)$ on the curve $y^2 = x^3 + 5x + 7 \mod 199$, assuming a group order of 212. The command

Example 366

```
EC_bsgs_decode:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 199 -end \
▷  ▷  -with F -do \
▷  ▷  -finite_field_activity \
▷  ▷  -EC_bsgs_decode 5 7 "129,176" 212 \
▷  ▷  "127,188,51,141,85,29,106,90,41,105,179,71,\
▷  ▷  171,2,16,197,183,72,27,129,37,10" \
▷  ▷  "50,179,169,13,153,169,115,116,188,110,176" \
▷  ▷  -end
```

decodes the ciphertext sequence

$$T_i = (127, 188), (51, 141), (85, 29), (106, 90), (41, 105), (179, 71), \\ (171, 2), (16, 197), (183, 72), (27, 129), (37, 10),$$

assuming round keys

$$k_i = 50, 179, 169, 13, 153, 169, 115, 116, 188, 110, 176,$$

using the base point $(147, 164)$ on the curve $y^2 = x^3 + 5x + 7 \mod 199$, and assuming a group order of 212.

The next sequence of examples discusses the NTRU cryptosystem (cf. Example 7.53 in [38]). In the example, we choose the parameters of the cryptosystem to be $(N, p, q, d) = (7, 41, 3, 2)$. Orbiter uses the following convention for polynomials over a finite field $\mathbb{F}_q$: The coefficients of $A(X) = a_0 + a_1 X + \cdots + a_d X^d$ are listed as a sequence, starting with the constant term and ending with the leading coefficient. The cryptosystem requires coefficients a_i in the range $-\frac{p}{2} \le a_i \le \frac{p}{2}$. So, in an extension to the conventionas for field elements in $\mathbb{F}_q$, Orbiter allows negative coefficients as well. The assumption is that q is prime and negative coefficients are considered modulo q. In the example, Alice picks the private polynomials $f(x) = x^6 - x^4 + x^3 + x^2 - 1$ (with $d+1$ coefficients equal to plus one and d coefficients equal to minus one) and $g(x) = x^6 + x^4 - x^2 - x$ with d coefficients plus one and d coefficients minus one. We also need the polynomial $x^N - 1$. The makefile commands

Example 367

```
NTRU_N=7
```

Example 368

```
NTRU_P=3
```

Example 369

```
NTRU_Q=41
```

Example 370

```
NTRU_D=2
```

Example 371

```
NTRU_XN1="-1,0,0,0,0,0,0,1,"
```

Example 372

```
ALICE_PRIVATE_F="-1,0,1,1,-1,0,1"
```

Example 373

```
ALICE_PRIVATE_G="0,-1,-1,0,1,0,1"
```

are used to set up the appropriate variables according to these choices.

Regarding the NTRU set-up, Alice needs to compute her private keys $F_p(x)$ and $F_q(x)$. These two polynomials are defined as follows:

1. $F_p(x)$ is the inverse of $f(x)$ in $\mathbb{F}_p[x]/(x^n - 1)$,

2. $F_q(x)$ the inverse of $f(x)$ in $\mathbb{F}_q[x]/(x^n - 1)$.

To this end, we can use the `extended_gcd_for_polynomials` command from Table 3.3. The following two makefile commands do the job:

Example 374

```
NTRU_Alice1:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q $(NTRU_Q) -end \
▷  ▷  -with F -do \
▷  ▷  -finite_field_activity \
▷  ▷  -extended_gcd_for_polynomials \
▷  ▷  ▷  $(NTRU_XN1) $(ALICE_PRIVATE_F) \
▷  ▷  -end
```

Example 375

```
ALICE_PRIVATE_FQ="37,2,40,21,31,26,8"
```

Example 376

```
NTRU_Alice2:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q $(NTRU_P) -end \
▷  ▷  -with F -do \
▷  ▷  -finite_field_activity \
▷  ▷  -extended_gcd_for_polynomials \
▷  ▷  ▷  $(NTRU_XN1) $(ALICE_PRIVATE_F) \
▷  ▷  -end
```

The resulting polynomials (indicated as comments by means of the # symbol) are again encoded as makefile variables.

Example 377

```
ALICE_PRIVATE_FP="1,1,1,1,0,2,1"
```

There is a chance that the polynommial $f(x)$ does not have an inverse in either $\mathbb{F}_p[x]$ or in $\mathbb{F}_q[x]$. In that case, Alice simply chooses a different polynomial $f(x)$ and tries again. Alice can now compute her public key:

Example 378

```
NTRU_Alice_public_key:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q $(NTRU_Q) -end \
▷  ▷  -with F -do \
▷  ▷  -finite_field_activity \
▷  ▷  -polynomial_mult_mod $(ALICE_PRIVATE_F) \
▷  ▷  ▷  $(ALICE_PRIVATE_G) $(NTRU_XN1) \
▷  ▷  -end
```

Example 379

```
ALICE_PUBLIC_KEY="30,26,8,38,2,40,20"
```

The public key is assigned to the makefile variable `ALICE_PUBLIC_KEY`. Now, Bob chooses his message to Alice and his one-time-key. The message must be the center lift of a polynomial in $\mathbb{F}_p[x]$. The round-key must have exactly d coefficients one and d coefficients -1 (rest zeroes).

Example 380

```
BOB_MESSAGE="1,-1,1,1,0,-1"
```

Example 381

```
BOB_ONE_TIME_KEY="-1,1,0,0,0,-1,1"
```

The encryption proceeds using the `NTRU_encrypt` command, and the result is stored in the makefile variable `BOB_ENCRYPT`:

Example 382

```
NTRU_encrypt:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q $(NTRU_Q) -end \
▷  ▷  -with F -do \
▷  ▷  -finite_field_activity \
▷  ▷  -NTRU_encrypt \
▷  ▷  $(NTRU_N) $(NTRU_P) $(ALICE_PUBLIC_KEY) \
▷  ▷  $(BOB_ONE_TIME_KEY) $(BOB_MESSAGE) \
▷  ▷  -end
```

Example 383

```
BOB_ENCRYPT= "25,3,40,2,4,19,31"
```

Decryption is done in five steps.

Example 384

```
NTRU_decrypt1:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q $(NTRU_Q) -end \
▷  ▷  -with F -do \
▷  ▷  -finite_field_activity \
▷  ▷  -polynomial_mult_mod $(ALICE_PRIVATE_F) \
▷  ▷  ▷  $(BOB_ENCRYPT) $(NTRU_XN1) \
▷  ▷  -end
```

Example 385

```
ALICE_C1="40,1,40,40,33,10,1"
```

Example 386

```
NTRU_decrypt2:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q $(NTRU_Q) -end \
▷  ▷  -with F -do \
▷  ▷  -finite_field_activity \
▷  ▷  -polynomial_center_lift $(ALICE_C1) -end
```

Example 387

```
ALICE_C2="-1,1,-1,-1,-8,10,1"
```

Example 388

```
NTRU_decrypt3:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q $(NTRU_P) -end \
▷  ▷  -with F -do \
▷  ▷  -finite_field_activity \
▷  ▷  -polynomial_reduce_mod_p $(ALICE_C2) -end
```

Example 389

```
ALICE_C3="2,1,2,2,1,1,1"
```

Example 390

```
NTRU_decrypt4:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q $(NTRU_Q) -end \
▷  ▷  -with F -do \
▷  ▷  -finite_field_activity \
▷  ▷  -polynomial_mult_mod $(ALICE_PRIVATE_FP) \
▷  ▷  ▷  $(ALICE_C3) $(NTRU_XN1) \
▷  ▷  -end
```

Example 391

```
ALICE_C4="1,2,1,1,0,2"
```

Example 392

```
NTRU_decrypt5:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q $(NTRU_P) -end \
▷  ▷  -with F -do \
▷  ▷  -finite_field_activity \
▷  ▷  -polynomial_center_lift $(ALICE_C4) -end
```

Decryption produces Bob's message to Alice.

Chapter 10

Coding Theory

10.1 Introduction

Table 10.1 lists global Orbiter commands related to coding theory.

The command

Example 393

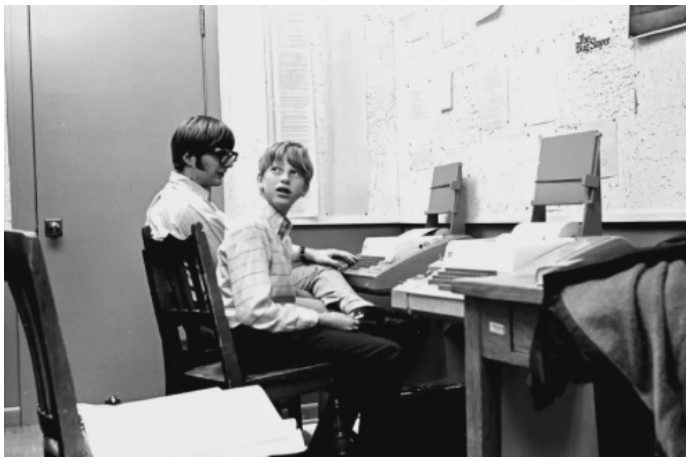
```
Allen_Gates_noise_1_percent:
▷  cp $(MY_PATH)/examples/users_guide/allen_Gates.bmp .
▷  $(ORBITER) -v 3 \
▷  ▷  -random_noise_in_bitmap_file \
▷  ▷  allen_Gates.bmp \
▷  ▷  allen_Gates_1.bmp \
▷  ▷  1 100
▷  $(OPEN) allen_Gates_1.bmp
```

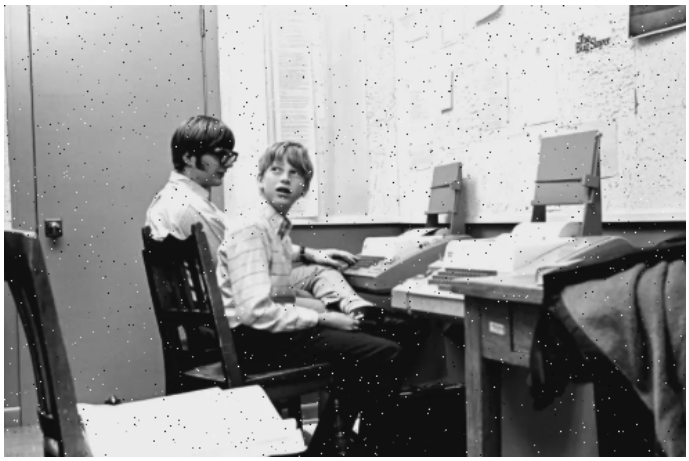
simulates random noise at the 1 percent level applied to the file `allen_Gates.bmp`, see below

Global Coding Theoretic Commands

Command	Arguments	Purpose
`-make_macwilliams_system`	$q\ n\ k$	Create the MacWilliams equations for the weight enumerator of the dual code.
`-table_of_bounds`	$n_{\max}\ q$	Make a table of bounds for q-ary linear codes for all $k \leq n \leq n_{\max}$
`-make_bounds_for_d_given_n_and_k_and_q`	$n\ k\ q$	Make bounds for the minimum distance of an $[n, k]_q$ code
`-introduce_errors`	CRC-options	Introduce errors to a file. See Table 10.8.
`-check_errors`	CRC-options	Find errors in a CRC coded file. See Table 10.8.
`-extract_block`	CRC-options	Extract a block from a CRC coded file. See Table 10.8.
`-random_noise_in_bitmap_file`	input output $n\ d$	Apply random noise at the d/n level to the bitmap file "input" and write to "output."
`-random_noise_of_burst_type_in_bitmap_file`	input output $n\ d\ b$	Apply random noise of type burst with max burst length b at the d/n level to the bitmap file "input" and write to "output"

Table 10.1: Global Coding Theoretic Commands





The original is on the top. The effect of noise can be seen in the picture below. The picture shows Paul Allen and Bill Gates in the early 1970s.

The command

Example 394

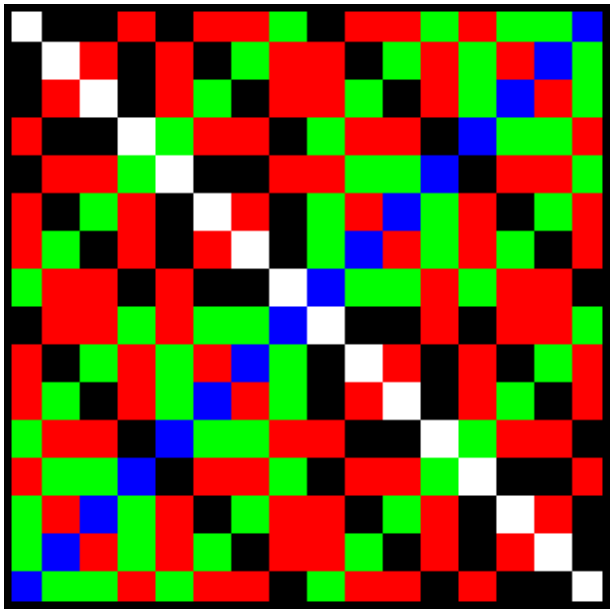
```
Hamming_space_4_2_distance_matrix:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -with F -do -coding_theoretic_activity \
▷  ▷  ▷  -Hamming_space_distance_matrix 4 \
▷  ▷  -end
```

creates the distance matrix of the Hamming graph $H(n, q)$. The data is written to the file `Hamming_n4_q2.csv`. The command

Example 395

```
Hamming_space_4_2_distance_matrix_draw:
▷  $(ORBITER) -v 2 \
▷  ▷  -draw_matrix \
▷  ▷  ▷  -input_csv_file Hamming_n4_q2.csv \
▷  ▷  ▷  -box_width 20 -bit_depth 24 \
▷  ▷  ▷  -partition 4 16 16 \
▷  ▷  -end
▷  $(OPEN) Hamming_n4_q2_draw.bmp
```

produces the bitmap graphic `Hamming_n4_q2_draw.bmp` shown below:



The command

Example 396

```
Hamming_code_macwilliams:
▷  $(ORBITER) -v 2 \
▷  ▷  -make_macwilliams_system 7 4 2
▷  pdflatex MacWilliams_n7_k4_q2.tex
▷  $(OPEN) MacWilliams_n7_k4_q2.pdf
```

creates the coefficient matrix of the MacWilliams system for the $[7, 4, 2]$ Hamming code:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 7 & 5 & 3 & 1 & -1 & -3 & -5 & -7 \\ 21 & 9 & 1 & -3 & -3 & 1 & 9 & 21 \\ 35 & 5 & -5 & -3 & 3 & 5 & -5 & -35 \\ 35 & -5 & -5 & 3 & 3 & -5 & -5 & 35 \\ 21 & -9 & 1 & 3 & -3 & -1 & 9 & -21 \\ 7 & -5 & 3 & -1 & -1 & 3 & -5 & 7 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix}$$

For examples concerning the bounds, see Section 10.9.

Consider the $[5, 2]_2$ code whose codewords are $\{0, 7, 25, 30\}$. We start with a makefile variable:

Example 397

```
CODE_5_2_3_CODEWORDS="0,7,25,30"
```

The following command creates the code and draws the code diagram in Hamming space:

Example 398

```
code_5_2_3_diagram:
▷ $(ORBITER) -v 2 \
▷ ▷ -define F -finite_field -q 2 -end \
▷ ▷ -with F -do -coding_theoretic_activity \
▷ ▷ ▷ -general_code_binary 5 "code_5_2_3" \
▷ ▷ ▷ $(CODE_5_2_3_CODEWORDS) \
▷ ▷ ▷ -metric_balls 1 \
▷ ▷ -end
▷ $(ORBITER) -v 2 \
▷ ▷ -draw_matrix \
▷ ▷ ▷ -input_csv_file code_5_2_3_char_func_5_4.csv \
▷ ▷ ▷ -box_width 25 -bit_depth 24 \
▷ ▷ ▷ -partition 4 8 4 \
▷ ▷ -end
```

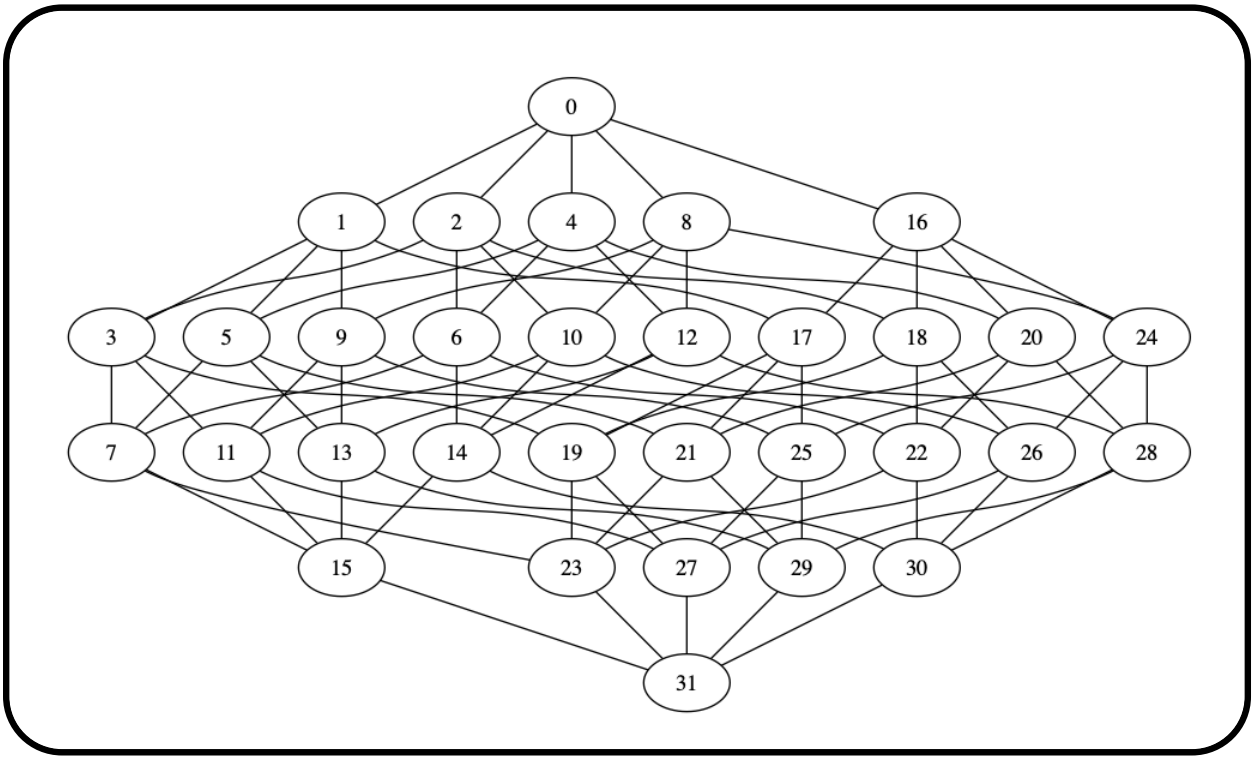
The Hamming graph $H(5, 2)$ can be created with the following command:

Example 399

```
Hamming_5_2_graph:
▷ $(ORBITER) -v 2 \
▷ ▷ -define G -graph -Hamming 5 2 -end \
▷ ▷ -with G -do \
▷ ▷ -graph_theoretic_activity -export_csv -end \
▷ ▷ -with G -do \
▷ ▷ -graph_theoretic_activity -export_graphviz -end \
```

```
▷  ▷  -with G -do \
▷  ▷  -graph_theoretic_activity -save -end
▷  $(ORBITER) -v 2 -draw_matrix \
▷  ▷  -input_csv_file Hamming_5_2.csv \
▷  ▷  -box_width 8 -bit_depth 24 -partition 4 32 32 -end
▷  dot -Tpng Hamming_5_2.gv >Hamming_5_2.png
```

Using the unix dot program, this command sequence creates the drawing of $H(5,2)$



The command

Example 400

```
Sylvester_Hadamard_code_8:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 3 -end \
▷  ▷  -with F -do -coding_theoretic_activity \
▷  ▷  ▷  -Sylvester_Hadamard_code 8 \
▷  ▷  -end
```

creates the code arising from a Sylvester-type Hadamard matrix of order 8.

10.2 Linear Codes

Table 10.2 lists Orbiter commands to create codes. Tables 10.3-10.5 list coding theoretic activities in Orbiter. Table 10.6 shows ways to create new codes from old.

The following command creates the first order Reed-Muller code in three variables:

Example 401

```
RM_3_1:
▷ $(ORBITER) -v 2 \
▷ ▷ -define F -finite_field -q 2 -end \
▷ ▷ -define C -code -field F \
▷ ▷ ▷ -Reed_Muller 3 \
▷ ▷ -end \
▷ ▷ -with C -and F -do -coding_theoretic_activity \
▷ ▷ ▷ -export_magma RM_3_1.magma \
▷ ▷ -end \
▷ ▷ -with C -do -coding_theoretic_activity \
▷ ▷ ▷ -report \
▷ ▷ -end
▷ pdflatex RM_3_code_n8_k4_q2.tex
▷ $(OPEN) RM_3_code_n8_k4_q2.pdf
```

Consider the linear code with generator matrix

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

The row vectors correspond to the integers 60, 50, 41 in binary. We can use the following command to create the code:

Example 402

```
code_6:
▷ $(ORBITER) -v 2 \
▷ ▷ -define F -finite_field -q 2 -end \
▷ ▷ -define C -code -field F \
▷ ▷ ▷ -basis 6 "60,50,41" \
▷ ▷ -end \
▷ ▷ -with C -and F -do -coding_theoretic_activity \
▷ ▷ ▷ -make_diagram \
▷ ▷ -end \
▷ ▷ -with C -do -coding_theoretic_activity \
▷ ▷ ▷ -report \
▷ ▷ -end
▷ $(ORBITER) -v 2 -draw_matrix \
▷ ▷ -input_csv_file by_basis_n6_k3_char_func_6_8.csv \
▷ ▷ -box_width 20 -bit_depth 24 \
▷ ▷ -partition 2 "1,1,1,1,1,1,1,1" "1,1,1,1,1,1,1,1" \
▷ ▷ -end
```

Creating Codes

Command	Arguments	Purpose
-field	F	The field $F = \mathbb{F}_q$ over which the code is defined.
-generator_matrix	M	A generator matrix over F defining the code as Orbiter object.
-basis	n basis	A basis for the code as Orbiter ranks. The rows of the generator matrix are coded as integers using the binary representation.
-long_code	n genma	Create a code over $\mathbb{F}_q$ defined by a generator matrix. The rows of the generator matrix are given as a list of strings in "words". Each string describes one row, by listing the columns of the generator matrix which have a one in the corresponding row.
-projective_set	nmk	A projective set defining the columns of a check matrix of a code. The projective set is in $\mathrm{PG}(n-k-1,q)$ where nmk = n - k is given.
-columns_of_generator_matrix	k cols	The columns of a check matrix defining the code are given, using the ranking of elements of $\mathrm{AG}(k,q)$.
-Reed_Muller	m	The first order Reed Muller code with m variables.
-BCH	n d	BCH-code over $\mathbb{F}_q$ of length n with designed distance d. The minimum distance will be at least the designed distance.
-Reed_Solomon	n d	A Reed-Solomon code of length n with designed distance d. This requires that n divides $q-1$.
-Gilbert_Varshamov	n k d	A Gilbert-Varshamov code with length n, dimension k and minimum distance d.
-ttpA	FQ	Twisted tensor product code of type A using the extension field FQ. The code arises from a hyperoval in $\mathbb{F}_{q^2}$ (Theorem 1 in [8])
-ttpB	FQ	Twisted tensor product code of type B using the extension field FQ. The code arises from the projective line over $\mathbb{F}_{q^3}$ (Theorem 2 in [8]).
-dual		Create the dual code instead.

Table 10.2: Creating Codes

Coding Theoretic Activities (Part 1)

Command	Arguments	Purpose
-report		Produce a latex report.
-general_code_binary	n label code-words	Produces a plot of the codewords in the Hamming space. The code is given by a set of codewords in binary. The code has length n. Produces several csv files. If -metric_balls r is given, the metric balls of radius r centered around codewords are included in the plot.
-encode_text_5bits	input fname	
-field_induction	fname-in fname-out nb-bits	
-weight_enumerator	matrix	Compute the complete weight enumerator of the code C.
-minimum_distance	code-object-label	Compute the minimum distance of the linear code object.
-generator_matrix_cyclic_code	n poly	
-Sylvester_Hadamard_code	n	Create the codewords of a Sylvester type Hadamard matrix of order n. Here, n must be a power of 2.
-fixed_code	perm	Create the subcode which is fix under the group generated by perm, which acts on the generator matrix by column permutations.
-export_magma	fname	
-export_codewords	fname	Export a list of codewords using coding as binary numbers.
-export_codewords_long	fname	Export a table of codewords.
-export_codewords_by_weight	fname	
-export_genma	fname	
-export_checkma	fname	
-f_export_checkma_as_projective_set	fname	

Table 10.3: Coding Theoretic Activities (Part 1)

Coding Theoretic Activities (Part 2)

Command	Arguments	Purpose
-make_diagram		Creates a diagram of the code in Hamming space. Uses -embellish r to emphasize codewords by drawing a circle of radius r. If -metric_balls r is given, the matric balls around codewords of radius r are indicated.
-boolean_function_of_code		Computes the boolean function of the code C.
-embellish	r	See -make_diagram.
-metric_balls	r	See -make_diagram.
-Hamming_space_distance_matrix	n	Make the distance matrix of the Hamming graph $H(n, q)$.

Table 10.4: Coding Theoretic Activities (Part 2)

```
▷ pdflatex by_basis_n6_k3_code_n6_k3_q2.tex
▷ $(OPEN) by_basis_n6_k3_code_n6_k3_q2.pdf
▷ $(OPEN) by_basis_n6_k3_char_func_6_8_draw.bmp
```

The command also draws the code in the Hamming space.

Let us create the Hamming code. The dual of the Hamming code is the simplex code, so we create the simplex code first. The following makefile variable is defined to hold the generator matrix of the simplex code:

Example 403

```
SIMPLEX_CODE_GENERATOR="\
1,0,1,0,1,0,1, \
0,1,1,0,0,1,1, \
0,0,0,1,1,1,1"
```

The following command computes the nullspace of this matrix, which is the Hamming code:

Example 404

```
simplex_code:
▷ $(ORBITER) -v 2 \
▷ ▷ -define F -finite_field -q 2 -end \
▷ ▷ -define v -vector -field F -format 3 \
▷ ▷ ▷ -dense $(SIMPLEX_CODE_GENERATOR) \
▷ ▷ -end \
▷ ▷ -define C -code -field F \
▷ ▷ ▷ -generator_matrix v \
▷ ▷ -end \
▷ ▷ -with C -do -coding_theoretic_activity \
▷ ▷ ▷ -report \
```

Coding Theoretic Activities (Part 3)		
Command	Arguments	Purpose
-crc32	text	
-crc32_hexdata	hexdata	
-crc32_test	block-length	
-crc256_test	message-length R k	
-crc32_remainders	msg-length	
-crc_encode_file_based	fname-in fname-out block-length crc_type block_length	
-crc_compare	fname code1 code2 error-weight nb_tests-per-block	
-crc_compare_read_output_file	fname nb-lines code1 code2	
-all_errors_of_a_given_weight	fname block-idx code1 code2 max-weight	
-weight_enumerator_bottom_up	code max-weight	
-convert_data_to_polynomials	fname-in fname-out block-length symbol-size	
-find_CRC_polynomials	nb-errors info-bits check-bits	
-write_code_for_division	fname A B	
-polynomial_division_from_file	fname $r1$	
-polynomial_division_from_ file_all_k_bit_error_patterns	fname $r1$ k	

Table 10.5: Coding Theoretic Activities (Part 3)

Code Modifications		
Command	Arguments	Purpose
-dual		Create the dual code.

Table 10.6: Code Modifications

```
▷  ▷  -end
▷  pdflatex by_genma_n7_k3_code_n7_k3_q2.tex
▷  $(OPEN) by_genma_n7_k3_code_n7_k3_q2.pdf
```

The following latex output is produced:

Input matrix:

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

RREF:

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Basis for Perp:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

It is possible to create the Hamming code by taking the dual of the simplex code. The following command does so:

Example 405

```
Hamming_code:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define v -vector -field F -format 3 \
▷  ▷  ▷  -dense $(SIMPLEX_CODE_GENERATOR) \
▷  ▷  -end \
▷  ▷  -define C -code -field F \
▷  ▷  ▷  -generator_matrix v \
▷  ▷  ▷  -dual -end \
▷  ▷  -end \
▷  ▷  -with C -do -coding_theoretic_activity \
▷  ▷  ▷  -export_magma Hamming.magma \
▷  ▷  -end
```

The command also exports the code to magma by means of the magma file `Hamming.magma`, shown below:

```
K<w> := GF(2);
V := VectorSpace(K, 7);
C := LinearCode(sub<V |
```

```
[1,1,1,0,0,0,0],
[1,0,0,1,1,0,0],
[0,1,0,1,0,1,0],
[1,1,0,1,0,0,1]>);
```

The next command creates the first order Reed-Muller code in 3 variables. All codewords are created. The codewords and the generator matrix are exported to files.

Example 406

```
RM_3_1_and_codewords:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define C -code -field F \
▷  ▷  ▷  -Reed_Muller 3 \
▷  ▷  -end \
▷  ▷  -with C -and F -do -coding_theoretic_activity \
▷  ▷  ▷  -export_magma RM_3_1.magma \
▷  ▷  -end \
▷  ▷  -with C -and F -do -coding_theoretic_activity \
▷  ▷  ▷  -export_codewords RM_3_1_codewords.csv \
▷  ▷  -end \
▷  ▷  -with C -and F -do -coding_theoretic_activity \
▷  ▷  ▷  -export_genma RM_3_1_genma.csv \
▷  ▷  -end
```

We can create the code from scratch, using a generator matrix. To do so, we use a makefile variable:

Example 407

```
CODE_RM_3_1_GENMA="\
11111111\
01010101\
00110011\
00001111"
```

The following command creates the Reed-Muller code from its generator matrix directly:

Example 408

```
RM_3_1_from_generator_matrix:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define genma -vector -format 8 -field F \
▷  ▷  ▷  -compact $(CODE_RM_3_1_GENMA) \
▷  ▷  -end \
▷  ▷  -define C -code -field F \
▷  ▷  ▷  -generator_matrix genma \
▷  ▷  -end
▷  #pdflatex code_n8_k4_q2.tex
```

```
▷  #$(OPEN) code_n8_k4_q2.pdf
```

Let us look at the Hamming code. The following command creates the $[7, 4]$ Hamming code.

Example 409

```
Hamming_code_by_rows:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define v -vector -dense $(HAMMING_CODE_ROWS_IN_BINARY_RANKS) -end \
▷  ▷  -define C -code -field F \
▷  ▷  ▷  -basis 7 v \
▷  ▷  -end
▷  #pdflatex code_n7_k4_q2.tex
▷  #$(OPEN) code_n7_k4_q2.pdf
```

A different way to create the same code is using the `long_code` command:

Example 410

```
Hamming_code_long:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define C -code -field F \
▷  ▷  ▷  -long_code 7 4 \
▷  ▷  ▷  "0,5,6" \
▷  ▷  ▷  "1,4,6" \
▷  ▷  ▷  "2,4,5" \
▷  ▷  ▷  "3,4,5,6" \
▷  ▷  -end
```

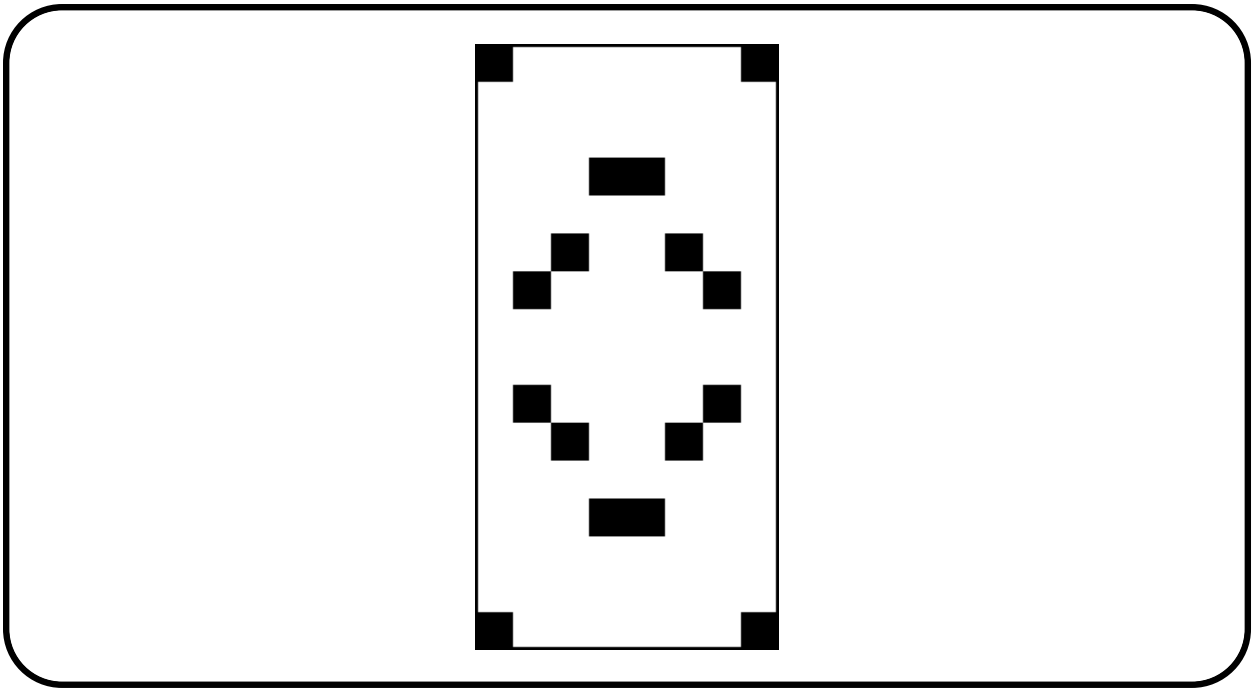
The following command creates a diagram for the Hamming code inside the Hamming space.

Example 411

```
Hamming_code_diagram:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define v -vector -dense $(HAMMING_CODE_ROWS_IN_BINARY_RANKS) -end \
▷  ▷  -define C -code -field F \
▷  ▷  ▷  -basis 7 v \
▷  ▷  -end \
▷  ▷  -with C -and F -do -coding_theoretic_activity \
▷  ▷  ▷  -make_diagram \
▷  ▷  -end
▷  $(ORBITER) -v 2 \
▷  ▷  -draw_matrix \
▷  ▷  ▷  -input_csv_file by_basis_n7_k4_char_func_7_16.csv \
```

```
▷ ▷ ▷ -box_width 25 -bit_depth 24 \
▷ ▷ ▷ -partition 4 16 8 \
▷ ▷ -end
▷ $(OPEN) by_basis_n7_k4_char_func_7_16_draw.bmp
```

Here is the output:



The next command creates the metric balls of radius one centered at each of the 16 codewords of the Hamming code. At first, we define the numerical values of the codewords of the Hamming code:

Example 412

```
HAMMING_CODE_CODEWORDS="0, 67, 37, 102, 22, 85, \
51, 112, 15, 76, 42, 105, 25, 90, 60, 127"
```

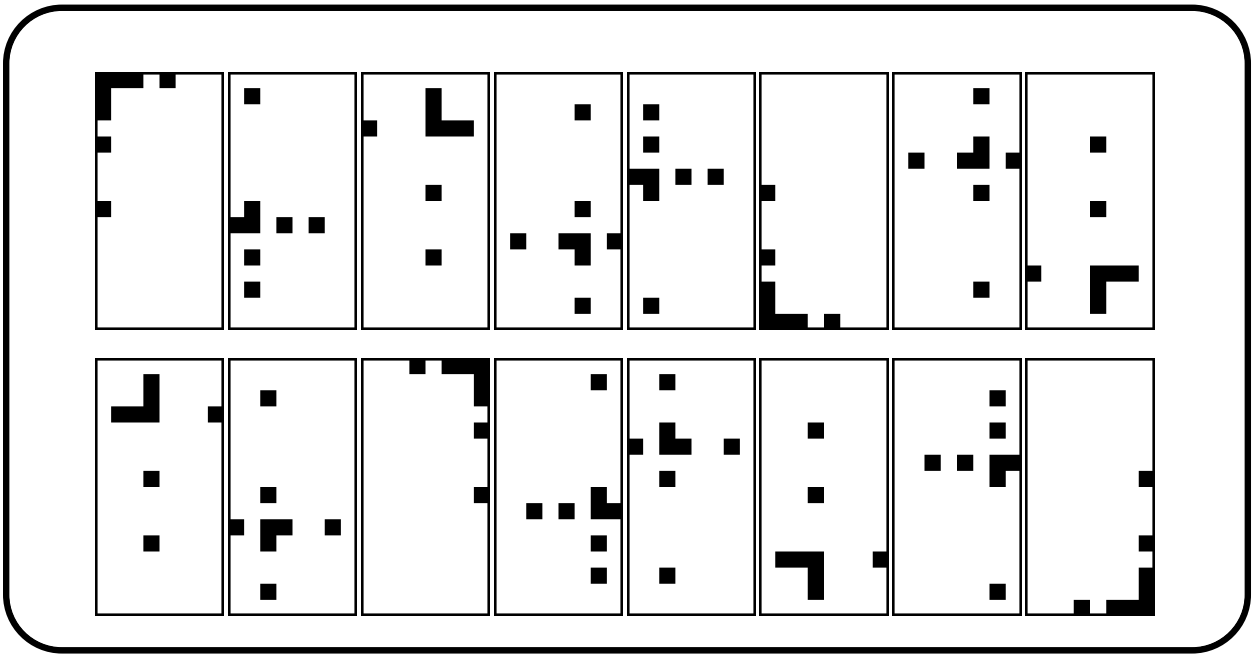
Then we create the diagrams of all codewords. We utilize a loop-over command for the codewords and a loop command for the graphical output.

Example 413

```
Hamming_code_words:
▷ $(ORBITER) -v 2 \
▷ ▷ -define v -vector -dense $(HAMMING_CODE_CODEWORDS) -end \
▷ ▷ -define F -finite_field -q 2 -end \
▷ ▷ -loop_over i v \
▷ ▷ ▷ -with F -do -coding_theoretic_activity \
▷ ▷ ▷ -general_code_binary 7 "Hamming_7_4_word_%i" "%i[v]" \
▷ ▷ ▷ -metric_balls 1 \
```

```
▷ ▷ ▷ -end \
▷ ▷ -end_loop_over i
▷ $(ORBITER) -v 2 \
▷ ▷ -loop i 0 16 1 -draw_matrix \
▷ ▷ ▷ -input_csv_file Hamming_7_4_word_%i_char_func_7_1.csv \
▷ ▷ ▷ -box_width 25 -bit_depth 8 \
▷ ▷ ▷ -partition 4 16 8 -end \
▷ ▷ ▷ -system "convert Hamming_7_4_word_%i_char_func_7_1_draw.bmp \
▷ ▷ ▷ -frame 8 Hamming_7_4_word_%i_char_func_7_1_draw.png" \
▷ ▷ -end_loop i
```

These metric balls form a partition of the Hamming space. The partition can be shown in binary Hamming space:



Orbiter can compute the weight enumerator and the minimum distance of codes. Let us consider the Hamming code, for example. We use a makefile variable for the generator matrix:

Example 414

```
HAMMING_CODE_GENERATOR="\
1,0,0,0,0,1,1, \
0,1,0,0,1,0,1, \
0,0,1,0,1,1,0, \
0,0,0,1,1,1,1"
```

The next command computes the weight enumerator:

Example 415

```
Hamming_weight_enumerator:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define v -vector -field F -format 4 \
▷  ▷  ▷  -dense $(HAMMING_CODE_GENERATOR) \
▷  ▷  -end \
▷  ▷  -define C -code -field F \
▷  ▷  ▷  -generator_matrix v \
▷  ▷  -end \
▷  ▷  -with C -do \
▷  ▷  -coding_theoretic_activity \
▷  ▷  ▷  -weight_enumerator \
▷  ▷  -end
```

We find that the weight enumerator is

$$(1, 0, 0, 7, 7, 0, 0, 1).$$

The next command computes the minimum distance of the code:

Example 416

```
Hamming_minimum_distance:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define v -vector -field F -format 4 \
▷  ▷  ▷  -dense $(HAMMING_CODE_GENERATOR) \
▷  ▷  -end \
▷  ▷  -with F -do \
▷  ▷  -coding_theoretic_activity \
▷  ▷  ▷  -minimum_distance v \
▷  ▷  -end
```

The following command computes the minimum distance of the Golay code of length 23:

Example 417

```
Golay23_minimum_distance:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define v -vector -field F -format 12 \
▷  ▷  ▷  -dense $(GOLAY23_CODE_GENERATOR) \
▷  ▷  -end \
▷  ▷  -with F -do \
▷  ▷  -coding_theoretic_activity \
▷  ▷  ▷  -minimum_distance v \
▷  ▷  -end
```

10.3 Golay Codes

The Golay code of length 23 is a perfect code of dimension 12 and minimum distance 7. The metric balls of radius three centered around codewords cover the whole Hamming space. We can create the code by listing the columns of a generator matrix in Orbiter ranks of points in PG(11, 2). The following makefile variable does that:

Example 418

```
GOLAY_23_COLUMN_RANKS_PROJECTIVELY="0, 1, 2, 3, 4, 5, 6, 7, \
8, 9, 10, 11, 132, 913, 1460, 1750, 1898, 2518, 2787, 2874, \
3320, 3357, 3662"
```

Suppose we want to list the code words. The following command can be used:

Example 419

```
Golay23_code_words:
▷  $(ORBITER) -v 2 \
▷  ▷  -define v -vector -dense $(GOLAY_23_COLUMN_RANKS_PROJECTIVELY) -end \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define C -code -field F \
▷  ▷  ▷  -projective_set 12 v -end \
▷  ▷  -with C -and F -do -coding_theoretic_activity \
▷  ▷  ▷  -export_magma Golay23.magma \
▷  ▷  -end \
▷  ▷  -with C -and F -do -coding_theoretic_activity \
▷  ▷  ▷  -export_codewords Golay23_codewords.csv \
▷  ▷  -end \
▷  ▷  -with C -and F -do -coding_theoretic_activity \
▷  ▷  ▷  -export_genma Golay23_genma.csv \
▷  ▷  -end
▷  #pdflatex code_n23_k12_q2.tex
▷  #$(OPEN) code_n23_k12_q2.pdf
```

Defining a CRC-process		
Command	Arguments	Purpose
-code	label	Set the a label.
-crc_options	options	Specify the CRC-code using commands from Table 10.8.

Table 10.7: Defining a CRC-process

Options for CRC-Codes		
Command	Arguments	Purpose
-input	fname	Input file name.
-output	fname	Output file name.
-crc_type	type	The type of CRC-code.
-block_length	L	Set the block length to L field elements.
-block_based_error_generator		Apply block-based error generator.
-file_based_error_generator	threshold	Apply file-based error generator.
-nb_repeats	N	Set the number of repeats to N.
-threshold	t	Set probability of error per experiment to $t/1000000$.
-error_log	fname	Set file name for error logging.
-selected_block	i	Set block number.

Table 10.8: Options for CRC-Codes

10.4 CRC Codes

A CRC code can be used to detect communication errors. It is a cyclic code, and hence generated by a polynomial over a finite field. The message is encoded as a string, which is then thought of as a polynomial, called the information polynomial. Assume that the check polynomial has degree d. The information polynomial is then divided by the check polynomial. The remainder is added to the information polynomial multiplied by X^d. This is the codeword, which is sent.

Table 10.7 lists the commands for defining a CRC-process.

Table 10.8 summarizes options associated with commands for CRC-codes.

Table 10.9 lists the available commands for defining a CRC-code.

The following command performs an exhaustive search over all binary CRC polynomials of degree $k = 10$ which can detect every error pattern of Hamming weight at most $t = 3$ in messages of length $n = 128$.

Defining a CRC-Code		
Command	Arguments	Purpose
-type	type	Select the type of CRC-code.
-block_length	length	Select the symbol length of the CRC-code.

Table 10.9: Defining a CRC-Code

Example 420

```
CRC_3_128_10:
▷  $(ORBITER) -v 1 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -with F -do -coding_theoretic_activity \
▷  ▷  ▷  -find_CRC_polynomials 3 128 10 \
▷  ▷  -end
```

The program finds 244 polynomials in about 1 minute.

Here is a collection of CRC polynomials from various sources:

Example 421

```
CRC4="1,4,1,2,1,1,1,0"
```

Example 422

```
CRC7="1,7,1,3,1,0"
```

Example 423

```
CRC8_ATM="1,8,1,2,1,1,1,0"
```

Example 424

```
CRC16_CCITT="1,16,1,12,1,5,1,0"
```

Example 425

```
CRC32_ETHERNET="1,32,1,26,1,23,1,22,1,16,1,12,1,11,1,10,1,8,1,7,\
1,5,1,4,1,2,1,1,1,0"
```

Example 426

```
CRC32_CASTAGNOLI="1,32,1,28,1,27,1,26,1,25,1,23,1,22,1,20,1,19,1,\
18,1,14,1,13,1,11,1,10,1,9,1,8,1,6,1,0"
```

Example 427

```
CRC64_ECMA182="1,64,1,62,1,57,1,55,1,54,1,53,1,52,1,47,1,46,1,45,\
1,40,1,39,1,38,1,37,1,35,1,33,1,32,1,31,1,29,1,27,1,24,1,23,1,22,\
```

```
1,21,1,19,1,17,1,13,1,12,1,10,1,9,1,7,1,4,1,1,1,0"
```

Example 428

```
CRC64_ROCKSOFT="1,64,1,63,1,61,1,59,1,58,1,56,1,55,1,52,1,49,1,48,\
1,47,1,46,1,44,1,41,1,37,1,36,1,34,1,32,1,31,1,28,1,26,1,23,1,22,1,\
19,1,16,1,13,1,12,1,10,1,9,1,6,1,4,1,3,1,0"
```

We test whether the polynomial crc32 is irreducible:

Example 429

```
crc32_Berlekamp_matrix:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define v -vector -field F -sparse 33 $(CRC32_ETHERNET) -end \
▷  ▷  -with F -do \
▷  ▷  -finite_field_activity \
▷  ▷  ▷  -Berlekamp_matrix v \
▷  ▷  -end
```

Now, we create some new CRC polynomials over the field $\mathbb{F}_{256}$. To begin with, we create the 771st roots over $\mathbb{F}_{256}$:

Example 430

```
CRC_F256_roots_771:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 256 -end \
▷  ▷  -with F -do -finite_field_activity \
▷  ▷  ▷  -nth_roots 771 \
▷  ▷  -end
```

We create a BCH code of length 771 over $\mathbb{F}_{256}$ with designed distance 2:

Example 431

```
CRC_F256_BCH_code_d2:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 256 -end \
▷  ▷  -define C -code -field F \
▷  ▷  ▷  -BCH 771 2 \
▷  ▷  -end \
▷  ▷  -with C -and F -do -coding_theoretic_activity \
▷  ▷  ▷  -export_magma BCH_lq8_n771_d2.magma \
▷  ▷  -end
```

```
▷  #pdflatex BCH_codes_q256_n771_d2.tex
▷  #$(OPEN) BCH_codes_q256_n771_d2.pdf
```

The polynomial in dense coding

Example 432

```
CRC_POLY_Q256_DEG2_DENSE="214,167,1"
```

We generate C++ source code for the use of this polynomial:

Example 433

```
CRC_F256_BCH_write_code_for_division_d2:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 256 -end \
▷  ▷  -define A -vector -field F -sparse 772 "1,771,1,0" -end \
▷  ▷  -define B -vector -field F -dense $(CRC_POLY_Q256_DEG2_DENSE) -end \
▷  ▷  -with F -do \
▷  ▷  -coding_theoretic_activity \
▷  ▷  ▷  -write_code_for_division \
▷  ▷  ▷  alfa A B \
▷  ▷  -end
▷  g++ crc_alfa.cpp -o crc_alfa.out
▷  ./crc_alfa.out
```

We create a BCH code of length 771 over $\mathbb{F}_{256}$ with designed distance 16:

Example 434

```
F256_BCH_code_d16:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 256 -end \
▷  ▷  -define C -code -field F \
▷  ▷  ▷  -BCH 771 16 \
▷  ▷  -end
▷  pdflatex BCH_codes_q256_n771_d16.tex
▷  $(OPEN) BCH_codes_q256_n771_d16.pdf
```

The polynomial in sparse coding is:

Example 435

```
POLY_Q256_DEG30_SPARSE="1,0,26,1,210,2,24,3,\
138,4,148,5,160,6,58,7,108,8,199,9,95,10,56,\
11,9,12,205,13,194,14,193,15,3,16,248,17,110,\
18,150,19,24,20,169,21,192,22,212,23,112,24,\
144,25,97,26,109,27,174,28,253,29,1,30"
```

The polynomial in dense coding is:

Example 436

```
POLY_Q256_DEG30_DENSE="1,26,210,24,138,148,\
160,58,108,199,95,56,9,205,194,193,3,248,110,\
150,24,169,192,212,112,144,97,109,174,253,1"
```

We generate C++ source code for the use of this polynomial:

Example 437

```
F256_BCH_write_code_for_division_d16:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 256 -end \
▷  ▷  -define A -vector -field F -sparse 772 "1,771,1,0" -end \
▷  ▷  -define B -vector -field F -dense $(POLY_Q256_DEG30_DENSE) -end \
▷  ▷  -with F -do \
▷  ▷  -coding_theoretic_activity \
▷  ▷  ▷  -write_code_for_division \
▷  ▷  ▷  check_q256_n771_r30 A B \
▷  ▷  -end
▷  g++ crc_check_q256_n771_r30.cpp -o crc_check_q256_n771_r30.out
▷  ./crc_check_q256_n771_r30.out
```

We confirm that the polynomial divides $X^{771} - 1$ as it should:

Example 438

```
F256_BCH_code_d16_division:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 256 -end \
▷  ▷  -define A -vector -field F -sparse 772 "1,771,1,0" -end \
▷  ▷  -define B -vector -field F -dense $(POLY_Q256_DEG30_DENSE) -end \
▷  ▷  -with F -do \
▷  ▷  -finite_field_activity \
▷  ▷  -polynomial_division A B -end
```

The next example introduces three errors. The remainder is not zero, so the errors are detected:

Example 439

```
F256_BCH_code_d16_error:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 256 -end \
▷  ▷  -define A -vector -field F -sparse 771 "2,30,3,31,55,770" -end \
▷  ▷  -define B -vector -field F -dense $(POLY_Q256_DEG30_DENSE) -end \
▷  ▷  -with F -do \
▷  ▷  -finite_field_activity \
▷  ▷  -polynomial_division A B -end
```

10.5 Reed-Muller Codes

The following command creates the Reed Muller code RM_3. The command exports the code to magma and writes a file of all codewords.

Example 440

```
RM_3_export:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define C -code -field F \
▷  ▷  ▷  -Reed_Muller 3 \
▷  ▷  -end \
▷  ▷  -with C -and F -do -coding_theoretic_activity \
▷  ▷  ▷  -export_magma RM_3.magma \
▷  ▷  -end \
▷  ▷  -with C -and F -do -coding_theoretic_activity \
▷  ▷  ▷  -export_codewords RM_3_codewords.csv \
▷  ▷  -end
```

The next command creates a diagram of the code in Hamming space, based on the list of codewords:

Example 441

```
RM_3_Hamming_space_diagram:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define C -code -field F \
▷  ▷  ▷  -Reed_Muller 3 \
▷  ▷  -end \
▷  ▷  -with C -do \
▷  ▷  -coding_theoretic_activity \
▷  ▷  ▷  -make_diagram \
▷  ▷  ▷  -metric_balls 1 \
▷  ▷  -end
```

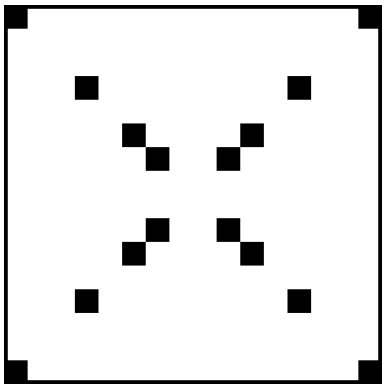
The following command produces a diagram of the characteristic function of the code in the Hamming space $H(8, 2)$.

Example 442

```
RM_3_draw:
▷  $(ORBITER) -v 2 \
▷  ▷  -draw_matrix \
▷  ▷  ▷  -input_csv_file RM_3_distance_H_8_16.csv \
▷  ▷  ▷  -box_width 25 -bit_depth 8 \
▷  ▷  ▷  -partition 4 16 16 \
▷  ▷  -end
▷  $(ORBITER) -v 2 \
▷  ▷  -draw_matrix \
▷  ▷  ▷  -input_csv_file RM_3_char_func_8_16.csv \
```

```
▷ ▷ ▷ -box_width 25 -bit_depth 8 \
▷ ▷ ▷ -partition 4 16 16 \
▷ ▷ -end
▷ $(ORBITER) -v 2 \
▷ ▷ -draw_matrix \
▷ ▷ ▷ -input_csv_file RM_3_idx_8_16.csv \
▷ ▷ ▷ -box_width 25 -bit_depth 8 \
▷ ▷ ▷ -partition 4 16 16 \
▷ ▷ -end
▷ convert RM_3_distance_H_8_16_draw.bmp RM_3_distance_H_8_16_draw.png
▷ convert RM_3_char_func_8_16_draw.bmp RM_3_char_func_8_16_draw.png
▷ convert RM_3_idx_8_16_draw.bmp RM_3_idx_8_16_draw.png
▷ $(OPEN) RM_3_distance_H_8_16_draw.bmp
```

The following boolean function representation of RM_3 in $H(8,2)$ is produced:



The next command splits the vectors according to their distance from the code:

Example 443

```
RM_3_split:
▷ #$(ORBITER) -split_by_values RM_3_distance_8_16.csv
▷ $(ORBITER) -split_by_values RM_3_distance_H_8_16.csv
```

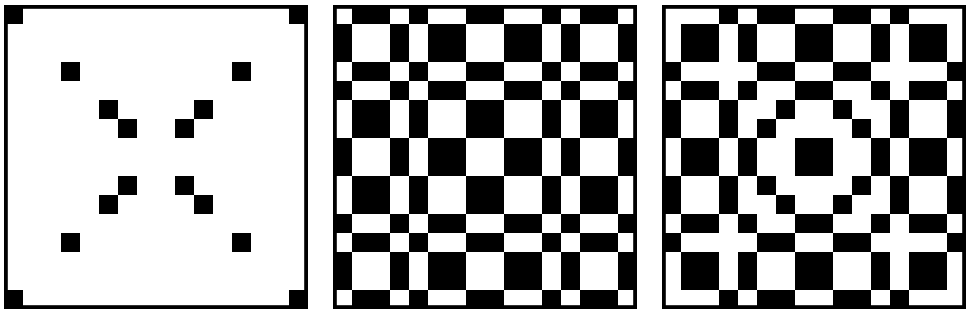
We produce a diagram of each distance set. We loop over all distances $\delta = 0, 1, 2$:

Example 444

```
RM_3_distance_draw:
▷ $(ORBITER) -v 2 \
▷ ▷ -loop L 0 3 1 \
▷ ▷ ▷ -draw_matrix \
▷ ▷ ▷ ▷ -input_csv_file RM_3_distance_H_8_16_value%L.csv \
▷ ▷ ▷ ▷ -box_width 25 -bit_depth 8 \
▷ ▷ ▷ ▷ -partition 5 16 16 \
▷ ▷ ▷ -end \
```

```
▷  ▷  -end_loop L
▷  convert RM_3_distance_H_8_16_value0_draw.bmp RM_3_distance_H_8_16_value0_draw.p
ng
▷  convert RM_3_distance_H_8_16_value1_draw.bmp RM_3_distance_H_8_16_value1_draw.p
ng
▷  convert RM_3_distance_H_8_16_value2_draw.bmp RM_3_distance_H_8_16_value2_draw.p
ng
```

The distance sets are shown below



$\delta = 0$ $\delta = 1$ $\delta = 2$

For each distance, we compute the algebraic normal form of the characteristic function of the distance set:

Example 445

```
RM_3_algebraic_normal_form:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -loop L 0 3 1 \
▷  ▷  ▷  -define v%L -vector -field F \
▷  ▷  ▷  ▷  -file RM_3_distance_8_16_value%L.csv \
▷  ▷  ▷  -end \
▷  ▷  ▷  -with F -do -finite_field_activity \
▷  ▷  ▷  ▷  -algebraic_normal_form 8 v%L \
▷  ▷  ▷  -end \
▷  ▷  -end_loop L
```

In the example, we will compute a graphical representation of the codewords of the Reed-Muller code of order 6. First, we use makefile variables to define the rows of the generator matrix:

Example 446

```
RM_6_GENERATOR_1="0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,\
22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,\
46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63"
```

Example 447

```
RM_6_GENERATOR_2="1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,\
41,43,45,47,49,51,53,55,57,59,61,63"
```

Example 448

```
RM_6_GENERATOR_3="2,3,6,7,18,19,22,23,10,11,14,15,26,27,30,31,34,35,38,\
39,42,43,46,47,50,51,54,55,58,59,62,63"
```

Example 449

```
RM_6_GENERATOR_4="4,6,12,14,36,38,52,54,5,7,13,15,37,39,53,55,20,22,28,\
30,44,46,60,62,21,23,29,31,45,47,61,63"
```

Example 450

```
RM_6_GENERATOR_5="8,9,12,13,24,25,28,29,10,11,14,15,26,27,30,31,40,41,\
44,45,56,57,60,61,42,43,46,47,58,59,62,63"
```

Example 451

```
RM_6_GENERATOR_6="16,18,24,26,48,50,56,58,17,19,25,27,49,51,57,59,20,22,\
28,30,52,54,60,62,21,23,29,31,53,55,61,63"
```

Example 452

```
RM_6_GENERATOR_7="32,34,48,50,33,35,49,51,36,38,52,54,37,39,53,55,40,42,\
56,58,41,43,57,59,44,46,60,62,45,47,61,63"
```

The following command creates the generator matrix, and exports the generator matrix and the set of codewords to two csv files.

Example 453

```
RM_6:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define C -code -field F \
▷  ▷  ▷  -long_code 64 7 \
▷  ▷  ▷  $(RM_6_GENERATOR_1) \
▷  ▷  ▷  $(RM_6_GENERATOR_2) \
▷  ▷  ▷  $(RM_6_GENERATOR_3) \
▷  ▷  ▷  $(RM_6_GENERATOR_4) \
```

```
▷ ▷ ▷ $(RM_6_GENERATOR_5) \
▷ ▷ ▷ $(RM_6_GENERATOR_6) \
▷ ▷ ▷ $(RM_6_GENERATOR_7) \
▷ ▷ -end \
▷ ▷ -with C -and F -do -coding_theoretic_activity \
▷ ▷ ▷ -export_genma RM6_genma.csv \
▷ ▷ -end \
▷ ▷ -with C -and F -do -coding_theoretic_activity \
▷ ▷ ▷ -export_codewords_long RM6_codewords.csv \
▷ ▷ -end \
```

The next command produces a drawing of the generator matrix:

Example 454

```
RM_6_genma_draw:
▷ $(ORBITER) -v 2 \
▷ ▷ -draw_matrix \
▷ ▷ ▷ -input_csv_file RM6_genma.csv \
▷ ▷ ▷ -box_width 25 -bit_depth 8 \
▷ ▷ ▷ -partition 3 7 64 \
▷ ▷ -end
▷ convert RM6_genma_draw.bmp RM6_genma_draw.png
▷ $(OPEN) RM6_genma_draw.png
```

The following command produces a drawing of the matrix of codewords:

Example 455

```
RM_6_codewords_matrix_draw:
▷ $(ORBITER) -v 2 \
▷ ▷ -draw_matrix \
▷ ▷ ▷ -input_csv_file RM6_codewords.csv \
▷ ▷ ▷ -box_width 25 -bit_depth 8 \
▷ ▷ ▷ -partition 3 128 64 \
▷ ▷ -end
▷ convert RM6_codewords_draw.bmp RM6_codewords_draw.png
▷ $(OPEN) RM6_codewords_draw.png
```

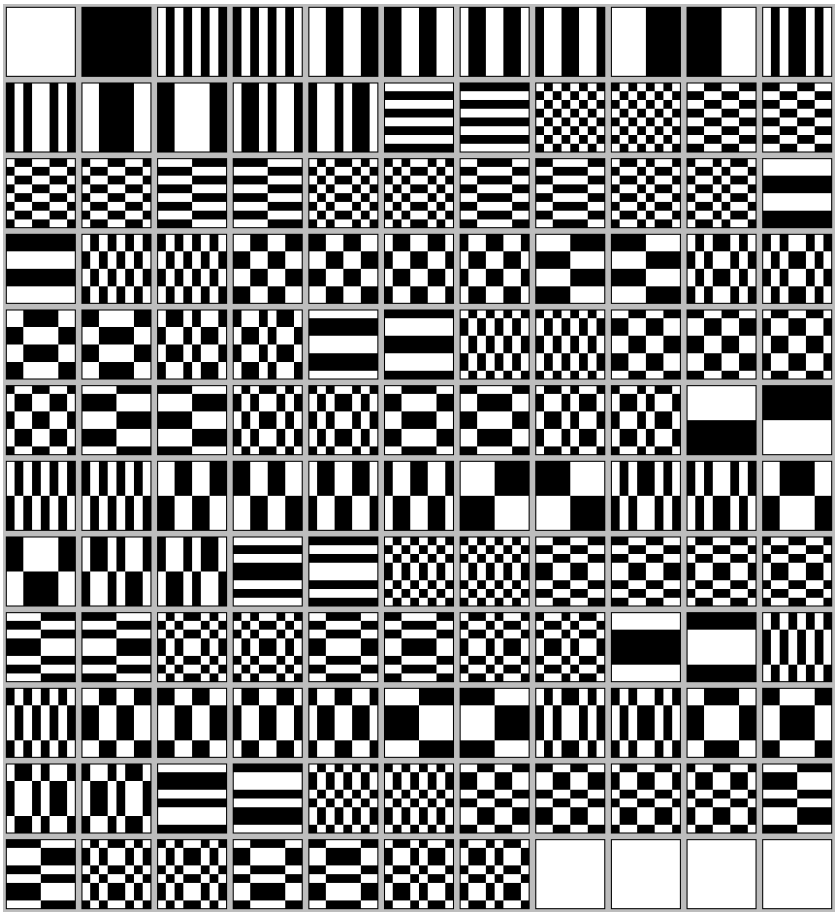
The next command produces individual pictures for each of the 128 codewords. The command utilizes the `-loop` command to automatize the processing. Afterwards, the codeword pictures are composed in one poster, using the `convert` tool from ImageMagix.

Example 456

```
RM_6_codewords_draw:
▷ $(ORBITER) -v 4 \
▷ ▷ -loop i 0 128 1 \
▷ ▷ ▷ -csv_file_select_rows RM6_codewords.csv %i \
```

```
▷ ▷ ▷ -reformat RM6_codewords_select_%i.csv RM6_codewords_ref_%i.csv 8 \
▷ ▷ ▷ -draw_matrix \
▷ ▷ ▷ ▷ -input_csv_file RM6_codewords_ref_%i.csv \
▷ ▷ ▷ ▷ -box_width 25 -bit_depth 8 \
▷ ▷ ▷ ▷ -partition 3 8 8 \
▷ ▷ ▷ -end \
▷ ▷ ▷ -system "convert RM6_codewords_ref_%i_draw.bmp -frame 8 %i.png" \
▷ ▷ -end_loop i
▷ convert 0.png 1.png 2.png 3.png 4.png 5.png \
▷ ▷ 6.png 7.png 8.png 9.png 10.png +append a0
▷ convert 11.png 12.png 13.png 14.png 15.png \
▷ ▷ 16.png 17.png 18.png 19.png 20.png 21.png +append a1
▷ convert 22.png 23.png 24.png 25.png 26.png \
▷ ▷ 27.png 28.png 29.png 30.png 31.png 32.png +append a2
▷ convert 33.png 34.png 35.png 36.png 37.png \
▷ ▷ 38.png 39.png 40.png 41.png 42.png 43.png +append a3
▷ convert 44.png 45.png 46.png 47.png 48.png \
▷ ▷ 49.png 50.png 51.png 52.png 53.png 54.png +append a4
▷ convert 55.png 56.png 57.png 58.png 59.png \
▷ ▷ 60.png 61.png 62.png 63.png 64.png 65.png +append a5
▷ convert 66.png 67.png 68.png 69.png 70.png \
▷ ▷ 71.png 72.png 73.png 74.png 75.png 76.png +append a6
▷ convert 77.png 78.png 79.png 80.png 81.png \
▷ ▷ 82.png 83.png 84.png 85.png 86.png 87.png +append a7
▷ convert 88.png 89.png 90.png 91.png 92.png \
▷ ▷ 93.png 94.png 95.png 96.png 97.png 98.png +append a8
▷ convert 99.png 100.png 101.png 102.png 103.png \
▷ ▷ 104.png 105.png 106.png 107.png 108.png 109.png +append a9
▷ convert 110.png 111.png 112.png 113.png 114.png \
▷ ▷ 115.png 116.png 117.png 118.png 119.png 120.png +append a10
▷ convert 121.png 122.png 123.png 124.png 125.png \
▷ ▷ 126.png 127.png 0.png 0.png 0.png 0.png +append a11
▷ convert a0 a1 a2 a3 a4 a5 a6 a7 a8 a9 a10 \
▷ ▷ a11 -append poster_RM_1_6.png
```

The codewords of RM_6 as boolean function in Hamming space are shown below:



10.6 BCH Codes

Let β be an n-th root of unity over $\mathbb{F}_q$. The minimum polynomial of β over $\mathbb{F}_q$ is denoted as $m_{\beta,\mathbb{F}_q}$. The BCH code of length n and designed distance d is the cyclic code with generator polynomial

$$\mathrm{lcm}\Big(m_{\beta^1,\mathbb{F}_q}, m_{\beta^2,\mathbb{F}_q}, \ldots, m_{\beta^{d-1},\mathbb{F}_q}\Big).$$

To create the polynomial $m_{\beta^a,\mathbb{F}_q}$, we consider the q-cyclotomic set of a modulo n, which is

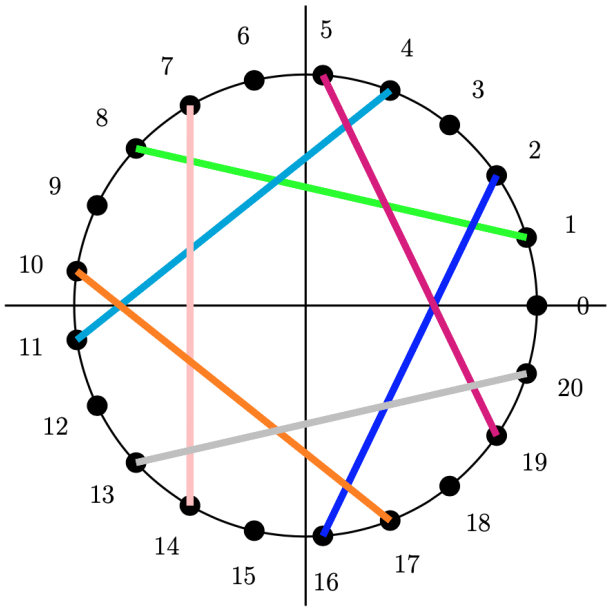
$$\{aq^i \mod n \mid i \in \mathbb{Z}\}.$$

Suppose we want to make a BCH-code of length 21 over $\mathbb{F}_8$. In Section 3.3, we considered the q-cyclotomic sets modulo 21 for $q = 8$. Let us produce a pictorial representation. Omitting the singletons, a transversal is given by the sets containing $1, 2, 4, 5, 7, 10, 13$. For this reason, we issue the command

Example 457

```
draw_cyclotomic_mod_21_q8:
▷  $(ORBITER) -v 2 \
▷  ▷  -draw_options \
▷  ▷  ▷  -radius 100 \
▷  ▷  ▷  -line_width 1.0 -embedded \
▷  ▷  -end \
▷  ▷  -draw_mod_n \
▷  ▷  ▷  -n 21 \
▷  ▷  ▷  -file mod_21_cyclotomic \
▷  ▷  ▷  -cyclotomic_sets 8 "1,2,4,5,7,10,13" \
▷  ▷  -end
▷  pdflatex mod_21_cyclotomic_draw.tex
▷  $(OPEN) mod_21_cyclotomic_draw.pdf
```

The output is shown below:



We will try BCH-codes with minimum distances 3, 5 and 7. Here is distance 3:

Example 458

```
F_8_BCH_code_d3:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 8 -override_polynomial 11 -end \
▷  ▷  -define C -code -field F \
▷  ▷  ▷  -BCH 21 3 \
▷  ▷  -end
▷  pdflatex BCH_codes_q8_n21_d3.tex
▷  $(OPEN) BCH_codes_q8_n21_d3.pdf
```

The code construction is shown in a latex report:

BCH-code:
$n = 21$, $k = 17$, $d_0 = 3$, $q = 8$,
$g(x) = m_1 m_2 = X^4 + 4X^3 + 4X^2 + 3X + 4$
Chosen cyclotomic sets:
{ 1, 8 }
{ 2, 16 }
The generator polynomial has degree 4

```
-dense "4,3,4,4,1"

-sparse "4,0,3,1,4,2,4,3,1,4"
```

The generator matrix is:

$$\begin{bmatrix}
4 & 3 & 4 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 4 & 3 & 4 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 4 & 3 & 4 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 4 & 3 & 4 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 4 & 3 & 4 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 4 & 3 & 4 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 4 & 3 & 4 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 3 & 4 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 3 & 4 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 3 & 4 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 3 & 4 & 4 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 3 & 4 & 4 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 3 & 4 & 4 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 3 & 4 & 4 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 3 & 4 & 4 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 3 & 4 & 4 & 1
\end{bmatrix}$$

Let us stick to the same parameters for the field and the length but increase the minimum distance to $d = 5$:

Example 459

```
F_8_BCH_code_d5:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 8 -override_polynomial 11 -end \
▷  ▷  -define C -code -field F \
▷  ▷  ▷  -BCH 21 5 \
▷  ▷  -end
▷  pdflatex BCH_codes_q8_n21_d5.tex
▷  $(OPEN) BCH_codes_q8_n21_d5.pdf
```

The output file is:

BCH-code:
$n = 21$, $k = 14$, $d_0 = 5$, $q = 8$,
$g(x) = m_1 m_2 m_3 m_4 = X^7 + 3X^6 + 3X^5 + 2X^4 + X^3 + 2X^2 + X + 2$
Chosen cyclotomic sets:
{ 1, 8 }
{ 2, 16 }
{ 3 }
{ 4, 11 }
The generator polynomial has degree 7

```
-dense "2,1,2,1,2,3,3,1"
```

```
-sparse "2,0,1,1,2,2,1,3,2,4,3,5,3,6,1,7"
```

The generator matrix is:

$$\begin{bmatrix}
2 & 1 & 2 & 1 & 2 & 3 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 2 & 1 & 2 & 1 & 2 & 3 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 2 & 1 & 2 & 1 & 2 & 3 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 2 & 1 & 2 & 1 & 2 & 3 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 2 & 1 & 2 & 1 & 2 & 3 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 2 & 1 & 2 & 1 & 2 & 3 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 & 2 & 1 & 2 & 3 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 & 2 & 1 & 2 & 3 & 3 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 & 2 & 1 & 2 & 3 & 3 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 & 2 & 1 & 2 & 3 & 3 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 & 2 & 1 & 2 & 3 & 3 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 & 2 & 1 & 2 & 3 & 3 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 & 2 & 1 & 2 & 3 & 3 & 1
\end{bmatrix}$$

We compute the minimum distance:

Example 460

```
F_8_BCH_code_d5_minimum_distance:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 8 -override_polynomial 11 -end \
▷  ▷  -define v -vector -format 14 -field F \
▷  ▷  ▷  -compact $(CODE_BCH_F8_N21_D5_GENMA_OVERRIDE_POLYNOMIAL11) \
▷  ▷  -end \
▷  ▷  -with F -do \
▷  ▷  -coding_theoretic_activity \
▷  ▷  ▷  -minimum_distance v \
▷  ▷  -end
# important: use the same polynomial as when creating the code.
#
# d=5
```

The minimum distance turns out to be $d = 5$.

Finally, we create the BCH code with minimum distance $d = 7$:

Example 461

```
F_8_BCH_code_d7:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 8 -override_polynomial 11 -end \
▷  ▷  -define C -code -field F \
▷  ▷  ▷  -BCH 21 7 \
▷  ▷  -end
```

The output file is:

BCH-code:
$n = 21$, $k = 11$, $d_0 = 7$, $q = 8$,
$g(x) = m_1 m_2 m_3 m_4 m_5 m_6 = X^{10} + X^9 + 2X^8 + 5X^7 + 2X^6 + 4X^4 + 6X^3 + 5X^2 + 6X + 6$
Chosen cyclotomic sets:
{ 1, 8 }
{ 2, 16 }
{ 3 }
{ 4, 11 }
{ 5, 19 }
{ 6 }
The generator polynomial has degree 10

```
-dense "6,6,5,6,4,0,2,5,2,1,1"

-sparse "6,0,6,1,5,2,6,3,4,4,2,6,5,7,2,8,1,9,1,10"
```

The generator matrix is:

$$\begin{bmatrix}
6 & 6 & 5 & 6 & 4 & 0 & 2 & 5 & 2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 6 & 6 & 5 & 6 & 4 & 0 & 2 & 5 & 2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 6 & 6 & 5 & 6 & 4 & 0 & 2 & 5 & 2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 6 & 6 & 5 & 6 & 4 & 0 & 2 & 5 & 2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 6 & 6 & 5 & 6 & 4 & 0 & 2 & 5 & 2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 6 & 6 & 5 & 6 & 4 & 0 & 2 & 5 & 2 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 6 & 6 & 5 & 6 & 4 & 0 & 2 & 5 & 2 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & 6 & 5 & 6 & 4 & 0 & 2 & 5 & 2 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & 6 & 5 & 6 & 4 & 0 & 2 & 5 & 2 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & 6 & 5 & 6 & 4 & 0 & 2 & 5 & 2 & 1 & 1
\end{bmatrix}$$

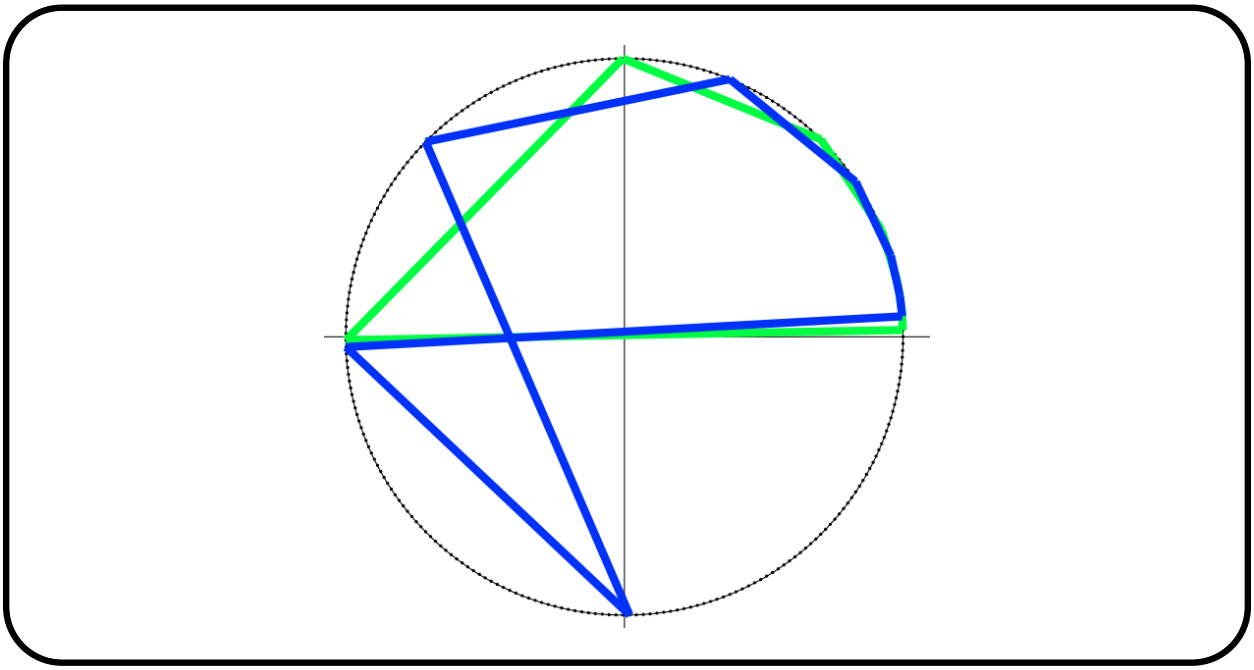
As a larger example, let us consider the 2-cyclotomic sets of 2 and 3 modulo 255. The following command produces a graphical representation on a circle (similar to the unit circle in complex analysis). The 255-th roots of unity are placed in the appropriate position.

Example 462

```
draw_mod_255_cyclotomic_1_and_3:
▷  $(ORBITER) -v 2 \
▷  ▷  -draw_options -nodes_empty -radius 10 \
▷  ▷  ▷  -line_width 0.4 -embedded -end \
▷  ▷  -draw_mod_n \
▷  ▷  ▷  -n 255 \
▷  ▷  ▷  -file mod_255_cyclotomic_1_and_3 \
▷  ▷  ▷  -cyclotomic_sets 2 "1,3" \
▷  ▷  -end
▷  pdflatex mod_255_cyclotomic_1_and_3_draw.tex
```

```
▷  $(OPEN) mod_255_cyclotomic_1_and_3_draw.pdf
```

The drawing is shown below



Suppose we want to make a BCH-code over $\mathbb{F}_{256}$. In order to keep the degree of the generator polynomial low, we try a quadratic field extension. This way, each cyclotomic set has size either 1 or 2. Since

$$256^2 - 1 = (256 + 1)(256 - 1) = 257 \cdot 3 \cdot 5 \cdot 17,$$

we can consider a code of length $n = 771 = 257 \cdot 3$. The following command computes the 256-cyclotomic cosets modulo 771:

Example 463

```
BCH_F256_roots_771:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 256 -end \
▷  ▷  -with F -do -finite_field_activity \
▷  ▷  ▷  -nth_roots 771 \
▷  ▷  -end
```

The next command creates a BCH-code of length 771 over $\mathbb{F}_{256}$ with minimum distance at least 16:

Example 464

```
BCH_F256_BCH_code_d16:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 256 -end \
▷  ▷  -define C -code -field F \
```

```
▷  ▷  ▷  -BCH 771 16 \
▷  ▷  -end
▷  pdflatex BCH_codes_q256_n771_d16.tex
▷  $(OPEN) BCH_codes_q256_n771_d16.pdf
```

We compute the minimum distance:

Example 465

```
F_8_BCH_code_d5_minimum_distance:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 8 -override_polynomial 11 -end \
▷  ▷  -define v -vector -format 14 -field F \
▷  ▷  ▷  -compact $(CODE_BCH_F8_N21_D5_GENMA_OVERRIDE_POLYNOMIAL11) \
▷  ▷  -end \
▷  ▷  -with F -do \
▷  ▷  -coding_theoretic_activity \
▷  ▷  ▷  -minimum_distance v \
▷  ▷  -end
# important: use the same polynomial as when creating the code.
#
# d=5
```

The minimum distance turns out to be $d = 5$.

10.7 Reed-Solomon Codes

Reed-Solomon codes are BCH-codes where the length n divides $q-1$. In particular, they are cyclic codes. They are almost never binary.

To create a Reed-Solomon code over $\mathbb{F}_7$ with minimum distance at least 3. The command

Example 466

```
RS_6_3:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 7 -end \
▷  ▷  -define C -code -field F -Reed_Solomon 6 3 -end \
▷  ▷  -with C -do -coding_theoretic_activity \
▷  ▷  ▷  -export_magma RS_6_3.magma \
▷  ▷  -end
▷  pdflatex RS_codes_q7_n6_d3.tex
▷  $(OPEN) RS_codes_q7_n6_d3.pdf
```

creates this code. The following report is produced:

RS-code:
$n = 6$, $k = 4$, $d_0 = 3$, $q = 7$,
$g(x) = m_1 m_2 = X^2 + 2X + 6$
The generator polynomial has degree 2

```
-dense "6,2,1"

-sparse "6,0,2,1,1,2"
```

The generator matrix is:

$$\begin{bmatrix} 6 & 2 & 1 & 0 & 0 & 0 \\ 0 & 6 & 2 & 1 & 0 & 0 \\ 0 & 0 & 6 & 2 & 1 & 0 \\ 0 & 0 & 0 & 6 & 2 & 1 \end{bmatrix}$$

As the report shows, the code is cyclic, generated by

$$(X-\alpha)(X-\alpha^2) = (X-3)(X-2) = X^2 + 2X + 6.$$

Let us investigate this code. We start with the weight enumerator. The command

Example 467

```
RS_6_3_weight_enumerator:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 7 -end \
▷  ▷  -define C -code -field F -Reed_Solomon 6 3 -end \
▷  ▷  -with C -do -coding_theoretic_activity \
▷  ▷  ▷  -export_magma RS_6_3.magma \
```

```
▷  ▷  -end \
▷  ▷  -with C -do \
▷  ▷  -coding_theoretic_activity \
▷  ▷  ▷  -weight_enumerator \
▷  ▷  -end
```

computes the weight enumerator, which turns out to be

$$(1, 0, 0, 120, 360, 972, 948).$$

In polynomial form, this is

$$1y^6 + 120x^3y^3 + 360x^4y^2 + 972x^5y + 948x^6.$$

This confirms that the minimum distance is three.

Let us consider an example of a Reed-Solomon code in characteristic two. We will create a Reed Solomon code of designed distance 3 over $\mathbb{F}_8$. The command

Example 468

```
RS_7_3:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 8 -end \
▷  ▷  -define C -code -field F -Reed_Solomon 7 3 -end \
▷  ▷  -with C -do -coding_theoretic_activity \
▷  ▷  ▷  -export_magma RS_7_3.magma \
▷  ▷  -end
▷  pdflatex RS_codes_q8_n7_d3.tex
▷  $(OPEN) RS_codes_q8_n7_d3.pdf
```

creates this code. The following report is produced:

RS-code:
$n = 7$, $k = 5$, $d_0 = 3$, $q = 8$,
$g(x) = m_1 m_2 = X^2 + 6X + 5$
The generator polynomial has degree 2

```
-dense "5,6,1"

-sparse "5,0,6,1,1,2"
```

The generator matrix is:

$$\begin{bmatrix} 5 & 6 & 1 & 0 & 0 & 0 & 0 \\ 0 & 5 & 6 & 1 & 0 & 0 & 0 \\ 0 & 0 & 5 & 6 & 1 & 0 & 0 \\ 0 & 0 & 0 & 5 & 6 & 1 & 0 \\ 0 & 0 & 0 & 0 & 5 & 6 & 1 \end{bmatrix}$$

As we can see, the code is a cyclic code generated by the polynomial

$$(X - \alpha)(X - \alpha^2) = X^2 + 6X + 5.$$

What is the minimum weight of the code? The following command

Example 469

```
RS_7_3_weight_enumerator:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 8 -end \
▷  ▷  -define C -code -field F -Reed_Solomon 7 3 -end \
▷  ▷  -with C -do -coding_theoretic_activity \
▷  ▷  ▷  -export_magma RS_7_3.magma \
▷  ▷  -end \
▷  ▷  -with C -do \
▷  ▷  -coding_theoretic_activity \
▷  ▷  ▷  -weight_enumerator \
▷  ▷  -end
▷  pdflatex RS_codes_q8_n7_d3.tex
▷  $(OPEN) RS_codes_q8_n7_d3.pdf
```

computes the weight enumerator, which turns out to be

$$y^7 + 245x^3y^4 + 1225x^4y^3 + 5586x^5y^2 + 12838x^6y + 12873x^7.$$

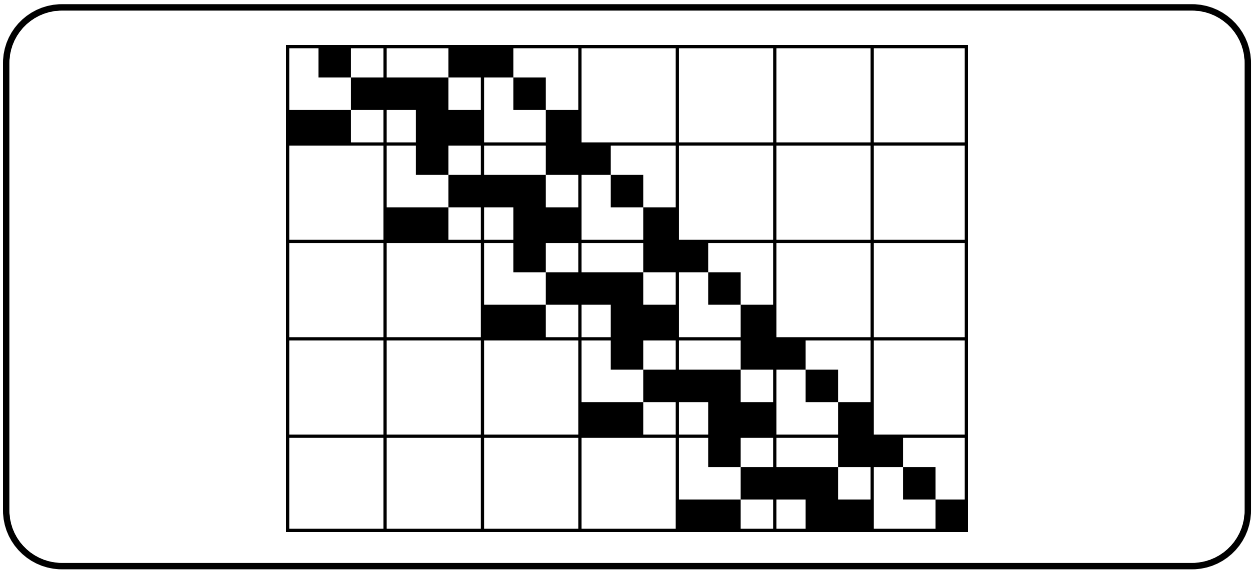
This shows that the minimum distance is 3, as expected. Computing the automorphism group of the code is computationally infeasible.

The next command performs field reduction of the code.

Example 470

```
RS_7_3_field_reduction:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 8 -end \
▷  ▷  -define C -code -field F -Reed_Solomon 7 3 -end \
▷  ▷  -with F -do \
▷  ▷  -finite_field_activity \
▷  ▷  -field_reduction "RS_8_red_2" \
▷  ▷  ▷  2 5 7 $(CODE_RS_F8_N7_K5_D3_GENMA) \
▷  ▷  -end
▷  $(ORBITER) -v 2 \
▷  ▷  -draw_matrix -input_csv_file RS_8_red_2.csv \
▷  ▷  -box_width 40 -bit_depth 24 \
▷  ▷  -partition 4 "3,3,3,3,3" "3,3,3,3,3,3,3" -end
▷  pdflatex field_reduction_Q8_q2_5_7.tex
▷  $(OPEN) field_reduction_Q8_q2_5_7.pdf
```

This produces a $[21, 15]_2$ code. The reduced matrix is below



We store the generator matrix in a makefile variable:

Example 471

```
RS_8_reduced="\
01000110000000000000\
00111001000000000000\
11001100100000000000\
00001000110000000000\
00000111001000000000\
00011001100100000000\
00000001000110000000\
00000000111001000000\
00000110011001000000\
00000000010001100000\
00000000001110010000\
00000000110011001000\
00000000000010001100\
00000000000001110010\
00000000000110011001"
```

Let us compute the weight enumerator of the reduced code. The command

Example 472

```
RS_7_3_reduced_weight_enumerator:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define v -vector -format 15 -field F \
▷  ▷  ▷  -compact $(RS_8_reduced) \
▷  ▷  -end \
▷  ▷  -define C -code -field F \
▷  ▷  ▷  -generator_matrix v \
▷  ▷  -end \
```

```
▷ ▷ -with C -do \
▷ ▷ -coding_theoretic_activity \
▷ ▷ ▷  -weight_enumerator \
▷ ▷ -end
```

computes the weight enumerator of the binary code. It is

$$1y^{21} + 28x^3y^{18} + 84x^4y^{17} + 273x^5y^{16} + 924x^6y^{15} + 1956x^7y^{14} + 2982x^8y^{13} + 4340x^9y^{12} + 5796x^{10}y^{11} + 5796x^{11}y^{10} + 4340x^{12}y^9 + 2982x^{13}y^8 + 1956x^{14}y^7 + 924x^{15}y^6 + 273x^{16}y^5 + 84x^{17}y^4 + 28x^{18}y^3 + 1x^{21}$$

In particular, the field reduced Reed-Solomon code has minimum distance three. Since there are other codes of minimum distance 4, the field-reduced code is not optimal.

10.8 Twisted Tensor Product Codes

The command

Example 473

```
TTP_A_4:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F4 -finite_field -q 4 -end \
▷  ▷  -define F16 -finite_field -q 16 -end \
▷  ▷  -define C -code -field F4 -ttpA F16 -end \
▷  ▷  -with C -do -coding_theoretic_activity \
▷  ▷  ▷  -export_magma TTP_A_q4.magma \
▷  ▷  -end \
▷  ▷  -with C -do -coding_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex code_n18_k9_q4.tex
▷  $(OPEN) code_n18_k9_q4.pdf
```

creates the twisted tensor product code arising from the hyperoval of PG(2, 16). The command also produces a report of the code, as shown below.

The generator matrix is:

$$\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 3 & 1 & 3 & 1 & 2 & 3 & 0 & 3 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 2 & 0 & 2 & 1 & 2 & 3 & 0 & 2 & 3 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 2 & 3 & 1 & 0 & 3 & 2 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 3 & 3 & 1 & 1 & 2 & 2 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 3 & 2 & 0 & 1 & 0 & 0 & 3 & 2 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 2 & 3 & 1 & 0 & 0 & 0 & 2 & 3 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 2 & 2 & 1 & 2 & 1 & 3 & 0 & 2 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 2 & 0 & 3 & 3 & 0 & 3 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}$$

The command also exports the code to magma. The magma file is shown below:

```
K<w> := GF(4);
V := VectorSpace(K, 18);
C := LinearCode(sub<V |
[1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0],
[1,w^2,1,w^2,1,w^1,w^2,0,w^2,w^1,1,0,0,0,0,0,0,0],
[0,w^1,0,w^1,1,w^1,w^2,0,w^1,w^2,0,1,0,0,0,0,0,0],
[1,w^1,w^2,1,0,w^2,w^1,0,1,0,0,0,1,0,0,0,0,0],
[0,w^2,w^2,1,1,w^1,w^1,0,0,1,0,0,0,1,0,0,0,0],
[1,1,w^2,w^1,0,1,0,0,w^2,w^1,0,0,0,0,1,0,0,0],
[1,1,w^1,w^2,1,0,0,0,w^1,w^2,0,0,0,0,0,1,0,0],
```

```
[1,w^1,w^1,1,w^1,1,w^2,0,w^1,w^1,0,0,0,0,0,0,1,0],
[1,w^1,0,w^2,w^2,0,w^2,0,1,1,0,0,0,0,0,0,0,1]>);
```

10.9 Bounds

In coding theory, one main question is to determine the best value of $d_{\max}$ for a fixed n, k and q such that a linear $[n, k, d]_q$ code exists. There are many bounds, both upper and lower bounds. An upper bound tells us that no code with $d \geq d_{\max}$ exists. A lower bound tells us that a code with $d \geq d_{\max}$ exists. The command

Example 474

```
bounds_for_d_given_n15_k6_q2:
▷  $(ORBITER) -v 2 \
▷  ▷  -make_bounds_for_d_given_n_and_k_and_q 15 6 2
```

gives upper and lower bounds on the optimal minimum distance d_{max} of a $[15, 6]_2$ code. The values of the Gilbert-Varshamov lower bound and the Singleton, Hamming, Plotkin and Griesmer upper bounds are computed. The output is:

```
n = 15 k=6 q=2
d_GV = 5
d_singleton = 10
d_hamming = 6
d_plotkin = 7
d_griesmer = 6
```

This shows that $5 \leq d_{\max} \leq 6$. The command

Example 475

```
coding_theory_bounds_q2:
▷  $(ORBITER) -v 2 -table_of_bounds 12 2
▷  $(ORBITER) -v 2 \
▷  ▷  -csv_file_latex 1 table_of_bounds_n12_q2.csv
▷  pdflatex table_of_bounds_n12_q2.tex
▷  $(OPEN) table_of_bounds_n12_q2.pdf
```

produces a table of bounds for binary codes with $n, k \leq 12$. The second command produces the latex report below:

n	k	q	GV	S	H	P	G
2	1	2	2	2	2	2	3
2	2	2	1	1	2	1	1
3	1	2	3	3	4	3	4
3	2	2	2	2	2	2	2
3	3	2	1	1	2	1	1
4	1	2	4	4	4	4	5
4	2	2	2	3	2	2	2
4	3	2	2	2	2	2	2
4	4	2	1	1	2	2	1
5	1	2	5	5	6	5	6
5	2	2	3	4	4	3	3
5	3	2	2	3	2	2	2
5	4	2	2	2	2	2	2
5	5	2	1	1	2	2	1
6	1	2	6	6	6	6	7
6	2	2	3	5	4	4	4
6	3	2	3	4	4	3	3
6	4	2	2	3	2	3	2
6	5	2	2	2	2	3	2
6	6	2	1	1	2	3	1
7	1	2	7	7	8	7	8
7	2	2	4	6	6	4	4
7	3	2	3	5	4	4	4
7	4	2	3	4	4	3	3
7	5	2	2	3	2	3	2
7	6	2	2	2	2	3	2
7	7	2	1	1	2	3	1
8	1	2	8	8	8	8	9
8	2	2	4	7	6	5	5
8	3	2	4	6	4	4	4
8	4	2	3	5	4	4	4
8	5	2	2	4	2	4	3
8	6	2	2	3	2	4	2
8	7	2	2	2	2	4	2
8	8	2	1	1	2	4	1
9	1	2	9	9	10	9	10
9	2	2	5	8	6	6	6
9	3	2	4	7	6	5	4
9	4	2	3	6	4	4	4
9	5	2	3	5	4	4	4
9	6	2	2	4	2	4	3
9	7	2	2	3	2	4	2
9	8	2	2	2	2	4	2
9	9	2	1	1	2	4	1
10	1	2	10	10	10	10	11
10	2	2	5	9	8	6	6
10	3	2	4	8	6	5	5
10	4	2	4	7	6	5	4
10	5	2	3	6	4	5	4
10	6	2	3	5	4	5	4
10	7	2	2	4	2	5	3
10	8	2	2	3	2	5	2
10	9	2	2	2	2	5	2
10	10	2	1	1	2	5	1
11	1	2	11	11	12	11	12
11	2	2	6	10	8	7	7
11	3	2	5	9	8	6	6
11	4	2	4	8	6	5	5
11	5	2	4	7	4	5	4
11	6	2	3	6	4	5	4
11	7	2	3	5	4	5	4
11	8	2	2	4	2	5	3
11	9	2	2	3	2	5	2
11	10	2	2	2	2	5	2
11	11	2	1	1	2	5	1
12	1	2	12	12	12	12	13
12	2	2	6	11	10	8	8
12	3	2	5	10	8	6	6
12	4	2	5	9	6	6	6
12	5	2	4	8	6	6	5
12	6	2	3	7	4	6	4
12	7	2	3	6	4	6	4
12	8	2	3	5	4	6	4
12	9	2	2	4	2	6	3
12	10	2	2	3	2	6	2
12	11	2	2	2	2	6	2
12	12	2	1	1	2	6	1

The command

Example 476

```
GV_n15_k6_d5:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define C -code -field F \
▷  ▷  ▷  -Gilbert_Varshamov 15 6 5 \
▷  ▷  -end
```

creates a $[15, 6, d]_2$ with minimum distance $g \geq 5$ using a greedy algorithm based on the proof of the Gilbert-Varshamov bound. The code that is produced has the following generator matrix:

```
1 1 1 1 1 1 1 1 1 1 0 0 0 0 0
1 1 1 1 1 0 0 0 0 0 1 0 0 0 0
1 1 1 0 0 1 1 0 0 0 0 1 0 0 0
1 1 0 1 0 1 0 1 0 0 0 0 1 0 0
1 0 1 0 1 0 1 1 0 0 0 0 0 1 0
1 0 1 1 0 1 0 0 1 0 0 0 0 0 1
```

To compute the minimum distance of the code, we do:

Example 477

```
CODE_GV_N15_K6="\
111111111100000\
111110000010000\
111001100001000\
110101010000100\
101010110000010\
101101001000001"
```

Example 478

```
GV_n15_k6_d5_weight_enumerator:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define v -vector -format 6 -field F \
▷  ▷  ▷  -compact $(CODE_GV_N15_K6) \
▷  ▷  -end \
▷  ▷  -define C -code -field F \
▷  ▷  ▷  -generator_matrix v \
▷  ▷  -end \
▷  ▷  -with C -do \
▷  ▷  -coding_theoretic_activity \
▷  ▷  ▷  -weight_enumerator \
▷  ▷  -end
```

The weight enumerator is computed to be:

$$1y^{15} + 27x^6y^9 + 24x^8y^7 + 9x^{10}y^5 + 3x^{12}y^3.$$

From this, we see that the code has minimum distance 6, which is better than predicted.

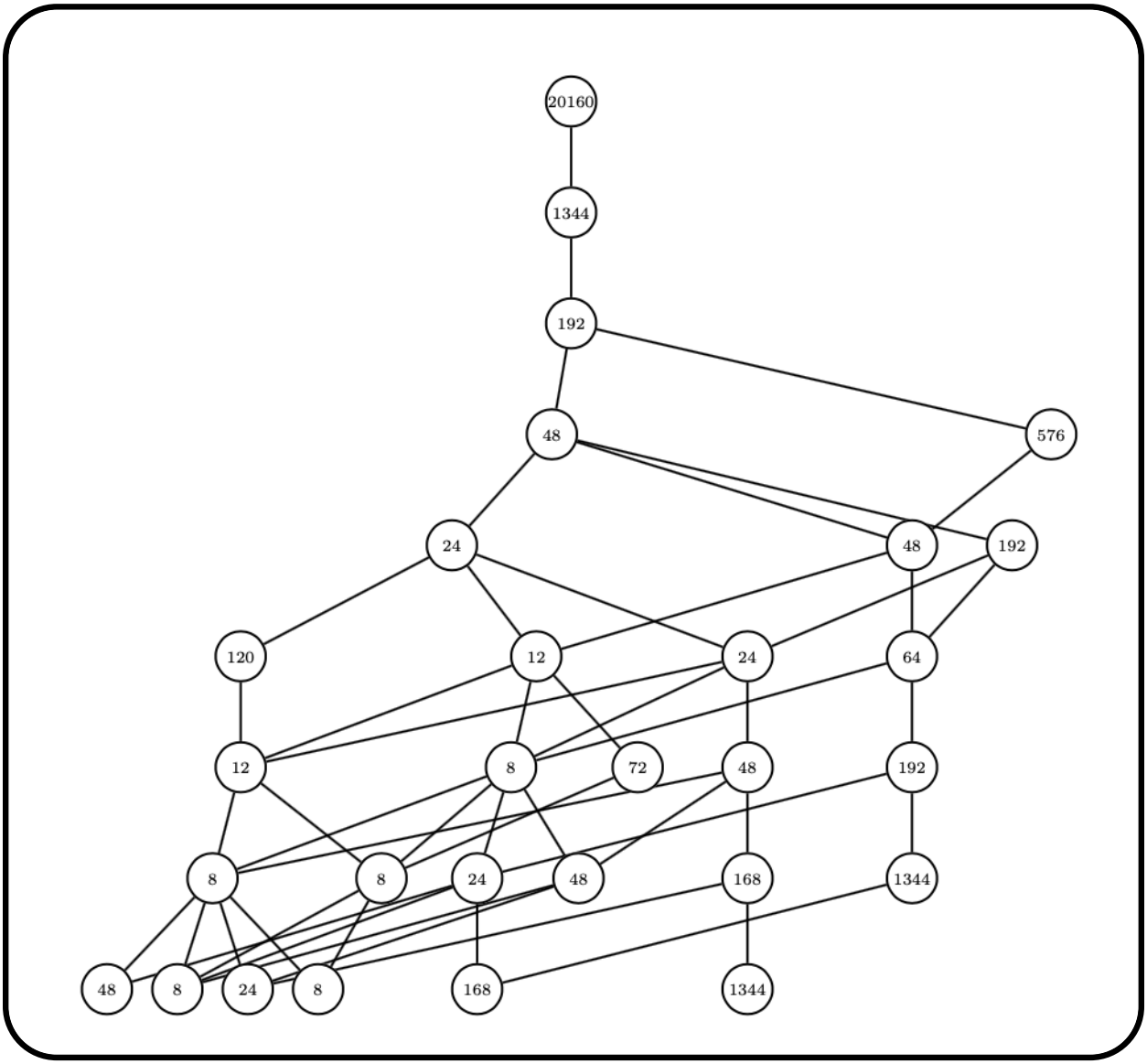
10.10 Classification of Optimal Linear Codes

The classification problem of optimal codes in coding theory is the problem of determining the equivalence classes of codes for a given set of values of n and k and q with a lower bound on d. Orbiter can be used to classify linear codes with given redundancy and bounded minimum distance. The redundancy of a linear $[n, k]$ code is the parameter $r = n-k$. Codes with redundancy r can be identified with subsets of $\mathrm{PG}(r-1, q)$. Under this correspondence, a code with minimum distance at least d corresponds to a subset such that any $d-1$ elements are independent. We use the notation $\Lambda_{r-1,s}(q)$ to denote the poset of subsets of $\mathrm{PG}(r-1, q)$ for which any $d-1$-subset (if any) is independent. Under the correspondence, the action of $\mathrm{P\Gamma L}(r, q)$ on $\Lambda_{r-1,s}(q)$ corresponds to the orbits of equivalent linear codes. For this reason, we are interested in determining the orbits of $\mathrm{P\Gamma L}(r, q)$ on $\Lambda_{r-1,s}(q)$. An orbit of size n represents an isometry class of $[n, n-r, d; q]$ codes with $d \geq s+1$. The projective stabilizer of the subset is the automorphism group of the code. The Orbiter command

Example 479

```
codes_8_4_4:
▷  $(ORBITER) -v 6 \
▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
▷  ▷  -define Control -poset_classification_control \
▷  ▷  ▷  -problem_label codes_8_4_4 \
▷  ▷  ▷  -draw_options \
▷  ▷  ▷  ▷  -embedded -radius 250 \
▷  ▷  ▷  ▷  -line_width 1.0 -spanning_tree \
▷  ▷  ▷  -end \
▷  ▷  -end \
▷  ▷  -define G \
▷  ▷  -linear_group -PGL 4 2 -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -linear_codes Control 3 8 \
▷  ▷  -end
▷  #pdflatex codes_8_4_4_poset_lvl_8.tex
▷  #$(OPEN) codes_8_4_4_poset_lvl_8.pdf
▷  #pdflatex codes_8_4_4_poset.tex
▷  #$(OPEN) codes_8_4_4_poset.pdf
```

classifies linear codes with redundancy 4 and minimum distance at least 4. Orbiter confirms that there is exactly one such code, and it computes the code together with the projective stabilizer. Orbiter creates the action of the group $\mathrm{PGL}(4, 2)$ on the poset $\Lambda_{3,3}(2)$. Using poset classification, Orbiter then produces the poset of orbits shown below



In this diagram, the numbers stand for Orbiter ranks of points in PG(3, 2). All nodes except for the root node have a number attached to it. The nodes represent subsets. In order to determine the set associated to a node, follow the path from the root node to the node and collect the points according to their labels. The root node represents the empty set. The $[8, 4, 4; 2]$-code is represented by the set $\{0, 1, 2, 3, 8, 11, 13, 14\}$. The fact that there is only one node at level 8 in the poset of orbits tells us that the code is unique up to equivalence. Let us look at the code. The elements of the set $\{0, 1, 2, 3, 8, 11, 13, 14\}$ are points in PG(3, 2). We write the coordinate vectors in the columns of a matrix H:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}.$$

This matrix is the parity check matrix H of the code $\mathcal{C}$. This means that the words of the code are the vectors $\mathbf{c}$ such that $\mathbf{c} \cdot H^\top = 0$. Observe that the vectors that we put in the columns of H all have odd weight. They are in fact the points of the hyperplane $x + y + z + w = 0$. This shows that the stabilizer of the code which is the stabilizer of the set is equal to AGL(3, 2), a group of order 1344.

Chapter 11

Combinatorics

11.1 Introduction

In Tables 11.1 and 11.2, global Orbiter commands for Combinatorics are summarized.

The command

Example 480

```
Sym_10_conj_classes:
▷  $(ORBITER) -v 2 -conjugacy_classes_Sym_n 10
▷  $(OPEN) classes_Sym_10.csv
```

produces a list of the conjugacy classes of Sym(10). The list is written to a csv file.

The next command computes the character table of the symmetric group Sym(4):

Example 481

```
Char_Sym_4:
▷  $(ORBITER) -v 2 -character_table_symmetric_group 4
```

The command produces the following output:

The character table of Sym(4) is the matrix

$$\begin{bmatrix} 1 & -1 & 1 & 1 & -1 \\ 3 & 1 & 0 & -1 & -1 \\ 2 & 0 & -1 & 2 & 0 \\ 3 & -1 & 0 & -1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Orbiter often uses the lexicographic order to arrange things. Consider the k-subsets of an n-set. The following example creates the 3-subsets of a 6-set in lexicographic order. A tree representation is produced. The leaves of the tree represent the subsets:

Combinatorics (Part 1)

Command	Arguments	Purpose
-random_permutation	n fname	Creates a random permutation in $\mathrm{Sym}(n)$ and stores it in the given file.
-create_random_k_subsets	n k N	Creates N random k-subsets of an n-set.
-read_poset_file	fname	Reads a poset from the given file.
-read_poset_file_with_grouping	fname x-stretch	Reads a poset from the given file and sets stretch factor for orbit grouping.
-list_parameters_of_SRG	$v_{\max}$	Performs a sift for putative parameter sets of SRGs.
-conjugacy_classes_Sym_n	n	Compute a list of conjugacy classes of $\mathrm{Sym}(n)$.
-tree_of_all_k_subsets	n k	Creates a tree-file for all k-subsets of an n-set.
-Delandtsheer_Doyen		See Section 13.4.
-tdo_refinement		See Section 11.5.
-tdo_print		See Section 11.5.
-convert_stack_to_tdo		See Section 11.5.
-maximal_arc_parameters		See Section 11.5.
-arc_parameters		See Section 11.5.
-pentomino_puzzle		

Table 11.1: Combinatorics (Part 1)

Combinatorics (Part 2)

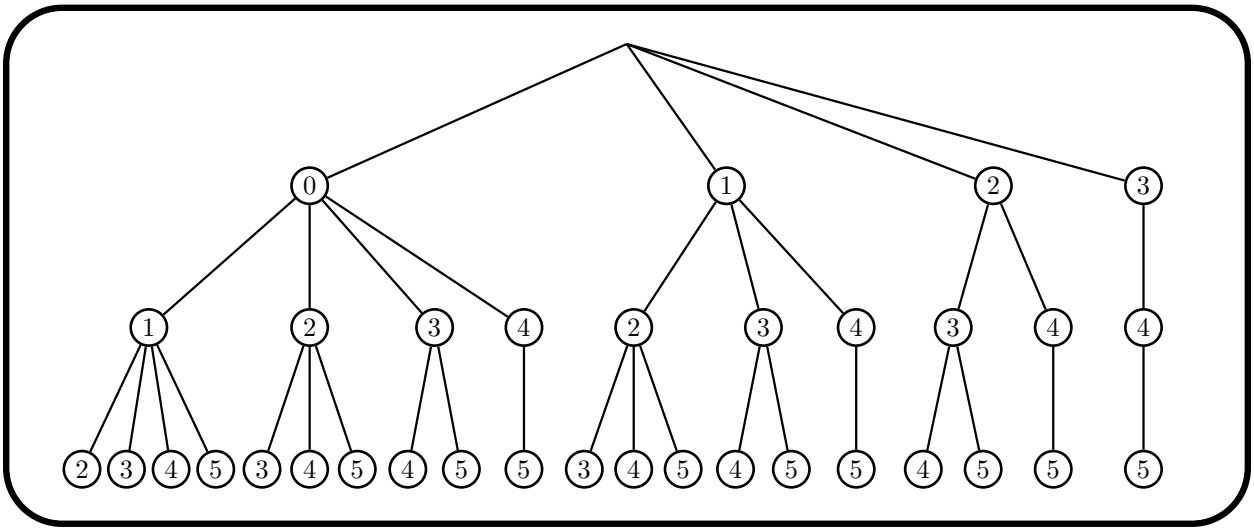
Command	Arguments	Purpose
-make_elementary_symmetric_functions	n $k_{\max}$	Computes the elementary symmetric functions in n variables of degree $1, \ldots, k_{\max}$
-Dedekind_numbers	$n_{\min}$ $n_{\max}$ $q_{\min}$ $q_{\max}$	Computes the Dedekind numbers $D_{n,q}$ for $n_{\min} \le n \le n_{\max}$ and $q_{\min} \le q \le q_{\max}$
-rank_k_subset	n k L	Computes the ranks of k-subsets chosen from an n-set. L is a list of k-sets taken from an n-set.
-geometry_builder		See Section 11.4.
-character_table_symmetric_group	n	Computes the character table of $\mathrm{Sym}(n)$ using the algorithm of Burnside.
-domino_portrait	D s fname	Computes a domino portrait for a graphics file in r/g/b format using double-D domino sets.

Table 11.2: Combinatorics (Part 2)

Example 482

```
all_subsets_6_3:
▷  $(ORBITER) -v 2 -tree_of_all_k_subsets 6 3
▷  $(ORBITER) -v 20 \
▷  ▷  -draw_options -embedded \
▷  ▷  ▷  -nodes -radius 80 \
▷  ▷  ▷  -xin 5000 -yin 5000 \
▷  ▷  ▷  -xout 1000000 -yout 500000 \
▷  ▷  ▷  -scale 0.5 -line_width 1.0 \
▷  ▷  -end \
▷  ▷  -tree_draw -file all_k_subsets_n6_k3.tree -end
▷  pdflatex all_k_subsets_n6_k3_draw.tex
▷  $(OPEN) all_k_subsets_n6_k3_draw.pdf
```

Here is the tree:



The following command illustrates how to create random k-subsets of a set of size n. In the example, we create 20 5-subsets of a 10-element set:

Example 483

```
random_k_subsets:
▷  $(ORBITER) -v 4 \
▷  ▷  -create_random_k_subsets 10 5 20
```

Using the lexicographic order, the k-subsets of an n-element set are ranked. The following command computes the ranks of a number of 3-subsets of a 10-element set:

Example 484

```
rank_k_subsets_test:
▷  $(ORBITER) -v 2 \
```

```
▷  ▷  -define Blocks -vector -format 7 \
▷  ▷  ▷  -dense "0,1,2,0,3,4,1,3,5,2,4,5,3,6,7,1,6,8,0,6,9" \
▷  ▷  ▷  -end \
▷  ▷  -rank_k_subset 10 3 Blocks
```

Orbiter can create the Sylvester type Hadamard matrix of size 2^n (also called the Walsh matrix). The following command creates the matrix of size $2^4 \times 2^4$ and produces a graphical representation:

Example 485

```
Walsh_matrix_4:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -with F -do -finite_field_activity \
▷  ▷  ▷  -Walsh_matrix 4 -end
▷  $(ORBITER) -v 2 -draw_matrix \
▷  ▷  -input_csv_file Walsh_01_4.csv \
▷  ▷  -box_width 10 -bit_depth 24 -partition 3 16 16 -end
▷  #pdflatex GF_2.tex
▷  #$(OPEN) GF_2.pdf
```

The following command creates the matrix of Dedekind numbers of order at most 10:

Example 486

```
Dedekind_10_10:
▷  $(ORBITER) -v 3 -Dedekind_numbers 2 10 2 10
▷  $(ORBITER) -v 3 -csv_file_latex 1 Dedekind_2_10_2_10.csv
▷  pdflatex Dedekind_2_10_2_10.tex
▷  $(OPEN) Dedekind_2_10_2_10.pdf
```

The following table is produced:

2	3	4	5	6	7	8	9	10
2	1	3	6	10	15	21	28	36
3	2	8	20	40	70	112	168	240
4	3	18	60	150	315	588	1008	1620
5	6	48	204	624	1554	3360	6552	11808
6	9	116	670	2580	7735	19544	43596	88440
7	18	312	2340	11160	39990	117648	299592	683280
8	30	810	8160	48750	209790	720300	2096640	5380020
9	56	2184	29120	217000	1119720	4483696	14913024	43046640
10	99	5880	104754	976248	6045837	28245840	107370900	348672528

The following command creates the elementary symmetric functions in 4 variables.

Example 487

```
elementary_symmetric_functions_4:
▷  $(ORBITER) -make_elementary_symmetric_functions 4 4
```

The output is:

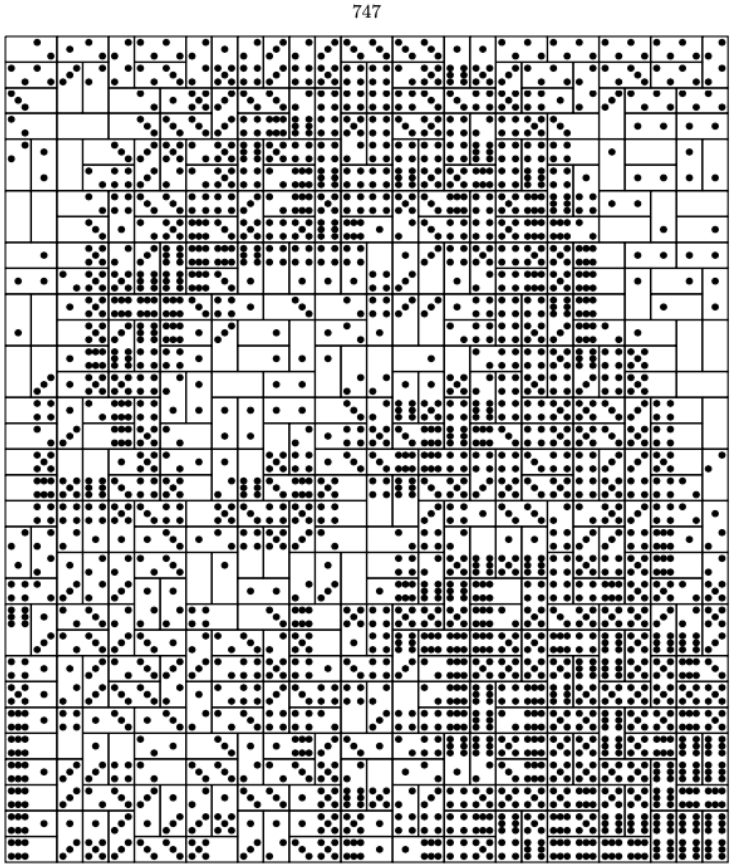
```
k=1 :
x0 + x1 + x2 + x3
k=2 :
x0*x1 + x0*x2 + x0*x3 + x1*x2 + x1*x3 + x2*x3
k=3 :
x0*x1*x2 + x0*x1*x3 + x0*x2*x3 + x1*x2*x3
k=4 :
x0*x1*x2*x3
```

Orbiter can compute Domino portraits. To do so, we need an input file in r/g/b format of size $(D+1)s \times Ds$, where $D = 7$ for double-six dominos.

Example 488

```
domino_portrait:
▷  cp $(MY_PATH)/examples/users_guide/anton_28x32.? .
▷  $(ORBITER) -v 3 -domino_portrait 7 4 anton_28x32 -end
```

The portrait is shown below

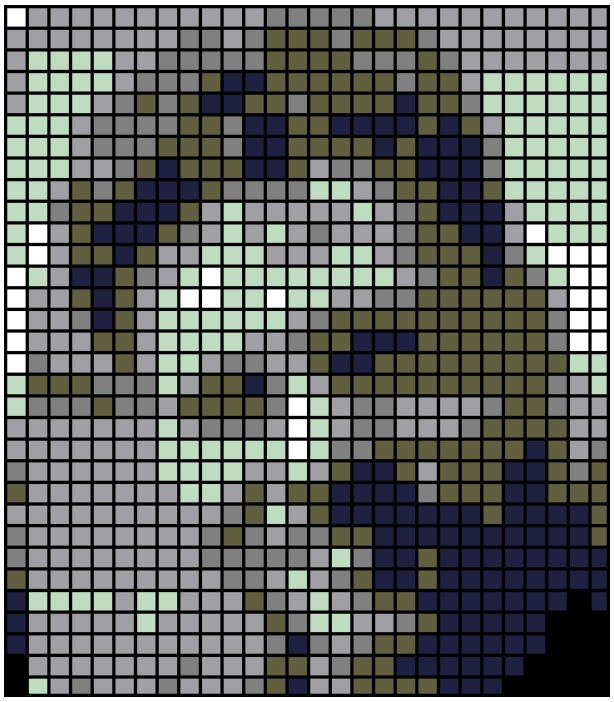


It is possible to compare the domino portrait with a grayscale verson of the input image. The following command creates a grayscale image of the input file that was written during the previous command.

Example 489

```
domino_portrait_input:
▷  cp $(MY_PATH)/examples/users_guide/anton_28x32.? .
▷  $(ORBITER) -v 2 \
▷  ▷  -define all_one_r -vector -repeat 1 28 -end \
▷  ▷  -define all_one_c -vector -repeat 1 32 -end \
▷  ▷  -draw_matrix \
▷  ▷  ▷  -grayscale \
▷  ▷  ▷  -invert_colors \
▷  ▷  ▷  -input_csv_file anton_28x32_m.csv \
▷  ▷  ▷  -box_width 20 -bit_depth 8 \
▷  ▷  ▷  -partition 3 \
▷  ▷  ▷  ▷  all_one_c all_one_r \
▷  ▷  -end
▷  $(OPEN) anton_28x32_m_draw.bmp
```

The grayscale version of the input file is



11.2 Combinatorial Objects

Combinatorial algorithms often need to process large quantities of combinatorial data. Orbiter facilitates batch processing by means of input streams. An input stream defines a sequence of combinatorial objects, all of the same type. The objects can be defined in files or on the command line, or any combination thereof. Once an input stream is defined, there are commands for activities that can be applied to the objects in the stream.

Table 11.3 shows the commands that are available to define input streams. In Table 11.4, activities for combinatorial objects are listed. Some commands require extra options. Table 11.5 shows options for classification. Table 11.6 shows options for the report which is generated after a classification.

Let us consider some examples. First, we create the Hirschfeld surface in PG$(3, 4)$, using point ranks as described in Section 4.2. The 45 points on the surface can be coded as in the following makefile variable:

Example 490

```
HIRSCHFELD_SURFACE_Q4_SET_OF_POINTS="0,1,2,3,4,5,6,7,8,9,\
10,11,12,13,14,23,26,27,30,31,34,35,38,39,42,47,48,51,52,\
53,54,59,60,61,62,67,68,69,70,75,76,79,80,81,82"
```

The next command creates a combinatorial object from this set of points:

Example 491

```
Hirschfeld_q4_from_set:
▷  $(ORBITER) -v 4 \
▷  ▷  -define H -set -here \
▷  ▷  ▷  $(HIRSCHFELD_SURFACE_Q4_SET_OF_POINTS) \
▷  ▷  -end \
▷  ▷  -define C -combinatorial_object \
▷  ▷  ▷  -label Hirschfeld_surface Hirschfeld\_surface \
▷  ▷  ▷  -set_of_points H \
▷  ▷  -end
```

The next example creates the two hyperovals in PG$(2, 16)$. The hyperovals are stored in two makefile variables:

Example 492

```
HYPEROVAL_16_144="0, 1, 2, 3, 52, 67, 89, 106, 126, \
141, 159, 176, 184, 199, 220, 235, 245, 262"
```

Example 493

```
HYPEROVAL_16_16320="0, 1, 2, 3, 52, 70, 83, 109, 127, \
139, 156, 174, 186, 199, 217, 229, 256, 264"
```

The next command creates combinatorial objects for these two hyperovals:

Defining Combinatorial Objects

Command	Arguments	Purpose
-label	label-txt label-tex	A label for the objects.
-set_of_points	S	A set S consisting of points.
-set_of_lines	S	A set S consisting of lines.
-set_of_points_and_lines	S_1 S_2	A set S_1 consisting of points and a second set S_2 consisting of lines.
-set_of_packing	S q	A set S of packings of $\mathrm{PG}(3,q)$.
-file_of_points	fname	A set consisting of points read from file.
-file_of_points_csv	fname col-header	A csv-file containing sets of points in a specific column. The column header is given.
-file_of_lines	fname	A set consisting of lines read from file.
-file_of_packings	fname	A set consisting of packings read from file.
-file_of_packings_through_spread_table	spread-table fname-packings q	A file of packings of $\mathrm{PG}(3,q)$. This relies on the presence of a spread-table.
-file_of_designs_through_block_orbits	fname1 fname2 v k	A set of designs (or configurations) built from a set of orbits on k-subsets. The file fname1 is a matrix of zero-one vectors, for instance the solutions of a Diophantine system (see Section 15.3). The file fname2 is a list of orbits of a group on k-subsets created using the -export_something activity from Table 7.2 using the keyword set_orbits. The design will be invariant under the group used to compute the k-orbits.
-file_of_point_set	fname	A file containing sets of points.
-file_of_designs	fname v b k c	A file containing designs or large sets. Here, v is the number of points in the design, b is the number of blocks, k is the block size, and c is the size of one class of the partition.
-file_of_incidence_geometries	fname v b f	A file of incidence geometries. The incidence structure is listed as set of flags. Here, v is the number of points, b is the number of blocks and f is the number of flags. If point i and block j are incident, this corresponds to the value $ib+j$ being listed as a flag (recall that i and j are zero-based).
-file_of_incidence_geometries_by_row_ranks	fname v b r	A file describing incidence geometries defined by their row ranks.
-incidence_geometry	fname v b f	An incidence geometry defined by a set of flags. Here, v is the number of points, b is the number of blocks and f is the number of flags.
-incidence_geometry_by_row_ranks	fname v b r	An incidence geometry defined by row ranks.
-from_parallel_search	fname-mask nb_cases cases_fname	

Table 11.3: Defining Combinatorial Objects

Activities for Combinatorial Objects		
Command	Arguments	Purpose
-save		Save the objects under the name that was assigned automatically.
-save_as	fname	Save the objects under the given name.
-extract_subset	set fname	Extract a subset of the objects.
-line_type_old		Compute the line type of a set in projective space.
-line_type		Compute the line type of a set in projective space.
-conic_type		Compute the conic type of a set in projective space.
-non_conical_type		
-ideal	R	Compute the ideal of the set, using the ring R.
-canonical_form_PG	P options	Requires a projective space P. See Tab. 11.5 for options.
-canonical_form	options	Classification by canonical form. See Tab. 11.5 for options.
-report	options	See Tab. 11.6 for options.
-draw_incidence_matrices	prefix	
-test_distinguishing_property	G	Test the distinguishing property for the graph G.
-unpack_from_restricted_action	prefix G	
-line_covering_type	prefix P L	Computes the line type of the lines in L in the projective space P.

Table 11.4: Activities for Combinatorial Objects

Classification of Combinatorial Objects		
Command	Arguments	Purpose
-save_classification	prefix	Save classification when finished.
-max_TDO_depth	d	Limit TDO depth to d in the report.
-save_canonical_labeling		Save the canonical labeling.
-save_ago		Save the automorphism group orders to file.
-save_transversal		Save the indices of the elements chosen for the transversal.

Table 11.5: Classification of Combinatorial Objects

Options for Reporting After Classification

Command	Arguments	Purpose
-export_flag_orbits		Export flag orbits to csv file.
-show_incidence_matrices		Show incidence matrices in the report.
-show_TDO		Compute the TDO for the report.
-show_TDA		Compute the TDA for the report.
-export_labels		
-export_group_orbiter		
-export_group_GAP		
-lex_least	geo	Show lex-least orbit representatives. Here, geo is a geometry builder object.

Table 11.6: Options for Reporting After Classification

Example 494

```
hyperoval_16_create:
▷  $(ORBITER) -v 2 \
▷  ▷  -define C -combinatorial_object \
▷  ▷  ▷  -label hyperoval_q16 hyperoval\_q16 \
▷  ▷  ▷  -set_of_points $(HYPEROVAL_16_16320) \
▷  ▷  ▷  -set_of_points $(HYPEROVAL_16_144) \
▷  ▷  -end \
```

In the next example, we read the points of an elliptic curve from file (see Section 4.2):

Example 495

```
EC_read: elliptic_curve_b1_c3_q11.txt
▷  $(ORBITER) -v 4 \
▷  ▷  -define C -combinatorial_object \
▷  ▷  ▷  -label EC EC \
▷  ▷  ▷  -file_of_points elliptic_curve_b1_c3_q11.txt \
▷  ▷  -end
```

In the next example, we read a packing, using a previously defined table of spreads, stored in a csv file.

Example 496

```
PG_3_5_assume_31_read:
▷  $(ORBITER) -v 2 \
▷  ▷  -define C -combinatorial_object \
▷  ▷  ▷  -label packings_25 packings_25 \
▷  ▷  ▷  -file_of_packings_through_spread_table \
▷  ▷  ▷  ▷  H31_packings.csv \
▷  ▷  ▷  ▷  SPREAD_TABLES_5_REG/spread_25_spreads.csv \
▷  ▷  ▷  ▷  5 \
▷  ▷  -end
```

The following command reads a file of large sets of designs:

Example 497

```
LS_AG_2_3_read:
▷  $(ORBITER) -v 2 \
▷  ▷  -define C -combinatorial_object \
▷  ▷  ▷  -label designs designs \
▷  ▷  ▷  -file_of_designs \
▷  ▷  ▷  solutions.csv 9 84 3 12 \
▷  ▷  -end
```

The next command reads incidence geometries from a file containing the flags:

Example 498

```
geo_7_3_read:
▷  $(ORBITER) -v 10 \
▷  ▷  -draw_incidence_structure_description \
▷  ▷  ▷  -width 60 -width_10 6 -end \
▷  ▷  -define C -combinatorial_object \
▷  ▷  ▷  -label geo_7_3 geo\_7\_3 \
▷  ▷  ▷  -file_of_incidence_geometries \
▷  ▷  ▷  ▷  7_3.inc 7 7 21 \
▷  ▷  -end
```

The next command creates incidence geometries from a file containing row-ranks:

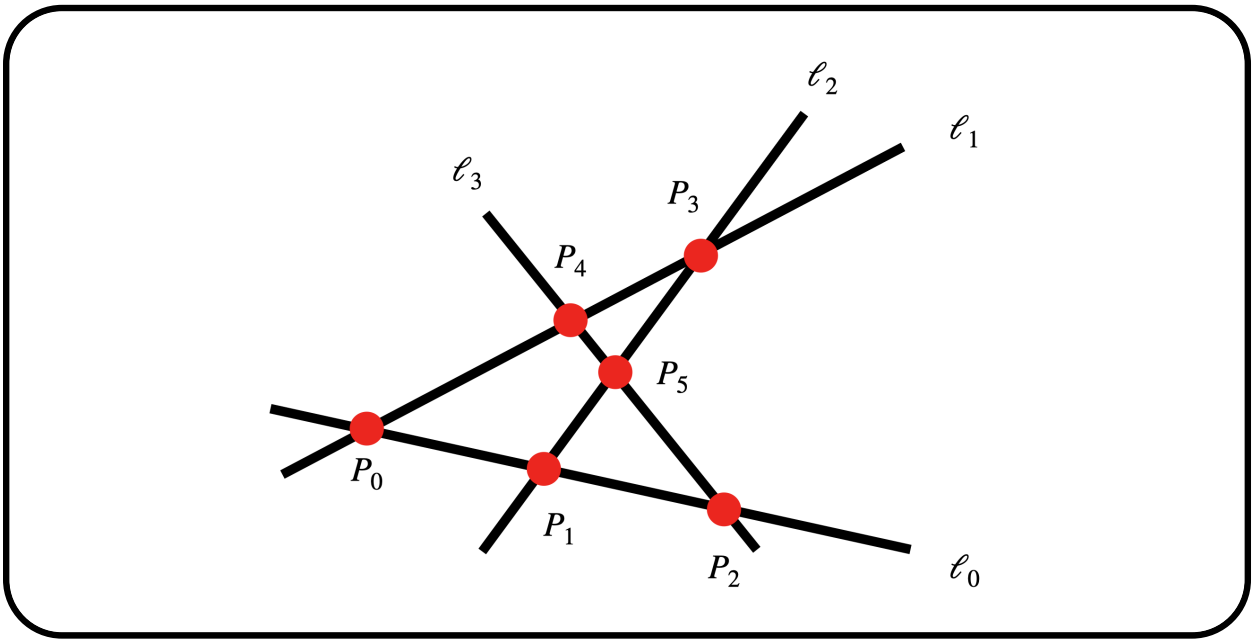
Example 499

```
Desargues_path_lex_least_read:
▷  echo $(DESARGUES_PATH_LEX_LEAST) >Desargues_path_lex_least.inc
▷  $(ORBITER) -v 10 \
▷  ▷  -draw_incidence_structure_description \
▷  ▷  ▷  -width 60 -width_10 6 -end \
▷  ▷  -define C -combinatorial_object \
▷  ▷  ▷  -label Desargues_path_lex_least Desargues\_path\_lex\_least \
▷  ▷  ▷  -file_of_incidence_geometries_by_row_ranks \
▷  ▷  ▷  ▷  Desargues_path_lex_least.inc 10 10 3 \
▷  ▷  -end
```

Combinatorial objects can be stored in text files. There can be any number of objects in one file. The objects themselves are coded. For instance, a set of points in projective space is given as a set of integers, using the Orbiter point ranks. Likewise, a set of lines is coded using Orbiter line ranks. For designs, there are several ways in which the object can be stored. One way is by listing the incidences (flags) in a numerical form. An incidence between point i and line j is recorded as $ib + j$. Here, i and j are the zero based index values of the point and line pair, and b is the number of lines. Another way is by recording the flags in a row of the incidence matrix as a subset. In this case, the number of incidences per row must be constant, say r. The flags in a given row thus comprise an r-subset of a set of size b, and the ranking of subsets is used to encode the flags in one row. Proceeding in a row-by-row fashion, the incidence matrix is encoded as a vector of ranks of length v. Yet another way is by listing the columns of the incidence matrix, provided the number

of inciences is constant across columns, say k. In this case, we encode the flags in one column as a rank of a k-subset of a set of size v.

The Pasch configuration is the configuration of 6 points and 4 lines shown below:



In the geometry, we have 4 lines, which we can identify with the index sets of the points as $\{0,1,2\}$, $\{0,3,4\}$, $\{1,3,5\}$ and $\{2,4,5\}$. The incidence matrix of the configuration is

	ℓ_0	ℓ_1	ℓ_2	ℓ_3
P_0	1	1	0	0
P_1	1	0	1	0
P_2	1	0	0	1
P_3	0	1	1	0
P_4	0	1	0	1
P_5	0	0	1	1

The rows correspond to the points, the columns correspond to the lines. The labels of points and lines are shown.

How to encode an incidence structure for computer storage? There are three ways to encode the incidence structure. One way encodes the flags of the geometry. This will be described next. The flag space is the set of all possible flags in the incidence matrix between the given number of points and lines. The space is totally ordered using the row-major ordering

	ℓ_0	ℓ_1	ℓ_2	ℓ_3
P_0	0	1	2	3
P_1	4	5	6	7
P_2	8	9	10	11
P_3	12	13	14	15
P_4	16	17	18	19
P_5	20	21	22	23

The Pasch configuration can now be coded as

$$\{0, 1, 4, 6, 8, 11, 13, 14, 17, 19, 22, 23\}.$$

The file `pasch.inc` contains:

```
6 4 12
0 1 4 6 8 11 13 14 17 19 22 23
-1 1
24
```

The first line lists the number of rows and columns of the incidence matrix, and the number of incidences. The geometry is encoded on the next line. After that, a marker of -1 shows that this is the only geometry in this file (the file format allows for any number of incidence geometries, all with the same parameters). The final row is the order of the automorphism group of the geometry. This row is optional. In case that there are several geometries in the file, the orders will all be listed. In this case, the possible values will be collected with multiplicities, and listed in decreasing order. The command

Example 500

```
geo_pasch_read:
▷  $(ORBITER) -v 10 \
▷  ▷  -define C -combinatorial_object \
▷  ▷  ▷  -label Pasch Pasch \
▷  ▷  ▷  -file_of_incidence_geometries \
▷  ▷  ▷  ▷  pasch.inc 6 4 12 \
▷  ▷  -end
```

reads the incidence geometry from the file `pasch.inc`. It is also possible to enter the incidence geometry directly from the command line. The following example uses the `-incidence_geometry` command to do so:

Example 501

```
geo_pasch_given:
▷  $(ORBITER) -v 10 \
▷  ▷  -define C -combinatorial_object \
▷  ▷  ▷  -label Pasch Pasch \
▷  ▷  ▷  -incidence_geometry \
▷  ▷  ▷  ▷  "0,1,4,6,8,11,13,14,17,19,22,23" \
▷  ▷  ▷  ▷  6 4 12 \
```

```
▷ ▷ -end
```

11.3 Combinatorial Linear Spaces

A linear space is a pair $(S, \mathcal{L})$ where S is a set and $\mathcal{L}$ is a set of subsets of S such that each set $L \in \mathcal{L}$ satisfies $|L| \geq 2$ and morover for any two $a, b \in S$ there is exactly one element $L \in \mathcal{L}$ such that both a and b belong to L. The usual notions of isomorphism and automorphism apply. For finite linear spaces, a first combinatorial property is the number a_i which counts the number of sets $L \in \mathcal{L}$ of size i. The vector $(a_2, \ldots, a_n)$ is the line type of $(S, \mathcal{L})$. The equation

$$\binom{n}{2} = \sum_{j=2}^{n} a_j \binom{j}{2} \tag{11.1}$$

is satisfied. The equation can be used to generate all possible line types of a putative linear space. Here is an example. For $|S| = 6$, (11.1) becomes

$$x_0\binom{6}{2} + x_1\binom{5}{2} + x_2\binom{4}{2} + x_3\binom{3}{2} + x_4\binom{2}{2} = \binom{6}{2}.$$

Here, $(x_0, x_1, \ldots, x_4)$ is the line type of a putative linear space on 6 points. That is, $x_i = a_{6-i}$ is the number of lines of size $6 - i$. The extended coefficient matrix of the system is

$$[15\ 10\ 6\ 3\ 1\ \ |\ 15\ ].$$

The Orbiter command

Example 502

```
linsp6:
▷  $(ORBITER) -v 4 \
▷  ▷  -define A -vector -format 1 -dense "15,10,6,3,1" -end \
▷  ▷  -define D -diophant -label linsp6 \
▷  ▷  -coefficient_matrix A \
▷  ▷  -RHS "mult=1,EQ=15" \
▷  ▷  -x_min_global 0 \
▷  ▷  -x_max_global 15 \
▷  ▷  -end \
▷  ▷  -with D -do \
▷  ▷  ▷  -diophant_activity -solve_mckay \
▷  ▷  -end
▷
# Found 15 solutions with 22 backtrack steps
```

solves the system using McKay's program possolve [54]. The program finds 15 solutions, written to the file `linsp6.sol`.

Let us consider a problem from [12]. Suppose we are interested in linear spaces on 30 points and with line type $(7, 5^{27}, 4^{24})$. This notation means that we assume one 7-line, 27 5-lines and 24 4-lines. The type of a point P is the set of integers

$$p_j = \#j\text{-lines though } P.$$

We are trying to precompute the matrix of point types

$$(p_{ij})$$

where $j = 7, 5, 4$ and i belongs to an index set of all possible point types. Fixing a point P, counting points $Q \neq P$ collinear with P yields

$$6p_7 + 4p_5 + 3p_4 = 29, \quad p_7 \leq 1,\ p_5 \leq 27, p_4 \leq 24.$$

Using the Orbiter commands

Example 503

```
linsp30_pt_types:
▷  $(ORBITER) -v 4 \
▷  ▷  -define A -vector -format 1 -dense "6,4,3" -end \
▷  ▷  -define D -diophant \
▷  ▷  ▷  -label linsp30_pt_types \
▷  ▷  ▷  -coefficient_matrix A \
▷  ▷  ▷  -RHS "mult=1,EQ=29" -x_bounds "0,1,0,27,0,24" \
▷  ▷  -end \
▷  ▷  -with D -do \
▷  ▷  ▷  -diophant_activity -solve_mckay \
▷  ▷  -end
```

we determine the possibilities

$$(p_7, p_5, p_4) = \begin{Bmatrix} 1 & 5 & 1 \\ 1 & 2 & 5 \\ 0 & 5 & 3 \\ 0 & 2 & 7 \end{Bmatrix}$$

The rows in this matrix are called the point types ($i = 0, 1, 2, 3$). Let b_i be the number of points of type i. By counting points, incident (point,line) pairs by j-lines and pairs of intersecting j-lines, we arrive at the following system:

$$\begin{aligned} b_0 + b_1 + b_2 + b_3 &= 30 \\ b_0 + b_1 &= 7 \\ 5b_0 + 2b_1 + 5b_2 + 2b_3 &= 135 = 27 \cdot 5 \\ b_0 + 5b_1 + 3b_2 + 7b_3 &= 96 = 24 \cdot 4 \\ 10b_0 + b_1 + 10b_2 + b_3 &\le 351 = \binom{27}{2} \\ 10b_1 + 3b_2 + 21b_3 &\le 276 = \binom{24}{2} \end{aligned}$$

Using the Orbiter commands

Example 504

```
linsp30_pt_distribution:
▷  $(ORBITER) -v 4 \
▷  ▷  -define A -vector -format 6 -dense \
▷  ▷  ▷  "1,1,1,1,1,1,0,0,5,2,5,2,1,5,3,7,10,1,10,1,0,10,3,21" \
▷  ▷  -end \
▷  ▷  -define D -diophant \
▷  ▷  ▷  -label linsp30_pt_distribution \
▷  ▷  ▷  -coefficient_matrix A \
▷  ▷  ▷  -RHS "mult=1,EQ=30" \
▷  ▷  ▷  -RHS "mult=1,EQ=7" \
▷  ▷  ▷  -RHS "mult=1,EQ=135" \
▷  ▷  ▷  -RHS "mult=1,EQ=96" \
▷  ▷  ▷  -RHS "mult=1,LE=351" \
▷  ▷  ▷  -RHS "mult=1,LE=276" \
```

```
▷ ▷ ▷ -x_min_global 0 -x_max_global 30 \
▷ ▷ -end \
▷ ▷ -with D -do \
▷ ▷ ▷ -diophant_activity -solve_mckay \
▷ ▷ -end \
▷ ▷ -with D -do \
▷ ▷ ▷ -diophant_activity -draw_as_bitmap 20 8 \
▷ ▷ -end
```

we determine the possibilities

$$(b_0, b_1, b_2, b_3) = \begin{cases} 2 & 5 & 23 & 0 \\ 3 & 4 & 22 & 1 \\ 4 & 3 & 21 & 2 \\ 5 & 2 & 20 & 3 \\ 6 & 1 & 19 & 4 \\ 7 & 0 & 18 & 5 \end{cases}$$

11.4 Classification of Configurations and Geometries

A partial linear space is a set system on a fixed set V. We write $\mathcal{L} = (V, \mathcal{B})$, where $\mathcal{B}$ is a set of distinct subsets of V, called lines. The members of $V \cup \mathcal{B}$ are called elements. For two elements x, y, we say that x is incident with y, written xIy, if either $x \in y$ or $y \in x$. We require that any line has at least two points and that any two points are contained in at most one line. A *decomposition* of a linear space is a partition $\Pi = (C_1, \ldots, C_n)$ of $V \cup \mathcal{B}$ such that each C_i either is a subset of V or a subset of $\mathcal{B}$. A decomposition is called *tactical* if for all i, the incidence number

$$\iota(C_i, C_j) = \#\{y \in C_j, xIy\}$$

does not depend on the choice of $x \in C_i$. Any linear space has a tactical decomposition, as the discrete partition (every element is in its own class) is tactical. Let $\mathrm{Aut}(\mathcal{L})$ be the automorphism group of the linear space, which is the subgroup of $\mathrm{Sym}(V)$ which preserves incidence. For $\alpha \in \mathrm{Aut}(\mathcal{L})$, we say that the decomposition Π preserves α if α fixes every class of Π. For $A \leq \mathrm{Aut}(\mathcal{L})$, we say that Π preserves A if Π preserves every element $\alpha \in A$. Mostly, we are interested in those decompositions Π which preserve $\mathrm{Aut}(\mathcal{L})$. In light of this, the discrete decomposition is not that interesting.

Any linear space has a coarsest tactical decomposition that preserves its automorphism group: The orbit partition of the automorphism group acting on $V \cup \mathcal{B}$ will do. Up to ordering of the classes, the coarsest tactical refinement is unique. Computing the orbit decomposition is challenging as it involves computing the automorphism group. Computationally, there are easier ways to get to admissible decompositions. One way is by means of successive refinements. If a class C_i does not have the property that $\iota(C_i, C_j)$ is well-defined for all $x \in C_i$, then a refinement of C_i will do. The coarsest refinement of C_i has the property that if C_i preserves some group A then the refinement will do, too. This shows that there is an algorithm to compute a tactical decompostion of any given linear space $\mathcal{P}$. Simply start with the decomposition of two classes, one the set of points and one the set of blocks, and refine. The output may or may not be equal to the decomposition arising from the orbit partition of $\mathrm{Aut}(\mathcal{L})$.

Let us consider the opposite question. Given a tactical decomposition, can we classify the linear spaces which admit a given tactical decomposition. The `-geometry_builder` option can help. Table 11.7 shows the options for the geometry builder.

The command

Example 505

```
geo_10_3:
▷  $(ORBITER) -v 2 \
▷  ▷  -define Test_lines -set -loop 4 11 1 -end \
▷  ▷  -define Geo -geometry_builder \
▷  ▷  ▷  -V 10 -B 10 -TDO 3 -fuse 1 \
▷  ▷  ▷  -fname_GEO 10_3 \
▷  ▷  ▷  -output_to_inc_file \
▷  ▷  ▷  -output_to_sage_file \
▷  ▷  ▷  -output_to_blocks_file \
▷  ▷  ▷  -test Test_lines \
▷  ▷  -end
```

classifies the configuations 10_3. It uses isomorphism tests after 4, 5, 6, 7, 8, 9 and 10 points. The positions of the tests is defined using a set called `Test_lines`. The set of test lines is defined using a loop command. The command shows that there are exactly 10 configurations of this kind. One of them is the Desargues configuration. Four different output files can be written. Each contains all geometries, but the file format is different.

Constructing and Classifying Geometries

Command	Arguments	Purpose
-V	part	The initial partition of points (rows).
-B	part	The initial partition of blocks (columns).
-TDO	tdo	The initial row-tactical decomposition scheme.
-fuse	fuse	The partition of row classes.
-girth_test	g	Require the girth of the collinearity graph to be at least g.
-lambda	λ	Set λ for two-design test. Every pair of points is required to lie in λ blocks.
-no_square_test		Disable the test for linear spaces.
-simple		Construct simple designs only (needs -lambda)
-search_tree		Write a file containing the search tree (at the level of rows of the partial geometry).
-search_tree_flags		Write a file containing the search tree (at the level of flags of the partial geometry). A flag is an incident point-block pair.
-orderly		Apply orderly generation.
-special_test_not_orderly		Apply the special test. This option only applies to generation which is not orderly.
-split	$l\ r\ m$	Split the search tree at line l. After l lines, continue only the cases which are congruent to r modulo m.
-fname_GEO	fname	Set the output file name base (no extension).
-output_to_inc_file		Set output to inc file.
-output_to_sage_file		Set output to sage file.
-output_to_blocks_file		Set output to a file containing the blocks in coded form.
-output_to_blocks_latex_file		Set output to a file containing the blocks in latex.

Table 11.7: Constructing and Classifying Geometries

1. The option `-output_to_inc_file` writes `10_3.inc`. The file contains the incidences in increasing order. The position in the incidence matrix is given. Apart from the header and footer, each row in the file describes exactly one geometry. The incidences are given in numeric form. Each incidence is the numerical position of the point/block pair in the incidence matrix. The position is the numbering of the matrix entries in the incidence matrix in row-major ordering, starting with zero for the top left entry. The index of the incidence in row i (zero-based) and column j is $b \cdot i + j$, where b is the number of blocks in the geometry. The header consists of one row, containing the number of points, the number of lines, and the number of incidences. The footer consists of two rows. A single -1 indicates the beginning of the footer. The second row in the footer summarizes the automorphism group orders of the germetries. Here is an example: The file `10_3.inc` lists the 10 configurations 10_3:

```
10 10 30
0 1 2 10 13 14 20 25 26 31 33 35 41 44 47 52 53 58 62 66 69 74 78 79 85 87 89 96 97 98
0 1 2 10 13 14 20 25 26 31 33 35 41 44 47 52 54 58 62 66 69 73 78 79 85 87 89 96 97 98
0 1 2 10 13 14 20 25 26 31 33 35 41 44 47 52 54 58 62 67 69 73 76 79 85 88 89 96 97 98
0 1 2 10 13 14 20 25 26 31 33 35 41 44 47 52 56 58 62 67 69 73 78 79 84 86 89 95 97 98
0 1 2 10 13 14 20 25 26 31 33 35 41 47 48 52 54 57 62 66 68 73 77 79 84 86 89 95 98 99
0 1 2 10 13 14 20 25 26 31 33 35 41 47 48 52 54 57 62 66 68 73 78 79 84 86 89 95 97 99
0 1 2 10 13 14 20 25 26 31 33 35 41 47 48 52 54 57 62 66 69 73 78 79 84 86 88 95 97 99
0 1 2 10 13 14 20 25 26 31 33 37 41 45 48 52 54 57 62 66 68 73 75 79 84 86 89 97 98 99
0 1 2 10 13 14 20 25 26 31 33 37 41 45 48 52 54 57 62 66 68 73 75 79 84 88 89 96 97 99
0 1 2 10 13 14 20 25 26 31 33 37 41 45 48 52 54 57 62 66 68 73 76 79 84 88 89 95 97 99
-1 10
120, 24, 12, 10, 6, 4^2, 3^2, 2
```

2. The option `-output_to_sage_file` writes `10_3.sage`. This file is meant to be read by Sage [66].

3. The option `-output_to_blocks_file` writes `10_3.blocks`. Here is the content of the file `10_3.blocks`:

```
10 10 3
0 15 26 44 51 68 81 109 114 116
0 15 26 46 49 68 81 109 114 116
0 15 26 46 49 68 83 106 115 116
0 15 26 46 52 69 77 106 114 116
0 15 26 46 56 69 80 101 106 119
0 15 26 46 56 69 80 103 104 119
0 15 26 46 56 69 80 103 107 117
0 15 26 46 56 72 80 93 106 119
0 15 26 46 56 72 81 93 105 119
0 15 26 46 56 74 79 93 105 119
-1 10
120, 24, 12, 10, 6, 4^2, 3^2, 2
```

   contains the blocks. Each number represents the rank of a 3-subset corresponding to a block in the lexicographic ordering of all 3-subsets.

4. The option `-output_to_blocks_latex_file` writes `10_3.blocks_long`. The file `10_3.blocks_long` contains a list of all blocks written out in a format ready for use in latex.

It is possible to create graphical representations of the search tree. The additional option `-search_tree` can be given, as shown in the following command:

Example 506

```
geo_10_3_tree:
▷  $(ORBITER) -v 20 \
▷  ▷  -define Test_lines -set -loop 0 11 1 -end \
▷  ▷  -define GEO -geometry_builder \
▷  ▷  ▷  -V 10 -B 10 -TDO 3 -fuse 1 \
▷  ▷  ▷  -fname_GEO 10_3 \
▷  ▷  ▷  -output_to_inc_file \
▷  ▷  ▷  -search_tree \
▷  ▷  ▷  -test Test_lines \
▷  ▷  -end
▷  $(ORBITER) -v 20 \
▷  ▷  -draw_options -embedded -radius 40 \
▷  ▷  ▷  -paperheight 220 \
▷  ▷  ▷  -paperwidth 330 \
▷  ▷  ▷  -xin 10000 -yin 10000 \
▷  ▷  ▷  -xout 1000000 -yout 500000 \
▷  ▷  ▷  -scale 2 -line_width 0.3 \
▷  ▷  ▷  -nodes_empty \
▷  ▷  -end \
▷  ▷  -tree_draw \
▷  ▷  ▷  -file 10_3_tree.txt \
▷  ▷  -end
▷  pdflatex 10_3_tree_draw.tex
▷  $(OPEN) 10_3_tree_draw.pdf
```

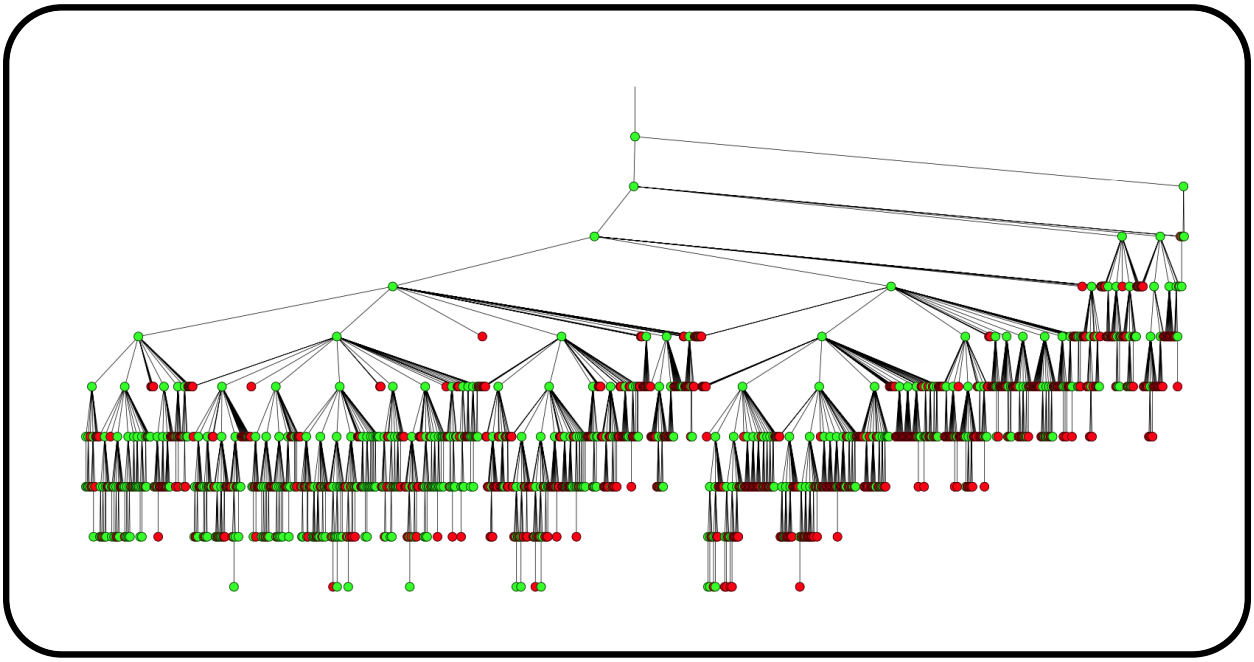
The `-search_tree` option causes Orbiter to create a file containing the search tree. The name of the file is derived from the file name given with the `fname_GEO` option. Here, the `fname_GEO` option sets the output file to `10_3`. The `-search_tree` option then creates the file `10_3_tree.txt`. Once the classification is completed, the `-tree_draw` command can be used to create a drawing of the search tree from the file just created. The vertex color represents the outcome of the isomorphism test. A green node is accepted. A red node is rejected. The search will continue for the green nodes only. A green node at the bottom of the tree corresponds to an isomorphism type of a geometry satisfying all the requirements. In the example of the configurations 10_3, there are 10 green nodes at the very bottom of the search tree. They represent the 10 isomorphism types of configurations 10_3. The size of the tree can be determined by counting the lines of the file `10_3_tree.txt` and subtracting one:

```
wc 10_3_tree.txt
```

The word count command yields the following output:

```
    1471   13543   37633 10_3_tree.txt
```

This means that there are 1471 lines in the file. Hence the search tree has 1470 nodes. The resulting tree is shown below



Any incidence structure defines a graph on its underlying set of points. The vertices of the collinearity graph are the points of the incidence structure. Two vertices are adjacent if the incidence structure contains a block which contains the associated points. The distance between two points in the geometry is the distance of the associated vertices in the collinearity graph. The girth is the length of the shortest cycle. Incidence structures with high girth are often interesting. Orbiter allows classifying incidence structures with a given girth. More specifically, Orbiter classifies all incidence structures whose girth is at least a given number, or proves that there are none. This is interesting because the girth can be used as a filter, and classifying with a given girth may be faster than classifying all incidence geometries. For instance, the following example classifies all cubic graphs on 10 vertices with girth at least 5:

Example 507

```
geo_petersen:
▷  $(ORBITER) -v 8 \
▷  ▷  -define Test_lines -set -loop 3 11 1 -end \
▷  ▷  -define Geo -geometry_builder \
▷  ▷  ▷  -V 10 -B 15 -TDO 3 -fuse 1 \
▷  ▷  ▷  -fname_GEO petersen -girth 5 \
▷  ▷  ▷  -output_to_inc_file \
▷  ▷  ▷  -search_tree \
▷  ▷  ▷  -test Test_lines \
▷  ▷  -end
```

There is a unique graph with these properties. It is the Petersen graph. Its automorphism group is Sym(5) of order 120.

Let us look at the girth of configurations. For instance, while there are 245342 isomorphism classes of configurations 15_3, only one of them has girth 4. This is the Cremona Richmond configuration with automorphism group Sym(6) of order 720. It is associated to a cubic surface. Let us compare the classification with and without girth condition. The following command classifies all configurations 15_3:

Example 508

```
15_3.inc:
▷  $(ORBITER) -v 2 \
▷  ▷  -define Test_lines -set -loop 4 16 1 -end \
▷  ▷  -define Geo -geometry_builder \
▷  ▷  ▷  -V 15 -B 15 -TDO 3 \
▷  ▷  ▷  -fuse 1 -fname_GEO 15_3 \
▷  ▷  ▷  -output_to_inc_file \
▷  ▷  ▷  -test Test_lines \
▷  ▷  ▷  -special_test_not_orderly \
▷  ▷  -end
```

The next command classifies the configurations 15_3 with girth 4.

Example 509

```
geo_15_3_g4:
▷  $(ORBITER) -v 2 \
▷  ▷  -define Test_lines -set -loop 4 16 1 -end \
▷  ▷  -define Geo -geometry_builder \
▷  ▷  ▷  -V 15 -B 15 -TDO 3 \
▷  ▷  ▷  -fuse 1 -fname_GEO 15_3_g4 \
▷  ▷  ▷  -output_to_inc_file \
▷  ▷  ▷  -girth 4 \
▷  ▷  ▷  -search_tree \
▷  ▷  ▷  -special_test_not_orderly \
▷  ▷  ▷  -test Test_lines \
▷  ▷  -end
▷  $(ORBITER) -v 2 \
▷  ▷  -draw_options -embedded -radius 50 \
▷  ▷  ▷  -nodes_empty \
▷  ▷  ▷  -scale 0.5 -line_width 0.3 -end \
▷  ▷  -tree_draw -file 15_3_g4_tree.txt -end
▷  pdflatex 15_3_g4_tree_draw.tex
▷  $(OPEN) 15_3_g4_tree_draw.pdf
```

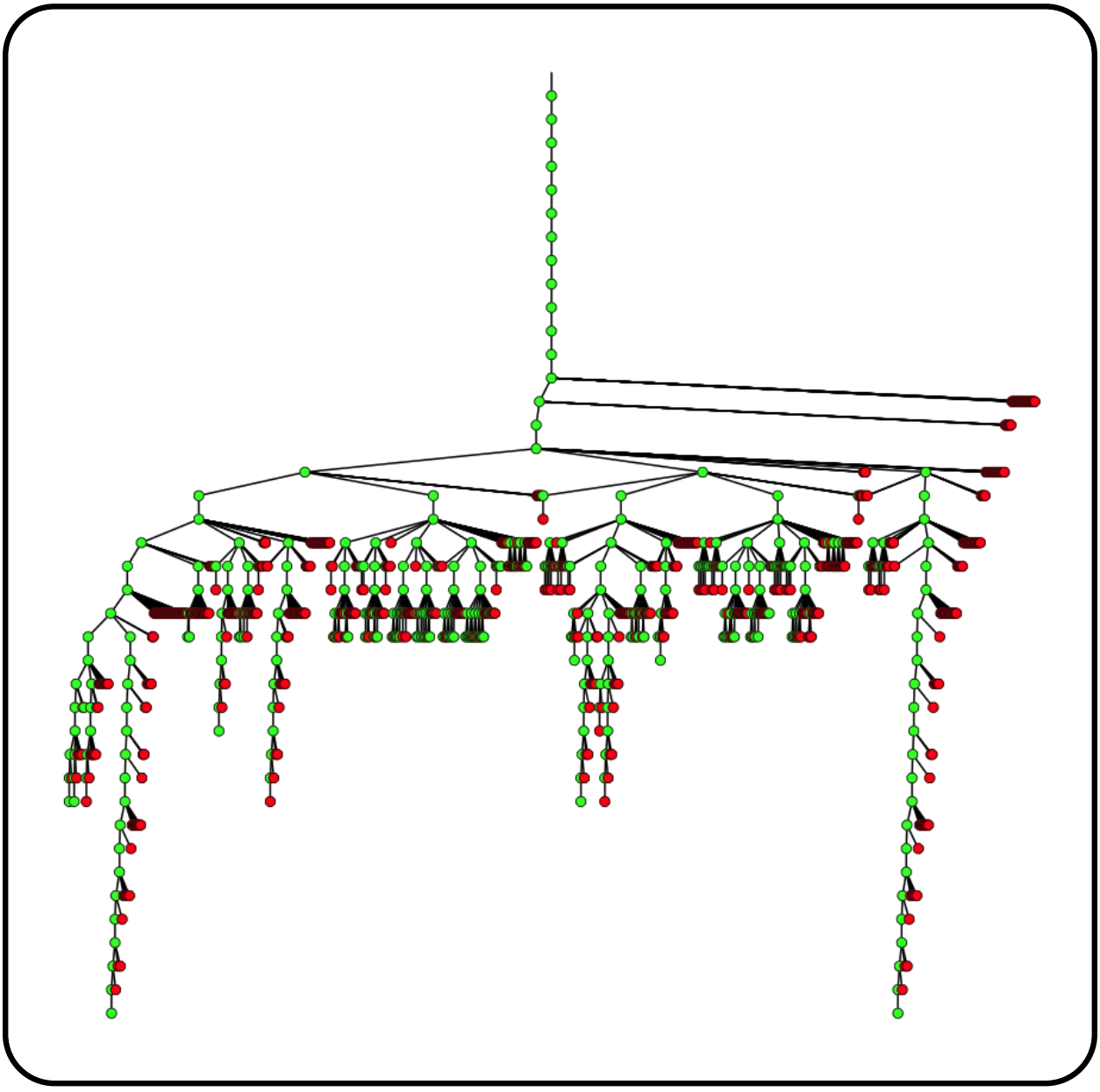
The next command classifies the configurations 40_4 with girth 4. Exactly two configuration arise, both with a group of order 51840. Note the extra option `-special_test_not_orderly` to speed up the search.

Example 510

```
geo_40_4_g4:
▷  $(ORBITER) -v 5 \
▷  ▷  -define Test_lines -set -loop 0 41 1 -end \
▷  ▷  -define Geo -geometry_builder \
▷  ▷  ▷  -V 40 -B 40 -TDO 4 \
▷  ▷  ▷  -fuse 1 \
▷  ▷  ▷  -fname_GEO 40_4_g4 \
▷  ▷  ▷  -girth 4 \
```

```
▷ ▷ ▷ -search_tree \
▷ ▷ ▷ -special_test_not_orderly \
▷ ▷ ▷ -test Test_lines \
▷ ▷ ▷ -output_to_sage_file \
▷ ▷ ▷ -output_to_inc_file \
▷ ▷ -end
▷ $(ORBITER) -v 2 \
▷ ▷ -draw_options -embedded -radius 50 \
▷ ▷ ▷ -xin 10000 -yin 10000 \
▷ ▷ ▷ -xout 1000000 -yout 1000000 \
▷ ▷ ▷ -nodes_empty \
▷ ▷ ▷ -scale 0.5 -line_width 0.3 -end \
▷ ▷ -tree_draw -file 40_4_g4_tree.txt -end
▷ pdflatex 40_4_g4_tree_draw.tex
▷ $(OPEN) 40_4_g4_tree_draw.pdf
```

The search tree is shown below



The next command classifies the configurations 63_3 with girth 6. Exactly two configuration arise, both with a group of order 12096. Note the extra option `-special_test_not_orderly` to speed up the search.

Example 511

```
geo_63_3_g6:
▷  $(ORBITER) -v 2 \
▷  ▷  -define Test_lines -set -loop 4 64 1 -end \
▷  ▷  -define Geo -geometry_builder \
▷  ▷  ▷  -V 63 -B 63 -TDO 3 \
▷  ▷  ▷  -fuse 1 -fname_GEO 63_3_g6 \
▷  ▷  ▷  -girth 6 \
▷  ▷  ▷  -test Test_lines \
```

```
▷ ▷ ▷ -special_test_not_orderly \
▷ ▷ ▷ -output_to_sage_file \
▷ ▷ ▷ -output_to_inc_file \
▷ ▷ ▷ -search_tree \
▷ ▷ -end
```

The configuration is also known as the split Cayley hexagon.

The next command classifies the Latin squares of order 6.

Example 512

```
geo_LSQ6.inc:
▷ $(ORBITER) -v 2 \
▷ ▷ -define Test_lines -set -loop 1 19 1 -end \
▷ ▷ -define Geo -geometry_builder \
▷ ▷ ▷ -V 6,6,6 -B 1,1,1,36 -TDO \
▷ ▷ ▷ "1,0,0,6, 0,1,0,6, 0,0,1,6" \
▷ ▷ ▷ -fuse 3 \
▷ ▷ ▷ -fname_GEO LSQ6 \
▷ ▷ ▷ -output_to_inc_file \
▷ ▷ ▷ -test Test_lines \
▷ ▷ -end
```

Up to isomorphism, there are exactly 12 such objects.

Refining Tactical Decompositions

Command	Arguments	Purpose
-lambda3	λ_3 s	Refine as 3-design with λ_3 and with block-size s
-solution	i fname	Use solutions to system i from file fname.
-range	f l	Refine cases i with $f \leq i < f + l$ only.
-select	label	Select the case for refinement by label.
-o1	s	Omit s variables from the first refinement system.
-o2	s	Omit s variables from the second refinement system.
-D1_upper_bound_x0	b	Add the bound $x_0 \leq b$ in the first refinement.
-reverse		Sort the distributions in reverse order.
-reverse_inverse		
-nopacking		Do not use packing inequalities.
-dual_is_linear_space		Assume that the dual incidence structure is a linear space also. This is valid for projective planes, for instance.
-geometric_test		Subject the distributions to the geometric test.
-once		Find at most one refinement in each case. This can be used to test which cases can be refined.
-mckay		Use McKay's solver instead (by default, a lexicographic solver is used).
-input_file	fname	Specify the input TDO-file for refinement.

Table 11.8: Refining Tactical Decompositions

11.5 Tactical Decompositions

Table 11.8 lists the Orbiter commands for decomposition refinement.

Suppose we want to study projective planes of order 16. It is a linear space with $16^2 + 16 + 1 = 273$ points and equally many lines. Each point lies on 17 lines and each line contains 17 points. Any two points lie on exactly one line and any two lines intersect in exactly one point.

We decide to study maximal arcs of degree 4 in this plane (the degree has to divide the order of the plane). A maximal arc of degree d is a set of points so that each line intersects in either d or zero points. A line which intersects in d points is called a secant. A line which intersects in no point is called an external line. The command

Example 513

```
max_arc_16_4_start:
▷  $(ORBITER) -v 4 -maximal_arc_parameters 16 4
```

creates a decomposition stack for the parameters of the arc and writes the file `max_arc_q16_r4.stack`

```
<HTDO type=pt ptanz=2 btanz=2 fuse=simple>
              221       52
       52      17        0
      221      13        4

  1  1
</HTDO>
```

This is a point-tactical decomposition with 2 point-classes and 2 block-classes. The point classes are associated with the rows. The block-classes are associated with the columns. The first row and column indicates the size of the classes. The entries a_{ij} count the number of blocks in the column class j that are incident with a given point in the ith row class. The fuse information at the bottom (1 1) is a partition of the row classes which indicates the ancestor decomposition which was column tactical. The next step is to convert the stack file to a tdo file. The command

Example 514

```
max_arc_16_4_convert_stack_tdo:
▷  $(ORBITER) -v 4 -convert_stack_to_tdo max_arc_q16_r4.stack
```

does that. It creates the file `max_arc_q16_r4.tdo`. It also prints the decomposition stack:

```
lambda_scheme at level 2 :
is 1 x 1
          | 273_{  1}
======================
273_{  0} |

row_scheme at level 4 :
is 2 x 2
          | 221_{  1} 52_{  2}
==============================
 52_{  0} |        17        0
221_{  3} |        13        4

col_scheme at level 3 :
is 1 x 2
          | 221_{  1} 52_{  2}
==============================
273_{  0} |        17       17
```

Next, we can compute all coarsest column-tactical refinements of the decomposition. To this end, the command

Example 515

```
max_arc_16_4_refine:
▷  $(ORBITER) -v 4 -tdo_refinement \
▷  ▷  -input_file max_arc_q16_r4.tdo -dual_is_linear_space -end
▷
```

```
max_arc_16_4r_print:
▷  $(ORBITER) -v 4 -tdo_print max_arc_q16_r4r.tdo
```

is used. Because the incidence structure is a projective plane, the dual is a linear space also. Hence the option `-dual_is_linear_space` can be used, which is helpful to reduce possibilities. As it turns out, there is exactly one refinement, and it is tactical. The file `max_arc_q16_r4r.tdo` is produced. Note the added letter `r` at the end of the file name (r for refinement). We can use the following command to display the decomposition stack in the file:

Example 516

```
max_arc_16_4r_print:
▷  $(ORBITER) -v 4 -tdo_print max_arc_q16_r4r.tdo
```

This produces the following output:

```
decomposition 0.1:
lambda_scheme at level 2 :
is 1 x 1
          | 273_{  1}
======================
273_{  0} |

row_scheme at level 4 :
is 2 x 2
          | 221_{  1} 52_{  2}
==============================
 52_{  0} |        17        0
221_{  3} |        13        4

col_scheme at level 4 :
is 2 x 2
          | 221_{  1} 52_{  2}
==============================
 52_{  0} |         4        0
221_{  3} |        13       17

extra_col_scheme at level 3 :
is 1 x 2
          | 221_{  1} 52_{  2}
==============================
273_{  0} |        17       17
```

Chapter 12

Graph Theory

12.1 Creating Graphs

Tables 12.1-12.2 list Orbiter commands to create graphs.

Let us consider some examples. The command

Example 517

```
Cycle_graph_13:
▷  $(ORBITER) -v 2 \
▷  ▷  -define Gamma -graph \
▷  ▷  ▷  -cycle 13 \
▷  ▷  -end
```

creates the cycle graph of degree 13.

There are many ways to read graphs from file. One way is by means of an adjacency matrix stored as a csv file. Consider an example. The `-load_csv_no_border` command can be used to create a graph from a csv file containing the adjacency matrix. The following command sequence uses a makefile variable to store the adjacency matrix of a graph. The matrix is then copied into a file and the graph is created from the file. Here is the makefile variable containing the adjacency matrix:

Example 518

```
TRIANGLE_GRAPH="0,1,1\n1,0,1\n1,1,0\n"
```

The following command writes the adjacency matrix to a csv file and then creates the graph from the csv file:

Example 519

```
make_triangle_graph:
▷  echo $(TRIANGLE_GRAPH) >triangle_graph.csv
▷  $(ORBITER) -v 6 \
▷  ▷  -define G -graph \
▷  ▷  ▷  -load_csv_no_border \
▷  ▷  ▷  triangle_graph.csv \
```

Creating Graphs (Part 1)

Command	Arguments	Purpose
-load	filename	Read a graph from file in Orbiter format.
-load_csv_no_border	filename	Read a graph from a csv file. The first row and the first column are part of the adjacency matrix.
-load_adjacency_matrix_from_csv_and_select_value	filename v	Read a graph from a csv file. The first row and the first column are ignored. Entries equal to v define adjacencies. All other values are ignored.
-load_dimacs	filename	Read a graph from file in dimacs format.
-edge_list	n list-of-edges	Create a graph on n vertices from a list of edges as ranked pairs.
-edges_as_pairs	n edges-as-pairs	Create a graph on n vertices from a list of edges as pairs.
-cycle	n	Cycle graph on n vertices.
-inversion_graph	text	Read inversion graph.
-Hamming	n q	Hamming graph $H(n, q)$
-Johnson	n k s	Johnson graph
-Paley	F	Paley graph over the field $F = \mathbb{F}_q$.
-Sarnak	p q	Lubotzky-Phillips-Sarnak graph [50]
-Schlaefli	F	Schlaefli graph
-Shrikhande		Shrikhande graph
-Winnie_Li	F i	Winnie-Li graph (see [49]) over the field $F = \mathbb{F}_q$ with subfield index i.
-Grassmann	n k F r	Grassmann graph over $F = \mathbb{F}_q$.
-coll_orthogonal	O S	Collinearity graph of the set S in the orthogonal geometry O.
-trihedral_pair_disjointness_graph		Trihedral pair disjointness graph.
-non_attacking_queens_graph	n	Create the graph for non-attacking queens on a $n \times n$ chess board.

Table 12.1: Creating Graphs (Part 1)

Creating Graphs (Part 2)

Command	Arguments	Purpose
-subset	label labeltex subset	Define vertex coloring with two colors based on a subset of vertices.
-disjoint_sets_graph	fname	Reads a set of sets from the given file. Define a graph on the sets. Vertices correspond to the given sets. Two vertices are adjacent if the associated sets are disjoint.
-orbital_graph	G i	Define orbital graph from the i-th orbit of the group G acting on pairs.
-collinearity_graph	inc-matrix	Collinearity graph of the given incidence matrix.
-chain_graph	P1 P2	Chain graph with respect to the partitions P1 and P2.
-Cayley_graph	G gens	Cayley graph with respect to group G and generating set gens.

Table 12.2: Creating Graphs (Part 2)

```
▷  ▷  -end
```

This will create the three-cycle graph.

The command

Example 520

```
Chain_232:
▷  $(ORBITER) -v 2 \
▷  ▷  -define P1 -vector -dense 2,3,2 -end \
▷  ▷  -define P2 -vector -dense 2,3,2 -end \
▷  ▷  -define Gamma -graph \
▷  ▷  ▷  -chain_graph P1 P2 \
▷  ▷  -end
```

creates the chain graph with respect to the partitions $(2, 3, 2)$ and $(2, 3, 2)$.

The command

Example 521

```
Paley_13_graph:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 13 -end \
▷  ▷  -define Gamma -graph -Paley F -end \
```

creates the Paley graph on 13 vertices. The command first creates the field $\mathbb{F}_{13}$ and then the associated Payley graph.

The command

Example 522

```
Winnie_Li_graph_q16_index2:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 16 -end \
▷  ▷  -define Gamma -graph -Winnie_Li F 2 -end \
```

creates the Winnie Li graph [49] on 16 vertices with index 2. The command first creates the field $\mathbb{F}_{16}$ and then the associated graph.

The command

Example 523

```
Sarnak_graph_29_5:
▷  $(ORBITER) -v 2 \
▷  ▷  -define Gamma -graph -Sarnak 29 5 -end \
```

creates the graph $X_{29,5}$ from [50] on 60 vertices.

The command

Example 524

```
trihedral_pair_graph:
▷  $(ORBITER) -v 2 \
▷  ▷  -define Gamma \
▷  ▷  ▷  -graph -trihedral_pair_disjointness_graph \
▷  ▷  -end
```

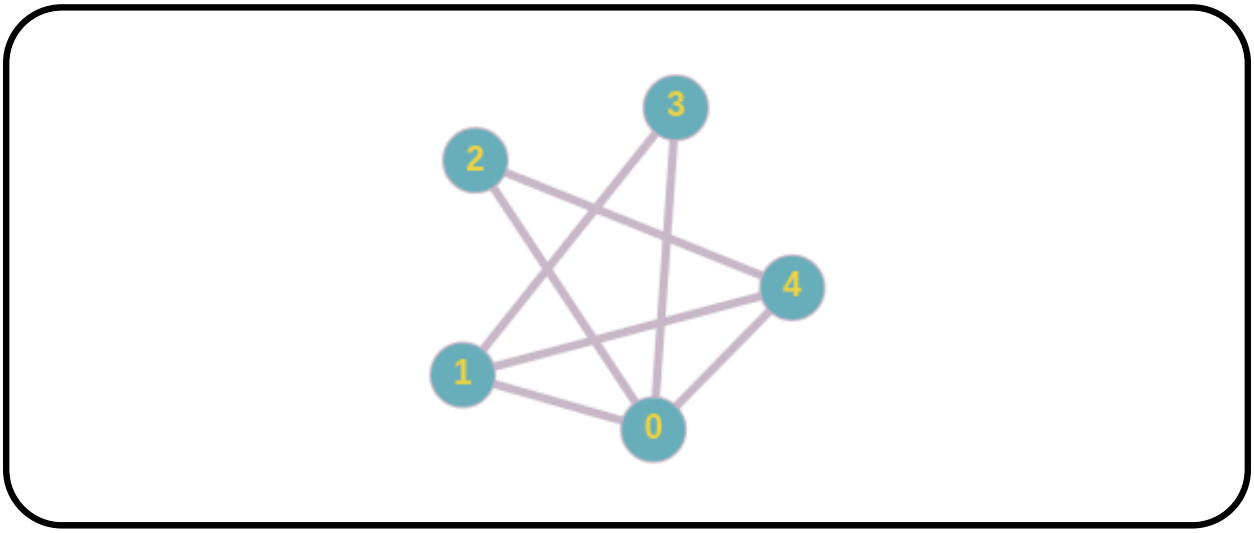
creates the graph of trihedral pairs in a general cubic surface with 27 lines. Two vertices are adjacent if the associated trihedral pairs are line-disjoint.

The command

Example 525

```
small_graph:
▷  $(ORBITER) -v 2 \
▷  ▷  -define G -graph -edges_as_pairs \
▷  ▷  ▷  5 "0,1,0,2,0,3,0,4,1,3,1,4,2,4" \
▷  ▷  -end
```

creates a graph by listing the edges in pairs. The following graph is created:



The command

Example 526

```
petersen:
▷  $(ORBITER) -v 2 \
▷  ▷  -define G -graph -Johnson 5 2 0 -end
```

creates the Petersen graph.

The command

Example 527

```
Johnson_6_2_0:
▷  $(ORBITER) -v 2 \
▷  ▷  -define G -graph -Johnson 6 2 0 -end
```

creates the Johnson graph $J(6, 2, 0)$.

The command

Example 528

```
Hamming_graph_3:
▷  $(ORBITER) -v 2 \
▷  ▷  -define G -graph -Hamming 3 2 -end
```

creates the Hamming graph of order 3.

The command

Example 529

```
Op_6_2_coll_graph:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 13 -end \
▷  ▷  -define O -orthogonal_space 1 6 F -end \
▷  ▷  -define v -vector -loop 0 35 1 -end \
▷  ▷  -define Gamma -graph -coll_orthogonal O v -end \
▷  ▷  -with Gamma -do \
▷  ▷  ▷  -graph_theoretic_activity -save \
▷  ▷  -end
```

creates the collinearity graph of the orthogonal space $O^+(6,2)$.

There is a graph on 315 vertices that arises from the Cohen-Tits near octagon (see [17]). The graph was first constructed in [20] and has automorphism group equal to Aut(HJ), the automorphism group of the Higman-Sims sporadic simple group. The graph is distance-regular. An incidence matrix can be found in Ascii format on the web site [7]. In the following, we assume that a file `halljanko315.csv` is present, containing the incidence matrix of the graph. The following command creates the graph from the file:

Example 530

```
HJ_graph:
▷  cp $(MY_PATH)/examples/users_guide/halljanko315.csv .
▷  $(ORBITER) -v 6 \
▷  ▷  -define G -graph \
▷  ▷  ▷  -load_csv_no_border \
▷  ▷  ▷  halljanko315.csv \
▷  ▷  -end
```

In Section 16.7, we will compute the automorphism group of the graph (of order 1209600). This will create a file `halljanko315_gens.csv` which we use here to create an orbital graph. An orbital graph is a graph whose adjacency matrix corresponds to an orbit of a permutation group in the action on pairs. The group is the automorphism group of the graph. The following command creates the third orbital graph:

Example 531

```
Group_Perm315_Orbital_3.colored_graph: halljanko315_gens.csv
▷  $(ORBITER) -v 2 \
▷  ▷  -define gens -vector -format 5 -file \
▷  ▷  ▷  halljanko315_gens.csv -end \
▷  ▷  -define G -permutation_group \
▷  ▷  ▷  -bsgs halljanko315 "File\_halljanko315" \
▷  ▷  ▷  315 1209600 "0,1,2" 5 gens \
▷  ▷  -end \
▷  ▷  -define Gamma -graph \
▷  ▷  ▷  -orbital_graph G 3 \
▷  ▷  -end \
▷  ▷  -with Gamma -do \
▷  ▷  ▷  -graph_theoretic_activity -save \
▷  ▷  -end
```

Graph Modifiers

Command	Arguments	Purpose
-complement		Complementary graph.
-distance_2		Distance two graph: Two vertices are adjacent if and only if they have distance two in the original graph.

Table 12.3: Graph Modifiers

Table 12.3 shows some Orbiter commands to modify graphs. The commands replace the given graph by the graph obtained after applying the specified modification.

For a graph Γ, the distance 2 graph Δ has the same vertices as Γ, and two vertices in Δ are adjacent if and only if the distance in Γ is two. The following command creates the distance 2 graph of the Cohen-Tits graph.

Example 532

```
HJ_d2_graph:
▷ $(ORBITER) -v 6 \
▷ ▷ -define G -graph \
▷ ▷ ▷ -load_csv_no_border \
▷ ▷ ▷ halljanko315.csv \
▷ ▷ ▷ -distance_2 \
▷ ▷ -end
```

Let us look at some examples of Cayley graphs. The first graph has $G = \mathbb{Z}_{11}$ and connection set all elements congruent 1 mod 3. We create the group as a subgroup of the one-dimensional affine group over $\mathbb{F}_{11}$. This means that the map $x \mapsto ax + b$ mod 11 is encoded as a pair (a, b).

Example 533

```
Cayley_Z11_1mod3:
▷ $(ORBITER) -v 2 \
▷ ▷ -define F -finite_field -q 11 -end \
▷ ▷ -define S -vector -dense \
▷ ▷ ▷ "1,1, 1,4, 1,7, 1,10" -end \
▷ ▷ -define G -linear_group -AGL 1 F \
▷ ▷ ▷ -subgroup_by_generators "Z11" 11 1 "1,1" \
▷ ▷ -end \
▷ ▷ -define Gamma -graph \
▷ ▷ ▷ -Cayley_graph G S \
▷ ▷ -end
```

The vertices of the Cayley graph are ordered. The ordering is determined by the stabilizer chain. This is a total ordering.

In the following example, we create a Cayley graph based on the symmetric group on 4 things. We take the Coxeter generators as connection set:

Example 534

```
Cayley_Sym4_coxeter:
▷  $(ORBITER) -v 2 \
▷  ▷  -define S -vector -dense "1,0,2,3, 0,2,1,3, 0,1,3,2" -end \
▷  ▷  -define G -permutation_group -symmetric_group 4 \
▷  ▷  -end \
▷  ▷  -define Gamma -graph \
▷  ▷  ▷  -Cayley_graph G S \
▷  ▷  -end
```

The star graph is another Cayley graph for the symmetric group. The connection set is given by the permutations $(0, i)$ for $i = 1, \ldots, n-1$. The next example creates the star graph on 4 vertices:

Example 535

```
Cayley_Sym4_star:
▷  $(ORBITER) -v 2 \
▷  ▷  -define S -vector -dense "1,0,2,3, 2,1,0,3, 3,1,2,0" -end \
▷  ▷  -define G -permutation_group -symmetric_group 4 \
▷  ▷  -end \
▷  ▷  -define Gamma -graph \
▷  ▷  ▷  -Cayley_graph G S \
▷  ▷  -end
```

Graph Theoretic Activities

Command	Arguments	Purpose
-find_cliques	options	Find all cliques. See Section 15.1.
-export_magma		Export to Magma [16].
-export_maple		Export to Maple [53].
-export_csv		Export to csv-file.
-export_graphviz		Export to graphviz-file.
-print		Print the graph.
-sort_by_colors		Sort the vertices by color classes.
-split	file	Split the graph into subgraphs.
-split_by_starters	file	Split the graph into subgraphs according to starters.
-split_by_clique	label clique	Compute the neighborhood graph of the given clique.
-save		Save the graph to file in binary format.
-automorphism_group		Compute the automorphism group and write a report. See Section 16.7.
-properties		Compute properties of the graph.
-eigenvalues		Compute the eigenvalues of the graph.
-draw		Draw the graph.

Table 12.4: Graph Theoretic Activities

12.2 Graph Theoretic Activities

Graph theoretic activities allow us to perform tasks on graphs. Table 12.4 shows the commands for graph theoretic activities. These are activities that can be applied to objects of type graph.

Let us continue the example of the three-cycle from Section 12.1. The command

Example 536

```
triangle_graph_properties:
▷  echo $(TRIANGLE_GRAPH) >triangle_graph.csv
▷  $(ORBITER) -v 6 \
▷  ▷  -define G -graph \
▷  ▷  ▷  -load_csv_no_border \
▷  ▷  ▷  triangle_graph.csv \
▷  ▷  -end \
▷  ▷  -with G -do \
▷  ▷  ▷  -graph_theoretic_activity -properties \
▷  ▷  -end
```

computes the degree type of the graph. This is the distribution of degrees in the degree sequence of the graph. It prints the distribution of degree values in exponential notation. The multiplicities are indicated as exponent. For instance, the graph in this example has three vertices of degree 2, so the degree type is written as 2^3.

The following command exports the adjacency matrix and creates a bitmap drawing of it.

Example 537

```
Cycle_13_draw:
▷  $(ORBITER) -v 2 \
▷  ▷  -define Gamma -graph -cycle 13 -end \
▷  ▷  -with Gamma -do \
▷  ▷  -graph_theoretic_activity -export_csv -end \
▷  ▷  -with Gamma -do \
▷  ▷  -graph_theoretic_activity -export_graphviz -end
▷  $(ORBITER) -v 2 -draw_matrix \
▷  ▷  -input_csv_file Cycle_13.csv \
▷  ▷  -box_width 20 -bit_depth 8 -partition 4 13 13 -end
▷  dot -Tpng Cycle_13.gv >Cycle_13.png
▷  #twopi -Tpng Cycle_13.gv >Cycle_13.png
▷  #$(OPEN) Cycle_13_draw.bmp
▷  #pdflatex Cycle_13_report.tex
▷  #$(OPEN) Cycle_13_report.pdf
```

The command first creates the cycle graph of order 13, and then exports the adjacency matrix as csv file. It then draws the adjacency matrix as a bitmap graphics.

Suppose we want to compute the eigenvalues of the adjacency matrix of a graph. In the following example, the command `-eigenvalues` is used to compute the eigenvalues (both regular and Laplace) of the 9-cycle:

Example 538

```
Cycle_9_eigenvalues:
▷  $(ORBITER) -v 2 \
▷  ▷  -define Gamma -graph \
▷  ▷  ▷  -cycle 9 \
▷  ▷  -end \
▷  ▷  -with Gamma -do \
▷  ▷  -graph_theoretic_activity -eigenvalues -end
▷  pdflatex Cycle_9_eigenvalues.tex
▷  $(OPEN) Cycle_9_eigenvalues.pdf
```

The following output is produced:

i	λ_i	θ_i
0	2	3.87939
1	1.53209	3.87939
2	1.53209	3
3	0.347296	3
4	0.347296	1.6527
5	-1	1.6527
6	-1	0.467911
7	-1.87939	0.467911
8	-1.87939	$-2.26243e-16$

The energy is 11.5175
Eigenvalues: λ_i
Laplace eigenvalues: θ_i

The command

Example 539

```
Paley_13_draw:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 13 -end \
▷  ▷  -define Gamma -graph -Paley F -end \
▷  ▷  -with Gamma -do \
▷  ▷  -graph_theoretic_activity -export_csv -end \
▷  ▷  -with Gamma -do \
▷  ▷  -graph_theoretic_activity -export_graphviz -end
▷  $(ORBITER) -v 2 -draw_matrix \
▷  ▷  -input_csv_file Paley_13.csv \
▷  ▷  -box_width 20 -bit_depth 8 -partition 4 13 13 -end
▷  dot -Tpng Paley_13.gv >Paley_13.png
▷  $(OPEN) Paley_13_draw.bmp
```

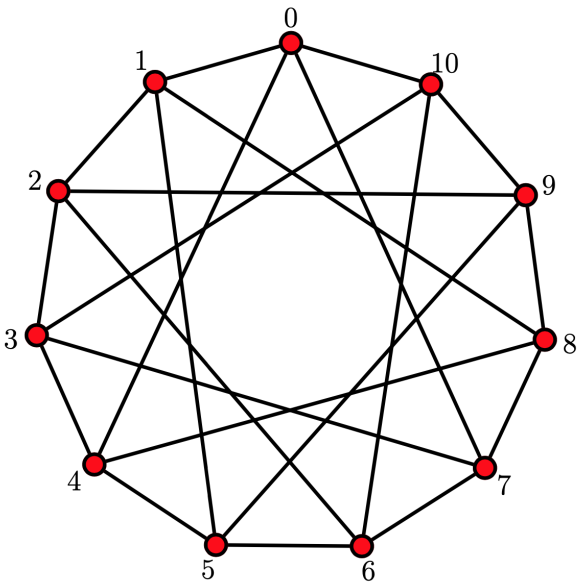
draws the Paley graph of order 13 created in Section 12.1 using the external tool graphviz.

Let us consider the Cayley graphs from Section 12.1. Here is a command that draws the first graph and computes the eigenvalues:

Example 540

```
Cayley_Z11_1mod3_eigenvalues_and_draw:
▷  $(ORBITER) -v 2 \
▷  ▷  -draw_options -xin 2000000 \
▷  ▷  ▷  -yin 2000000 -embedded -radius 20000 -end \
▷  ▷  -define F -finite_field -q 11 -end \
▷  ▷  -define S -vector -dense \
▷  ▷  ▷  "1,1, 1,4, 1,7, 1,10" -end \
▷  ▷  -define G -linear_group -AGL 1 F \
▷  ▷  ▷  -subgroup_by_generators "Z11" 11 1 "1,1" \
▷  ▷  -end \
▷  ▷  -define Gamma -graph \
▷  ▷  ▷  -Cayley_graph G S \
▷  ▷  -end \
▷  ▷  -with Gamma -do \
▷  ▷  -graph_theoretic_activity -eigenvalues -end \
▷  ▷  -with Gamma -do \
▷  ▷  -graph_theoretic_activity -draw -end
▷  pdflatex Cayley_graph_AGL_1_11_draw.tex
▷  $(OPEN) Cayley_graph_AGL_1_11_draw.pdf
```

The drawing is shown below

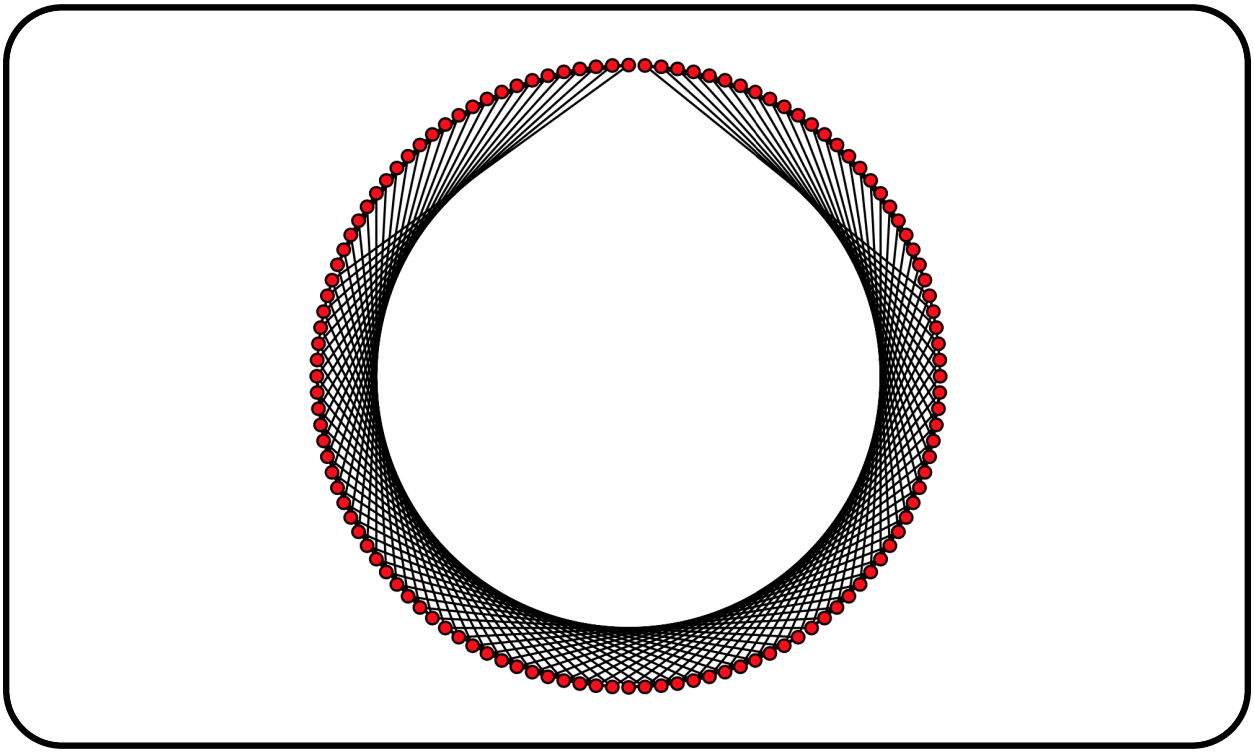


Let us draw the Cayley graph of Sym(5) with respect to the Coxeter generators:

Example 541

```
Cayley_Sym5_coxeter_draw:
▷  $(ORBITER) -v 2 \
▷  ▷  -draw_options -xin 1000000 -yin 1000000 \
▷  ▷  ▷  -embedded -radius 10000 -nodes_empty -end \
▷  ▷  -define S -vector -dense \
▷  ▷  ▷  "1,0,2,3,4, 0,2,1,3,4, 0,1,3,2,4, 0,1,2,4,3" -end \
▷  ▷  -define G -permutation_group -symmetric_group 5 \
▷  ▷  -end \
▷  ▷  -define Gamma -graph \
▷  ▷  ▷  -Cayley_graph G S \
▷  ▷  -end \
▷  ▷  -with Gamma -do \
▷  ▷  -graph_theoretic_activity -draw -end
▷  pdflatex Cayley_graph_Sym_5_draw.tex
▷  $(OPEN) Cayley_graph_Sym_5_draw.pdf
```

The drawing is shown below



It is possible to create the collinearity graph of an incidence structure. The incidence structure must be encoded by means of an incidence matrix. Let us continue an example from Section 4.7, where the incidence matrix of $Q(4,2)$ was created. At that time, we wrote the incidence matrix to file. Here, we read the incidence matrix from file and create the collinearity graph of it:

Example 542

```
PGO_5_2_collinearity_graph: O_5_2_incidence_matrix.csv
▷  $(ORBITER) -v 3 \
▷  ▷  -define Inc -vector -file O_5_2_incidence_matrix.csv -end \
▷  ▷  -define Gamma -graph -collinearity_graph Inc -end \
▷  ▷  -with Gamma -do \
▷  ▷  -graph_theoretic_activity \
▷  ▷  ▷  -properties \
▷  ▷  -end
```

The command also computes properties of the graph. The graph has 15 vertices of degree 6. This is because in the geometry, each point lies on three lines, and hence is collinar with 6 other points.

Classifying Graphs		
Command	Arguments	Purpose
-regular	r	Regular of degree r
-n	n	Number of vertices n
-girth	d	Girth at least d
-tournament		Classify tournaments.
-no_transmitter		Tournament without transmitter (requires -tournament)
-test_multi_edge		
-identify		
-depth		

Table 12.5: Classifying Graphs

Activities after Graph Classification		
Command	Arguments	Purpose
-draw_level_graph	k	
-draw_graphs		
-list_graphs_at_level	min max	
-draw_graphs_at_level	k	
-draw_options		
-recognize_graphs_from_ adjacency_matrix_csv		

Table 12.6: Activities after Graph Classification

12.3 Classification of Graphs and Tournaments

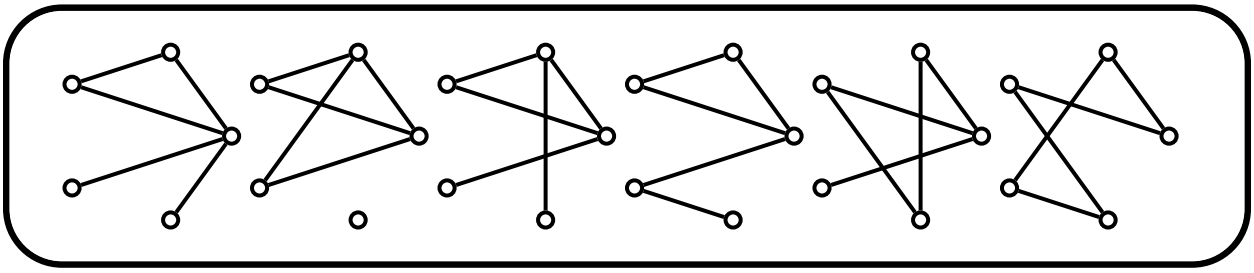
Table 12.5 lists the Orbiter commands to classify graphs and tournaments. Table 12.6 lists activities after the classification has been completed.
The following command classifies all graphs on 5 vertices:

Example 543

```
graph_classify_5:
▷  $(ORBITER) -v 2 \
▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
▷  ▷  -define GC -graph_classification \
▷  ▷  ▷  -n 5 \
▷  ▷  ▷  -poset_classification_control \
▷  ▷  ▷  ▷  -problem_label graphs_v5 \
▷  ▷  ▷  ▷  -depth 10 \
▷  ▷  ▷  ▷  -draw_options -radius 250 \
▷  ▷  ▷  ▷  -embedded \
▷  ▷  ▷  ▷  -end \
▷  ▷  ▷  -end \
▷  ▷  -end \
▷  ▷  -with GC -do \
```

```
▷  ▷  -graph_classification_activity \
▷  ▷  ▷  -list_graphs_at_level 5 5 \
▷  ▷  -end \
▷  ▷  -with GC -do \
▷  ▷  -graph_classification_activity \
▷  ▷  ▷  -draw_options \
▷  ▷  ▷  ▷  -radius 300 -nodes_empty \
▷  ▷  ▷  ▷  -line_width 1.5  \
▷  ▷  ▷  ▷  -scale 0.1 \
▷  ▷  ▷  -end \
▷  ▷  ▷  -draw_graphs_at_level 5 \
▷  ▷  -end \
▷  ▷  -print_symbols
▷  pdflatex graphs_v5_level_5_reps.tex
▷  $(OPEN) graphs_v5_level_5_reps.pdf
▷  #pdflatex graphs_v5_poset.tex
▷  #$(OPEN) graphs_v5_poset.pdf
```

After the classification, the graphs with 5 edges are shown.



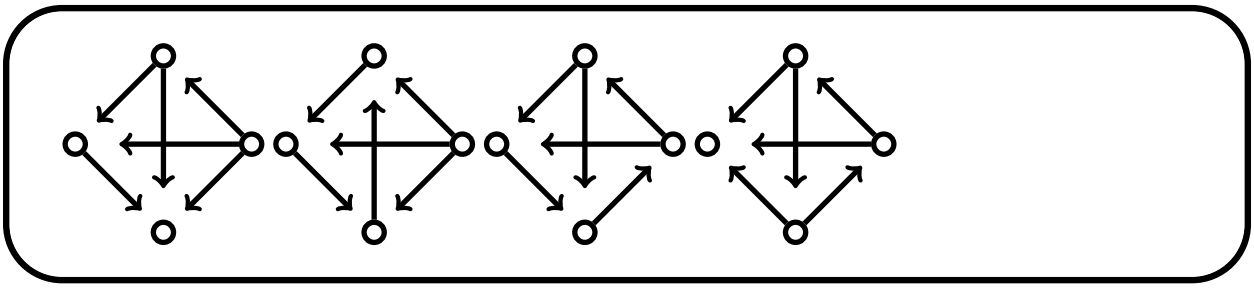
The next command classifies all tournaments on 4 vertices:

Example 544

```
tournament_classify_4:
▷  $(ORBITER) -v 2 \
▷  ▷  -define GC -graph_classification \
▷  ▷  ▷  -n 4 -tournament \
▷  ▷  ▷  -poset_classification_control \
▷  ▷  ▷  ▷  -problem_label tournament_4 \
▷  ▷  ▷  ▷  -depth 6 \
▷  ▷  ▷  ▷  -draw_options \
▷  ▷  ▷  ▷  ▷  -radius 250 -embedded \
▷  ▷  ▷  ▷  -end \
▷  ▷  ▷  -end \
▷  ▷  -end \
▷  ▷  -with GC -do \
▷  ▷  -graph_classification_activity \
▷  ▷  ▷  -draw_options \
▷  ▷  ▷  ▷  -radius 400 \
▷  ▷  ▷  ▷  -line_width 2 -scale 0.10 \
▷  ▷  ▷  -end \
```

```
▷  ▷  ▷  -draw_graphs_at_level 6 \
▷  ▷  -end \
▷  ▷  -print_symbols
▷  pdflatex tournament_4_level_6_reps.tex
▷  $(OPEN) tournament_4_level_6_reps.pdf
▷
```

There are four tournaments, shown below:

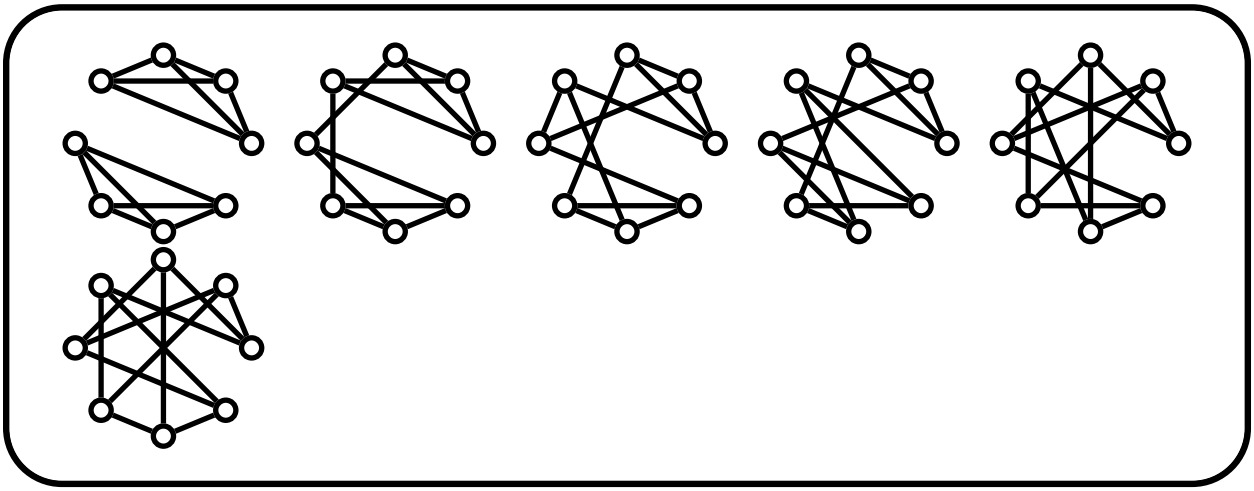


The next command classifies all cubic graphs on 8 vertices:

Example 545

```
graph_classify_8_r3:
▷  $(ORBITER) -v 3 \
▷  ▷  -define GC -graph_classification \
▷  ▷  ▷  -n 8 -regular 3 \
▷  ▷  ▷  -poset_classification_control \
▷  ▷  ▷  ▷  -problem_label graphs_v8_r3 \
▷  ▷  ▷  ▷  -depth 12 \
▷  ▷  ▷  ▷  -draw_options -radius 250 \
▷  ▷  ▷  ▷  ▷  -line_width 0.2 -embedded \
▷  ▷  ▷  ▷  -end \
▷  ▷  ▷  -end \
▷  ▷  -end \
▷  ▷  -with GC -do \
▷  ▷  -graph_classification_activity \
▷  ▷  ▷  -draw_options \
▷  ▷  ▷  ▷  -radius 400 \
▷  ▷  ▷  ▷  -line_width 2 -scale 0.10 \
▷  ▷  ▷  -end \
▷  ▷  ▷  -draw_graphs_at_level 12 \
▷  ▷  -end \
▷  ▷  -print_symbols
▷  #pdflatex graphs_v8_r3_poset_lvl_12.tex
▷  #$(OPEN) graphs_v8_r3_poset_lvl_12.pdf
```

There are six cubic graphs. The graph drawings are shown below:



Chapter 13

Design Theory

13.1 Creating Designs

A design is an incidence structure of points and blocks. The incidence matrix of a design has rows corresponding to the points and columns corresponding to the blocks. An entry in a certain row and column is one if and only if the point associated with the row is contaned in the block associated with the column, zero otherwise. A decomposition of the design is a partition of the points and blocks such that each class consists either exclusively of points or exclusively of blocks.

A decomposition is point-tactical if for all points, the number of incident lines in the jth block class depends only on the class of the point. It the point belongs to class i, this number is denoted as a_{ij}. A decomposition is block-tactical if for all blocks, the number of incident points in the ith point class depends only on the class of the block. If the block belongs to class j, this number is denoted as b_{ij}.

A projective plane of order n is a design with n^2+n+1 points and equally many blocks (also called lines), each of size $n+1$ such that any two points lie in exactly one block and any two blocks have exactly one point in common. Projective planes are known to exist for all $n = q$ which are a power of a prime. This follows from a construction which utilizes the projective geometry $\mathrm{PG}(2, q)$. Points are the one-dimensional subspaces of $\mathbb{F}_q^3$, blocks are the two-dimensional subspaces of $\mathbb{F}_q^3$, and incidence is natural (inclusion of subspaces). The automorphism group of this design is the collineation group of the projective space. Projective planes other than these exist, though none are known when n is not a prime power. The number of lines through a point equals the number of points on a line. The fact that these numbers exist imply that there is a tactical decomposition. Namely, the trivial decomposition with two classes, one containing all points and one containing all lines. The structure constants of the decomposition are the numbers just described.

Table 13.1 lists Orbiter commands to create designs. Table 13.2 lists Orbiter activities for designs.
The command

Example 546

```
design_PG_2_3:
▷  $(ORBITER) -v 8 \
▷  ▷  -define F -finite_field -q 3 -end \
▷  ▷  -define D -design -field F -family PG_2_q -end \
▷  ▷  -with D -do \
▷  ▷  ▷  -design_activity \
▷  ▷  ▷  ▷  -export_inc \
▷  ▷  -end \
▷  ▷  -with D -do \
▷  ▷  ▷  -design_activity \
```

Creating Designs

Command	Arguments	Purpose
-field	F	Some constructions of designs require a finite field. Set the field to be F.
-catalogue	iso	Recall a design of a given family from the Orbiter catalogue. The parameter iso is the isomorphism type in the catalogue. It is zero-based.
-family	family	Select the family name of the design to be constructed. The following family names are supported: PG_2_q for the projective plane $\mathrm{PG}(2, q)$.
-list_of_base_blocks	G fname col	Creates a design by spanning the orbits of the base blocks given in column col in file fname under the group G. See Section 13.4.
-list_of_blocks_coded	v k L	Create a design on v points and with block size k from the coded list of sets in L.
-list_of_sets_coded	v label	Create a design on v points and with block codes in the list in L.
-list_of_blocks_coded_from_file	v k file	Create a design on v points and with block size k from the coded list of sets read from the given file.
-list_of_blocks_from_file	v fname	Create a design on v points and with block codes in the given file.
-wreath_product_designs	n k	Create a class of designs with wreath product action.
-linear_space_from_latin_square	matrix	Create a linear space associated with a Latin Square given as matrix. The matrix must be an Orbiter object of type vector.
-no_group		

Table 13.1: Creating Designs

Design Activities

Command	Arguments	Purpose
-load_table	label group H_label H_group_order H_gens l	
-canonical_form	options	
-extract_solutions_by_index_csv	label group fname-solutions-in fname-solutions-out prefix	
-extract_solutions_by_index_txt	label group fname-solutions-in fname-solutions-out prefix	
-export_inc		
-intersection_matrix		
-export_blocks		
-row_sums		
-tactical_decomposition		

Table 13.2: Design Activities

```
▷ ▷ ▷ ▷ -tactical_decomposition \
▷ ▷ -end
```

creates the design PG(2, 3).

We have created the following design:

$$\{19, 79, 126, 219, 256, 284, 371, 392, 465, 541, 619, 627, 653\}$$

The stabilizer is generated by:
Strong generators for a group of order 5616:

$$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix},$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

1,0,0,0,2,0,0,0,2,
1,0,0,0,2,0,0,0,1,
1,0,0,0,1,0,1,0,1,
1,0,0,0,1,0,0,1,1,
1,0,0,0,0,1,0,1,0,
0,1,0,1,0,0,0,0,1,

The blocks of the design are encoded in the lexicographic ordering of k-subsets (here $k = 4$). The program also displays the tactical decomposition scheme of the design, which is

$\rightarrow$	13_1
13_0	4

$\downarrow$	13_1
13_0	4

In Section 16.4, we will show how to compute further properties of the design.

The command

Example 547

wreath_product_designs_n4_k2_inc.txt:

```
▷ $(ORBITER) -v 8 \
▷ ▷ -define D -design -wreath_product_designs 4 2 -end \
▷ ▷ -with D -do \
▷ ▷ ▷ -design_activity \
▷ ▷ ▷ ▷ -export_inc \
▷ ▷ -end
```

creates a design on 8 points invariant under the wreath product Sym(4) ≀ Sym(2). The design has 12 blocks of size 4. The command

Example 548

```
wreath_product_designs_n8_k6_inc.txt:
▷  $(ORBITER) -v 8 \
▷  ▷  -define D -design -wreath_product_designs 8 6 -end \
▷  ▷  -with D -do \
▷  ▷  ▷  -design_activity \
▷  ▷  ▷  ▷  -export_inc \
▷  ▷  -end
```

creates a design on 16 points invariant under the wreath product Sym(8) ≀ Sym(2). The design has 3920 blocks of size 6. We will compute the automorphism groups of these two designs in Section 16.3.

A design can be created by listing the blocks in coded fashion. The codes correspond to the ranking of k-subsets of the v set which is the underlying set of the design. The following command creates a configuration $15_4 20_3$ from its set of blocks.

Example 549

```
design_20:
▷  $(ORBITER) -v 8 \
▷  ▷  -define D -design -list_of_blocks_coded 15 3 \
▷  ▷  "61, 67, 72, 76, 129, 147, 152, 156, 197, 204, 215, 224, 249, 261, 267, 276,
296, 303, 309, 319" \
▷  ▷  -end \
▷  ▷  -with D -do \
▷  ▷  ▷  -design_activity \
▷  ▷  ▷  ▷  -export_inc \
▷  ▷  -end
```

It is also possible to create a design by listing the blocks as sets. Here is an example. At first, we create a makefile variable for a csv file containing the blocks:

Example 550

```
GEO_BLOCKS_600="Row,C0,C1,C2\n\
0, 0, 5, 14, \n\
1, 1, 6, 10, \n\
2, 2, 7, 11, \n\
3, 3, 8, 12, \n\
4, 4, 9, 13, \n\
5, 0, 6, 13, \n\
6, 1, 7, 14, \n\
7, 2, 8, 10, \n\
8, 3, 9, 11, \n\
9, 4, 5, 12, \n\
10, 0, 7, 12, \n\
11, 1, 8, 13, \n\
```

```
12, 2, 9, 14, \n\
13, 3, 5, 10, \n\
14, 4, 6, 11, \n\
15, 0, 8, 11, \n\
16, 1, 9, 12, \n\
17, 2, 5, 13, \n\
18, 3, 6, 14, \n\
19, 4, 7, 10, \n\
20, 0, 9, 10, \n\
21, 1, 5, 11, \n\
22, 2, 6, 12, \n\
23, 3, 7, 13, \n\
24, 4, 8, 14, \n\
END\n"
```

The following command creates the csv file from the makefile variable using the unix echo command. The sed command is necessary to remove space characters that are inserted by the echo command. The Orbiter command then reads the list of blocks from the csv file. After that, it creates the design from the list of blocks. As we will see below, the design has an automorphism group of order 600 and is related to the Latin Square of order 5 associated with the cyclic group of order 5.

Example 551

```
design_600:
▷  echo $(GEO_BLOCKS_600) | sed 's/ //' >geo_600_blocks.csv
▷  $(ORBITER) -v 8 \
▷  ▷  -define D -design -list_of_blocks_from_file \
▷  ▷  ▷  15 geo_600_blocks.csv \
▷  ▷  -end \
▷  ▷  -with D -do \
▷  ▷  ▷  -design_activity \
▷  ▷  ▷  ▷  -export_inc \
▷  ▷  -end \
▷  ▷  -with D -do \
▷  ▷  ▷  -design_activity \
▷  ▷  ▷  ▷  -intersection_matrix \
▷  ▷  -end
```

The command also computes the intersection matrix of the design. The design is a transversal design. The configuration graph is three copies of the complete graph K_5.

How can we convince ourselves that the design created previously is indeed associated to a certain Latin Square? There are exactly two Latin Squares of order 5. One is the Cayley table of the cyclic group of order 5, the other is the multiplication table of the non-associative loop of order 5, see Table 13.3.
We create makefile variables for the two tables:

Example 552

```
LSQ_5A_TABLE="0,1,2,3,4,1,2,3,4,0,2,3,4,0,1,3,4,0,1,2,4,0,1,2,3"
```

Latin Squares of Order 5

Cyclic 5

0	1	2	3	4
1	2	3	4	0
2	3	4	0	1
3	4	0	1	2
4	0	1	2	3

Nonassociative Loop

0	1	2	3	4
1	0	4	2	3
2	4	3	1	0
3	2	0	4	1
4	3	1	0	2

Table 13.3: Latin Squares of Order 5

Example 553

```
LSQ_5B_TABLE="0,1,2,3,4,1,0,4,2,3,2,4,3,1,0,3,2,0,4,1,4,3,1,0,2"
```

The next two commands create the associated linear spaces:

Example 554

```
LSQ_5A:
▷  $(ORBITER) -v 8 \
▷  ▷  -define LSQ5A -vector -format 5 -dense $(LSQ_5A_TABLE) -end \
▷  ▷  -define D -design -linear_space_from_latin_square \
▷  ▷  ▷  LSQ5A \
▷  ▷  -end \
▷  ▷  -with D -do \
▷  ▷  ▷  -design_activity \
▷  ▷  ▷  ▷  -export_inc \
▷  ▷  -end
```

Example 555

```
LSQ_5B:
▷  $(ORBITER) -v 8 \
▷  ▷  -define LSQ5B -vector -format 5 -dense $(LSQ_5B_TABLE) -end \
▷  ▷  -define D -design -linear_space_from_latin_square \
▷  ▷  ▷  LSQ5B \
▷  ▷  -end \
▷  ▷  -with D -do \
▷  ▷  ▷  -design_activity \
```

```
▷ ▷ ▷ ▷ -export_inc \
▷ ▷ -end
```

Finally, we perform the canonical form algorithm on these two linear spaces, noting that the isomorphisms / automorphisms of the Latin Squares corresponds to the isomorphisms / automorphism of the associated linear spaces.

Example 556

```
LSQ_5_c:
▷ $(ORBITER) -v 3 \
▷ ▷ -define C -combinatorial_object \
▷ ▷ ▷ -label LSQ_5 LSQ\_5 \
▷ ▷ ▷ -file_of_incidence_geometries latin_square_order5_LSQ5A_inc.txt 15 28 90 \
▷ ▷ ▷ -file_of_incidence_geometries latin_square_order5_LSQ5B_inc.txt 15 28 90 \
▷ ▷ -end \
▷ ▷ -with C -do \
▷ ▷ -combinatorial_object_activity \
▷ ▷ ▷ -canonical_form \
▷ ▷ ▷ ▷ -save_ago \
▷ ▷ ▷ ▷ -max_TDO_depth 4 \
▷ ▷ ▷ -end \
▷ ▷ -end \
▷ ▷ -with C -do \
▷ ▷ -combinatorial_object_activity \
▷ ▷ ▷ -report \
▷ ▷ ▷ ▷ -export_flag_orbits \
▷ ▷ ▷ ▷ -show_TDO \
▷ ▷ ▷ ▷ -show_TDA \
▷ ▷ ▷ ▷ -dont_show_incidence_matrices \
▷ ▷ ▷ -end \
▷ ▷ -end
▷ pdflatex LSQ_5_classification.tex
▷ $(OPEN) LSQ_5_classification.pdf
```

This establishes the fact that the geometries are not isomorphic, and it also produces the automorphism groups of each geometry. It turns out that the Latin Square of order 5 arising from the cyclic group has a symmetry group of order 600. The second Latin Square arising from the loop has a symmetry group of order 72. Orbiter produces a report listing generators for each group. For more information about the canonical form of designs, see Section 16.4. Since there is only one linear space of Latin Square 5 type with a group of order 600, the object constructed earlier must be that same space which is associated to the Cayley table of the cyclic group of order 5.

13.2 Assuming Symmetry

One way to construct designs is by assuming a suitable group of symmetries. Let us consider an example. It is possible to construct t-(v, k, λ) designs invariant under a permutation group G acting on a set V with $|V| = v$ as follows: Classify the orbits of G on subsets of size k and less. Construct a matrix which describes the relationship between the orbits on t-sets and the orbits on k-sets. This matrix is often referred to as the Kramer-Mesner matrix (cf. [44]). For each pair of t-orbit and k-orbit, for instance with representatives T and K, say, we count the number of elements in the orbit of K which contain T. The rows of the matrix are in correspondence to the t-orbits, while the columns are in correspondence to the k-orbits. The matrix entry a_{ij} is the number just defined where T is the representative of the i-th orbit on t-sets, and where K is the representative of the j-th orbit on k-sets. Let $M_{t,k}(G)$ be the Kramer-Mesner matrix for the group $G \leq \mathrm{Sym}(V)$ defined in this way. The t-(v, k, λ) designs invariant under G are in one-to-one correspondence to the solutions of

$$M_{t,k}(G) \cdot \mathbf{x} = \lambda \mathbf{1},$$

where $\mathbf{x}$ is a column vector of zeros and ones and $\mathbf{1}$ is the column vector of all ones. The length of $\mathbf{x}$ is the number of k-orbits of G on V, while the length of $\mathbf{1}$ is the number of t-orbits of G on V. Any vector $\mathbf{x}$ satisfying the matrix equation corresponds to a design invariant under G. Simply take the blocks of the design to be the union of those orbits of G on k-subsets whose associated entry in $\mathbf{x}$ is one. We assume the group $\mathrm{P\Gamma L}(2, 32)$ in the action on points of the projective line $\mathrm{PG}(1, 32)$ over the field $\mathbb{F}_{32}$. The parameters of the design are 7-$(33, 8, 10)$, that is, each 7-subset of $\mathrm{PG}(1, 32)$ is covered exactly 10 times by the chosen 8-subsets comprising the design. Let us consider the following Orbiter command:

Example 557

```
KM_PGGL_2_32:
▷ $(ORBITER) -v 3 \
▷ ▷ -orbiter_path $(ORBITER_EXE_PATH) \
▷ ▷ -define Control -poset_classification_control \
▷ ▷ ▷ -problem_label KM_PGGL_2_32 \
▷ ▷ ▷ -W -depth 8 \
▷ ▷ ▷ -draw_options -embedded -sideways -radius 50 \
▷ ▷ ▷ ▷ -scale 0.5 -line_width 0.3 -end \
▷ ▷ -end \
▷ ▷ -define G -linear_group -PGGL 2 32 -end \
▷ ▷ -define Orb -orbits -group G \
▷ ▷ ▷ -on_subsets 8 Control \
▷ ▷ -end \
▷ ▷ -with Orb -do -orbits_activity \
▷ ▷ ▷ -Kramer_Mesner_matrix 7 8 \
▷ ▷ -end \
▷ ▷ -with Orb -do -orbits_activity \
▷ ▷ ▷ -report \
▷ ▷ ▷ -report_options \
▷ ▷ ▷ ▷ -draw_poset -type_ordinary \
▷ ▷ ▷ -end \
▷ ▷ -end
▷ $(ORBITER) -v 2 -draw_matrix \
▷ ▷ -input_csv_file KM_PGGL_2_32_KM_7_8.csv \
▷ ▷ -box_width 20 -bit_depth 24 \
▷ ▷ -partition 3 32 97 -end
▷ pdflatex KM_PGGL_2_32_poset_lvl_8.tex
▷ $(OPEN) KM_PGGL_2_32_poset_lvl_8.pdf
▷ $(OPEN) KM_PGGL_2_32_KM_7_8_draw.bmp
```

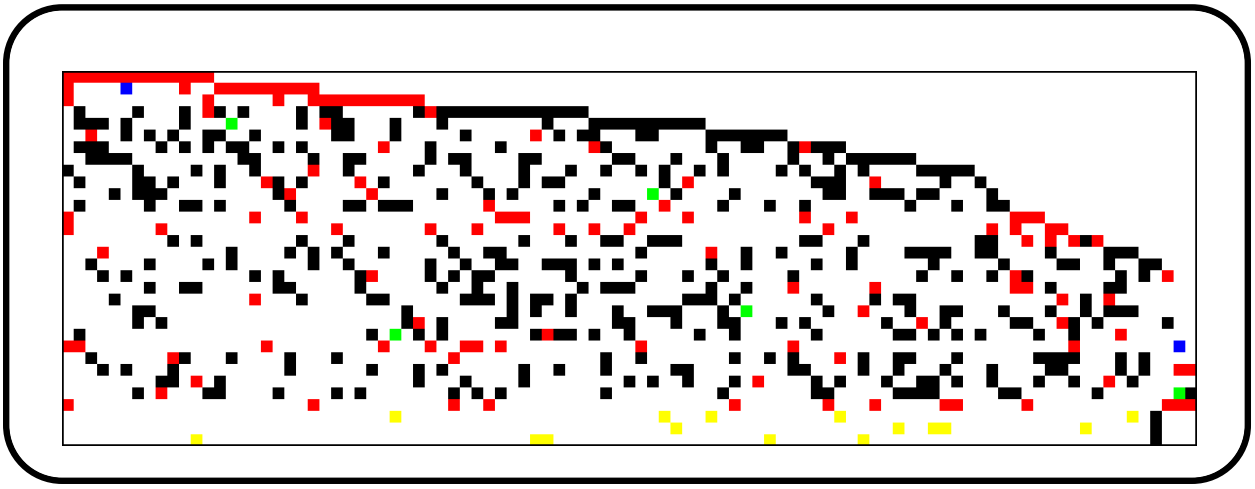
The first command creates the group $\mathrm{P\Gamma L}(2,32)$ and computes the Kramer-Mesner matrix

$$M_{7,8}(\mathrm{P\Gamma L}(2,32)).$$

The number of 7-orbits is 32. The number of 8-orbits is 97. Correspondingly, the Kramer-Menser matrix has 32 rows and 97 columns. The matrix is stored in the csv-file

KM_PGGL_2_32_KM_7_8.csv.

The second command produces the graphical representation of the matrix shown below



(different colors represent different values of the entries in the matrix). The command

Example 558

```
KM_PGGL_2_32_solve:
▷  $(ORBITER) -v 4 \
▷  ▷  -define A -vector -file KM_PGGL_2_32_KM_7_8.csv -end \
▷  ▷  -define D -diophant \
▷  ▷  -label "KM_PGGL_2_32_KM_7_8_system" \
▷  ▷  -coefficient_matrix A \
▷  ▷  -RHS_constant "EQ=10" \
▷  ▷  -x_min_global 0 -x_max_global 1 \
▷  ▷  -end \
▷  ▷  -with D -do \
▷  ▷  ▷  -diophant_activity -solve_mckay \
▷  ▷  -end
```

creates the diophantine system associated with the Kramer-Mesner matrix and solves it. It performs a complete enumeration of all solutions by solving the system and producing the solution vectors $\mathbf{x}$ corresponding to the designs.

Let us construct configurations with an assumed symmetry group. Suppose we are interested in configurations with $v = 15$ points, and 25 blocks of size 3. We assume a group of order 5 acting semi regularly. In cyclce notation, the generator may be taken as

$$(0,1,2,3,4)(5,6,7,8,9)(10,11,12,13,14).$$

In two row notation, we write

$$\begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 \\ 1 & 2 & 3 & 4 & 0 & 6 & 7 & 8 & 9 & 5 & 11 & 12 & 13 & 14 & 10 \end{bmatrix}$$

A makefile variable will hold the generator (the second row in the two-row notation):

Example 559

```
GEN_C5="1,2,3,4,0,6,7,8,9,5,11,12,13,14,10"
```

Next, we need to compute the orbits on pairs and triples. We also need the Kramer-Mesner matrices

$$M_{1,3}, \quad M_{2,3}$$

Every point lies on 5 blocks, and every pair orbit is covered at most once. This leads to the system

$$M_{1,3}\mathbf{x} = 5 \cdot \mathbf{1}, \qquad M_{2,3}\mathbf{x} \leq \mathbf{1}, \qquad \mathbf{0} \leq \mathbf{x} \leq \mathbf{1},$$

where $\mathbf{0}$ is the all-zero vector and $\mathbf{1}$ is the all-one vector. The following command computes $M_{1,3}$:

Example 560

```
C5_KM_1_3:
▷  $(ORBITER) -v 3 \
▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
▷  ▷  -define gens -vector -dense $(GEN_C5) -end \
▷  ▷  -define G -permutation_group -symmetric_group 15 \
▷  ▷  ▷  -subgroup_by_generators "C5" 5 1 gens \
▷  ▷  -end \
▷  ▷  -define Control -poset_classification_control \
▷  ▷  ▷  -problem_label C5 -W -depth 3 \
▷  ▷  ▷  -draw_options -embedded -sideways -radius 50 \
▷  ▷  ▷  ▷  -scale 0.5 -line_width 0.3 -end \
▷  ▷  -end \
▷  ▷  -define Orb -orbits -group G \
▷  ▷  ▷  -on_subsets 3 Control \
▷  ▷  -end \
▷  ▷  -with Orb -do -orbits_activity \
▷  ▷  ▷  -Kramer_Mesner_matrix 1 3 \
▷  ▷  -end \
▷  ▷  -with Orb -do -orbits_activity \
▷  ▷  ▷  -export_something set_orbits 3 \
▷  ▷  -end \
```

The matrix has 3 rows and 91 columns. The command also exports the orbits on 3-subsets to a file with the name `C5_set_orbits_level_3.csv`. The following command computes $M_{2,3}$:

Example 561

```
C5_KM_2_3:
▷  $(ORBITER) -v 3 \
▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
```

```
▷  ▷  -define gens -vector -dense $(GEN_C5) -end \
▷  ▷  -define G -permutation_group -symmetric_group 15 \
▷  ▷  ▷  -subgroup_by_generators "C5" 5 1 gens \
▷  ▷  -end \
▷  ▷  -define Control -poset_classification_control \
▷  ▷  ▷  -problem_label C5 -W -depth 3 \
▷  ▷  ▷  -draw_options -embedded -sideways -radius 50 \
▷  ▷  ▷  ▷  -scale 0.5 -line_width 0.3 -end \
▷  ▷  -end \
▷  ▷  -define Orb -orbits -group G \
▷  ▷  ▷  -on_subsets 3 Control \
▷  ▷  -end \
▷  ▷  -with Orb -do -orbits_activity \
▷  ▷  ▷  -Kramer_Mesner_matrix 2 3 \
▷  ▷  -end
```

The matrix has 21 rows and 91 columns. The next command stacks the two matrices together, one on top of the other:

Example 562

```
C5_concatenate:
▷  $(ORBITER) -v 3 \
▷  ▷  -csv_file_concatenate C5_KM_combined.csv 2 C5_KM_1_3.csv C5_KM_2_3.csv
```

Next, we need to solve the system. To this end, we use the following command.

Example 563

```
C5_solve:
▷  $(ORBITER) -v 4 \
▷  ▷  -define A -vector -file C5_KM_combined.csv -end \
▷  ▷  -define D -diophant \
▷  ▷  ▷  -label "C5_KM_combined_system" \
▷  ▷  ▷  -coefficient_matrix A \
▷  ▷  ▷  -x_min_global 0 \
▷  ▷  ▷  -x_max_global 1 \
▷  ▷  ▷  -RHS "mult=3,EQ=5" \
▷  ▷  ▷  -RHS "mult=21,LE=1" \
▷  ▷  -end \
▷  ▷  -with D -do \
▷  ▷  ▷  -diophant_activity -solve_mckay \
▷  ▷  -end
```

Orbiter finds 21540 solutions in 61001 backtrack steps. Next, we need to assemble the designs by concatenating the set orbits on 3-subsets that are selected by the solution vector $\mathbf{x}$. Once the designs are created, we need to classify them up to isomorphism. We utilize the classifcation algorithm in Orbiter which is based on canonical forms computed by Nauty:

Example 564

```
C5_design_classify:
▷  $(ORBITER) -v 2 \
▷  ▷  -define C -combinatorial_object \
▷  ▷  ▷  -label C5_config C5\_config \
▷  ▷  ▷  -file_of_designs_through_block_orbits \
▷  ▷  ▷  ▷  C5_KM_combined_system_sol.csv \
▷  ▷  ▷  ▷  C5_set_orbits_level_3.csv \
▷  ▷  ▷  ▷  15 3 \
▷  ▷  -end \
▷  ▷  -with C -do \
▷  ▷  -combinatorial_object_activity \
▷  ▷  ▷  -canonical_form \
▷  ▷  ▷  -end \
▷  ▷  -end \
▷  ▷  -with C -do \
▷  ▷  -combinatorial_object_activity \
▷  ▷  ▷  -report \
▷  ▷  ▷  ▷  -show_incidence_matrices \
▷  ▷  ▷  -end \
▷  ▷  -end
▷  pdflatex C5_config_classification.tex
▷  $(OPEN) C5_config_classification.pdf
```

A latex report is produced. Orbiter finds 42 isomorphism types of designs. The largest automorphism group associated with these designs has order 600.

Creating Large Sets with Assumed Symmetry		
Command	Arguments	Purpose
-H	go gens	Specify the assumed group H.
-N	go gens	Specify the normalizer N of the group H.
-report		Create a latex report.
-prefix	prefix	Set a prefix for output files.
-selected_orbit_length	l	Select orbit length l.

Table 13.4: Creating Large Sets with Assumed Symmetry

Large Set Activity		
Command	Arguments	Purpose
-normalizer_on_orbits_of_ a_given_length	l nb_orbits control	
-create_graph_on_orbits_of_length	fname l	
-create_graph_on_orbits_ of_length_based_on_N_orbits	fname-mask l nb_orbits r m	
-read_solution_file	l fname	

Table 13.5: Large Set Activity

13.3 Design Theory – Large Sets

Fix a set of size v and an integer k with $1 < k < v$. Is it possible to partition the set of k-subsets of v into designs, all with the same parameters? If so, the resulting set of designs is called a large set (of designs). So, a large set of designs is a set of designs, all of the same types, on a fixed v-element set whose block sets are pairwise disjoint and partition the set of k-subsets. Let us see how Orbiter can help construct and classify small large sets.

Tables 13.4 shows Orbiter commands to construct large sets with assumed symmetry group. Table 13.5 lists Orbiter activities for objects of type large-set-with-assumed-symmetry.

Suppose we consider AG$(2, 3)$, the affine plane of order 3. It is a configuration with 9 points, 12 lines, 4 lines on each point and 3 points on each line. To see whether or not it is unique, we use the following command:

Example 565

```
AG_2_3.inc:
▷  $(ORBITER) -v 2 \
▷  ▷  -define Geo -geometry_builder \
▷  ▷  ▷  -V 9 -B 12 \
▷  ▷  ▷  -TDO 4 -fuse 1 \
▷  ▷  ▷  -fname_GEO AG_2_3 \
▷  ▷  ▷  -test 3,4,5,6,7,8,9 \
▷  ▷  ▷  -output_to_inc_file \
▷  ▷  -end
```

The command produces the file AG_2_3.inc, which contains the following lines:

```
9 12 36
0 1 2 3 12 16 17 18 24 31 32 33 37 40 43 46 49 53 56 59 62 64 69 71 74 78 80 82 87 89 93 94 99 102 103 107
-1 1
432
```

This shows that the design is unique, and has an automorphism group of order 432. For the following commnds, we will treat blocks of the design as sets of ranks of k-subsets. Next, we create a table of all designs AG$(2,3)$, as orbit under the group Sym(9). The following command does that:

Example 566

```
AG_2_3_BLOCKS="0,13,22,27,35,41,47,53,55,59,71,76"
```

Example 567

```
LS_AG_2_3_design_table_create:
▷  $(ORBITER) -v 5 \
▷  ▷  -define B -vector -dense $(AG_2_3_BLOCKS) -end \
▷  ▷  -define D -design -list_of_blocks_coded 9 3 B -end \
▷  ▷  -define Sym9 -permutation_group -symmetric_group 9 -end \
▷  ▷  -define T -design_table D "AG_2_3" Sym9 -end
```

The number of designs is $|\text{Sym}(9)|/432 = 362880/432 = 840$. To find all large sets, we establish the block-disjointness graph on this set of designs. After that, we find all cliques of size 7:

Example 568

```
LS_AG_2_3_disjoint_sets_graph_and_cliques:
▷  $(ORBITER) -v 2 \
▷  ▷  -define Gamma -graph \
▷  ▷  ▷  -disjoint_sets_graph \
▷  ▷  ▷  AG_2_3_design_table.csv \
▷  ▷  -end \
▷  ▷  -with Gamma -do \
▷  ▷  -graph_theoretic_activity \
▷  ▷  ▷  -save \
▷  ▷  -end \
▷  ▷  -with Gamma -do \
▷  ▷  -graph_theoretic_activity \
▷  ▷  ▷  -find_cliques -target_size 7 -end \
▷  ▷  -end \
▷  ▷  -print_symbols
```

The files `AG_2_3_design_table_disjoint_sets_sol.txt` and `AG_2_3_design_table_disjoint_sets_sol.csv` are created, each containing the cliques of size 7. There are exactly 15360 cliques of size 7. It remains to classify the resulting 15360 large sets up to isomorphism. To do that, we first need to create the actual large sets from the cliques. The following command does that:

Example 569

```
LS_AG_2_3_export_solutions:
▷  $(ORBITER) -v 20 \
▷  ▷  -define B -vector -dense $(AG_2_3_BLOCKS) -end \
▷  ▷  -define D -design -list_of_blocks 9 3 B -end \
▷  ▷  -define Sym9 -permutation_group -symmetric_group 9 -end \
▷  ▷  -define T -design_table D "AG_2_3" Sym9 -end \
▷  ▷  -with D -do \
▷  ▷  -design_activity \
▷  ▷  ▷  -extract_solutions_by_index "AG_2_3" Sym9 \
▷  ▷  ▷  ▷  AG_2_3_design_table_disjoint_sets_sol.csv \
▷  ▷  ▷  ▷  solutions.csv \
▷  ▷  ▷  ▷  "" \
▷  ▷  -end
```

The final step is the classification of these large sets up to isomorphism. This will be discussed in Section 16.4.

13.4 Design Theory – Delandtsheer-Doyen

Delandtsheer and Doyen in [25] study line-transitive and point-imprimitive designs and show that they are rare in a certain sense. Orbiter can be used to construct such designs assuming that there is a grid structure on the set of points and assuming that the design is invariant under a chosen group G. The group G is assumed to be a subgroup of the group $\mathrm{AGL}(d_1, q_1) \times \mathrm{AGL}(d_2, q_2)$ acting on a grid of size $q_1^{d_1} \times a_2^{d_2}$ in product action.

Table 13.6 shows Orbiter commands to search for linear spaces of Delandtsheer/Doyen type. Finite projective planes often arise in this context. However, not all examples are projective planes. Orbiter can help to classify small examples.

Let us consider an example. Suppose we want to classify all designs on 21 points with blocks of size $k = 5$ invariant under a cyclic group of order 21 preserving a grid of type 3×7. To this end, we consider the group $\mathrm{AGL}(1, 3) \times \mathrm{AGL}(1, 7)$. The subgroup is generated by the map

$$(\tau_1, \tau_2), \mathbb{Z}_3 \times \mathbb{Z}_7 \to \mathbb{Z}_3 \times \mathbb{Z}_7,$$

where

$$\tau_1 : \mathbb{Z}_3 \to \mathbb{Z}_3,\ x \mapsto x + 1 \mod 3, \quad \tau_2 : \mathbb{Z}_7 \to \mathbb{Z}_7,\ y \mapsto y + 1 \mod 7.$$

With blocks of size 5, we cover 10 pairs each. The group of order 21 allows to cover each of the $210 = \binom{21}{2}$ pairs exactly once using a single orbit of a block. Under the group G, there are 55 orbits on pairs. The question remains to construct all blocks and to classify the resulting designs.
At first, three makefile variables are defined:

Example 570

```
PP4=-d1 1 -q1 3 -d2 1 -q2 7 -K 5 -problem_label PP4
```

Example 571

```
PP4_GROUP1=-subgroup "1,1,1,1, " "21" -group_label "cyclic21"
```

Example 572

```
PP4_MASK1=\
▷  -nb_orbits_on_blocks 1 \
▷  -mask_label "no_mask"
```

The next command performs a classification of the sets of size 5 under the given group and using the given mask conditions:

Example 573

```
DD_PP4:
▷  $(ORBITER) -v 6 \
▷  ▷  -define pair_search_control -poset_classification_control \
▷  ▷  ▷  -problem_label PP4_pairs -W \
▷  ▷  ▷  -draw_options -end \
▷  ▷  -end \
```

Creating Linear Spaces of Delandtsheer-Doyen Type		
Command	Arguments	Purpose
-d1	d1	Affine group parameter.
-d2	d2	Affine group parameter.
-q1	q1	Affine group parameter.
-q2	q2	Affine group parameter.
-group_label	label	
-mask_label	label	
-problem_label	label	
-DDx	x	Deladtsheer-Doyen parameter x.
-DDy	y	Secondary Deladtsheer-Doyen parameter y.
-K	k	Specify the size of the base-line to generate the linear space under the affine group.
-pair_search_control	label	Options for the classification of pair orbits, see Tab. 7.4.
-search_control	label	Options for the classification of the base lines by means of subsets, see Tab. 7.4. The depth of the search for starters (partial base-lines) is set here, using the -depth command.
-R	nb_row_types row_type	
-C	nb_col_types col_type	
-nb_orbits_on_blocks	n	
-masktest	options	Specify conditions on the mask, which is the pattern induced by the base-line when considering the grid structure.
-search_partial_base_lines		Search for partial base-lines. The size of the partial base-line is specified in the search_control data structure, using the -depth argument. Once the classification of partial base-lines is finished (of size s, say), the covering matrix of pair orbits is written. For each orbit on partial base-lines of size s, the $\binom{s}{2}$ disjoint pair-orbits orbits covered by that representative are written.
-singletons	sz	Search for singletons. This requires that a classification of partial base-lines of size sz has been completed before. This command initiates the completion of the partial base-lines.
-subgroup	gens order	Specify generators for the subgroup G which acts line-transitive and point-imprimitive on the linear space.
-search_wrt_subgroup		

Table 13.6: Creating Linear Spaces of Delandtsheer-Doyen Type

```
▷  ▷  -define search_control -poset_classification_control \
▷  ▷  ▷  -problem_label PP4_search -W \
▷  ▷  ▷  -draw_options -end \
▷  ▷  ▷  -depth 5 \
▷  ▷  -end \
▷  ▷  -Delandtsheer_Doyen \
▷  ▷  ▷  -pair_search_control pair_search_control \
▷  ▷  ▷  -search_control search_control \
▷  ▷  ▷  -search_partial_base_lines \
▷  ▷  ▷  $(PP4) $(PP4_GROUP1) $(PP4_MASK1) \
▷  ▷  ▷  -end \
```

We find exactly one solution. The following command spans the design from the base line by computing the orbit under G:

Example 574

```
DD_PP4_span_design:
▷  $(ORBITER) -v 6 \
▷  ▷  -define D -design -list_of_base_blocks \
▷  ▷  ▷  group \
▷  ▷  ▷  PP4_no_mask_cyclic21_cases.csv
▷  ▷  ▷  Starter \
▷  ▷  -end
```

The next command creates a design object from the base line under the group G:

Example 575

```
DD_PP4_design_orbit:
▷  $(ORBITER) -v 4 \
▷  ▷  -define G1 -linear_group -AGL 1 3 \
▷  ▷  ▷  -end \
▷  ▷  -define G2 -linear_group -AGL 1 7 \
▷  ▷  ▷  -end \
▷  ▷  -define G -modified_group -direct_product "G1,G2" \
▷  ▷  ▷  "21" "1,1,1,1" \
▷  ▷  -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end \
▷  ▷  -define D -design -list_of_base_blocks \
▷  ▷  ▷  G \
▷  ▷  ▷  PP4_no_mask_cyclic21_cases.csv \
▷  ▷  ▷  Starter \
▷  ▷  -end
```

As PG$(2, 4)$ admits an action by a cyclic group of order 21 in product action, and we find only one design, the design we found must be PG$(2, 4)$. It is the only line-transitive point-imprimitive design with these

parameters.

Chapter 14

Finite Geometry

14.1 Spreads

A t-spread of PG(n, q) is a set of disjoint PG(t, q) that cover all of PG(n, q) pointwise. t-spreads in PG(n, q) exist if and only if $t + 1$ divides $n + 1$. Table 14.1 lists the Orbiter commands to create spreads. Table 14.2 lists Orbiter spread activities. Table 14.3 list Orbiter commands to classify spreads. Table 14.4 lists the Orbiter commands that apply to a object of type spread classify. Table 14.5 lists the Orbiter activities for spread tables.
The following two commands create the two spreads of order 9, relying on the Orbiter knowledge base.

Example 576

```
create_spread_9a:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 3 -end \
▷  ▷  -define G -linear_group -PGL 4 F -end \
▷  ▷  -define S -spread -kernel_field F \
▷  ▷  ▷  -group G -k 2 -catalogue 0 \
▷  ▷  ▷  -end
```

Example 577

```
create_spread_9b:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 3 -end \
▷  ▷  -define G -linear_group -PGL 4 F -end \
▷  ▷  -define S -spread -kernel_field F \
▷  ▷  ▷  -group G -k 2 -catalogue 1 \
▷  ▷  ▷  -end
```

The first spread is the Desarguesian spread, with automorphism group of order 5760. The second spread is the Hall spread with automorphism group of order 1920.

Spreads can be defined using spread sets. A spread set is a set of q^k matrices of size $k \times k$ over $\mathbb{F}_q$ such that $A_i - A_j$ is nonsingular for all $i \neq j$. Let us look at an example. The spread due to Rao and Rao [58] can be defined using the following makefile variable.

Creating a Spread		
Command	Arguments	Purpose
-kernel_field	F	Define the kernel of the spread. F must be an object of type finite field.
-group	G	Define the group acting on the spread. Should be $\mathrm{P\Gamma L}(2k, F)$.
-group_on_subspaces	G	
-k	k	Set the dimension of the spread.
-catalogue	i	Pull spread number i from the catalogue of spreads associated with the given field and the given dimension.
-family	L	Define a spread from a named family L. So far, no family has been provided.
-spread_set	label-txt label-tex S	Define a spread from the named spreadset S. The spreadset S must be a vector object. It must contain q^3k^2 entries over F.
-transform	elt	Apply the transformation given by the group element.
-transform_inverse	elt	Apply the inverse transformation given by the group element.

Table 14.1: Creating a Spread

Spread Activities		
Command	Arguments	Purpose
-report		Create a latex report.

Table 14.2: Spread Activities

Classification of Spreads		
Command	Arguments	Purpose
-projective_space	P	Select the projective space object P.
-starter_size	k	Choose starter size k.
-k	k	Choose dimension of the spread elements.
-poset_classification_control	control	Select poset classification control.
-output_prefix	prefix	Set output prefix for file i/o.
-recoordinatize		Turn recoordinatization on (recommended).

Table 14.3: Classification of Spreads

Activities for the Classification of Spreads		
Command	Arguments	Purpose
-compute_starter		Perform classification of partial spreads.
-prepare_lifting_single_case	k	Prepare to lift single case k. Here (and below) case means isomorphism type in the classification of partial spreads.
-prepare_lifting_all_cases		Prepare to lift all cases of partial spreads.
-split	r m	Lift only those cases k with $k \equiv r \mod m$.
-isomorph	prefix1 prefix2 options	Perform isomorphism testing after lifting is complete.

Table 14.4: Activities for the Classification of Spreads

Activities for Spread Tables		
Command	Arguments	Purpose
-find_spread	spread	Find a given spread in the table.
-find_spread_and_dualize	spread	Find a given spread in the table and dualize.
-dualize_packing	packing	Dualize a packing.
-print_spreads	spreads	Print the given spreads.
-export_spreads_to_csv	fname index	Export the spread whose index is given to csv-file.
-find_spreads_containing_one_line	index	Find all spreads which contain the given line.

Table 14.5: Activities for Spread Tables

Example 578

```
SPREAD_SET_27_RAO_RAO="\
0,0,0,0,0,0,0,0,0, \
1,1,0,2,1,1,0,0,2, \
1,0,1,1,2,2,0,1,0, \
1,2,2,1,2,0,2,2,2, \
0,0,2,2,2,0,1,2,0, \
1,1,2,0,2,1,2,1,0, \
0,1,0,1,0,1,0,2,1, \
2,0,2,0,0,2,1,1,0, \
2,2,2,0,1,1,0,1,2, \
2,0,0,1,0,2,1,2,1, \
0,2,2,2,2,2,2,0,2, \
2,1,2,0,2,0,2,0,1, \
0,1,2,2,0,1,0,1,1, \
1,0,0,0,1,0,0,0,1, \
2,1,0,1,2,1,0,2,0, \
0,2,0,0,2,2,1,1,2, \
0,0,1,0,1,2,2,2,1, \
2,0,1,2,2,1,1,0,1, \
0,1,1,1,1,0,1,2,2, \
2,2,0,2,0,0,0,2,2, \
2,1,1,1,1,2,2,1,2, \
2,2,1,2,1,0,2,0,0, \
1,2,0,2,0,2,1,0,0, \
1,2,1,1,0,0,1,1,1, \
0,2,1,1,1,1,2,2,0, \
1,1,1,0,0,1,1,0,2, \
1,0,2,2,1,2,2,1,1 \
"
```

Each line represents one matrix of the spread set, with matrix entries being listed consecutively. The following command can be used to define the spread:

Example 579

```
create_spread_Rao_Rao_27:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 3 -end \
▷  ▷  -define SS -vector -dense $(SPREAD_SET_27_RAO_RAO) -end \
▷  ▷  -define G -linear_group -PGL 6 F -end \
▷  ▷  -define S -spread -kernel_field F \
▷  ▷  ▷  -group G -k 3 -spread_set "Rao_Rao" "Rao\_Rao" SS \
▷  ▷  ▷  -end
```

The following command creates the Desarguesian line-spread in PG$(3, 2)$:

Example 580

```
desarguesian_spread_in_PG_3_2:
▷  $(ORBITER) -v 3 \
▷  ▷  -define FQ -finite_field -q 4 -end \
▷  ▷  -define Fq -finite_field -q 2 -end \
▷  ▷  -with FQ -and Fq -do -finite_field_activity \
▷  ▷  ▷  -cheat_sheet_desarguesian_spread 2 -end
▷  pdflatex Desarguesian_Spread_3_2.tex
▷  $(OPEN) Desarguesian_Spread_3_2.pdf
```

The cheat sheet contains the following spread:

Spread element 0 is $(1,0) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}_0$

Spread element 1 is $(0,1) = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_{34}$

Spread element 2 is $(1,1) = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}_9$

Spread element 3 is $(2,1) = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}_{17}$

Spread element 4 is $(3,1) = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}_{22}$

Spread elements by rank: (0, 34, 9, 17, 22).

The following command creates the Desarguesian plane-spread in PG$(5,2)$:

Example 581

```
desarguesian_spread_in_PG_5_2:
▷  $(ORBITER) -v 3 \
▷  ▷  -define FQ -finite_field -q 8 -end \
▷  ▷  -define Fq -finite_field -q 2 -end \
▷  ▷  -with FQ -and Fq -do -finite_field_activity \
▷  ▷  ▷  -cheat_sheet_desarguesian_spread 2 -end
▷  pdflatex Desarguesian_Spread_5_2.tex
▷  $(OPEN) Desarguesian_Spread_5_2.pdf
```

Spread element 0 is $(1,0) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}_0$

$$\text{Spread element 1 is } (0,1) = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}_{1394}$$

$$\text{Spread element 2 is } (1,1) = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}_{189}$$

$$\text{Spread element 3 is } (2,1) = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}_{671}$$

$$\text{Spread element 4 is } (3,1) = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}_{562}$$

$$\text{Spread element 5 is } (4,1) = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}_{1040}$$

$$\text{Spread element 6 is } (5,1) = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}_{792}$$

$$\text{Spread element 7 is } (6,1) = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}_{1161}$$

$$\text{Spread element 8 is } (7,1) = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}_{373}$$

Spread elements by rank: (0, 1394, 189, 671, 562, 1040, 792, 1161, 373)

Two t-spreads are isomorphic if there is a collineation which maps one to the other. The classification problem for t-spreads is the problem of determining a complete set of pairwise non-isomorphic t-spreads. Orbiter can be used to classify spreads for small parameters. For greater classification power, the method of classification by substructure is used. Let us look at some examples.

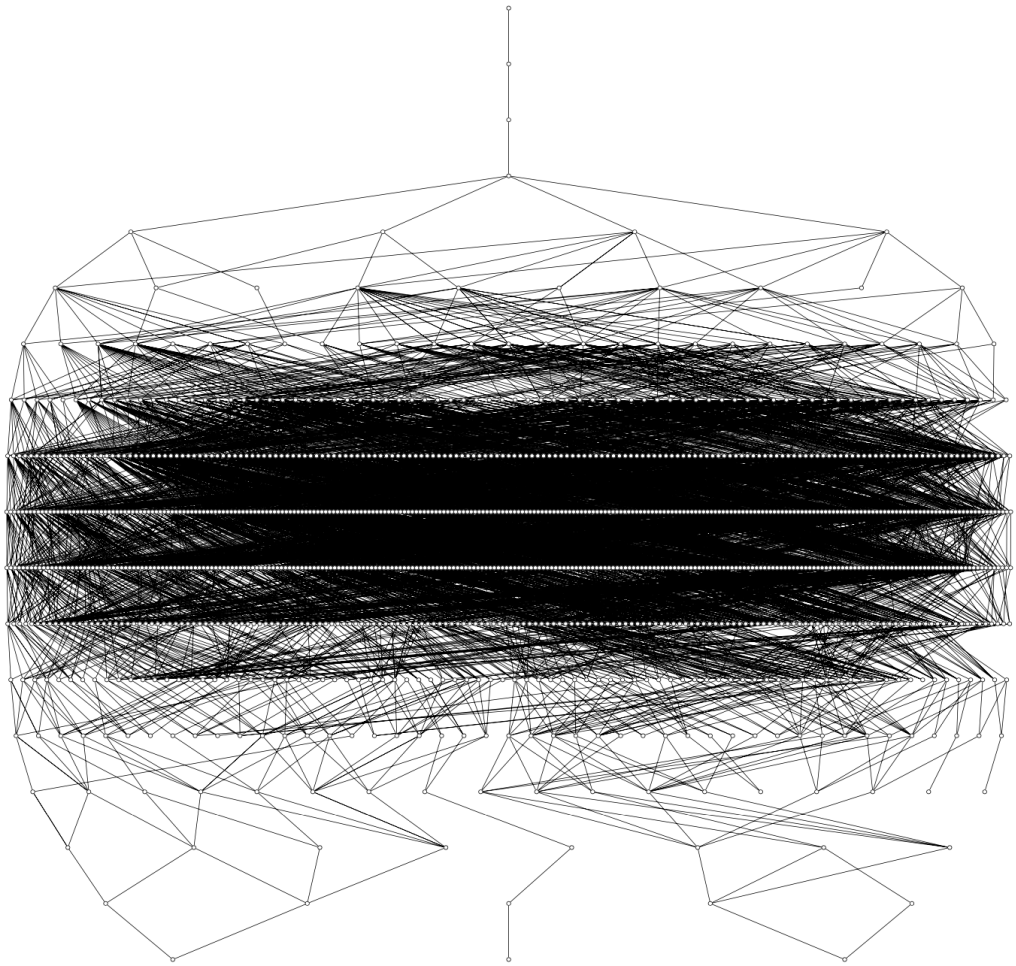
At first, we look at an example which is sufficiently small and can be solved using the standard method. Here, the standard method is poset classification algorithm for partial spreads. Suppose we want to classify the line spreads in PG$(3,4)$ under the action of PΓL$(4,4)$. Under the André, Bruck-Bose construction [3, 18], these spreads correspond to translation planes of order 16 with kernel $\mathbb{F}_4$. In order to classify the spreads of PG$(3,4)$, we utilize uses poset classification, as shown in the following command

Example 582

```
classify_spreads_16_4:
▷  $(ORBITER) -v 4 \
▷  ▷  -define Control -poset_classification_control \
▷  ▷  ▷  -draw_options \
▷  ▷  ▷  ▷  -radius 20 \
▷  ▷  ▷  ▷  -nodes_empty \
▷  ▷  ▷  ▷  -line_width 0.2 \
```

```
▷ ▷ ▷ ▷  -embedded \
▷ ▷ ▷  -end \
▷ ▷ ▷  -problem_label spreads_16_4 \
▷ ▷  -end \
▷ ▷  -define F -finite_field -q 4 -end \
▷ ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷ ▷  -define C -spread_classifier \
▷ ▷ ▷  -projective_space P \
▷ ▷ ▷  -poset_classification_control Control \
▷ ▷ ▷  -k 2 \
▷ ▷ ▷  -starter_size 17 \
▷ ▷ ▷  -output_prefix "." \
▷ ▷  -end \
▷ ▷  -with C -do -spread_classify_activity \
▷ ▷ ▷  -compute_starter \
▷ ▷  -end
▷  #pdflatex spreads_16_4_poset_lvl_17.tex
▷  #$(OPEN) spreads_16_4_poset_lvl_17.pdf
```

The algorithm computes the poset of orbits for the group $G = \mathrm{P\Gamma L}(4,4)$ acting on the poset of partial spreads in $\mathrm{PG}(3,4)$. The poset of orbits is shown below:



Up to isomorphism, there are exactly three line-spreads in PG$(3, 4)$ (corresponding to the three nodes at the bottom of the poset of orbits in the figure). These three spreads are the dearguesian spread, the Hall spread, and the semifield spread, respectively. Here is the relevant output taken from the latex report:

There are 3 orbits at level 17.

Orbit 0 / 3 at Level 17

Node number: 1126

$$\{0, 25, 50, 75, 90, 107, 122, 140, 144, 157, 179, 204, 213, 238, 268, 334, 345\}_{1200}$$

Strong generators for a group of order 1200:

$$\begin{bmatrix} \omega^2 & 0 & 0 & 0 \\ 0 & \omega^2 & 0 & 0 \\ 1 & \omega & 0 & 1 \\ \omega^2 & \omega^2 & \omega & 1 \end{bmatrix}_0, \begin{bmatrix} \omega^2 & 0 & 0 & 0 \\ \omega & \omega^2 & 0 & 0 \\ \omega & \omega & 1 & \omega^2 \\ 0 & 1 & 1 & 0 \end{bmatrix}_1, \begin{bmatrix} \omega & 1 & \omega & \omega \\ \omega & \omega^2 & \omega^2 & 0 \\ \omega & 0 & 0 & 1 \\ 0 & \omega & \omega & 1 \end{bmatrix}_0$$

1,0,0,0,0,1,0,0,2,3,0,2,1,1,3,2,0,
1,0,0,0,3,1,0,0,3,3,2,1,0,2,2,0,1,
1,3,1,1,1,2,2,0,1,0,0,3,0,1,1,3,0,
There are 0 extensions
Number of generators 3

Orbit 1 / 3 at Level 17

Node number: 1127

$$\{0, 25, 50, 75, 90, 107, 140, 157, 179, 204, 213, 238, 265, 282, 299, 316, 356\}_{81600}$$

Strong generators for a group of order 81600:

$$\begin{bmatrix} \omega & 0 & 0 & 0 \\ 0 & \omega & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_0, \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \omega & \omega^2 \\ 0 & 0 & 1 & 1 \end{bmatrix}_0, \begin{bmatrix} \omega^2 & 0 & 0 & 0 \\ 0 & \omega^2 & 0 & 0 \\ 1 & \omega^2 & \omega & \omega^2 \\ 1 & \omega & 1 & 1 \end{bmatrix}_0,$$

$$\begin{bmatrix} \omega^2 & 0 & 0 & 0 \\ \omega & \omega^2 & 0 & 0 \\ 0 & 0 & 0 & \omega^2 \\ 0 & 0 & \omega & 1 \end{bmatrix}_1, \begin{bmatrix} \omega^2 & 0 & 0 & \omega \\ \omega & \omega^2 & 1 & \omega^2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \omega^2 & 1 \end{bmatrix}_1, \begin{bmatrix} 0 & \omega^2 & \omega^2 & 0 \\ 1 & 0 & \omega^2 & \omega^2 \\ 0 & 0 & 1 & \omega^2 \\ 0 & 0 & 0 & 1 \end{bmatrix}_0,$$

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & \omega & 1 \\ 0 & 1 & \omega & \omega^2 \\ \omega & 1 & 1 & 1 \end{bmatrix}_0$$

1,0,0,0,0,1,0,0,0,0,3,0,0,0,0,3,0,
1,0,0,0,0,1,0,0,0,0,2,3,0,0,1,1,0,
1,0,0,0,0,1,0,0,2,1,3,1,2,3,2,2,0,
1,0,0,0,3,1,0,0,0,0,0,1,0,0,3,2,1,
1,0,0,3,3,1,2,1,0,0,2,0,0,0,1,2,1,
0,1,1,0,2,0,1,1,0,0,2,1,0,0,0,2,0,
0,0,0,1,0,0,2,1,0,1,2,3,2,1,1,1,0,
There are 0 extensions
Number of generators 7

Orbit 2 / 3 at Level 17

Node number: 1128

$$\{0, 25, 50, 75, 90, 108, 122, 140, 158, 183, 199, 217, 233, 250, 268, 312, 345\}_{576}$$

Spreads in the Orbiter Catalogue

OCN	|Aut|	Name
0	1200	Hall spread
1	81600	Desarguesian spread
2	576	Semifield spread

Table 14.6: Spreads in PG(3, 4) in the Orbiter Catalogue

Strong generators for a group of order 576:

$$\begin{bmatrix} \omega & 0 & 0 & 0 \\ 0 & \omega^2 & 0 & 0 \\ 0 & 0 & \omega^2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_1, \begin{bmatrix} \omega^2 & 0 & 0 & 0 \\ 0 & \omega^2 & 0 & 0 \\ \omega & 0 & \omega & 1 \\ \omega^2 & 0 & 0 & 1 \end{bmatrix}_0, \begin{bmatrix} \omega^2 & 0 & 0 & 0 \\ \omega & \omega^2 & 0 & 0 \\ \omega & 0 & 1 & 1 \\ \omega^2 & 0 & \omega^2 & 1 \end{bmatrix}_0,$$

$$\begin{bmatrix} \omega & \omega & \omega & \omega \\ \omega^2 & 0 & \omega^2 & 0 \\ \omega^2 & 0 & \omega^2 & \omega \\ 0 & \omega^2 & \omega^2 & 1 \end{bmatrix}_0, \begin{bmatrix} 1 & 0 & \omega^2 & 1 \\ 1 & \omega^2 & 1 & 0 \\ 1 & 0 & \omega & \omega \\ 0 & 0 & 0 & 1 \end{bmatrix}_1, \begin{bmatrix} 0 & \omega^2 & \omega^2 & 0 \\ 0 & 0 & 0 & \omega^2 \\ 1 & 0 & 1 & \omega^2 \\ \omega & 1 & \omega & 1 \end{bmatrix}_0$$

1,0,0,0,0,2,0,0,0,0,2,0,0,0,0,3,1,
1,0,0,0,0,1,0,0,3,0,3,2,1,0,0,2,0,
1,0,0,0,3,1,0,0,3,0,2,2,1,0,1,2,0,
1,1,1,1,2,0,2,0,2,0,2,1,0,2,2,3,0,
1,0,3,1,1,3,1,0,1,0,2,2,0,0,0,1,1,
0,1,1,0,0,0,0,1,2,0,2,1,3,2,3,2,0,
There are 0 extensions
Number of generators 6

The three spreads in PG(3, 4) can be distinguished by their stabilizer orders. Table 14.6 lists the line spreads in PG(3, 4) according to their orbiter catalogue number (OCN).

Classification by substructure is a powerful method to classify objects. In this method, we do not classify the whole poset. Instead, we classify the substructures, and then move directly to the objects at the target level, thereby skipping over many intermediate levels. The jump is facilitated by an auxiliary algorithm such as clique finding. Table 14.7 shows the commands available for classification by substructure.

Let us now look at spreads in PG(3, 5), with substructures the partial spreads of size $s = 5$. We perform the lifting, which means we will construct for each partial spreads of size s all spreads of PG(3, 5) containing the chosen partial spread as a substructure. In a final step, we will perform an isomorph classification on the set of liftings. This will furnish the desired classification of spreads of PG(3, 5). From a computational point of view, the lifting process is the bottleneck in this procedure. To enhance the performance of the lifting, we use specialized algorithms from graph theory, based on cliques. A clique in a graph is a complete subgraph, i.e. a subset of the vertices such that any two in the subset are adjacent. A variant of this is ranbow clique finding. Here, we have a coloring of the vertices, so that edges are only present between vertices of different color. The goal of rainbow clique finding is to find a clique whose vertices represent all colors. Many problems in the field of combinatorial designs can be attacked using rainbow cliques.

To illustrate the technique, let us discuss some examples. Regarding the substructure, we choose the parameter $s = 5$. This is based on some experimentation. The command

Isomorph Testing		
Command	Arguments	Purpose
-prefix_iso	prefix	Set a prefix for files related to isomorph testing.
-prefix_with_directory	prefix	Set a prefix for files related to isomorph testing.
-prefix_classify	prefix	Set a prefix for files related to classification.
-solution_prefix	prefix	Set a prefix for solution files.
-base_fname	base	Set the main file name.
-use_database_for_starter		Use database to store subobjects.
-implicit_fusion		Use implicit fusion during the isomorphism testing.
-build_db		Initialize the database.
-read_solutions		Read solutions.
-list_of_cases	fname	Read a list of cases from the given file.
-read_solutions_after_split	m	Read the solution files after split modulo m.
-read_statistics_after_split	m	Read the statistics files after split modulo m.
-recognize	label	
-compute_orbits		Compute the orbits of the stabilizer. These are the flag orbits.
-isomorph_testing		Apply isomorphism testing to the flag orbits to produce the classification.
-classification_graph		Comp-ute the classification class (after classification).
-event_file	fname	Reuse a previously written event file during the classification.
-print_mod	N	Set print internal to N.
-isomorph_report		Write a report on the classification.
-export_source_code		Export the classification as C++ source code.
-subset_orbits		
-subset_orbits_file	fname	
-eliminate_graphs_if_possible		
-down_orbits		

Table 14.7: Isomorph Testing

Example 583

```
classify_spreads_25_starter_lift_case_0:
▷  $(ORBITER) -v 3 \
▷  ▷  -define Control -poset_classification_control \
▷  ▷  ▷  -draw_options \
▷  ▷  ▷  ▷  -radius 20 \
▷  ▷  ▷  ▷  -nodes_empty \
▷  ▷  ▷  ▷  -line_width 0.2 \
▷  ▷  ▷  ▷  -embedded \
▷  ▷  ▷  -end \
▷  ▷  ▷  -W \
▷  ▷  ▷  -problem_label spreads_25 \
▷  ▷  -end \
▷  ▷  -define F -finite_field -q 5 -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷  ▷  -define C -spread_classifier \
▷  ▷  ▷  -projective_space P \
▷  ▷  ▷  -k 2 \
▷  ▷  ▷  -starter_size 5 \
▷  ▷  ▷  -recoordinatize \
▷  ▷  ▷  -poset_classification_control Control \
▷  ▷  ▷  -output_prefix "" \
▷  ▷  -end \
▷  ▷  -with C -do -spread_classify_activity \
▷  ▷  ▷  -compute_starter \
▷  ▷  -end \
▷  ▷  -with C -do -spread_classify_activity \
▷  ▷  ▷  -prepare_lifting_single_case 0 \
▷  ▷  -end
```

classifies the partial spreads of size $s = 5$ and prepares for the lifting of the first case only. In order to prepare for the lifting, a graph is constructed which describes the lines that can be added to the first partial spread. The vertices of the graph are the lines disjoint from the initial set of 5 lines in the partial spread. Two vertices are joined by an edge if the associated lines are disjoint. The vertices of the graph are colored according to the very first basis vector in the generator matrix of the subspace in reduced row echelon form. In order to find the rainbow cliques in the graph, the command

Example 584

```
spreads_25_starter_0_cliques:
▷  $(ORBITER) -v 2 \
▷  ▷  ▷  -define G -graph -load spreads_25_graph_0.bin -end \
▷  ▷  ▷  -with G -do \
▷  ▷  ▷  -graph_theoretic_activity \
▷  ▷  ▷  ▷  -find_cliques -rainbow -target_size 21 -end \
▷  ▷  ▷  -end \
```

can be used.

The command

Example 585

```
classify_spreads_25_starter_lift_all_cases:
▷  $(ORBITER) -v 3 \
▷  ▷  -define Control -poset_classification_control \
▷  ▷  ▷  -draw_options \
▷  ▷  ▷  ▷  -radius 20 \
▷  ▷  ▷  ▷  -nodes_empty \
▷  ▷  ▷  ▷  -line_width 0.2 \
▷  ▷  ▷  ▷  -embedded \
▷  ▷  ▷  -end \
▷  ▷  ▷  -W \
▷  ▷  ▷  -problem_label spreads_25 \
▷  ▷  -end \
▷  ▷  -define F -finite_field -q 5 -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷  ▷  -define C -spread_classifier \
▷  ▷  ▷  -projective_space P \
▷  ▷  ▷  -k 2 \
▷  ▷  ▷  -starter_size 5 \
▷  ▷  ▷  -recoordinatize \
▷  ▷  ▷  -poset_classification_control Control \
▷  ▷  ▷  -output_prefix "" \
▷  ▷  -end \
▷  ▷  -with C -do -spread_classify_activity \
▷  ▷  ▷  -compute_starter \
▷  ▷  -end \
▷  ▷  -with C -do -spread_classify_activity \
▷  ▷  ▷  -prepare_lifting_all_cases \
▷  ▷  -end
```

recomputes the partial spreads of size $s = 5$ and prepares for the lifting of all orbit representatives (there are 28). This leads to 28 graphs, each of which is written to a file. The next command performs the rainbow clique finding in each of the 28 graphs:

Example 586

```
spreads_25_starter_cliques:
▷  $(ORBITER) -v 2 \
▷  ▷  -loop L 0 29 1 \
▷  ▷  ▷  -define G -graph -load spreads_25_graph_%L.bin -end \
▷  ▷  ▷  -with G -do \
▷  ▷  ▷  -graph_theoretic_activity \
▷  ▷  ▷  ▷  -find_cliques -rainbow -target_size 21 -end \
▷  ▷  ▷  -end \
▷  ▷  -end_loop L
```

The resulting cliques are again stored in files. The command

Example 587

```
classify_spreads_25_isomorph:
▷  $(ORBITER) -v 4 \
▷  ▷  -define Control -poset_classification_control \
▷  ▷  ▷  -draw_options \
▷  ▷  ▷  ▷  -radius 20 \
▷  ▷  ▷  ▷  -nodes_empty \
▷  ▷  ▷  ▷  -line_width 0.2 \
▷  ▷  ▷  ▷  -embedded \
▷  ▷  ▷  -end \
▷  ▷  ▷  -W \
▷  ▷  ▷  -problem_label spreads_25 \
▷  ▷  -end \
▷  ▷  -define F -finite_field -q 5 -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷  ▷  -define C -spread_classifier \
▷  ▷  ▷  -projective_space P \
▷  ▷  ▷  -k 2 \
▷  ▷  ▷  -starter_size 5 \
▷  ▷  ▷  -recoordinatize \
▷  ▷  ▷  -poset_classification_control Control \
▷  ▷  ▷  -output_prefix "" \
▷  ▷  -end \
▷  ▷  -with C -do -spread_classify_activity \
▷  ▷  ▷  -compute_starter \
▷  ▷  -end \
▷  ▷  -with C -do -spread_classify_activity \
▷  ▷  ▷  -isomorph \
▷  ▷  ▷  ▷  -prefix_iso "./spreads_25" \
▷  ▷  ▷  ▷  -use_database_for_starter \
▷  ▷  ▷  ▷  -build_db \
▷  ▷  ▷  ▷  -solution_prefix "" \
▷  ▷  ▷  ▷  -base_fname "" \
▷  ▷  ▷  -end \
▷  ▷  -end \
▷  ▷  -with C -do -spread_classify_activity \
▷  ▷  ▷  -isomorph \
▷  ▷  ▷  ▷  -prefix_iso "./spreads_25" \
▷  ▷  ▷  ▷  -use_database_for_starter \
▷  ▷  ▷  ▷  -read_solutions \
▷  ▷  ▷  ▷  -solution_prefix "" \
▷  ▷  ▷  ▷  -base_fname "spreads_25_graph" \
▷  ▷  ▷  -end \
▷  ▷  -end \
▷  ▷  -with C -do -spread_classify_activity \
▷  ▷  ▷  -isomorph \
▷  ▷  ▷  ▷  -prefix_iso "./spreads_25" \
▷  ▷  ▷  ▷  -use_database_for_starter \
▷  ▷  ▷  ▷  -compute_orbits \
▷  ▷  ▷  ▷  -solution_prefix "" \
▷  ▷  ▷  ▷  -base_fname "spreads_25_graph" \
▷  ▷  ▷  -end \
```

Spreads in the Orbiter Catalogue

OCN	|Aut|	Name
0	1008	
1	1008	
2	1728	
3	216	
4	360	
5	288	
6	3600	
7	244800	

Table 14.8: Spreads in PG(7, 2) in the Orbiter Catalogue

```
▷  ▷  -end \
▷  ▷  -with C -do -spread_classify_activity \
▷  ▷  ▷  -isomorph \
▷  ▷  ▷  ▷  -prefix_iso "./spreads_25" \
▷  ▷  ▷  ▷  -use_database_for_starter \
▷  ▷  ▷  ▷  -isomorph_testing \
▷  ▷  ▷  ▷  -solution_prefix "" \
▷  ▷  ▷  ▷  -base_fname "spreads_25_graph" \
▷  ▷  ▷  -end \
▷  ▷  -end \
▷  ▷  -with C -do -spread_classify_activity \
▷  ▷  ▷  -isomorph \
▷  ▷  ▷  ▷  -prefix_iso "./spreads_25" \
▷  ▷  ▷  ▷  -use_database_for_starter \
▷  ▷  ▷  ▷  -isomorph_report \
▷  ▷  ▷  ▷  -solution_prefix "" \
▷  ▷  ▷  ▷  -base_fname "spreads_25_graph" \
▷  ▷  ▷  -end \
▷  ▷  -end
▷  pdflatex spreads_25_isomorphism_types.tex
▷  $(OPEN) spreads_25_isomorphism_types.pdf
```

performs the final isomorph rejection on the spreads arising from the rainbow cliques in all cases. It results in a transversal of the isomorphism classes of spreads of PG(3, 5). In total, 21 spreads are found. This agrees with the results in the literature, see [24].

Table 14.8 lists the solid spreads in PG(7, 2) according to their Orbiter catalogue number (OCN).

Activities for Translation Planes

Command	Arguments	Purpose
-export_incma		Export the incidence matrix to file.
-p_rank	p	Compute the p-rank of the incidence matrix for the given prime p.
-report		Produce a latex report.

Table 14.9: Activities for Translation Planes

14.2 Translation Planes

Orbiter can create translation planes from spreads. The construction of translation planes from spreads is due to André and Bruck, Bose (cf. [3, 18]). In order to perform the construction, we need a field $F = \mathbb{F}_q$ which is the kernel of the plane, a spread of k-subspaces, and the groups $\mathrm{P\Gamma L}(2k, F)$ and $\mathrm{P\Gamma L}(2k+1, F)$.

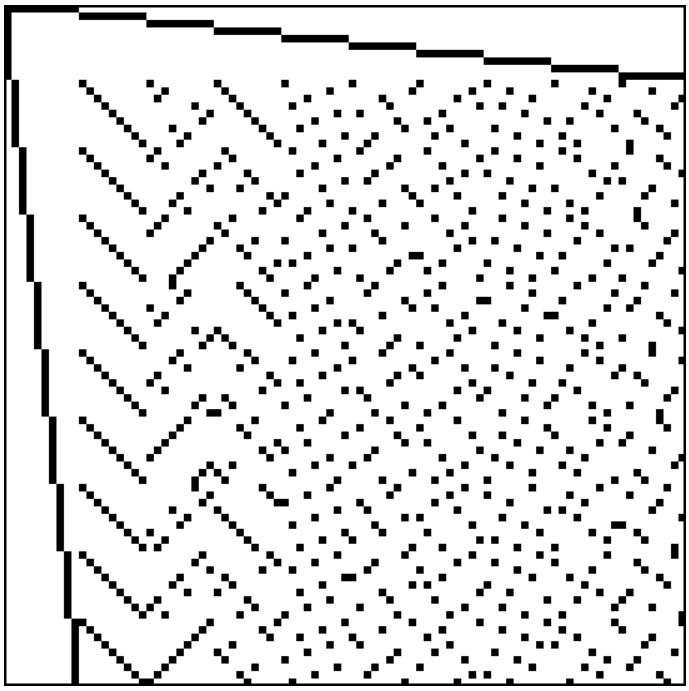
Table 14.9 lists Orbiter activities for translation planes.
For instance, the command

Example 588

```
create_translation_plane_9b:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 3 -end \
▷  ▷  -define G -linear_group -PGL 4 F -end \
▷  ▷  -define G1 -linear_group -PGL 5 F -end \
▷  ▷  -define S -spread -kernel_field F \
▷  ▷  ▷  -group G -k 2 -catalogue 1 \
▷  ▷  ▷  -end \
▷  ▷  -define T -translation_plane S G G1 -end \
▷  ▷  -with T -do -translation_plane_activity \
▷  ▷  ▷  -export_incma \
▷  ▷  -end \
▷  ▷  -with T -do -translation_plane_activity \
▷  ▷  ▷  -report \
▷  ▷  -end \
▷  ▷  -define A -linear_group -import_group_of_plane T -end \
▷  ▷  -define Orb -orbits -group A \
▷  ▷  ▷  -on_points \
▷  ▷  -end \
▷  ▷  -with Orb -do -orbits_activity \
▷  ▷  ▷  -report \
▷  ▷  -end \
▷  ▷  -with Orb -do -orbits_activity \
▷  ▷  ▷  -stabilizer 92 \
▷  ▷  -end \
▷  ▷  -with Orb -do -orbits_activity \
▷  ▷  ▷  -export_trees \
▷  ▷  -end
▷  $(ORBITER) -v 2 \
▷  ▷  -draw_matrix \
▷  ▷  ▷  -input_csv_file plane_catalogue_q3_k2_1_incma.csv \
▷  ▷  ▷  -box_width 6 -bit_depth 8 \
```

```
▷ ▷ ▷ -partition 2 91 91 \
▷ ▷ -end
▷ $(ORBITER) -v 3 \
▷ ▷ -draw_layered_graph \
▷ ▷ ▷ orbit_PGL_5_3_on_andre_3.layered_graph \
▷ ▷ ▷ -radius 250 -spanning_tree -embedded -nodes_empty \
▷ ▷ ▷ -line_width 1.1 -x_stretch 2.4 -scale 0.15 \
▷ ▷ -end
▷ pdflatex orbit_PGL_5_3_on_andre_3_draw.tex
▷ $(OPEN) orbit_PGL_5_3_on_andre_3_draw.pdf
▷ pdflatex group_of_plane_plane_catalogue_q3_k2_1_orbits_report.tex
▷ $(OPEN) group_of_plane_plane_catalogue_q3_k2_1_orbits_report.pdf
▷ pdflatex group_of_plane_plane_catalogue_q3_k2_1_stab_pt_92_report.tex
▷ $(OPEN) group_of_plane_plane_catalogue_q3_k2_1_stab_pt_92_report.pdf
```

creates the (projective) Hall plane of order 9 from the Hall spread. In this example, we use the fact that $\mathrm{P\Gamma L}(n,q) = \mathrm{PGL}(n,q)$ if q is prime. The example also creates a bitmap drawing of the incidence matrix of the plane, shown below:



In the next example, we will create a translation plane of order 16 with kernel of order 4:

Example 589

```
create_translation_plane_16_4_0:
▷ $(ORBITER) -v 3 \
▷ ▷ -define F -finite_field -q 4 -end \
▷ ▷ -define G -linear_group -PGGL 4 F -end \
```

```
▷  ▷  -define G1 -linear_group -PGGL 5 F -end \
▷  ▷  -define S -spread -kernel_field F \
▷  ▷  ▷  -group G -k 2 -catalogue 0 \
▷  ▷  ▷  -end \
▷  ▷  -define T -translation_plane S G G1 -end \
▷  ▷  -with T -do -translation_plane_activity \
▷  ▷  ▷  -export_incma \
▷  ▷  -end
▷  $(ORBITER) -v 2 \
▷  ▷  -draw_matrix \
▷  ▷  ▷  -input_csv_file plane_catalogue_q4_k2_0_incma.csv \
▷  ▷  ▷  -box_width 6 -bit_depth 8 \
▷  ▷  ▷  -partition 4 273 273 \
▷  ▷  -end
▷  $(OPEN) plane_catalogue_q4_k2_0_incma_draw.bmp
```

This plane is the Hall plane, and the spread is the Hall spread. The stabilizer of the spread has order 1200.

In the next example, we will create a translation plane of order 16 with kernel of order 2:

Example 590

```
create_translation_plane_16_2_0:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define G -linear_group -PGL 8 F -end \
▷  ▷  -define G1 -linear_group -PGL 9 F -end \
▷  ▷  -define S -spread -kernel_field F \
▷  ▷  ▷  -group G -k 4 -catalogue 0 \
▷  ▷  ▷  -end \
▷  ▷  -define T -translation_plane S G G1 -end \
▷  ▷  -with T -do -translation_plane_activity \
▷  ▷  ▷  -export_incma \
▷  ▷  -end
▷  $(ORBITER) -v 2 \
▷  ▷  -draw_matrix \
▷  ▷  ▷  -input_csv_file plane_catalogue_q2_k4_0_incma.csv \
▷  ▷  ▷  -box_width 6 -bit_depth 8 \
▷  ▷  ▷  -partition 4 273 273 \
▷  ▷  -end
▷  $(OPEN) plane_catalogue_q2_k4_0_incma_draw.bmp
```

The spread has a stabilizer of order 1008, which means that the associated translation plane has a stabilizer of order $1008 \cdot 256 = 258045$. According to [56], there are two planes whose associated spreads have an automorphism group of this order. They can be distinguished by the 2-rank of their incidence matrices. The Johnson-Walker plane has a 2-rank of 100. The Lorimer-Rahilly plane has a 2-rank of 106. Using Orbiter, we compute the 2-rank of the translation plane that we have created:

Example 591

```
RREF_plane_16_2_0_rank_of_incma:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define v -vector -field F  \
▷  ▷  ▷  -file plane_catalogue_q2_k4_0_incma.csv \
▷  ▷  -end \
▷  ▷  -with F -do -finite_field_activity \
▷  ▷  -RREF v \
▷  ▷  -end
```

It turns out that the 2-rank of our plane is 106, so the plane is Lorimer-Rahilly.

Let us investigate the Rao / Rao plane from Section 14.1, which we know is isomorphic to the spread in the Orbiter catalogue with number 0 and projective stabilizer of order 84. The command

Example 592

```
create_translation_plane_27_Rao_Rao:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 3 -end \
▷  ▷  -define SS -vector -dense $(SPREAD_SET_27_RAO_RAO) -end \
▷  ▷  -define G -linear_group -PGL 6 F -end \
▷  ▷  -define G1 -linear_group -PGL 7 F -end \
▷  ▷  -define S -spread -kernel_field F \
▷  ▷  ▷  -group G -k 3 -spread_set "Rao_Rao" "Rao\_Rao" SS \
▷  ▷  ▷  -end \
▷  ▷  -define T -translation_plane S G G1 -end \
▷  ▷  -with T -do -translation_plane_activity \
▷  ▷  ▷  -export_incma \
▷  ▷  -end
```

creates the translation plane from the spread. To compute the 3-rank of the incidence matrix, we issue the following comand:

Example 593

```
RREF_Rao_Rao_plane_incma_rank:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 3 -end \
▷  ▷  -define v -vector -field F  \
▷  ▷  ▷  -file plane_Rao_Rao_incma.csv \
▷  ▷  -end \
▷  ▷  -with F -do -finite_field_activity \
▷  ▷  ▷  -RREF v \
▷  ▷  -end
```

The 3-rank turns out to be 271. According to the Moorhouse tables [57], the plane has Moorhouse number IV.

Suppose we want to study the orbits of the group of a translation plane on points and blocks. The following command creates the translation plane of order 27 with Orbiter number 4. Once done, it creates the action on points and blocks and invokes the Schreier orbit algorithm (see Section 7.1).

Example 594

```
translation_plane_27_4_orbits:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 3 -end \
▷  ▷  -define G -linear_group -PGL 6 F -end \
▷  ▷  -define G1 -linear_group -PGL 7 F -end \
▷  ▷  -define S -spread -kernel_field F \
▷  ▷  ▷  -group G -k 3 -catalogue 4 \
▷  ▷  ▷  -end \
▷  ▷  -define T -translation_plane S G G1 -end \
▷  ▷  -with T -do -translation_plane_activity \
▷  ▷  ▷  -export_incma \
▷  ▷  -end \
▷  ▷  -with T -do -translation_plane_activity \
▷  ▷  ▷  -report \
▷  ▷  -end \
▷  ▷  -define A -linear_group -import_group_of_plane T -end \
▷  ▷  -define Orb -orbits -group A \
▷  ▷  ▷  -on_points \
▷  ▷  -end
```

The group of the plane has order 1592136. The degree of the action is 1514. The orbit lengths are

1, 28, 729, 756.

Creating Packings with Assumed Symmetry		
Command	Arguments	Purpose
-H	description	Specify the assumed group H of symmetries. The orbits of H on the set of spreads are considered. The packings will be constructed as union of orbits.
-N	description	Specify the normalizer of H.
-cliques_on_fixpoint_graph	s	Using poset classification, classify the orbits of N on cliques of size $\leq s$ in the graph on fixed points.
-cliques_on_fixpoint_graph_control	descr	Specify poset classification options related to the classification of cliques on the fixed point graph as in Table 7.4.
-fixp_clique_types_save_individually		Sort the cliques on fixed points by the type of their spreads and write one csv file for each possible type containing the index of the cliques of the given type.
-process_long_orbits	descr	Proceed on to long orbits using Table 14.13.
-spread_tables_prefix	P	Use prefix P to access spread tables.
-report		Create a report of the classification process.
-regular_packing		Initialize Klein correspondence and identify (regular) spreads with external lines to the Klein quadric using the polarity of the Klein quadric.

Table 14.10: Creating Packings with Assumed Symmetry

14.3 Packings

A packing of PG$(3, q)$ is a set of pairwise line-disjoint spreads of PG$(3, q)$ of size $q^2 + q + 1$. Each spread contains $q^2 + 1$ lines. A simple counting argument shows that every line is contained in exactly one spread of the packing. The classification problem for packings is the problem of determining a complete set of pairwise non-isomorphic packings. Orbiter can be used to classify packings for small parameters. It is sometimes useful to make a symmetry assumption. This means that only those packings will be found that satisfy the symmetry assumption. The reason for making such an assumption is that the problem becomes easier and hence more tractable. Often, an assumption is made that the packings are invariant under a (nontrivial) group H. This section describes various ways in which Orbiter can help find and classify packings, with or without symmetry assumption.

Table 14.10 list Orbiter commands related to the construction of packings with assumed symmetry.

Table 14.11 list Orbiter activities for packings.

Table 14.12 list Orbiter activities for packings with assumed symmetry and with fixpoints.

Table 14.13 list Orbiter commands related to the construction of packings with assumed symmetry by picking long orbits.

The following command creates a table of all labeled spreads in PG$(3, 4)$. There are three isomorphism types of spreads in PG$(3, 4)$. The command computes the orbits of each. In total, this gives 5096448 labeled spreads.

Activities for Packings		
Command	Arguments	Purpose
-report		Produce a latex report
-export_reduced_spread_orbits	fname	Export reduced spread orbits.
-create_graph_on_mixed_orbits	L	Create the graph of orbits whose length is listed in L.

Table 14.11: Activities for Packings

Activities for Packings with Fixpoints		
Command	Arguments	Purpose
-report		Produce a latex report
-print_packing	text	Print the packing.
-compare_files_of_packings	fname1 fname2	Compare packings in the two given files.

Table 14.12: Activities for Packings with Fixpoints

Creating Packings from Long Orbits		
Command	Arguments	Purpose
-split	r m	Define a subset of cases of fixed point cliques to be worked on. Only those cases whose number is congruent to r modulo m are considered.
-orbit_length	l	Use orbits of length l.
-mixed_orbits	length	
-list_of_cases_from_file	fname	Define a subset of cases of fixed point cliques to be worked on. Only the cases listed the given file are considered.
-solution_path	P	Use P as a prefix for all solution files.
-create_graphs		For each case, create the graph that describes whether two orbits of length l are compatible.
-solve		Perform clique finding and write solutions to file.
-read_solutions		Read solutions from file.

Table 14.13: Creating Packings from Long Orbits

Example 595

```
spread_table_PG_3_4:
▷  - mkdir SPREAD_TABLES_4
▷  $(ORBITER) -v 6 \
▷  ▷  -define Control -poset_classification_control \
▷  ▷  ▷  -problem_label spreads_PG_3_4 \
▷  ▷  -end \
▷  ▷  -define F -finite_field -q 4 -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷  ▷  -define T -spread_table P 2 "0,1,2" "SPREAD_TABLES_4/" Control
```

There are 21 isomorphism types of spreads in PG$(3,5)$. The regular spread has Orbiter catalogue number equal to 12. The following command creates a table of all regular spreads in PG$(3,5)$:

Example 596

```
spread_table_PG_3_5_regular:
▷  - mkdir SPREAD_TABLES_5_REG
▷  $(ORBITER) -v 6 \
▷  ▷  -define Control -poset_classification_control \
▷  ▷  ▷  -problem_label spreads_PG_3_5 \
▷  ▷  -end \
▷  ▷  -define F -finite_field -q 5 -end \
▷  ▷  -define P -projective_space -n 3 -field F -end \
▷  ▷  -define T -spread_table P 2 "12" "SPREAD_TABLES_5_REG/" Control \
▷  ▷  -print_symbols
```

There are 155000 regular spreads.

Suppose we are interested in packings invariant under a group of order 31. We pick the matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 3 & 4 & 3 \\ 0 & 3 & 3 & 4 \\ 0 & 3 & 2 & 3 \end{bmatrix}$$

of order 124 in GL$(4,5)$. At first, we store the element in a makefile variable:

Example 597

```
PGL_4_5_SUBGROUP_31_ME=-PGL 4 5 \
▷  -subgroup_by_generators "31" 31 1 \
▷  "1,0,0,0, 0,3,4,3, 0,3,3,4, 0,3,2,3"
```

We compute the normalizer of the cyclic subgroup generated by this element inside GL$(4,5)$:

Example 598

```
PG_3_5_element_of_order_31_GL_normalizer:
▷  $(ORBITER) -v 6 -define G \
▷  ▷  -linear_group -GL 4 5 -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -normalizer_of_cyclic_subgroup "124" \
▷  ▷  ▷  "1,0,0,0, 0,3,4,3, 0,3,3,4, 0,3,2,3" \
▷  ▷  -end
▷  pdflatex normalizer_of_124_in_GL_4_5.tex
▷  $(OPEN) normalizer_of_124_in_GL_4_5.pdf
```

We compute the normalizer of the cyclic subgroup generated by this element inside $\mathrm{PGL}(4,5)$:

Example 599

```
PG_3_5_element_of_order_31_ME_normalizer:
▷  $(ORBITER) -v 6 -define G \
▷  ▷  -linear_group -PGL 4 5 -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -normalizer_of_cyclic_subgroup "31" \
▷  ▷  ▷  "1,0,0,0, 0,3,4,3, 0,3,3,4, 0,3,2,3" \
▷  ▷  -end
▷  mv normalizer_of_31_in_PGL_4_5.tex normalizer_of_31_ME_in_PGL_4_5.tex
▷  pdflatex normalizer_of_31_ME_in_PGL_4_5.tex
▷  $(OPEN) normalizer_of_31_ME_in_PGL_4_5.pdf
```

We store the normalizer in a makefile variable:

Example 600

```
PGL_4_5_SUBGROUP_31_ME_NORMALIZER=-PGL 4 5 \
▷  -subgroup_by_generators "normalizer_31" "372" 4 \
▷  "1,0,0,0,0,4,0,0,0,0,4,0,0,0,0,4, \
▷  1,0,0,0,0,3,0,0,0,0,3,0,0,0,0,3, \
▷  1,0,0,0,0,4,0,0,0,0,2,1,0,3,2,4, \
▷  1,0,0,0,0,0,1,0,0,0,0,1,0,1,1,3,"
```

We create packings invariant under the cyclic group of order 31 generated by this element.

Example 601

```
PG_3_5_assume_31_graph:
▷  $(ORBITER) -v 5 \
▷  ▷  -define Control -poset_classification_control \
▷  ▷  ▷  -problem_label spreads_PG_3_5 \
▷  ▷  -end \
▷  ▷  -define F -finite_field -q 5 -end \
```

```
▷ ▷ -define P3 -projective_space -n 3 -field F -v 0 -end \
▷ ▷ -define P5 -projective_space -n 5 -field F -v 0 -end \
▷ ▷ -define T -spread_table P3 2 "12" "SPREAD_TABLES_5_REG/" Control \
▷ ▷ -define PC -packing_classify P3 P5 T \
▷ ▷ -define PW -packing_with_symmetry_assumption PC \
▷ ▷ ▷ ▷ -H "H31" $(PGL_4_5_SUBGROUP_31_ME) -end \
▷ ▷ ▷ ▷ -N "N31" $(PGL_4_5_SUBGROUP_31_ME_NORMALIZER) -end \
▷ ▷ ▷ -end \
▷ ▷ -define PWF -packing_choose_fixed_points PW 0 -end \
▷ ▷ -define L -packing_long_orbits PWF \
▷ ▷ ▷ -orbit_length 31 -create_graphs \
▷ ▷ ▷ -end \
▷ ▷ -print_symbols
```

Creating BLT-Sets		
Command	Arguments	Purpose
-catalogue	i	Create the BLT-set number i from the Orbiter catalogue (i is zero-based).
-family	name	Create a BLT-set from the named family. See Table 14.15 for possibilities for name.
-flock	flock	Create a BLT-set from a given flock.
-space	space	Set the space.
-invariants		Compute isomorphism invariants.

Table 14.14: Creating BLT-Sets

Families of BLT-sets		
Command	Condition	Purpose
Linear		Linear BLT-set.
Fisher		Fisher BLT-set [29].
Mondello	$q \equiv \pm 1 \mod 10$	Mondello BLT-set due to Penttila [59].
FTWKB	$q \equiv \pm 2 \mod 3$	Fisher, Thas, Walker [71], Kantor, Betten [15] BLT-set.
Kantor1	$q = p^e,\ e > 1$	Kantor's first family.
Kantor2	$q \equiv \pm 2 \mod 5$	Kantor's second family.
LP_37_72	$q = 37$	BLT-set for $q = 37$ with ago=72 due to Law and Penttila [48].
LP_37_41a	$q = 37$	First BLT-set for $q = 37$ with ago=4, due to Law and Penttila [48].
LP_37_41b	$q = 37$	Second BLT-set for $q = 37$ with ago=4, due to Law and Penttila [48].
LP_71	$q = 71$	BLT-set for $q = 71$ due to Law and Penttila [48].

Table 14.15: Families of BLT-sets

14.4 BLT-Sets

A BLT-set of $Q(4, q)$ is a set of $q + 1$ point on the quadric such that no point on the quadric is collinear to more than two points of the set. BLT sets are related to spreads of PG$(3, q)$, to flocks of the quadratic cone in PG$(3, q)$, and other objects in finite geometry. These sets have been introduced in [4] for the purposes of creating new flocks from old. A BLT-set exists if and only if q is odd. It is an interesting problem to classify BLT-sets of $Q(4, q)$ under the orthogonal group. Some references are Law [46], Penttila-Royle [60], Penttila-Law [47, 48], Betten [9], AlAzemi-Betten-Chowdhury [1]. For small values of q, computer search has been used extensively. As we will discuss below, Orbiter is useful to supporting such computations.

Before we go into the problem of classification, let us first discuss how known BLT-sets can be constructed in Orbiter. Orbiter maintains a catalogue of BLT-sets over small fields. Besides that, Orbiter can be used to create instances of known families of BLT-sets.

Table 14.14 shows options to create known BLT-sets.

Table 14.15 shows names for known families of sporadic sets.

BLT-Set Activity		
Command	Arguments	Purpose
-report		Produce a latex report.
-export_gap	index	Export to GAP/fining based on the indexed point.
-create_flock		Create a flock.
-BLT_test		Test of the set has the BLT-property.
-export_set_in_PG		Export the set in ranks of the underlying projective space.
-plane_invariant		Compute the plane invariant.

Table 14.16: BLT-Set Activity

Classifying BLT-Sets		
Command	Arguments	Purpose
-starter_size	s	Set the size of partial BLT-sets that is used as subobjects.

Table 14.17: Classifying BLT-Sets

Table 14.16 shows Orbiter activities for a BLT-set object.
Table 14.17 shows options to prepare for the classification of BLT-sets.
Table 14.18 shows commands for the classification of BLT-sets.

Let us demonstrate some examples. The command

Example 602

```
BLT_5_1:
▷  $(ORBITER) -v 10 \
▷  ▷  -define F -finite_field -q 5 -end \
▷  ▷  -define O -orthogonal_space 0 5 F -end \
▷  ▷  -define BLT -BLT_set \
▷  ▷  ▷  -space O -catalogue 1 -invariants \
▷  ▷  -end \
```

Activities for the Classification of BLT-sets		
Command	Arguments	Purpose
-compute_starter		Perform classification of partial spreads.
-poset_classification_activity	label	
-create_graphs		Prepare to lift all cases of partial BLT-sets by creating the associated colored graphs.
-split	r m	Lift only those cases k with $k \equiv r \mod m$.
-isomorph	options	Perform isomorphism testing after lifting is complete.

Table 14.18: Activities for the Classification of BLT-sets

```
▷  ▷  -with BLT -do -blt_set_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex BLT_catalogue_q5_iso1.tex
▷  $(OPEN) BLT_catalogue_q5_iso1.pdf
```

creates the first BLT-set of order 5 from the catalogue. Suppose we want to export the BLT-set to GAP/fining. The following command can do that:

Example 603

```
BLT_5_1_export_gap:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 5 -end \
▷  ▷  -define O -orthogonal_space 0 5 F -end \
▷  ▷  -define BLT -BLT_set \
▷  ▷  ▷  -space O -catalogue 1 \
▷  ▷  -end \
▷  ▷  -with BLT -do -blt_set_activity \
▷  ▷  ▷  -export_gap \
▷  ▷  -end
```

The command creates the following GAP/fining file:

```
LoadPackage("fining");
# BLT-set catalogue_q5_iso1
# Quadratic form: X_0^2 + X_1X_2 + X_3X_4
# Group of order 720
mat1 := [[Z(5)^2,0*Z(5),0*Z(5),Z(5)^3,0*Z(5)],
[0*Z(5),Z(5)^2,0*Z(5),0*Z(5),0*Z(5)],
[0*Z(5),0*Z(5),Z(5)^2,0*Z(5),0*Z(5)],
[0*Z(5),0*Z(5),0*Z(5),Z(5)^0,0*Z(5)],
[Z(5)^2,0*Z(5),0*Z(5),Z(5)^2,Z(5)^0]];
frob1 := FrobeniusAutomorphism(GF(5))^0;
psi1 := ProjectiveSemilinearMap(mat1, frob1,GF(5));
mat2 := [[Z(5)^0,0*Z(5),0*Z(5),Z(5)^2,0*Z(5)],
[0*Z(5),Z(5)^2,0*Z(5),0*Z(5),0*Z(5)],
[0*Z(5),0*Z(5),Z(5)^2,0*Z(5),0*Z(5)],
[0*Z(5),0*Z(5),0*Z(5),Z(5)^1,0*Z(5)],
[Z(5)^2,0*Z(5),0*Z(5),Z(5)^3,Z(5)^3]];
frob2 := FrobeniusAutomorphism(GF(5))^0;
psi2 := ProjectiveSemilinearMap(mat2, frob2,GF(5));
mat3 := [[Z(5)^3,0*Z(5),0*Z(5),Z(5)^1,Z(5)^0],
[0*Z(5),Z(5)^2,0*Z(5),0*Z(5),0*Z(5)],
[0*Z(5),0*Z(5),Z(5)^2,0*Z(5),0*Z(5)],
[Z(5)^3,0*Z(5),0*Z(5),Z(5)^2,Z(5)^2],
[Z(5)^0,0*Z(5),0*Z(5),Z(5)^0,Z(5)^2]];
frob3 := FrobeniusAutomorphism(GF(5))^0;
psi3 := ProjectiveSemilinearMap(mat3, frob3,GF(5));
mat4 := [[Z(5)^3,0*Z(5),0*Z(5),Z(5)^0,Z(5)^1],
```

```
[0*Z(5),Z(5)^2,0*Z(5),0*Z(5),0*Z(5)],
[0*Z(5),0*Z(5),Z(5)^2,0*Z(5),0*Z(5)],
[Z(5)^2,0*Z(5),0*Z(5),Z(5)^0,Z(5)^2],
[Z(5)^1,0*Z(5),0*Z(5),Z(5)^0,Z(5)^0]];
frob4 := FrobeniusAutomorphism(GF(5))^0;
psi4 := ProjectiveSemilinearMap(mat4, frob4,GF(5));
mat5 := [[Z(5)^1,Z(5)^0,0*Z(5),Z(5)^1,Z(5)^0],
[0*Z(5),Z(5)^0,0*Z(5),0*Z(5),0*Z(5)],
[Z(5)^3,Z(5)^3,Z(5)^0,Z(5)^0,Z(5)^3],
[Z(5)^3,Z(5)^3,0*Z(5),Z(5)^0,Z(5)^0],
[Z(5)^0,Z(5)^0,0*Z(5),Z(5)^2,Z(5)^0]];
frob5 := FrobeniusAutomorphism(GF(5))^0;
psi5 := ProjectiveSemilinearMap(mat5, frob5,GF(5));
mat6 := [[Z(5)^1,0*Z(5),0*Z(5),Z(5)^3,Z(5)^2],
[Z(5)^0,Z(5)^0,Z(5)^1,Z(5)^1,Z(5)^0],
[0*Z(5),0*Z(5),Z(5)^0,0*Z(5),0*Z(5)],
[Z(5)^3,0*Z(5),Z(5)^1,Z(5)^3,Z(5)^1],
[Z(5)^0,0*Z(5),Z(5)^2,Z(5)^3,Z(5)^3]];
frob6 := FrobeniusAutomorphism(GF(5))^0;
psi6 := ProjectiveSemilinearMap(mat6, frob6,GF(5));
gens := [psi1, psi2, psi3, psi4, psi5, psi6];
G := Group(gens);
Size(G);
pg := ProjectiveSpace(4,5);
S:=[
[0*Z(5),Z(5)^0,0*Z(5),0*Z(5),0*Z(5)],
[0*Z(5),0*Z(5),Z(5)^0,0*Z(5),0*Z(5)],
[0*Z(5),Z(5)^0,Z(5)^1,Z(5)^2,Z(5)^1],
[Z(5)^0,Z(5)^0,Z(5)^1,Z(5)^1,Z(5)^0],
[Z(5)^0,Z(5)^3,Z(5)^0,Z(5)^2,Z(5)^2],
[Z(5)^0,Z(5)^3,Z(5)^0,Z(5)^3,Z(5)^1]
];
S := List(S,x -> VectorSpaceToElement(pg,x));
```

The command

Example 604

```
BLT_11_0:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 11 -end \
▷  ▷  -define O -orthogonal_space 0 5 F -end \
▷  ▷  -define BLT -BLT_set \
▷  ▷  ▷  -space O -catalogue 0 -invariants \
▷  ▷  -end \
▷  ▷  -with BLT -do -blt_set_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex BLT_catalogue_q11_iso0.tex
▷  $(OPEN) BLT_catalogue_q11_iso0.pdf
```

creates the BLT-set #0 in $Q(4, 11)$. The command

Example 605

```
BLT_11_Mondello:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 11 -end \
▷  ▷  -define O -orthogonal_space 0 5 F -end \
▷  ▷  -define BLT -BLT_set \
▷  ▷  ▷  -space O -family "Mondello" -invariants \
▷  ▷  -end \
▷  ▷  -with BLT -do -blt_set_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex BLT_Mondello_q11.tex
▷  $(OPEN) BLT_Mondello_q11.pdf
```

creates the Mondello BLT-set in $Q(4,11)$. Orbiter creates the following report:

The quadratic form is:

$$X_0^2 + X_1X_2 + X_3X_4 = 0$$

The BLT-set is:

i	Rank	Point	(a,b,c)
0	846	$(1,6,4,10,3)$	$(22,11,1)$
1	851	$(1,5,7,10,3)$	$(22,110,1)$
2	1234	$(1,5,1,7,7)$	$(37,11,1)$
3	613	$(1,6,10,5,1)$	$(73,110,1)$
4	1307	$(1,1,3,8,5)$	$(59,36,1)$
5	1418	$(1,3,9,6,10)$	$(95,36,1)$
6	1022	$(1,9,5,10,2)$	$(99,96,1)$
7	835	$(1,2,6,3,3)$	$(99,36,1)$
8	950	$(1,10,8,2,9)$	$(95,96,1)$
9	789	$(1,8,2,4,4)$	$(59,96,1)$
10	611	$(1,7,7,5,1)$	$(73,11,1)$
11	1236	$(1,4,4,7,7)$	$(37,110,1)$

Plane intersection type is $4^{18}\ 3^{148}$
Plane invariant is too big (18 planes)

$\rightarrow$	18_1
12_0	6

$\downarrow$	18_1
12_0	4

$C_0 = \{0,1,2,3,4,5,6,7,8,9,10,11\}_{12}$
$C_1 = \{0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17\}_{18}$

$\rightarrow$	18_1
12_0	6

$\downarrow$	18_1
12_0	4

$C_0 = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}_{12}$
$C_1 = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17\}_{18}$

The classification of BLT-sets is a difficult problem. One approach is by means of the poset of partial BLT-sets. The orbits at level $q + 1$ in the poset of partial BLT-sets are the isomorphims classes of BLT-sets. We wish to illustrate this method by an example. The example is quite small, in order to allow us to compare different construction algorithms.
We begin with a classification of the complete poset of partial BLT sets. This method is very limited, as the number of orbits of partial BLT-sets gets out of hand quickly as q increases.

Example 606

```
BLT_13_deep_search:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 13 -end \
▷  ▷  -define O -orthogonal_space 0 5 F -end \
▷  ▷  -define C -BLT_set_classifier O -starter_size 14 -end \
▷  ▷  -with C -do -BLT_set_classify_activity \
▷  ▷  ▷  -compute_starter \
▷  ▷  ▷  ▷  -problem_label BLT_q13 \
▷  ▷  ▷  ▷  -W -depth 14 \
▷  ▷  ▷  -end \
▷  ▷  -end
▷  #pdflatex BLT_q13_poset.tex
▷  #$(OPEN) BLT_q13_poset.pdf
```

The technique of classification via substructure can help. Various authors have described different versions of this technique, see [1, 9, 46]. As an example, we classify the BLT-sets of order 13 with the technique of substructures and rainbow cliques.

We consider the classification problem of BLT-sets of order 13. We use the technique of substructures of size 5. So, to begin with, we classify all partial BLT-sets of size 5. For each isomorphism type of partial BLT-set of size 5, we create a colored graph. The vertices in the graph are the points in $Q(4, 13)$ which can be added to the partial BLT-set. Two vertices are connected if they can both be added. Here is the command:

Example 607

```
BLT_13_classify_starter:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 13 -end \
▷  ▷  -define O -orthogonal_space 0 5 F -end \
▷  ▷  -define C -BLT_set_classifier O -starter_size 5 -end \
▷  ▷  -with C -do -BLT_set_classify_activity \
▷  ▷  ▷  -compute_starter \
▷  ▷  ▷  ▷  -problem_label BLT_q13 \
▷  ▷  ▷  ▷  -W -depth 5 \
▷  ▷  ▷  -end \
▷  ▷  -end \
```

```
▷  ▷  -with C -do -BLT_set_classify_activity \
▷  ▷  ▷  -create_graphs \
▷  ▷  -end
```

In the next step, we compute all rainbow cliques in each of the graphs:

Example 608

```
BLT_13_cliques:
▷  $(ORBITER) -v 2 \
▷  ▷  -loop L 0 38 1 \
▷  ▷  ▷  -define G -graph \
▷  ▷  ▷  ▷  -load BLT_q13_graph_5_%L.bin \
▷  ▷  ▷  -end \
▷  ▷  ▷  -with G -do \
▷  ▷  ▷  -graph_theoretic_activity \
▷  ▷  ▷  ▷  -find_cliques -rainbow -target_size 9 -end \
▷  ▷  ▷  -end \
▷  ▷  -end_loop L
```

Next, we create a data structure for isomorphism testing. The first step is to create a database of all partial BLT-sets of order at most 5:

Example 609

```
BLT_13_isomorph_read_DB:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 13 -end \
▷  ▷  -define O -orthogonal_space 0 5 F -end \
▷  ▷  -define C -BLT_set_classifier O -starter_size 5 -end \
▷  ▷  -with C -do -BLT_set_classify_activity \
▷  ▷  ▷  -compute_starter \
▷  ▷  ▷  ▷  -problem_label BLT_q13 \
▷  ▷  ▷  ▷  -W -depth 5 \
▷  ▷  ▷  -end \
▷  ▷  -end \
▷  ▷  -with C -do -BLT_set_classify_activity \
▷  ▷  ▷  -isomorph \
▷  ▷  ▷  ▷  -prefix_iso "./BLT_q13" \
▷  ▷  ▷  ▷  -use_database_for_starter \
▷  ▷  ▷  ▷  -build_db \
▷  ▷  ▷  ▷  -solution_prefix "" \
▷  ▷  ▷  ▷  -base_fname "" \
▷  ▷  ▷  -end \
▷  ▷  -end
```

The next step is to read the rainbow cliques from the clique finding process:

Example 610

```
BLT_13_isomorph_read_solutions:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 13 -end \
▷  ▷  -define O -orthogonal_space 0 5 F -end \
▷  ▷  -define C -BLT_set_classifier O -starter_size 5 -end \
▷  ▷  -with C -do -BLT_set_classify_activity \
▷  ▷  ▷  -compute_starter \
▷  ▷  ▷  ▷  -problem_label BLT_q13 \
▷  ▷  ▷  ▷  -W -depth 5 \
▷  ▷  ▷  -end \
▷  ▷  -end \
▷  ▷  -with C -do -BLT_set_classify_activity \
▷  ▷  ▷  -isomorph \
▷  ▷  ▷  ▷  -prefix_iso "./BLT_q13" \
▷  ▷  ▷  ▷  -use_database_for_starter \
▷  ▷  ▷  ▷  -read_solutions \
▷  ▷  ▷  ▷  -list_of_cases BLT_q13_list_of_cases_5_0_1.csv \
▷  ▷  ▷  ▷  -solution_prefix "" \
▷  ▷  ▷  ▷  -base_fname "BLT_q13_graph" \
▷  ▷  ▷  -end \
▷  ▷  -end
```

Once the cliques have been read from file, we compute the flag orbits in the relation between partial BLT-sets and full BLT-sets. For each orbit representative of a partial BLT-set, we compute the orbits of the stabilizer on the BLT-sets incident with it. The following command can be used.

Example 611

```
BLT_13_isomorph_stabilizer_orbits:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 13 -end \
▷  ▷  -define O -orthogonal_space 0 5 F -end \
▷  ▷  -define C -BLT_set_classifier O -starter_size 5 -end \
▷  ▷  -with C -do -BLT_set_classify_activity \
▷  ▷  ▷  -compute_starter \
▷  ▷  ▷  ▷  -problem_label BLT_q13 \
▷  ▷  ▷  ▷  -W -depth 5 \
▷  ▷  ▷  -end \
▷  ▷  -end \
▷  ▷  -with C -do -BLT_set_classify_activity \
▷  ▷  ▷  -isomorph \
▷  ▷  ▷  ▷  -prefix_iso "./BLT_q13" \
▷  ▷  ▷  ▷  -use_database_for_starter \
▷  ▷  ▷  ▷  -compute_orbits \
▷  ▷  ▷  ▷  -list_of_cases BLT_q13_list_of_cases_5_0_1.csv \
▷  ▷  ▷  ▷  -solution_prefix "" \
▷  ▷  ▷  ▷  -base_fname "BLT_q13_graph" \
▷  ▷  ▷  -end \
▷  ▷  -end
```

Finally, we perform isomorphism testing:

Example 612

```
BLT_13_isomorph_testing:
▷  $(ORBITER) -v 4 \
▷  ▷  -define F -finite_field -q 13 -end \
▷  ▷  -define O -orthogonal_space 0 5 F -end \
▷  ▷  -define C -BLT_set_classifier O -starter_size 5 -end \
▷  ▷  -with C -do -BLT_set_classify_activity \
▷  ▷  ▷  -compute_starter \
▷  ▷  ▷  ▷  -problem_label BLT_q13 \
▷  ▷  ▷  ▷  -W -depth 5 \
▷  ▷  ▷  -end \
▷  ▷  -end \
▷  ▷  -with C -do -BLT_set_classify_activity \
▷  ▷  ▷  -isomorph \
▷  ▷  ▷  ▷  -prefix_iso "./BLT_q13" \
▷  ▷  ▷  ▷  -use_database_for_starter \
▷  ▷  ▷  ▷  -isomorph_testing \
▷  ▷  ▷  ▷  -solution_prefix "" \
▷  ▷  ▷  ▷  -base_fname "BLT_q13_graph" \
▷  ▷  ▷  -end \
▷  ▷  -end \
▷  ▷  -with C -do -BLT_set_classify_activity \
▷  ▷  ▷  -isomorph \
▷  ▷  ▷  ▷  -prefix_iso "./BLT_q13" \
▷  ▷  ▷  ▷  -use_database_for_starter \
▷  ▷  ▷  ▷  -isomorph_report \
▷  ▷  ▷  ▷  -solution_prefix "" \
▷  ▷  ▷  ▷  -base_fname "BLT_q13_graph" \
▷  ▷  ▷  -end \
▷  ▷  -end
▷  pdflatex BLT_q13_isomorphism_types.tex
▷  $(OPEN) BLT_q13_isomorphism_types.pdf
```

This last command results in exactly three isomorphism types of BLT-sets of order 13. Of course, these results match with the results from the classification of the complete poset of partial BLT-sets before. The difference is that the second approach is computationally much less demanding, as the number of orbits to be considered is much smaller. This is because many partial BLT-sets do not complete to BLT-sets. This means that the classification of poset of partial BLT-sets wastes a lot if time classifying things that are irrelevant.

Chapter 15

Computer Science Primitives

15.1 Clique Finding

A clique in a graph $\Gamma = (V, E)$ is a subset S of the vertices such that any two elements of S are adjacent in Γ. Finding and classifying cliques in graphs is important for many applications of graph theory. Orbiter can help. The command `-find_cliques` command from Table 12.4 can be used to find all cliques in a graph. Additional options for this command are shown in Table 15.1. For instance, the cliques of size 3 in the graph `graph_v5_e7.colored_graph` from Section 12.1 can all be found using the following command:

Example 613

```
small_graph_cliques: graph_v5_e7.colored_graph
▷  $(ORBITER) -v 2 \
▷  ▷  -define G -graph -load graph_v5_e7.colored_graph -end \
▷  ▷  -with G -do \
▷  ▷  -graph_theoretic_activity \
▷  ▷  ▷  -find_cliques -target_size 3 \
▷  ▷  -end
```

It is also possible to classify all cliques under the automorphism group of the graph. This is a multi-

Finding Cliques

Command	Arguments	Purpose
`-rainbow`		Find all rainbow cliques. The size of the cliques is the number of vertex colors.
`-target_size`	s	Find all cliques of size s.
`-weighted`	s	Find weighted cliques.
`-Sajeeb`		Use the implementation by Sajeeb Chowdhury.
`-output_file`	fname	Write cliques to the named file.
`-restrictions`	$l\ r\ m$	Restricted search: At level l, restrict to all branches congruent to r modulo m. Here, $0 \leq r < m$.

Table 15.1: Finding Cliques

step process, though. At first, the automorphism group of the graph has to be computed. Then, poset classification can be invoked to classify the cliques of a certain size. Here is an example. We wish to classify the cliques in the Paley graph of order 13. The first command creates the graph and computes the automorphism group:

Example 614

```
Paley_13_aut:
▷ $(ORBITER) -v 2 \
▷ ▷ -define F -finite_field -q 13 -end \
▷ ▷ -define Gamma -graph -Paley F -end \
▷ ▷ -with Gamma -do \
▷ ▷ -graph_theoretic_activity \
▷ ▷ ▷ -automorphism_group \
▷ ▷ -end
```

The command writes a file `Paley_13_group.makefile`. The group is given using a base and strong generating set. The base consists of the two points $0, 1$. Three strong generators with respect to this base are given in a csv file. Using this group, the next command classifies all cliques of size at most 5 in the Paley graph of order 13 under the action of the automorphism group. It turns out that there are no 5-cliques, and that the largest cliques have size 3. The command shows that there is a unique orbit of 3-cliques:

Example 615

```
Paley_13_cliques_classify:
▷ $(ORBITER) -v 4 \
▷ ▷ -define Control -poset_classification_control \
▷ ▷ ▷ -W \
▷ ▷ ▷ -problem_label Paley13_cliques \
▷ ▷ ▷ -clique_test Gamma \
▷ ▷ ▷ -depth 5 \
▷ ▷ -end \
▷ ▷ -define F -finite_field -q 13 -end \
▷ ▷ -define gens -vector -file Paley_13_gens.csv -end \
▷ ▷ -define G -permutation_group \
▷ ▷ ▷ -bsgs Paley_13 "Paley\_13" 13 78 "0,1" 3 gens -end \
▷ ▷ -define Gamma -graph -Paley F -end \
▷ ▷ -define Orb -orbits -group G \
▷ ▷ ▷ -on_subsets 5 Control \
▷ ▷ -end
```

For comparison, the command

Example 616

```
Paley_13_cliques_all:
▷ $(ORBITER) -v 10 \
▷ ▷ -define F -finite_field -q 13 -end \
▷ ▷ -define Gamma -graph -Paley F -end \
▷ ▷ -with Gamma -do \
▷ ▷ -graph_theoretic_activity \
```

```
▷ ▷ ▷ -find_cliques -target_size 3 \
▷ ▷ -end
```

finds all cliques of size 3. There are exactly 26 of them. Because of the previous command, we know that they are all equivalent under the automorphism group of the graph.

Let us consider the orbital graph created in Section 12.1. We wish to study the 5-cliques. We first determine the number of 5-cliques, and then the number of orbits of 5-cliques under the automorphism group. The following command computes all 5-cliques:

Example 617

```
HJ64_cliques5:
▷ $(ORBITER) -v 6 \
▷ ▷ -define Gamma -graph \
▷ ▷ ▷ -load \
▷ ▷ ▷ Group_Perm315_Orbital_3.colored_graph \
▷ ▷ -end \
▷ ▷ -with Gamma -do \
▷ ▷ -graph_theoretic_activity \
▷ ▷ ▷ -find_cliques -target_size 5 -end \
▷ ▷ -end
```

It turns out that there are exactly 1008 5-cliques. Concerning the classification with respect to the automorphism group of the graph, we apply the following command:

Example 618

```
HJ64_cliques5_classify:
▷ $(ORBITER) -v 6 \
▷ ▷ -define Control -poset_classification_control \
▷ ▷ ▷ -W \
▷ ▷ ▷ -problem_label HJ64_cliques \
▷ ▷ ▷ -clique_test Gamma \
▷ ▷ ▷ -depth 5 \
▷ ▷ -end \
▷ ▷ -define Gamma -graph \
▷ ▷ ▷ -load \
▷ ▷ ▷ Group_Perm315_Orbital_3.colored_graph \
▷ ▷ -end \
▷ ▷ -define gens -vector \
▷ ▷ ▷ -file halljanko315_gens.csv \
▷ ▷ -end \
▷ ▷ -define G -permutation_group \
▷ ▷ -bsgs halljanko315 "File\_halljanko315" \
▷ ▷ ▷ 315 1209600 "0,1,42,95" 5 gens -end \
▷ ▷ -define Orb -orbits -group G \
▷ ▷ ▷ -on_subsets 5 Control \
▷ ▷ -end
```

This command shows that all of the 1008 5-cliques lie in one orbit under the group. The orbit representative picked by Orbiter is $\{0, 8, 31, 110, 283\}$. These numbers refer to the vertices of the graph. They are zero-based. The stabilizer of the clique has order 1200.

Let us look at the collinearity graph of $Q(4, 2)$ one more time. The following command computes the cliques of size 3:

Example 619

```
PGO_5_2_cliques: O_5_2_incidence_matrix.csv
▷  $(ORBITER) -v 3 \
▷  ▷  -define Inc -vector -file O_5_2_incidence_matrix.csv -end \
▷  ▷  -define Gamma -graph -collinearity_graph Inc -end \
▷  ▷  -with Gamma -do \
▷  ▷  -graph_theoretic_activity \
▷  ▷  ▷  -find_cliques -target_size 3 -end \
▷  ▷  -end
```

There are 15 cliques of size 3. They correspond to the lines in the geometry.

15.2 Rainbow Cliques

Consider a graph $\Gamma = (V, E)$. A vertex coloring of Γ is a map $f : V \to C$ where C is a finite set whose elements we call colors. Let $k = |C|$. A rainbow clique in Γ is a clique X such that $|X \cap f^{-1})(c)|= 1$ for all $c \in C$. This means that the clique contains exactly one vertex from every color. Rainbow cliques can be helpful in the search for combinatorial objects. This is because the conditions for rainbow cliques are stronger than the conditions for ordinary cliques, which leads to better pruning in a backtrack search algorithm for cliques. Orbiter offers two algorithms for rainbow cliques. Let us consider an example. The classification of BLT-sets can be facilitated using rainbow cliques. The following algorithm was pointed out in [9], building on earlier work of Penttila. At first, the partial BLT-sets of a certain size (usually 5) are computed. The orbit representatives of this classification are called starter sets. Then, for each orbit representative, a certain graph is defined. The graph helps us to find the BLT-sets that contain the given starter set. The graph has a vertex coloring, and the extensions correspond to rainbow cliques. Let us look at an example. We classify the BLT-sets of size 11. The first command classifies the starters of size 5 and creates the graphs. The graphs are $\Gamma_0, \ldots, \Gamma_{s-1}$, where s is the number of orbits on partial BLT-sets.

Example 620

```
BLT_11_classify_starter:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 11 -end \
▷  ▷  -define O -orthogonal_space 0 5 F -end \
▷  ▷  -define C -BLT_set_classifier O -starter_size 5 -end \
▷  ▷  -with C -do -BLT_set_classify_activity \
▷  ▷  ▷  -compute_starter \
▷  ▷  ▷  ▷  -problem_label BLT_q11 \
▷  ▷  ▷  ▷  -W -depth 5 \
▷  ▷  ▷  -end \
▷  ▷  -end \
▷  ▷  -with C -do -BLT_set_classify_activity \
▷  ▷  ▷  -create_graphs \
▷  ▷  -end
```

Consider just one graph, say the first. The following command computes the rainbow cliques in Γ_0:

Example 621

```
BLT_11_clique_0:
▷  $(ORBITER) -v 2 \
▷  ▷  -define G -graph \
▷  ▷  ▷  -load BLT_q11_graph_5_0.bin \
▷  ▷  -end \
▷  ▷  -with G -do \
▷  ▷  -graph_theoretic_activity \
▷  ▷  ▷  -find_cliques -rainbow -target_size 12 -end \
▷  ▷  -end
```

There are three rainbow cliques. The following command computes the rainbow cliques in Γ_0 using the second algorithm. It is the same algorithm as before, but the implementationn is due to Sajeeb Chowdhury. It also uses multithreading, and hence is often a lot faster than the first algorithm:

Example 622

```
BLT_11_clique_0_Sajeeb:
▷  $(ORBITER) -v 2 \
▷  ▷  -define G -graph \
▷  ▷  ▷  -load BLT_q11_graph_5_0.bin \
▷  ▷  -end \
▷  ▷  -with G -do \
▷  ▷  -graph_theoretic_activity \
▷  ▷  ▷  -find_cliques -rainbow -Sajeeb -end \
▷  ▷  -end
```

Let us look at the graph Γ_0. At first we sort the vertices by color class:

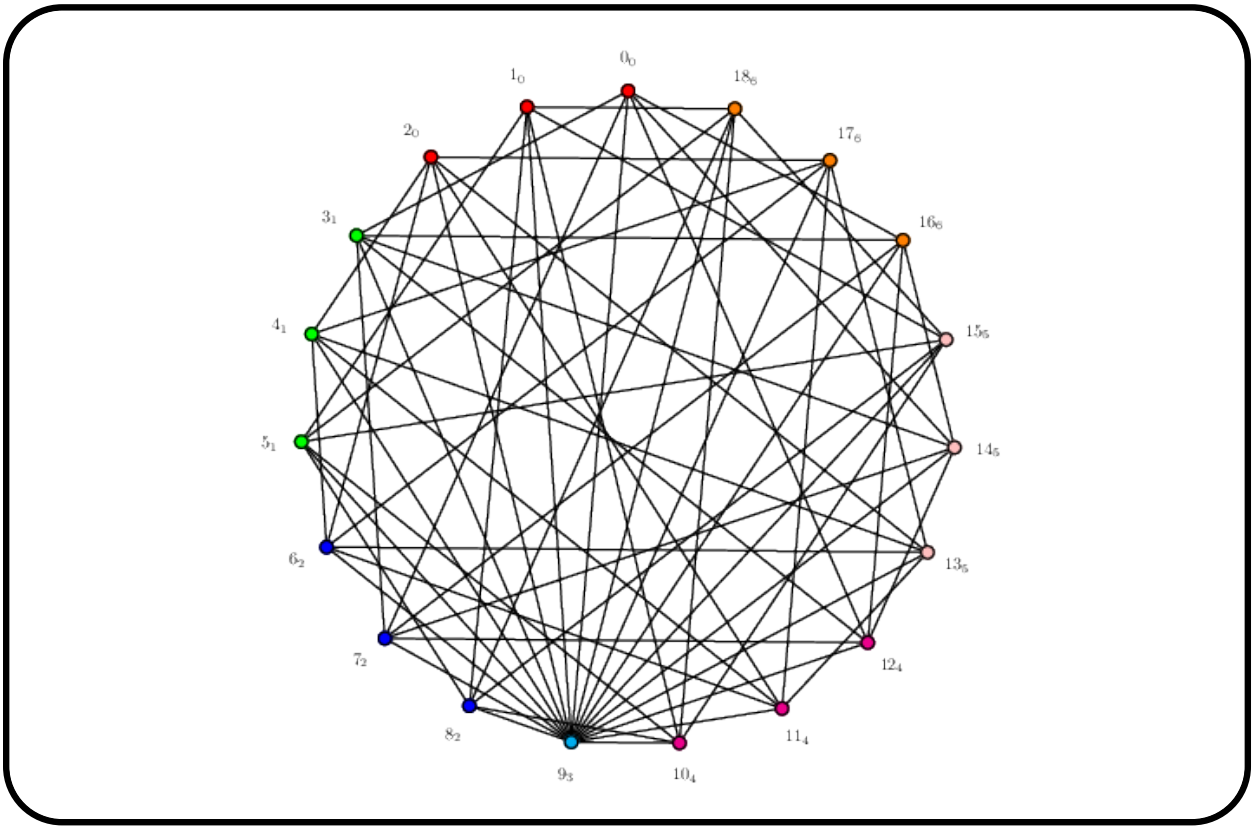
Example 623

```
BLT_11_0_sort:
▷  $(ORBITER) -v 2 \
▷  ▷  -define G -graph \
▷  ▷  ▷  -load BLT_q11_graph_5_0.bin \
▷  ▷  -end \
▷  ▷  -with G -do \
▷  ▷  -graph_theoretic_activity \
▷  ▷  ▷  -sort_by_colors \
▷  ▷  -end
```

Next, we produce a drawing of the graph. The subscripts at the vertex label indicate the vertex color:

Example 624

```
BLT_11_0_sorted_draw:
▷  $(ORBITER) -v 2 \
▷  ▷  -draw_options \
▷  ▷  ▷  -radius 100 \
▷  ▷  ▷  -scale 0.4 \
▷  ▷  ▷  -line_width 1.0 -embedded \
▷  ▷  -end \
▷  ▷  -define G -graph \
▷  ▷  ▷  -load BLT_q11_graph_5_0_sorted.bin \
▷  ▷  -end \
▷  ▷  -with G -do \
▷  ▷  -graph_theoretic_activity \
▷  ▷  ▷  -draw \
▷  ▷  -end
▷  pdflatex BLT_q11_graph_5_0_sorted_draw.tex
▷  $(OPEN) BLT_q11_graph_5_0_sorted_draw.pdf
```


0_0
1_0
18_6
2_0
17_6
3_1
16_6
4_1
15_5
5_1
14_5
6_2
13_5
7_2
12_4
8_2
11_4
9_3
10_4

15.3 Diophantine Systems

Diophantine systems of equations and inequalities arise frequently in Combinatorics. In Table 15.2, Orbiter commands for creating and solving diophantine systems are shown. In Table 15.3, Orbiter activities for diophantine systems are shown.
Consider the matrix

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Suppose we want to find all column vectors $\mathbf{x}$ with entries in $0, 1$ such that

$$A\mathbf{x} = \mathbf{1}$$

where $\mathbf{1}$ is the all-one column vector. Orbiter offers two algorithms to do this. One is McKay's possolve, the other is Knuth's dancing links (DLX). In order to get started, we need to create a diophant object. In the following example, we use the makefile variable `TEST_SYSTEM` for the coefficient matrix.

Example 625

```
TEST_SYSTEM="\
0,1,0,1,0,0, \
0,0,1,0,1,0, \
1,0,1,0,0,0, \
0,1,0,1,0,1, \
1,0,0,0,0,1, \
1,0,1,0,0,0, \
0,1,0,0,1,1"
```

The right hand side will be specified using the `RHS` command below:

Example 626

```
solve_test_system:
▷  $(ORBITER) -v 4 \
▷  ▷  -define A -vector -format 7 -dense $(TEST_SYSTEM) -end \
▷  ▷  -define D -diophant \
▷  ▷  ▷  -label test_system \
▷  ▷  ▷  -coefficient_matrix A \
▷  ▷  ▷  -RHS "mult=7,EQ=1" \
▷  ▷  ▷  -x_min_global 0 -x_max_global 1 \
▷  ▷  -end
```

There are two commands to solve a diophantine system: `-solve_mckay` and `-solve_DLX`. The latter is more restrictive, as it allows only 0,1 variables. Here is the McKay solver:

Creating Diophantine Systems		
Command	Arguments	Purpose
-label	label	Use the given name as file name.
-coefficient_matrix	A	Set the coefficient matrix to the previously created vector with format information.
-coefficient_matrix_csv	fname	Read the coefficient matrix from the given csv-file.
-RHS	options	Set RHS info for a set of m rows. Possible types of conditions are EQ, LE, INT, ZOR. This command can be repeated. Example: mult=2,EQ=1 is condition for 2 rows where the RHS is equal to zero. Example: mult=1,LE=3 is one condition where the RHS is at most three. Example: mult=1,INT=1to2 is one condition where the RHS is between 1 and 2. Example: mult=1,ZOR=2 is one condition where the RHS is either 0 or 2.
-RHS_csv	fname	Read the RHS information from the given csv file.
-RHS_constant	options	Set the RHS as in -RHS (without the mult component).
-x_max_global	a	Set the upper bound for all variables to a
-x_min_global	a	Set the lower bound for all variables to a
-x_bounds	list-of-values	Set the lower and upper bounds for all variables.
-x_bounds_csv	fname	Read the lower and upper bounds for all variables from the given file.
-has_sum	s	Force the sum of the variables to be s.
-problem_of_Steiner_type	N fname	Create an instance of an exact cover problem based on the matrix in the given file. The underlying set has size N.
-maximal_arc	s d secants subset	Create system for a maximal arc of size s and degree d in $\mathrm{PG}(2,q)$. Use the given set of two pencil lines. The subset picks the lines from the given pencils which are external.
-field	F	Use the field F. Related to -maximal_arc.

Table 15.2: Creating Diophantine Systems

Diophantine System Activities

Command	Arguments	Purpose
`-print`		Print the system.
`-solve_mckay`		Solve the system using McKay's possolve.
`-solve_DLX`		Solve the system using Knuth's dancing links.
`-solve_standard`		Solve the system using the standard solver.
`-draw`		Produce a drawing of the coefficient matrix of the system.
`-draw_as_bitmap`	w b	Produce a bitmap drawing of the coefficient matrix of the system, using boxes of w pixels with. Set the color bit-depth to b ($b = 8$ or $b = 24$). The output is a bmp-file.
`-perform_column_reductions`		Eliminate variables which must be zero and write a reduced system.
`-test_single_equation`		For each row of the system, compute the number of solutions of the system restricted to the nonzero coefficients.
`-project_to_single_equation_and_solve`	i j	Solve the system assuming the jth solution to the restricted system consisting of the ith row.
`-project_to_two_equations_and_solve`	i j r m	Solve the system assuming any solution to the restricted system consisting of the ith and the j-th row whose number is congruent to r mod m.

Table 15.3: Diophantine System Activities

Example 627

```
McKay_test:
▷  $(ORBITER) -v 4 \
▷  ▷  -define A -vector -format 7 -dense $(TEST_SYSTEM) -end \
▷  ▷  -define D -diophant \
▷  ▷  ▷  -label test_system \
▷  ▷  ▷  -coefficient_matrix A \
▷  ▷  ▷  -RHS "mult=7,EQ=1" \
▷  ▷  ▷  -x_min_global 0 -x_max_global 1 \
▷  ▷  -end \
▷  ▷  -with D -do \
▷  ▷  ▷  -diophant_activity -solve_mckay \
▷  ▷  -end
```

The solutions are written to the file `DLX_test.sol`. Next, we invoke the dancing links solver:

Example 628

```
DLX_test:
▷  $(ORBITER) -v 4 \
▷  ▷  -define A -vector -format 7 -dense $(TEST_SYSTEM) -end \
▷  ▷  -define D \
▷  ▷  -diophant -label test_system \
▷  ▷  ▷  -coefficient_matrix A \
▷  ▷  ▷  -RHS "mult=7,EQ=1" \
▷  ▷  ▷  -x_min_global 0 -x_max_global 1 \
▷  ▷  -end \
▷  ▷  -with D -do \
▷  ▷  ▷  -diophant_activity -solve_DLX \
▷  ▷  -end
```

Chapter 16

Canonical Forms with Nauty

16.1 Overview of Canonical Forms

Table 16.1 lists Orbiter commands related to canonical labelings of combinatorial objects, including objects in projective spaces.
The graph theory package Nauty [55] provides a canonical form algorithm for graphs. Problems in projective spaces can be reduced to equivalent problems graphs, so that Nauty can help here as well. For an instance of this technique to the classification of arcs, see [2].

Classification by Canonical Form

Command	Arguments	Purpose
-space	L	A projective space is required for some types of combinatorial objects.
-ring	R	A polynomial ring is required for algebraic varieties.
-input_fname_mask	fname-mask	Input file name mask.
-nb_files	n	Number of input files.
-output_fname	fname	Output file name.
-label_po_go	label	Column label for po-go columns. This is the stabilizer of the orbit representative of the primary orbit.
-label_po_index	label	Column label for the subgroup index of the secondary orbit representative inside the stabilizer of the representative of the primary orbit.
-label_po	label	Column label for the primary orbit number.
-label_so	label	Column label for the secondary orbit number.
-label_equation	label	Column label for the algebraic equation (in case of a variety).
-label_equation2	label	Column label for the optional second algebraic equation (in case of a variety).
-label_points	label	Column label for the set of points of the object (optional).
-label_lines	label	Column label for the set of lines of the object (optional).
-carry_through	label	Column label of a property that is carried through from the input file to the output file. This option can be repeated.
-algorithm_nauty		Select canonical forms computed by Nauty (recommended).
-algorithm_substructure		Select canonical forms computed by an algorithm based on substructures (not recommended).
-substructure_size	s	Select the size of the substructure (for the substructure based algorithm).
-skip	list	List of cases to be skipped. Case numbers are assigned consecutively. They are 0-based.

Table 16.1: Classification by Canonical Form

16.2 Canonical Forms of Objects in Projective Space

Suppose we want to compute the stabilizer of an elliptic curve. In Section 4.1, we have created an elliptic curve over $\mathbb{F}_{11}$ and stored the set of $\mathbb{F}_q$-points in the file

elliptic_curve_b1_c3_q11.txt.

The following example computes the set stabilizer of the curve. This means it computes the set stabilizer of the points on the curve. In order to do so, an input stream is created which referst to the file containing the Orbiter point ranks of points on the curve.

Example 629

```
EC_canon: elliptic_curve_b1_c3_q11.txt
▷  $(ORBITER) -v 3 \
▷  ▷  -define C -combinatorial_object \
▷  ▷  ▷  -label EC EC \
▷  ▷  ▷  -file_of_points elliptic_curve_b1_c3_q11.txt \
▷  ▷  -end \
▷  ▷  -define F -finite_field -q 11 -end \
▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
▷  ▷  -with C -do \
▷  ▷  -combinatorial_object_activity \
▷  ▷  ▷  -canonical_form_PG P \
▷  ▷  ▷  ▷  -save_ago \
▷  ▷  ▷  ▷  -save_transversal \
▷  ▷  ▷  ▷  -max_TDO_depth 400 \
▷  ▷  ▷  -end \
▷  ▷  -end \
▷  ▷  -with C -do \
▷  ▷  -combinatorial_object_activity \
▷  ▷  ▷  -report \
▷  ▷  ▷  ▷  -export_flag_orbits \
▷  ▷  ▷  ▷  -show_TDO \
▷  ▷  ▷  ▷  -show_TDA \
▷  ▷  ▷  ▷  -dont_show_incidence_matrices \
▷  ▷  ▷  ▷  -export_group_GAP \
▷  ▷  ▷  -end \
▷  ▷  -end
▷  pdflatex EC_classification.tex
▷  $(OPEN) EC_classification.pdf
▷  $(ORBITER) -v 2 -draw_matrix \
▷  ▷  -input_csv_file EC_object0_TDA_flag_orbits.csv \
▷  ▷  -secondary_input_csv_file EC_object0_TDA.csv \
▷  ▷  -box_width 20 -bit_depth 24 \
▷  ▷  -end
▷  $(OPEN) EC_object0_TDA_flag_orbits_draw.bmp
```

Orbiter shows that the curve has a collineation stabilizer of order 6, generated by

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 8 \\ 5 & 9 & 5 \\ 8 & 1 & 1 \end{bmatrix}.$$

The following example computes the canonical form and the automorphism group of the Hirschfeld surface in PG$(3, 4)$. Using indexing of points in PG$(3, 4)$, we encode the surface as a set of points using Orbiter ranks. We use a makefile variable to provide these ranks as input for the canonical form procedure.

Example 630

```
HIRSCHFELD_SURFACE_Q4_SET_OF_POINTS="0,1,2,3,4,5,6,7,8,9,\
10,11,12,13,14,23,26,27,30,31,34,35,38,39,42,47,48,51,52,\
53,54,59,60,61,62,67,68,69,70,75,76,79,80,81,82"
```

Example 631

```
Hirschfeld_q4_c:
▷  $(ORBITER) -v 6 \
▷  ▷  -define C -combinatorial_object \
▷  ▷  ▷  -label Hirschfeld_surface_q4 Hirschfeld\_surface\_q4 \
▷  ▷  ▷  -file_of_points Hirschfeld_surface_q4.txt \
▷  ▷  -end \
▷  ▷  -define F -finite_field -q 4 -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷  ▷  -with C -do \
▷  ▷  -combinatorial_object_activity \
▷  ▷  ▷  -canonical_form_PG P \
▷  ▷  ▷  ▷  -save_ago \
▷  ▷  ▷  ▷  -max_TDO_depth 10 \
▷  ▷  ▷  -end \
▷  ▷  -end \
▷  ▷  -with C -do \
▷  ▷  -combinatorial_object_activity \
▷  ▷  ▷  -report \
▷  ▷  ▷  ▷  -export_flag_orbits \
▷  ▷  ▷  ▷  -show_TDO \
▷  ▷  ▷  ▷  -show_TDA \
▷  ▷  ▷  ▷  -export_labels \
▷  ▷  ▷  ▷  -dont_show_incidence_matrices \
▷  ▷  ▷  ▷  -export_group_GAP \
▷  ▷  ▷  -end \
▷  ▷  -end
▷  pdflatex Hirschfeld_surface_q4_classification.tex
▷  $(OPEN) Hirschfeld_surface_q4_classification.pdf
```

Example 632

```
Hirschfeld_q4_set_c:
▷  $(ORBITER) -v 4 \
▷  ▷  -define H -set -here \
▷  ▷  ▷  $(HIRSCHFELD_SURFACE_Q4_SET_OF_POINTS) \
▷  ▷  -end \
▷  ▷  -define C -combinatorial_object \
▷  ▷  ▷  -label Hirschfeld_surface_q4 Hirschfeld\_surface\_q4 \
▷  ▷  ▷  -set_of_points H \
▷  ▷  -end \
▷  ▷  -define F -finite_field -q 4 -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷  ▷  -with C -do \
▷  ▷  -combinatorial_object_activity \
▷  ▷  ▷  -canonical_form_PG P \
▷  ▷  ▷  ▷  -save_ago \
▷  ▷  ▷  ▷  -max_TDO_depth 4 \
▷  ▷  ▷  -end \
▷  ▷  -end \
▷  ▷  -with C -do \
▷  ▷  -combinatorial_object_activity \
▷  ▷  ▷  -report \
▷  ▷  ▷  ▷  -show_TDO \
▷  ▷  ▷  ▷  -show_TDA \
▷  ▷  ▷  -end \
▷  ▷  -end
▷  pdflatex Hirschfeld_surface_q4_classification.tex
▷  $(OPEN) Hirschfeld_surface_q4_classification.pdf
```

In the next example, we compute the canonical form of the two hyperovals in $\mathrm{PG}(2, 16)$.

Example 633

```
hyperoval_16_canonical_form:
▷  $(ORBITER) -v 2 \
▷  ▷  -define C -combinatorial_object \
▷  ▷  ▷  -label hyperoval_q16 hyperoval\_q16 \
▷  ▷  ▷  -set_of_points $(HYPEROVAL_16_16320) \
▷  ▷  ▷  -set_of_points $(HYPEROVAL_16_144) \
▷  ▷  -end \
▷  ▷  -define F -finite_field -q 16 -end \
▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
▷  ▷  -with C -do \
▷  ▷  -combinatorial_object_activity \
▷  ▷  ▷  -canonical_form_PG P \
▷  ▷  ▷  ▷  -save_ago \
▷  ▷  ▷  ▷  -save_transversal \
▷  ▷  ▷  ▷  -max_TDO_depth 10 \
▷  ▷  ▷  -end \
▷  ▷  -end \
```

```
▷ ▷ -with C -do \
▷ ▷ -combinatorial_object_activity \
▷ ▷ ▷ -report \
▷ ▷ ▷ ▷ -export_flag_orbits \
▷ ▷ ▷ ▷ -show_TDO \
▷ ▷ ▷ ▷ -show_TDA \
▷ ▷ ▷ ▷ -dont_show_incidence_matrices \
▷ ▷ ▷ ▷ -export_group_GAP \
▷ ▷ ▷ -end \
▷ ▷ -end
▷ pdflatex hyperoval_q16_classification.tex
▷ $(OPEN) hyperoval_q16_classification.pdf
▷ $(ORBITER) -v 2 -draw_matrix \
▷ ▷ -input_csv_file hyperoval_q16_object0_TDA_flag_orbits.csv \
▷ ▷ -secondary_input_csv_file hyperoval_q16_object0_TDA.csv \
▷ ▷ -box_width 4 -bit_depth 24 \
▷ ▷ -end
▷ $(OPEN) hyperoval_q16_object0_TDA_flag_orbits_draw.bmp
▷ $(ORBITER) -v 2 -draw_matrix \
▷ ▷ -input_csv_file hyperoval_q16_object1_TDA_flag_orbits.csv \
▷ ▷ -secondary_input_csv_file hyperoval_q16_object1_TDA.csv \
▷ ▷ -box_width 4 -bit_depth 24 \
▷ ▷ -end
▷ $(OPEN) hyperoval_q16_object1_TDA_flag_orbits_draw.bmp
```

In the next example, we compute the set stabilizers of orbits of $\mathrm{PGL}(4,2)$ on subsets of $\mathrm{PG}(3,2)$, as computed earlier in Section 7.3, using the command `PG_3_2_subsets`. Concerning the work in Dickson [26] only subsets whose size is odd are relevant, so we restrict to those subsets:

Example 634

```
Dickson_sets_stabilizer:
▷ $(ORBITER) -v 3 \
▷ ▷ -define C -combinatorial_object \
▷ ▷ ▷ -label Dickson_sets Dickson\_sets \
▷ ▷ ▷ -set_of_points "0,1,2,5,6" \
▷ ▷ ▷ -set_of_points "0,1,2,3,6" \
▷ ▷ ▷ -set_of_points "0,1,2,3,4" \
▷ ▷ ▷ -set_of_points "0,1,2,3,8" \
▷ ▷ ▷ -set_of_points "0,1,2,5,6,7,8" \
▷ ▷ ▷ -set_of_points "0,1,2,3,5,6,7" \
▷ ▷ ▷ -set_of_points "0,1,2,3,5,6,9" \
▷ ▷ ▷ -set_of_points "0,1,2,3,5,6,10" \
▷ ▷ ▷ -set_of_points "0,1,2,3,5,6,4" \
▷ ▷ ▷ -set_of_points "0,1,2,3,8,11,13" \
▷ ▷ ▷ -set_of_points "3,6,9,7,10,12,8,11,13,14,4" \
▷ ▷ ▷ -set_of_points "3,5,6,9,7,10,12,11,13,14,4" \
▷ ▷ ▷ -set_of_points "0,1,2,3,5,6,9,7,10,12,4" \
▷ ▷ -end \
▷ ▷ -define F -finite_field -q 2 -end \
```

```
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷  ▷  -with C -do \
▷  ▷  -combinatorial_object_activity \
▷  ▷  ▷  -canonical_form_PG P \
▷  ▷  ▷  ▷  -save_ago \
▷  ▷  ▷  ▷  -max_TDO_depth 4 \
▷  ▷  ▷  -end \
▷  ▷  -end \
▷  ▷  -with C -do \
▷  ▷  -combinatorial_object_activity \
▷  ▷  ▷  -report \
▷  ▷  ▷  ▷  -show_TDO \
▷  ▷  ▷  ▷  -show_TDA \
▷  ▷  ▷  -end \
▷  ▷  -end
▷  pdflatex Dickson_sets_classification.tex
▷  $(OPEN) Dickson_sets_classification.pdf
```

There are two ovoids in PG$(3, 2)$. The classical ovoid is the elliptic quadric. It was created using the command `elliptic_quadric_ovoid_q8` in Section 4.10. The following command computes the stabilizer of the ovoid:

Example 635

```
ovoid_q8_canon: ovoid_q8.txt
▷  $(ORBITER) -v 6 \
▷  ▷  -define C -combinatorial_object \
▷  ▷  ▷  -label ovoid ovoid \
▷  ▷  ▷  -file_of_points ovoid_q8.txt \
▷  ▷  -end \
▷  ▷  -define F -finite_field -q 8 -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷  ▷  -with C -do \
▷  ▷  -combinatorial_object_activity \
▷  ▷  ▷  -canonical_form_PG P \
▷  ▷  ▷  ▷  -save_ago \
▷  ▷  ▷  ▷  -max_TDO_depth 4 \
▷  ▷  ▷  -end \
▷  ▷  -end \
▷  ▷  -with C -do \
▷  ▷  -combinatorial_object_activity \
▷  ▷  ▷  -report \
▷  ▷  ▷  ▷  -show_TDO \
▷  ▷  ▷  ▷  -show_TDA \
▷  ▷  ▷  ▷  -dont_show_incidence_matrices \
▷  ▷  ▷  ▷  -export_group_GAP \
▷  ▷  ▷  -end \
▷  ▷  -end
▷  #pdflatex ovoid_classification.tex
▷  #$(OPEN) ovoid_classification.pdf
```

The other ovoid is the Suzuki Tits ovoid, which was created using the command `ovoid_ST_q8` in Section 4.10. The stabilizer of the Suzuki Tits ovoid is the Suzuki group. The following command computes this group for $q = 8$.

Example 636

```
ovoid_ST_q8_canon: ovoid_ST_q8.txt
▷  $(ORBITER) -v 6 \
▷  ▷  -define C -combinatorial_object \
▷  ▷  ▷  -label ovoid_ST ovoid\_ST \
▷  ▷  ▷  -file_of_points ovoid_ST_q8.txt \
▷  ▷  -end \
▷  ▷  -define F -finite_field -q 8 -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷  ▷  -with C -do \
▷  ▷  -combinatorial_object_activity \
▷  ▷  ▷  -canonical_form_PG P \
▷  ▷  ▷  ▷  -save_ago \
▷  ▷  ▷  ▷  -max_TDO_depth 4 \
▷  ▷  ▷  -end \
▷  ▷  -end \
▷  ▷  -with C -do \
▷  ▷  -combinatorial_object_activity \
▷  ▷  ▷  -report \
▷  ▷  ▷  ▷  -show_TDO \
▷  ▷  ▷  ▷  -show_TDA \
▷  ▷  ▷  ▷  -dont_show_incidence_matrices \
▷  ▷  ▷  ▷  -export_group_GAP \
▷  ▷  ▷  -end \
▷  ▷  -end
▷  #pdflatex ovoid_ST_classification.tex
▷  #$(OPEN) ovoid_ST_classification.pdf
```

We can store the generators in a makefile variable as follows:

Example 637

```
SUZUKI_8_GENERATORS="\
1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1, \
1,0,0,0,0,6,0,0,0,0,2,0,0,0,0,3,0, \
1,0,0,0,1,1,1,0,0,0,1,0,1,0,0,1,0, \
1,0,0,0,3,6,2,2,5,0,2,0,3,0,6,3,2, \
0,1,0,0,1,0,0,0,0,0,0,1,0,0,1,0,2"
# group order 87360 = 3 * 29120
```

We can now recover the Suzuki group using the command:

Example 638

```
Suzuki_8:
▷  $(ORBITER) -v 6 \
```

```
▷  ▷  -define F -finite_field -q 8 -end \
▷  ▷  -define gens -vector -field F \
▷  ▷  ▷  -compact $(SUZUKI_8_GENERATORS) -end \
▷  ▷  -define G -linear_group -PGGL 4 8 \
▷  ▷  -subgroup_by_generators "Sz8" "87360" 5 gens \
▷  ▷  ▷  -end \
▷  ▷  -with G -do \
▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex PGGL_4_8_Subgroup_Sz8_87360_report.tex
▷  $(OPEN) PGGL_4_8_Subgroup_Sz8_87360_report.pdf
```

Let us go back to Section 5.3 and compute the stabilizer of the Veronese variety. The following command can be used. We utilize the list of point ranks computed earlier:

Example 639

```
Veronese_1_3_q7_c:
▷  $(ORBITER) -v 6 \
▷  ▷  -define C -combinatorial_object \
▷  ▷  ▷  -label veronese_1_3_q7 veronese\_1\_3\_q7 \
▷  ▷  ▷  -set_of_points "3, 0, 4, 268, 336, 184, 224, 364" \
▷  ▷  -end \
▷  ▷  -define F -finite_field -q 7 -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷  ▷  -with C -do \
▷  ▷  -combinatorial_object_activity \
▷  ▷  ▷  -canonical_form_PG P \
▷  ▷  ▷  ▷  -save_ago \
▷  ▷  ▷  ▷  -max_TDO_depth 4 \
▷  ▷  ▷  -end \
▷  ▷  -end \
▷  ▷  -with C -do \
▷  ▷  -combinatorial_object_activity \
▷  ▷  ▷  -report \
▷  ▷  ▷  ▷  -show_TDO \
▷  ▷  ▷  ▷  -show_TDA \
▷  ▷  ▷  ▷  -dont_show_incidence_matrices \
▷  ▷  ▷  ▷  -export_group_GAP \
▷  ▷  ▷  -end \
▷  ▷  -end
▷  pdflatex veronese_1_3_q7_classification.tex
▷  $(OPEN) veronese_1_3_q7_classification.pdf
```

The list of points is unpacked and points are printed with their homogeneous coordinates:

i	Rank	Point
0	3	$(0,0,0,1)$
1	0	$(1,0,0,0)$
2	4	$(1,1,1,1)$
3	268	$(1,2,4,1)$
4	336	$(6,4,5,1)$
5	184	$(1,4,2,1)$
6	224	$(6,2,3,1)$
7	364	$(6,1,6,1)$

The automorphism group is determined to have order 336. Generators are produced:

The stabilizer of order 336 is generated by:

$$g_1 = \begin{bmatrix} 6 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 6 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ of order 2 and with 16 fixed points.

$$g_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ of order 3 and with 10 fixed points.

$$g_3 = \begin{bmatrix} 6 & 0 & 0 & 0 \\ 4 & 2 & 0 & 0 \\ 4 & 4 & 3 & 0 \\ 6 & 2 & 3 & 1 \end{bmatrix}$$ of order 6 and with 4 fixed points.

$$g_4 = \begin{bmatrix} 1 & 5 & 4 & 6 \\ 6 & 1 & 0 & 3 \\ 5 & 0 & 2 & 4 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$ of order 8 and with 0 fixed points.

1,0,0,0,0,6,0,0,0,0,1,0,0,0,0,6,
1,0,0,0,0,2,0,0,0,0,4,0,0,0,0,1,
1,0,0,0,3,5,0,0,3,3,4,0,1,5,4,6,
1,5,4,6,6,1,0,3,5,0,2,4,1,1,1,1,

The array of numbers at the end is the coding of the generators. This can be used to recreate the group. Finally, a tactical decomposition is produced. This is the extended incidence matrix of $\mathrm{PG}(3,7)$, using combinatorial refinement. One artificial point and one artificial line are introduced to encode the variety as a combinatorial object.

Decomposition by combinatorial refinement:

$\rightarrow$	1_1	28_2	400_5	2422_6
8_0	1	7	50	0
1_4	1	0	0	0
168_3	0	1	6	50
224_7	0	0	8	49

$\downarrow$	1_1	28_2	400_5	2422_6
8_0	8	2	1	0
1_4	1	0	0	0
168_3	0	6	2	3
224_7	0	0	5	5

16.3 Canonical Forms of Incidence Geometries

Let us consider incidence structures, which are sets of subsets. This subsets are chosen from the same set, which we call the underlying set. The elements of the group set are often called points. In many cases, there are conditions that restrict the way in which the sets can be chosen. There is a notion of isomorphism on such set systems. Two set systems are isomorphic is there is a bijection between the underlying sets which takes one to the other. The incidence matrix is the 0/1 matrix whose rows correspond to the elements of the group set, and whose columns correspond to the chosen subsets. An entry 1 indicates that the corresponding point belongs to the corresponding set.

An incidence geometry is a set system with the following properties: No set appears twice, and no pair of elements in the set appear in two different sets. The elements of the set are called points. The sets are called lines (or sometimes planes). A flag is an incident point-line pair. An anti-flag is a non-incident point-line pair. Two points are said to be collinear of there is a line in the geometry containing both points.

It is interesting to study the action of the automorphism group on the elements of a gemetry. Properties of interest are various levels of transitivity on the elements of the geometry. For instance, a geometry is line-transitive if the automorphism group is transitive on lines. Likewise, it is flag transitive if the automorphism group is transitive on flags. The collinearity graph of a geometry is the graph whose vertices correspond to the points, with two vertices adjacent of the associated points are collinear. The girth of the incidence geometry is the girth of the associated collineation graph. A geometry is triangle free if its girth is at least 4.

A configuration $v_r b_k$ is an incidence geometry on a set of size v and with b lines such that each line has size k and each point is contained in exactly r lines. In the special case where $b = v$ and $k = r$, the name symmetric configuration v_r is used (the term symmetric is somewhat misleading because the incidence matrix of a symmetric configuration need not be symmetric). Orbiter can be used to classify incidence geometries. One of the important steps in this process is computing a canonical form of the incidence gometry.

We will also be producing drawings of the incidence matrices of geometries. In these drawings, flags are indicated as heavy squares while anti-flags are drawn as small squares. The coloring will indicate the orbits of the automorphism group on flags and anti-flags. Objects with the same color belong to the same orbit. For a flag-transitive geometry, there is only one color for the incidences.

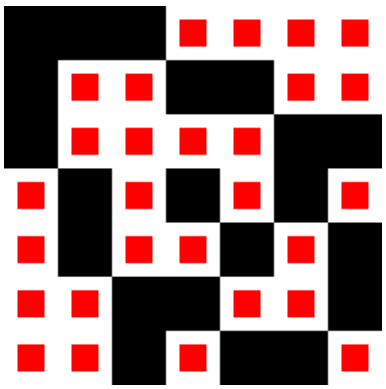
The following command computes the canonical form and a report of the projective plane PG$(2, 2)$, which is a configuration 7_3.

Example 640

```
geo_7_3_c:
▷  $(ORBITER) -v 10 \
▷  ▷  -draw_incidence_structure_description \
▷  ▷  ▷  -width 60 -width_10 6 -end \
▷  ▷  -define C -combinatorial_object \
▷  ▷  ▷  -label geo_7_3 geo\_7\_3 \
▷  ▷  ▷  -file_of_incidence_geometries 7_3.inc 7 7 21 \
▷  ▷  -end \
▷  ▷  -with C -do \
▷  ▷  -combinatorial_object_activity \
▷  ▷  ▷  -canonical_form \
▷  ▷  ▷  ▷  -save_ago \
▷  ▷  ▷  ▷  -save_transversal \
▷  ▷  ▷  -end \
▷  ▷  -end \
```

```
▷  ▷  -with C -do \
▷  ▷  -combinatorial_object_activity \
▷  ▷  ▷  -report \
▷  ▷  ▷  ▷  -export_flag_orbits \
▷  ▷  ▷  ▷  -show_incidence_matrices \
▷  ▷  ▷  ▷  -export_group_GAP \
▷  ▷  ▷  -end \
▷  ▷  -end
▷  pdflatex 7_3_classification.tex
▷  $(OPEN) 7_3_classification.pdf
▷  $(ORBITER) -v 2 -draw_matrix \
▷  ▷  -input_csv_file 7_3_object0_TDA_flag_orbits.csv \
▷  ▷  -secondary_input_csv_file 7_3_object0_TDA.csv \
▷  ▷  -box_width 32 -bit_depth 24 \
▷  ▷  -end
▷  $(ORBITER) -v 2 -draw_matrix \
▷  ▷  -input_csv_file 7_3_object0_INP_flag_orbits.csv \
▷  ▷  -secondary_input_csv_file 7_3_object0_INP.csv \
▷  ▷  -box_width 32 -bit_depth 24 \
▷  ▷  -end
▷  $(OPEN) 7_3_object0_INP_flag_orbits_draw.bmp
```

A bitmap drawing is produced, shown below:



The command also produces the following report of the geometry:

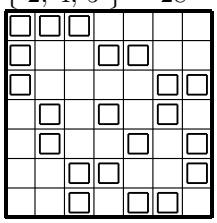
Summary of Orbits

	Rep	#	Ago	Objects
0	0	1	168	0

Ago : 168

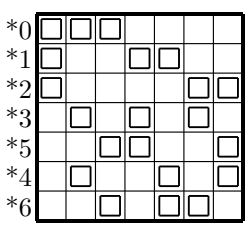
Isomorphism type 0 / 1

Isomorphism type 0 / 1 is original object 0 and appears 1 times:
This isomorphism type appears 1 times, namely for the following 1 input objects: {0}
incidence structure:
(0, 1, 2, 7, 10, 11, 14, 19, 20, 22, 24, 26, 29, 32, 34, 37, 38, 41, 44, 46, 47)
Column sets of the encoded object:
{ 0, 1, 2 }
{ 0, 3, 4 }
{ 0, 5, 6 }
{ 1, 3, 5 }
{ 1, 4, 6 }
{ 2, 3, 6 }
{ 2, 4, 5 }
Row sets of the encoded object:
{ 0, 1, 2 } = 0
{ 0, 3, 4 } = 9
{ 0, 5, 6 } = 14
{ 1, 3, 5 } = 20
{ 1, 4, 6 } = 23
{ 2, 3, 6 } = 27
{ 2, 4, 5 } = 28



Generators for the automorphism group:
The stabilizer of order 168 is generated by:
g_1 = (3, 5)(4, 6)(8, 9)(12, 13) of order 2 and with 6 fixed points.
g_2 = (3, 4)(5, 6)(10, 11)(12, 13) of order 2 and with 6 fixed points.
g_3 = (1, 2)(5, 6)(10, 12)(11, 13) of order 2 and with 6 fixed points.
g_4 = (1, 3)(2, 4)(7, 8)(11, 12) of order 2 and with 6 fixed points.
g_5 = (0, 1)(4, 5)(8, 10)(9, 11) of order 2 and with 6 fixed points.

Canonical labeling:
canonical row = 6
canonical orbit number = 0
Flags : (0, 1, 2, 7, 10, 11, 14, 19, 20, 22, 24, 26, 30, 31, 34, 36, 39, 41, 44, 46, 47)



Flag orbits:
orbit length : number of orbits of that length:

21 1

Anti-Flag orbits:
orbit length : number of orbits of that length:

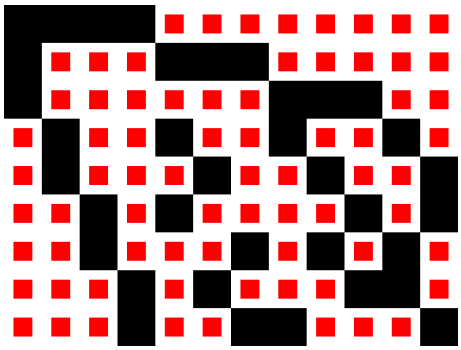
28 1

The following command computes the canonical form and a report of the affine plane AG$(2, 3)$, which is a configuration $9_4 12_3$.

Example 641

```
AG_2_3_c: AG_2_3.inc
▷  $(ORBITER) -v 2 \
▷  ▷  -define C -combinatorial_object \
▷  ▷  ▷  -label AG_2_3 AG\_2\_3 \
▷  ▷  ▷  -file_of_incidence_geometries \
▷  ▷  ▷  AG_2_3.inc 9 12 36 \
▷  ▷  -end \
▷  ▷  -with C -do \
▷  ▷  -combinatorial_object_activity \
▷  ▷  ▷  -canonical_form \
▷  ▷  ▷  ▷  -save_ago \
▷  ▷  ▷  ▷  -max_TDO_depth 10 \
▷  ▷  ▷  -end \
▷  ▷  -end \
▷  ▷  -with C -do \
▷  ▷  -combinatorial_object_activity \
▷  ▷  ▷  -report \
▷  ▷  ▷  ▷  -export_flag_orbits \
▷  ▷  ▷  ▷  -show_TDO \
▷  ▷  ▷  ▷  -show_TDA \
▷  ▷  ▷  ▷  -show_incidence_matrices \
▷  ▷  ▷  -end \
▷  ▷  -end
▷  pdflatex AG_2_3_classification.tex
▷  $(OPEN) AG_2_3_classification.pdf
▷  $(ORBITER) -v 2 -draw_matrix \
▷  ▷  -input_csv_file AG_2_3_object0_INP_flag_orbits.csv \
▷  ▷  -secondary_input_csv_file AG_2_3_object0_INP.csv \
▷  ▷  -box_width 40 -bit_depth 24 \
▷  ▷  -end
▷  $(OPEN) AG_2_3_object0_INP_flag_orbits_draw.bmp
```

The following bitmap drawing is produced:



Because the geometry is flag transitive, there is only one color being used for the incidence (black). In addition, the geometry is also anti-flag transitive. The smallish boxes all have the same color (red). Orbiter also produces the following report of the geometry:

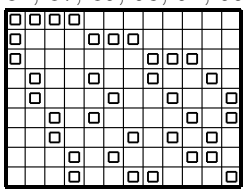
Summary of Orbits

	Rep	#	Ago	Objects
0	0	1	432	0

Ago :432

Isomorphism type 0 / 1

Isomorphism type 0 / 1 is original object 0 and appears 1 times:
This isomorphism type appears 1 times, namely for the following 1 input objects: {0}
incidence structure:
(0, 1, 2, 3, 12, 16, 17, 18, 24, 31, 32, 33, 37, 40, 43, 46, 49, 53, 56, 59, 62, 64, 69, 71, 74, 78, 80, 82, 87, 89, 93, 94, 99, 102, 103, 107)



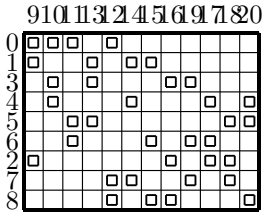
Generators for the automorphism group:
The stabilizer of order 432 is generated by:
$g_1 = (3, 4)(5, 7)(6, 8)(11, 12)(13, 14)(16, 17)(19, 20)$ of order 2 and with 7 fixed points.
$g_2 = (3, 5)(4, 6)(7, 8)(10, 11)(14, 15)(16, 18)(19, 20)$ of order 2 and with 7 fixed points.
$g_3 = (1, 3)(2, 4)(7, 8)(9, 10)(14, 16)(15, 19)(18, 20)$ of order 2 and with 7 fixed points.
$g_4 = (0, 1)(4, 5)(6, 7)(10, 13)(11, 14)(12, 15)(17, 18)$ of order 2 and with 7 fixed points.

Decomposition by combinatorial refinement:

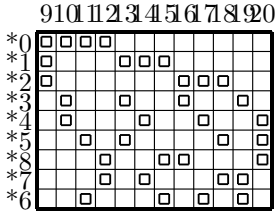
$\rightarrow$	12_1
9_0	4

$\downarrow$	12_1
9_0	3

Decomposition by automorphism group:



Canonical labeling:
canonical row = 6
canonical orbit number = 0
Flags : (0, 1, 2, 3, 12, 16, 17, 18, 24, 31, 32, 33, 37, 40, 43, 46, 49, 53, 56, 59, 62, 64, 69, 71, 75, 78, 79, 83, 87, 89, 93, 94, 98, 102, 104, 106)



Flag orbits:
orbit length : number of orbits of that length:

36 1

Anti-Flag orbits:
orbit length : number of orbits of that length:

72 1

It is possible to perform isomorph classification for configurations based on incidence files. Suppose we want to check that the configurations in `10_3` are in fact all nonisomorphic. We apply the canonical form algorithm given by Nauty. This produces a transversal of the isomorphism types of incidence geometries from the given list of input objects. The objects are specified by means of the `combinatorial_objects` command. The classification algorithm can print a report which lists the transversal and all elements in it in latex form.

Example 642

```
geo_10_3_c:
▷  $(ORBITER) -v 10 \
▷  ▷  -draw_incidence_structure_description \
▷  ▷  ▷  -width 60 -width_10 6 -end \
▷  ▷  -define C -combinatorial_object \
▷  ▷  ▷  -label geo_10_3 geo\_10\_3 \
▷  ▷  ▷  -file_of_incidence_geometries 10_3.inc 10 10 30 \
▷  ▷  -end \
▷  ▷  -with C -do \
▷  ▷  -combinatorial_object_activity \
▷  ▷  ▷  -canonical_form \
▷  ▷  ▷  ▷  -save_ago \
```

```
▷ ▷ ▷ ▷ -save_transversal \
▷ ▷ ▷ -end \
▷ ▷ -end \
▷ ▷ -with C -do \
▷ ▷ -combinatorial_object_activity \
▷ ▷ ▷ -report \
▷ ▷ ▷ ▷ -export_flag_orbits \
▷ ▷ ▷ ▷ -show_incidence_matrices \
▷ ▷ ▷ ▷ -export_group_GAP \
▷ ▷ ▷ -end \
▷ ▷ -end
▷ pdflatex 10_3_classification.tex
▷ $(OPEN) 10_3_classification.pdf
▷ $(ORBITER) -v 2 -draw_matrix \
▷ ▷ -input_csv_file 10_3_object7_TDA_flag_orbits.csv \
▷ ▷ -secondary_input_csv_file 10_3_object7_TDA.csv \
▷ ▷ -box_width 16 -bit_depth 24 \
▷ ▷ -end
▷ $(ORBITER) -v 2 -draw_matrix \
▷ ▷ -input_csv_file 10_3_object7_INP_flag_orbits.csv \
▷ ▷ -secondary_input_csv_file 10_3_object7_INP.csv \
▷ ▷ -box_width 16 -bit_depth 24 \
▷ ▷ -end
```

The report is shown below. It is truncated for reasons of space. Only the first two geometries are shown. Note that the ordering of geometries in the report may be different from the ordering in the input file. This is because the classification program sorts the geometries according to the canonical form. Also, note that the report includes the incidence geometry in the original form as well as the tactical decomposition induced by the orbits of the automorphism group.

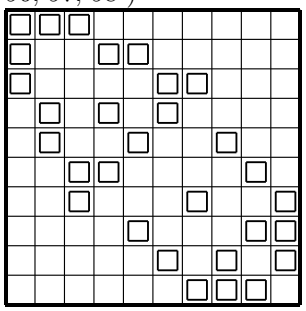
Summary of Orbits

	Rep	#	Ago	Objects
0	9	1	3	9
1	1	1	2	1
2	6	1	10	6
3	5	1	4	5
4	4	1	24	4
5	7	1	4	7
6	8	1	3	8
7	3	1	120	3
8	2	1	12	2
9	0	1	6	0

Ago :2, 3^2, 4^2, 6, 10, 12, 24, 120

Isomorphism type 0 / 10

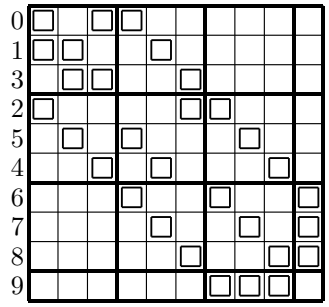
Isomorphism type 0 / 10 is original object 9 and appears 1 times:
This isomorphism type appears 1 times, namely for the following 1 input objects: {9}
incidence structure:
(0, 1, 2, 10, 13, 14, 20, 25, 26, 31, 33, 35, 41, 44, 47, 52, 53, 58, 62, 66, 69, 74, 78, 79, 85, 87, 89, 96, 97, 98)



Generators for the automorphism group:
The stabilizer of order 3 is generated by:
g_1 = (0, 1, 3)(2, 5, 4)(6, 7, 8)(10, 13, 11)(12, 14, 15)(16, 18, 17) of order 3 and with 2 fixed points.

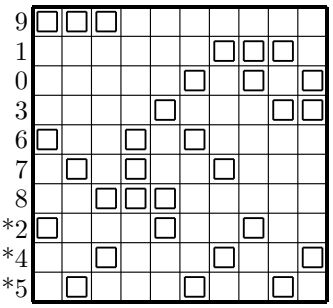
Decomposition by automorphism group:

10 13 11 12 14 15 16 18 17 19



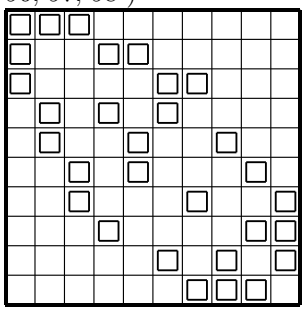
Canonical labeling:
canonical row = 5
canonical orbit number = 1
Flags : 0,1,2,16,17,18,25,27,29,34,38,39,40,43,45,51,53,56,62,63,64,70,74,77,82,86,89,91,95,98,

16 18 17 19 15 12 14 10 13 11



Isomorphism type 1 / 10

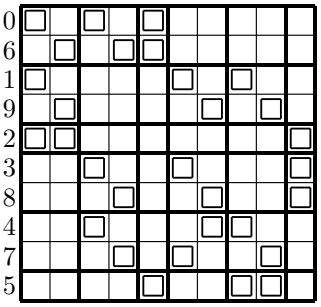
Isomorphism type 1 / 10 is original object 1 and appears 1 times:
This isomorphism type appears 1 times, namely for the following 1 input objects: {1}
incidence structure:
(0, 1, 2, 10, 13, 14, 20, 25, 26, 31, 33, 35, 41, 44, 47, 52, 54, 58, 62, 66, 69, 73, 78, 79, 85, 87, 89, 96, 97, 98)



Generators for the automorphism group:
The stabilizer of order 2 is generated by:
g_1 = (0, 6)(1, 9)(3, 8)(4, 7)(10, 16)(11, 19)(13, 17)(14, 18) of order 2 and with 4 fixed points.

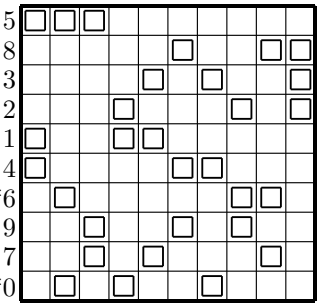
Decomposition by automorphism group:

10 16 11 19 12 13 17 14 18 15



Canonical labeling:
canonical row = 0
canonical orbit number = 0
Flags : 0,1,2,15,18,19,24,26,29,33,37,39,40,43,44,50,55,56,61,67,68,72,75,77,82,84,88,91,93,96,

14 12 18 10 13 17 11 16 19 15



The following command computes the canonical form for the three triangle free configurations 24_3 found by Abdullah Alazemi. These configurations have 24 points, 24 lines, each line consists of 3 points and each

point is on 3 lines.

Example 643

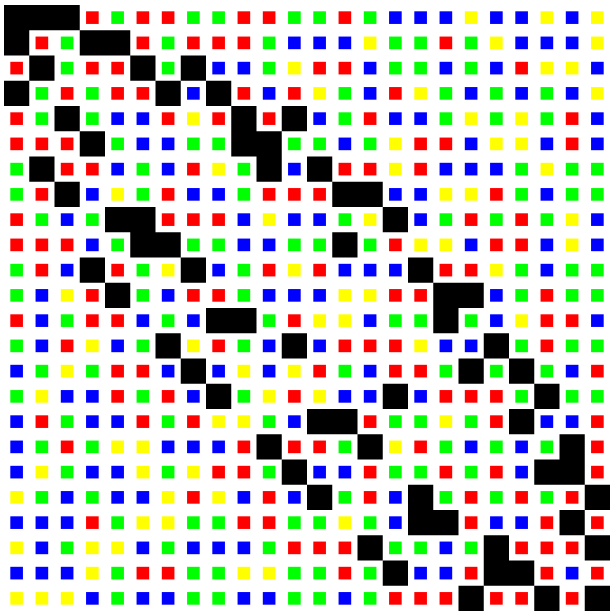
```
FILE_24_3_TFC_INC="24 24 72\
\n0 1 2 24 27 28 48 53 54 73 79 80 97 105 106 122 131 \
132 146 157 158 171 175 183 195 203 208 220 225 233 244 \
258 259 269 272 282 293 300 308 318 325 333 342 352 358 \
367 379 381 392 398 400 417 428 429 442 443 450 466 471 \
479 492 497 502 517 519 521 542 548 551 571 574 575 \
             48\
\n0 1 2 24 27 28 48 53 54 73 79 80 97 105 106 122 131 \
132 146 157 158 171 175 183 195 203 208 220 225 233 244 \
258 259 269 272 281 293 301 308 318 324 327 342 354 357 \
367 373 378 392 400 403 417 419 430 442 446 447 466 472 \
479 492 500 503 518 525 526 545 549 551 571 572 574 \
             48\
\n0 1 2 24 27 28 48 53 54 73 79 80 97 105 106 122 131 \
132 146 157 158 171 175 179 195 201 207 220 226 232 244 \
257 258 269 274 277 293 300 307 318 323 329 342 352 356 \
367 374 381 392 397 406 416 423 431 441 450 454 468 476 \
477 494 499 503 519 521 525 544 547 550 570 572 575 \
            144\
\n-1 3"
```

Example 644

```
TFC_24_3_c:
▷  echo $(FILE_24_3_TFC_INC) >24_3_TFC.inc
▷  $(ORBITER) -v 20 \
▷  ▷  -define C -combinatorial_object \
▷  ▷  ▷  -label 24_3_TFC 24\_3\_TFC \
▷  ▷  ▷  -file_of_incidence_geometries 24_3_TFC.inc 24 24 72 \
▷  ▷  -end \
▷  ▷  -with C -do \
▷  ▷  -combinatorial_object_activity \
▷  ▷  ▷  -canonical_form \
▷  ▷  ▷  -save_ago \
▷  ▷  ▷  -end \
▷  ▷  -end \
▷  ▷  -with C -do \
▷  ▷  -combinatorial_object_activity \
▷  ▷  ▷  -report \
▷  ▷  ▷  ▷  -export_flag_orbits \
▷  ▷  ▷  ▷  -show_TDO \
▷  ▷  ▷  ▷  -show_TDA \
▷  ▷  ▷  ▷  -show_incidence_matrices \
▷  ▷  ▷  -end \
▷  ▷  -end
▷  pdflatex 24_3_TFC_classification.tex
▷  $(OPEN) 24_3_TFC_classification.pdf
▷  $(ORBITER) -v 2 -draw_matrix \
```

```
▷  ▷  -input_csv_file 24_3_TFC_object2_TDA_flag_orbits.csv \
▷  ▷  -secondary_input_csv_file 24_3_TFC_object2_TDA.csv \
▷  ▷  -box_width 40 -bit_depth 24 \
▷  ▷  -end
▷  $(OPEN) 24_3_TFC_object2_TDA_flag_orbits_draw.bmp
```

The command also computes the tactical decomposition induced by the automorphism group. In addition, the command also computes the orbits on flags and on anti-flags. The third of the three geometries is flag transitive. A bitmap drawing is produced, shown below:



The geometry is flag transitive, so all incidences are drawn in the same color (black).

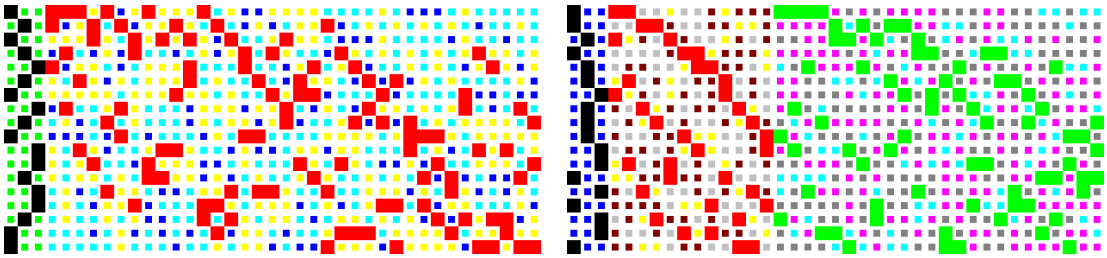
The next command produces a diagram of the flag orbits for each of the twelve Latin squares of order 6.

Example 645

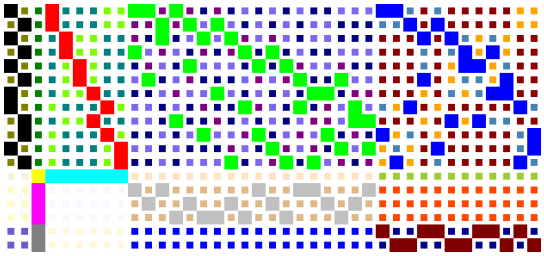
```
geo_LSQ6_c: LSQ6.inc
▷  $(ORBITER) -v 3 \
▷  ▷  -draw_incidence_structure_description \
▷  ▷  ▷  -width 60 -width_10 6 -end \
▷  ▷  -define C -combinatorial_object \
▷  ▷  ▷  -label LSQ6 LSQ6 \
▷  ▷  ▷  -file_of_incidence_geometries \
▷  ▷  ▷  ▷  LSQ6.inc 18 39 126 \
▷  ▷  -end \
▷  ▷  -with C -do \
▷  ▷  -combinatorial_object_activity \
▷  ▷  ▷  -canonical_form \
▷  ▷  ▷  ▷  -save_ago \
▷  ▷  ▷  ▷  -save_transversal \
```

```
▷ ▷ ▷ -end \
▷ ▷ -end \
▷ ▷ -with C -do \
▷ ▷ -combinatorial_object_activity \
▷ ▷ ▷ -report \
▷ ▷ ▷ ▷ -export_flag_orbits \
▷ ▷ ▷ ▷ -show_incidence_matrices \
▷ ▷ ▷ ▷ -export_group_GAP \
▷ ▷ ▷ -end \
▷ ▷ -end
▷ pdflatex LSQ6_classification.tex
▷ $(OPEN) LSQ6_classification.pdf
▷ $(ORBITER) -v 2 \
▷ ▷ -loop L 0 12 1 \
▷ ▷ ▷ -draw_matrix \
▷ ▷ ▷ -input_csv_file LSQ6_object%L_TDA_flag_orbits.csv \
▷ ▷ ▷ -secondary_input_csv_file LSQ6_object%L_TDA.csv \
▷ ▷ ▷ -box_width 32 -bit_depth 24 \
▷ ▷ ▷ -end \
▷ ▷ ▷ -system "convert \
▷ ▷ ▷ ▷ LSQ6_object%L_TDA_flag_orbits_draw.bmp \
▷ ▷ ▷ ▷ LSQ6_object%L_TDA_flag_orbits_draw.png" \
▷ ▷ -end_loop L
▷ $(ORBITER) -v 2 \
▷ ▷ -loop L 0 12 1 \
▷ ▷ ▷ -draw_matrix \
▷ ▷ ▷ -input_csv_file LSQ6_object%L_INP_flag_orbits.csv \
▷ ▷ ▷ -secondary_input_csv_file LSQ6_object%L_INP.csv \
▷ ▷ ▷ -box_width 32 -bit_depth 24 \
▷ ▷ ▷ -end \
▷ ▷ -end_loop L
```

The output is shown below

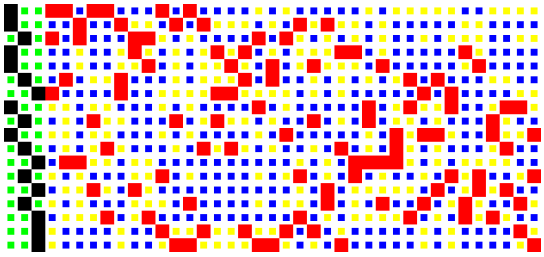


0 1

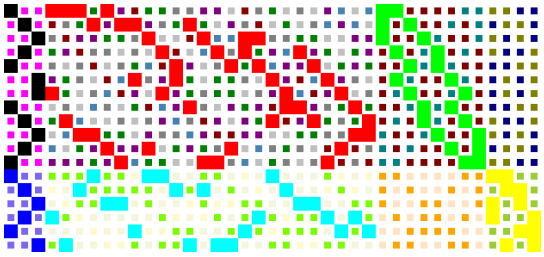


2

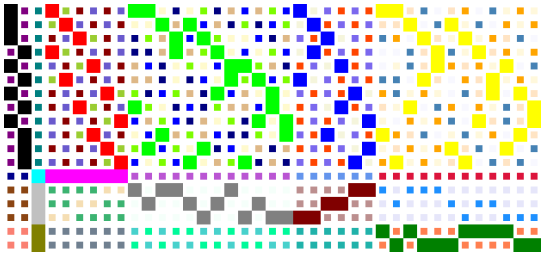
3



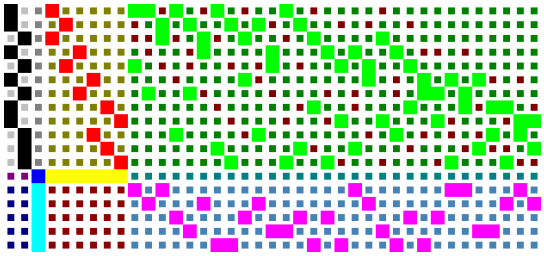
4



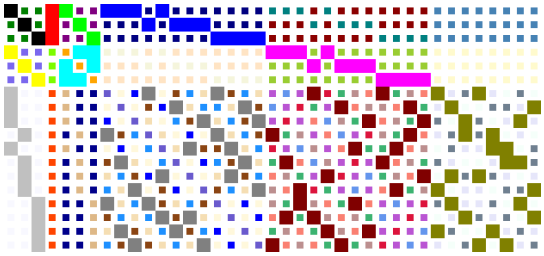
5



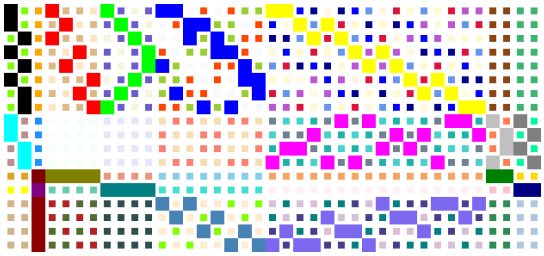
6



7

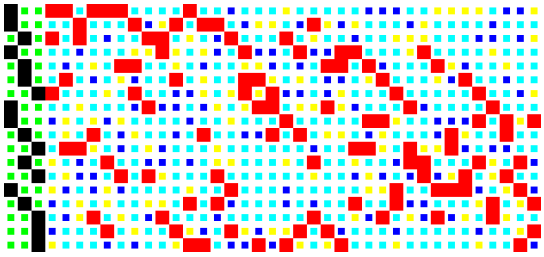


8



9

10



11

16.4 Canonical Forms of Objects from Design Theory

In Section 13.1, we discussed how to create designs in Orbiter. In this section, we wish to compute properties of designs related to the automorphism group. The first step is to export the incidence matrix of the design to file. After that, we compute the canonical form of the design, which allows us to determine many properties. The following example computes the properties of PG$(2, 3)$:

Example 646

```
design_PG_2_3_canonical:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 3 -end \
▷  ▷  -define D -design -field F -family PG_2_q -end \
▷  ▷  -with D -do \
▷  ▷  ▷  -design_activity \
▷  ▷  ▷  ▷  -export_inc \
▷  ▷  ▷  -end \
▷  ▷  -end
▷  $(ORBITER) -v 3 \
▷  ▷  -draw_incidence_structure_description \
▷  ▷  ▷  -width 60 -width_10 6 -end \
▷  ▷  -define C -combinatorial_object \
▷  ▷  ▷  -label PG_2_3 PG\_2\_3 \
▷  ▷  ▷  -file_of_incidence_geometries \
▷  ▷  ▷  ▷  PG_2_3_inc.txt 13 13 52 \
▷  ▷  -end \
▷  ▷  -with C -do \
▷  ▷  -combinatorial_object_activity \
▷  ▷  ▷  -canonical_form \
▷  ▷  ▷  ▷  -save_ago \
▷  ▷  ▷  ▷  -save_transversal \
▷  ▷  ▷  -end \
▷  ▷  -end \
▷  ▷  -with C -do \
▷  ▷  -combinatorial_object_activity \
▷  ▷  ▷  -report \
▷  ▷  ▷  ▷  -export_flag_orbits \
▷  ▷  ▷  ▷  -show_incidence_matrices \
▷  ▷  ▷  ▷  -export_group_GAP \
▷  ▷  ▷  -end \
▷  ▷  -end
▷  pdflatex PG_2_3_classification.tex
▷  $(OPEN) PG_2_3_classification.pdf
▷  $(ORBITER) -v 2 -draw_matrix \
▷  ▷  -input_csv_file PG_2_3_object0_TDA_flag_orbits.csv \
▷  ▷  -secondary_input_csv_file PG_2_3_object0_TDA.csv \
▷  ▷  -box_width 32 -bit_depth 24 \
▷  ▷  -end
▷  $(OPEN) PG_2_3_object0_TDA_flag_orbits_draw.bmp
▷
```

The command

Example 647

```
wreath_product_designs_n4_k2_c: wreath_product_designs_n4_k2_inc.txt
▷  $(ORBITER) -v 10 \
▷  ▷  -draw_incidence_structure_description \
▷  ▷  ▷  -width 60 -width_10 6 -end \
▷  ▷  -define C -combinatorial_object \
▷  ▷  ▷  -label wreath_4_2 wreath\_4\_2 \
▷  ▷  ▷  -file_of_incidence_geometries \
▷  ▷  ▷  wreath_product_designs_n4_k2_inc.txt \
▷  ▷  ▷  8 12 24 \
▷  ▷  -end \
▷  ▷  -with C -do \
▷  ▷  -combinatorial_object_activity \
▷  ▷  ▷  -canonical_form \
▷  ▷  ▷  ▷  -save_ago \
▷  ▷  ▷  ▷  -save_transversal \
▷  ▷  ▷  -end \
▷  ▷  -end \
▷  ▷  -with C -do \
▷  ▷  -combinatorial_object_activity \
▷  ▷  ▷  -report \
▷  ▷  ▷  ▷  -export_flag_orbits \
▷  ▷  ▷  ▷  -show_incidence_matrices \
▷  ▷  ▷  ▷  -export_group_GAP \
▷  ▷  ▷  -end \
▷  ▷  -end
▷  pdflatex wreath_4_2_classification.tex
▷  $(OPEN) wreath_4_2_classification.pdf
```

computes the automorphism group of the design on 8 points created in Section 13.1. The group is Sym(4) ≀ Sym(2). The command

Example 648

```
wreath_product_designs_n8_k6_c: wreath_product_designs_n8_k6_inc.txt
▷  $(ORBITER) -v 10 \
▷  ▷  -draw_incidence_structure_description \
▷  ▷  ▷  -width 60 -width_10 6 -end \
▷  ▷  -define C -combinatorial_object \
▷  ▷  ▷  -label wreath_8_6 wreath\_8\_6 \
▷  ▷  ▷  -file_of_incidence_geometries \
▷  ▷  ▷  wreath_product_designs_n8_k6_inc.txt \
▷  ▷  ▷  16 3920 23520 \
▷  ▷  -end \
▷  ▷  -with C -do \
▷  ▷  -combinatorial_object_activity \
▷  ▷  ▷  -canonical_form \
▷  ▷  ▷  ▷  -save_ago \
▷  ▷  ▷  ▷  -save_transversal \
▷  ▷  ▷  -end \
▷  ▷  -end \
```

```
▷  ▷  -with C -do \
▷  ▷  -combinatorial_object_activity \
▷  ▷  ▷  -report \
▷  ▷  ▷  ▷  -export_flag_orbits \
▷  ▷  ▷  ▷  -export_group_GAP \
▷  ▷  ▷  -end \
▷  ▷  -end
▷  pdflatex wreath_8_6_classification.tex
▷  $(OPEN) wreath_8_6_classification.pdf
```

computes the automorphism group of the design on 16 points created in Section 13.1. The group is $\mathrm{Sym}(8) \wr \mathrm{Sym}(2)$.

In Section 13.3, some large sets of $\mathrm{AG}(2,3)$ were constructed. The final isomorphism classifcation is performed using the Nauty interface. A list of combinatorial objects is created, and the `-canonical_form` command is applied as activity. This will produce a list of pairwise non-isomorphic designs. The size of this list is the number of isomorphism types of large sets of $\mathrm{AG}(2,3)$.

Example 649

```
LS_AG_2_3_solutions_classify:
▷  $(ORBITER) -v 2 \
▷  ▷  -draw_incidence_structure_description \
▷  ▷  ▷  -width 20 -width_10 2 -end \
▷  ▷  -define C -combinatorial_object \
▷  ▷  ▷  -label LS_AG_2_3 LS\_AG\_2\_3 \
▷  ▷  ▷  -file_of_designs \
▷  ▷  ▷  solutions.csv 9 84 3 12 \
▷  ▷  -end \
▷  ▷  -with C -do \
▷  ▷  -combinatorial_object_activity \
▷  ▷  ▷  -canonical_form \
▷  ▷  ▷  ▷  -save_ago \
▷  ▷  ▷  ▷  -save_transversal \
▷  ▷  ▷  ▷  -max_TDO_depth 10 \
▷  ▷  ▷  -end \
▷  ▷  -end \
▷  ▷  -with C -do \
▷  ▷  -combinatorial_object_activity \
▷  ▷  ▷  -report \
▷  ▷  ▷  ▷  -export_flag_orbits \
▷  ▷  ▷  ▷  -show_TDO \
▷  ▷  ▷  -end \
▷  ▷  -end
▷  pdflatex LS_AG_2_3_classification.tex
▷  $(OPEN) LS_AG_2_3_classification.pdf
▷  $(ORBITER) -v 2 -draw_matrix \
▷  ▷  -input_csv_file LS_AG_2_3_object0_INP_flag_orbits.csv \
▷  ▷  -secondary_input_csv_file LS_AG_2_3_object0_INP.csv \
▷  ▷  -box_width 12 -bit_depth 24 \
▷  ▷  -end
```

```
▷ $(OPEN) LS_AG_2_3_object0_INP_flag_orbits_draw.bmp
▷ $(ORBITER) -v 2 -draw_matrix \
▷ ▷ -input_csv_file LS_AG_2_3_object1_INP_flag_orbits.csv \
▷ ▷ -secondary_input_csv_file LS_AG_2_3_object1_INP.csv \
▷ ▷ -box_width 12 -bit_depth 24 \
▷ ▷ -end
▷ $(OPEN) LS_AG_2_3_object1_INP_flag_orbits_draw.bmp
```

It turns out that there are exactly two isomorphism types, with automorphism groups of order 54 and 42, respectively.

16.5 Canonical Forms of Linear Codes

Using Nauty, Orbiter can compute canonical forms and automorphism groups of codes. For linear codes, the semilinear automorphism group can be computed.

Consider the $[3, 2, 2]$ code generated by

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

Using the following command, the semilinear automorphism group of the code can be computed:

Example 650

```
code_3_2_aut:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define genma -vector -field F -format 2 \
▷  ▷  ▷  -dense $(CODE_N3_K2_Q2_GENMA) \
▷  ▷  -end \
▷  ▷  -define P -projective_space -n 1 -field F -v 0 -end \
▷  ▷  -with P -do \
▷  ▷  -projective_space_activity \
▷  ▷  ▷  -canonical_form_of_code \
▷  ▷  ▷  ▷  "3_2" genma -save_ago -label "3_2" \
▷  ▷  ▷  ▷  -classification_prefix "3_2" \
▷  ▷  ▷  -end \
▷  ▷  -end
▷  pdflatex 3_2_classification.tex
▷  $(OPEN) 3_2_classification.pdf
▷  $(ORBITER) -v 2 -draw_matrix \
▷  ▷  -input_csv_file 3_2_object0_TDA_flag_orbits.csv \
▷  ▷  -secondary_input_csv_file 3_2_object0_TDA.csv \
▷  ▷  -box_width 16 -bit_depth 24 \
▷  ▷  -end
▷  $(OPEN) 3_2_object0_TDA_flag_orbits_draw.bmp
```

The code has a semilinear automorphism group of order 6. The following report is written:

Summary of Orbits

	Rep	#	Ago	Objects
0	0	1	6	0

Ago : 6

Isomorphism type 0 / 1

Isomorphism type 0 / 1 is original object 0 and appears 1 times:
This isomorphism type appears 1 times, namely for the following 1 input objects: {0}

set of points of size 3: (0, 1, 2)

i	Rank	Point
0	0	$(1,0)$
1	1	$(0,1)$
2	2	$(1,1)$

Column sets of the encoded object:
{ 0, 1, 2 }
{ 0, 1, 2, 3 }
Row sets of the encoded object:
{ 0, 1 } = 0
{ 0, 1 } = 0
{ 0, 1 } = 0
{ 1 } = 1

Generators for the automorphism group:
The stabilizer of order 6 is generated by:
g_1 = (1, 2) of order 2 and with 4 fixed points.
g_2 = (0, 1) of order 2 and with 4 fixed points.

Generators for the automorphism group as matrix group:
The stabilizer of order 6 is generated by:

$$g_1 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 10 \\ 11 \end{bmatrix}$$ of order 2 and with 1 fixed points.

$$g_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 01 \\ 10 \end{bmatrix}$$ of order 2 and with 1 fixed points.

Decomposition by combinatorial refinement:

$\rightarrow$	2_1
4_0	2

$\downarrow$	2_1
4_0	3

Decomposition by automorphism group:

Canonical labeling:
canonical row = 3
canonical orbit number = 1
Flags : (0, 1, 2, 3, 4, 5, 7)

Flag orbits:
orbit length : number of orbits of that length:

1 1
3 2

Anti-Flag orbits:
orbit length : number of orbits of that length:

1 1

The command

Example 651

```
CODE_RM_3_1_GENMA="\
11111111\
01010101\
00110011\
00001111"
```

Example 652

```
RM_3_1_group:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define genma -vector -field F -format 4 \
▷  ▷  ▷  -compact $(CODE_RM_3_1_GENMA) \
▷  ▷  -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷  ▷  -with P -do \
▷  ▷  -projective_space_activity \
▷  ▷  ▷  -canonical_form_of_code \
▷  ▷  ▷  ▷  "RM_3_1" genma -save_ago -label "RM_3_1" \
▷  ▷  ▷  ▷  -classification_prefix "RM_3_1" \
▷  ▷  ▷  -end \
▷  ▷  -end
▷  pdflatex RM_3_1_classification.tex
▷  $(OPEN) RM_3_1_classification.pdf
```

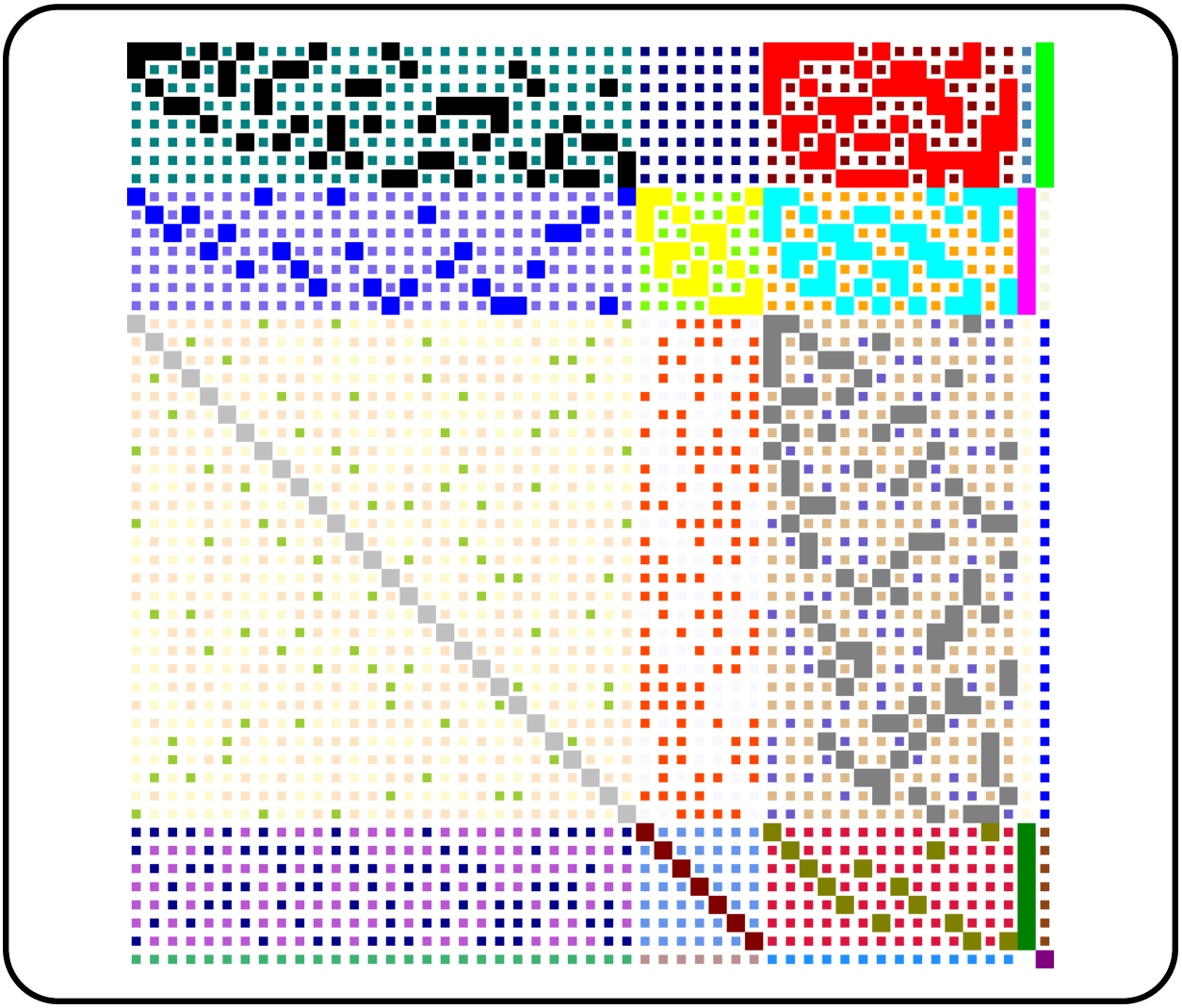
computes the automorphism group of the Reed-Muller code. The group is an affine linear group AGL(3, 2) of order 1344. A report is created, showing the automorphism group and the action on PG(3, 2), with the Reed-Muller code distinguished.

The following command creates a drawing of the incidence matrix between points and lines in PG(3, 2), with the Reed-Muller code distinguished:

Example 653

```
RM_3_1_group_and_diagram:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define genma -vector -field F -format 4 \
▷  ▷  ▷  -compact $(CODE_RM_3_1_GENMA) \
▷  ▷  -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷  ▷  -with P -do \
▷  ▷  -projective_space_activity \
▷  ▷  ▷  -canonical_form_of_code \
▷  ▷  ▷  ▷  "RM_3_1" genma -save_ago -label "RM_3_1" \
▷  ▷  ▷  ▷  -classification_prefix "RM_3_1" \
▷  ▷  ▷  -end \
▷  ▷  -end
▷  pdflatex RM_3_1_classification.tex
▷  $(OPEN) RM_3_1_classification.pdf
▷  $(ORBITER) -v 2 -draw_matrix \
▷  ▷  -input_csv_file RM_3_1_object0_INP_flag_orbits.csv \
▷  ▷  -secondary_input_csv_file RM_3_1_object0_INP.csv \
▷  ▷  -box_width 16 -bit_depth 24 \
▷  ▷  -end
▷  $(ORBITER) -v 2 -draw_matrix \
▷  ▷  -input_csv_file RM_3_1_object0_TDA_flag_orbits.csv \
▷  ▷  -secondary_input_csv_file RM_3_1_object0_TDA.csv \
▷  ▷  -box_width 16 -bit_depth 24 \
▷  ▷  -end
▷  $(OPEN) RM_3_1_object0_INP_flag_orbits_draw.bmp
▷  $(OPEN) RM_3_1_object0_TDA_flag_orbits_draw.bmp
```

The drawing is shown below:



The command

Example 654

```
RS_6_4_7_group:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 7 -end \
▷  ▷  -define genma -vector -field F -format 4 \
▷  ▷  ▷  -compact $(CODE_RS_6_4_7) \
▷  ▷  -end \
▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
▷  ▷  -with P -do \
▷  ▷  -projective_space_activity \
▷  ▷  ▷  -canonical_form_of_code \
▷  ▷  ▷  ▷  "RS_6" genma -save_ago -label "RS_6" \
▷  ▷  ▷  ▷  -classification_prefix "RS_6" \
▷  ▷  ▷  -end \
▷  ▷  -end
```

shows that the automorphism group has order 12. After some shortening, the output is:

Isomorphism type 0 / 1 is original object 0 and appears 1 times:
set of points of size 6: $\{(0,9,51,344,253,3)\}$

i	Rank	Point
0	0	$(1,0,0,0)$
1	9	$(5,1,0,0)$
2	51	$(6,5,1,0)$
3	344	$(0,6,5,1)$
4	253	$(0,0,4,1)$
5	3	$(0,0,0,1)$

Group order 12
This isomorphism type appears 1 times, namely for the following 1 input objects: $\{0\}$
Stabilizer:
Strong generators for a group of order 12:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 \\ 5 & 0 & 6 & 0 \\ 5 & 1 & 0 & 0 \end{bmatrix},$$

$$\begin{bmatrix} 0 & 0 & 0 & 3 \\ 4 & 0 & 0 & 6 \\ 0 & 4 & 0 & 1 \\ 0 & 0 & 4 & 1 \end{bmatrix}$$

1,0,0,0,2,0,0,1,5,0,6,0,5,1,0,0,
0,0,0,1,6,0,0,2,0,6,0,5,0,0,6,5,

$\rightarrow$	2850_1	1_2
401_0	57	1

The command

Example 655

```
GV_n15_k6_d5_group:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -define genma -vector -field F -format 6 \
▷  ▷  ▷  -compact $(CODE_GV_N15_K6) \
▷  ▷  -end \
▷  ▷  -define P -projective_space -n 5 -field F -v 0 -end \
▷  ▷  -with P -do \
▷  ▷  -projective_space_activity \
▷  ▷  ▷  -canonical_form_of_code \
```

```
▷ ▷ ▷ ▷ "GV_n15_k6_d5" genma -save_ago -label "GV_n15_k6_d5" \
▷ ▷ ▷ ▷ -classification_prefix "GV_n15_k6_d5" \
▷ ▷ ▷ -end \
▷ ▷ -end
▷ #pdflatex GV_n15_k6_d5_classification.tex
▷ #$(OPEN) GV_n15_k6_d5_classification.pdf
```

computes the automorphism group of the Gilbert-Varshamov code from Section 10.9. It has order 12.

16.6 Canonical Forms of General Codes

The command

Example 656

```
HAMMING_CODE_CODEWORDS="0, 67, 37, 102, 22, 85, \
51, 112, 15, 76, 42, 105, 25, 90, 60, 127"
```

Example 657

```
Hamming_graph_7_with_Hamming_code:
▷  $(ORBITER) -v 2 \
▷  ▷  -define G -graph -Hamming 7 2 \
▷  ▷  ▷  -subset "_Hamming_code" "\\_with\\_Hamming\\_code" \
▷  ▷  ▷  $(HAMMING_CODE_CODEWORDS) -end \
▷  ▷  -with G -do \
▷  ▷  -graph_theoretic_activity -export_csv -end \
▷  ▷  -with G -do \
▷  ▷  -graph_theoretic_activity -export_graphviz -end \
▷  ▷  -with G -do \
▷  ▷  -graph_theoretic_activity -save -end \
▷  ▷  -with G -do \
▷  ▷  -graph_theoretic_activity -automorphism_group -end
▷  pdflatex Hamming_7_2_Hamming_code_report.tex
▷  $(OPEN) Hamming_7_2_Hamming_code_report.pdf
```

computes the set stabilizer of the Hamming code inside the automorphism group of the Hamming graph. The group has order $2688 = 16 \cdot 168$.

16.7 Canonical Forms of Graphs

Orbiter can compute isomorphism and automorphism between graphs. Here are some examples.

Suppose we want to compute the automorphism group of the cycle graph of order 13:

Example 658

```
Cycle_13_aut:
▷  $(ORBITER) -v 2 \
▷  ▷  -define Gamma -graph -cycle 13 -end \
▷  ▷  -with Gamma -do \
▷  ▷  ▷  -graph_theoretic_activity -automorphism_group \
▷  ▷  -end \
```

The output is two files: The first one, `Cycle_13_group.makefile` is a makefile containing an Orbiter command to create the automorphism group: The second file is `Cycle_13_gens.csv`, which contains the permutation representation of the group, and which is needed for the makefile.

The next command computes the automorphism group of the chain graph with respect to the partition $(2, 3, 2)$.

Example 659

```
Chain_232_aut:
▷  $(ORBITER) -v 2 \
▷  ▷  -define P1 -vector -dense 2,3,2 -end \
▷  ▷  -define P2 -vector -dense 2,3,2 -end \
▷  ▷  -define Gamma -graph \
▷  ▷  ▷  -chain_graph P1 P2 \
▷  ▷  -end \
▷  ▷  -with Gamma -do \
▷  ▷  ▷  -graph_theoretic_activity -automorphism_group \
▷  ▷  -end
▷  pdflatex chain_graph_report.tex
▷  $(OPEN) chain_graph_report.pdf
```

The following report is written:

The automorphism group of *chain_graph* has order 1152 and is generated by:
Strong generators for a group of order 1152:

$$(12, 13),$$

$$(3, 4),$$

$$(2, 3),$$

$$(10, 11),$$

$$(9, 10),$$

$$(5, 6),$$

$$(7,8),$$

$$(0,1),$$

$$(0,12)(1,13)(2,9)(3,10)(4,11)(5,7)(6,8)$$

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 12, 14,
0, 1, 2, 4, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
0, 1, 3, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 10, 12, 13, 14,
0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 9, 11, 12, 13, 14,
0, 1, 2, 3, 4, 6, 5, 7, 8, 9, 10, 11, 12, 13, 14,
0, 1, 2, 3, 4, 5, 6, 8, 7, 9, 10, 11, 12, 13, 14,
1, 0, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
12, 13, 9, 10, 11, 7, 8, 5, 6, 2, 3, 4, 0, 1, 14,

Junttila and Kaski maintain a collection of graphs that can be used as test cases. The graphs are stored in Dimacs format and can be read in through the Orbiter `-load_dimacs` command. For instance, the following command computes the automorphism group of the Levi graph of the desarguesian projective plane of order 16:

Example 660

```
JK_graph_pp16_1:
▷  $(ORBITER) -v 2 \
▷  ▷  -define Gamma -graph -load_dimacs \
▷  ▷  ../JUNTTILA_KASKI/benchmarks/pp/pp16-1 \
▷  ▷  -end \
▷  ▷  -with Gamma -do \
▷  ▷  -graph_theoretic_activity -save -end \
▷  ▷  -with Gamma -do \
▷  ▷  -graph_theoretic_activity -automorphism_group -end \
```

The command shows a group of order 34217164800. As a measurement of the complexity, the number of backtrack nodes in Nauty is recorded:

```
nb_backtrack1 = 6
nb_backtrack2 = 134
nb_backtrack3 = 134
nb_backtrack4 = 1
```

Here, `nb_backtrack1` is the number of calls to `firstpathnode`, `nb_backtrack2` is the number of calls to `othernode`, `nb_backtrack3` is the number of calls to `processnode`, `nb_backtrack4` is the number of calls to `firstterminal`. These are the four recursive functions in Nauty.

Unfortunately, the complexity of graph isomorphism is not well-understood. We can see this here. While the first projective plane of order 16 can be handled relatively easily, the second one causes problems. The following command hardly finishes:

Example 661

```
JK_graph_pp16_2:
▷  $(ORBITER) -v 2 \
▷  ▷  -define Gamma -graph -load_dimacs \
▷  ▷  ../JUNTTILA_KASKI/benchmarks/pp/pp16-2 \
▷  ▷  -end \
▷  ▷  -with Gamma -do \
▷  ▷  -graph_theoretic_activity -save -end \
▷  ▷  -with Gamma -do \
▷  ▷  -graph_theoretic_activity -automorphism_group -end \
```

The difference between the two planes is that the first plane has a very large automorphism group, while the second one has not. For any q, the Desarguesian plane $\mathrm{PG}(2, q)$ has the largest automorphism group of all projective planes of order q.

The following example considers the block intersection graph of a Steiner triple system ("STS") of order 13. There are exactly two STS(13). The one we consider here has a group of order 39. The block intersection graph has the same automorphism group.

Example 662

```
JK_graph_sts_13:
▷  $(ORBITER) -v 2 \
▷  ▷  -define Gamma -graph -load_dimacs \
▷  ▷  ▷  ../JUNTTILA_KASKI/benchmarks/srg/sts-13 \
▷  ▷  -end \
▷  ▷  -with Gamma -do \
▷  ▷  -graph_theoretic_activity -save -end \
▷  ▷  -with Gamma -do \
▷  ▷  -graph_theoretic_activity -automorphism_group -end
▷  make ORBITER_PATH=$(ORBITER_EXE_PATH) -f sts-13_group.makefile sts-13
```

The automorphism group has order 39 and is generated by:

```
(
(1, 25, 16)(2, 18, 20)(3, 7, 15)(4, 13, 11)(5, 6, 17)
(9, 12, 19)(10, 14, 24)(21, 23, 22),
(0, 1, 2)(3, 4, 5)(7, 8, 9)(11, 12, 13)(14, 16, 18)
(15, 17, 19)(20, 22, 24)(21, 23, 25))
```

Graphs can be created from groups by means of orbitals. An orbital is an orbit of a permutation group G on the set of pairs. Here is an example. We start from the Coxeter-Tits graph on 315 vertices, whose automorphism group is the Hall-Janko group $J_2 : 2$. We first read the graph from file `halljanko315.csv` and compute the automorphism group using Nauty:

Example 663

```
halljanko315_gens.csv:
▷  $(ORBITER) -v 6 \
▷  ▷  -define G -graph \
▷  ▷  ▷  -load_csv_no_border \
▷  ▷  ▷  halljanko315.csv \
▷  ▷  -end \
▷  ▷  -with G -do \
▷  ▷  ▷  -graph_theoretic_activity -automorphism_group \
▷  ▷  -end \
▷  ▷  -with G -do \
▷  ▷  ▷  -graph_theoretic_activity -properties \
▷  ▷  -end
```

The next step is to compute the orbits of the automorphism group on pairs, using the following command:

Example 664

```
HJ_group_and_orbits:
▷  $(ORBITER) -v 2 \
▷  ▷  -define Control -poset_classification_control \
▷  ▷  ▷  -W \
▷  ▷  ▷  -problem_label HJ_orbits \
▷  ▷  ▷  -depth 2 \
▷  ▷  -end \
▷  ▷  -define gens -vector -file \
▷  ▷  ▷  halljanko315_gens.csv -end \
▷  ▷  -define G -permutation_group \
▷  ▷  ▷  -bsgs halljanko315 "File\_halljanko315" \
▷  ▷  ▷  315 1209600 "0,1,2" 6 gens \
▷  ▷  -end \
▷  ▷  -define Orb -orbits -group G \
▷  ▷  ▷  -on_subsets 2 Control \
▷  ▷  -end
```

There are 4 orbits on pairs. We decide to pick the fourth orbit to create a new graph. Because indexing is zero-based, we give the orbit index of 3:

Example 665

```
HJ_orbital_graph_3:
▷  $(ORBITER) -v 2 \
▷  ▷  -define gens -vector -file \
▷  ▷  ▷  halljanko315_gens.csv -end \
▷  ▷  -define G -permutation_group \
▷  ▷  ▷  -bsgs halljanko315 "File\_halljanko315" \
▷  ▷  ▷  315 1209600 "0,1,2" 6 gens \
▷  ▷  -end \
▷  ▷  -define Gamma -graph \
▷  ▷  ▷  -orbital_graph G 3 \
```

```
▷  ▷  -end \
▷  ▷  -with Gamma -do \
▷  ▷  ▷  -graph_theoretic_activity \
▷  ▷  ▷  -properties \
▷  ▷  -end \
▷  ▷  -with Gamma -do \
▷  ▷  ▷  -graph_theoretic_activity \
▷  ▷  ▷  -save \
▷  ▷  -end
```

The graph is regular of degree 64.

The next command computes the automorphism group of the collinearity graph of the $Q(4,2)$ quadric.

Example 666

```
PGO_5_2_graph_group: O_5_2_incidence_matrix.csv
▷  $(ORBITER) -v 3 \
▷  ▷  -define Inc -vector -file O_5_2_incidence_matrix.csv -end \
▷  ▷  -define Gamma -graph -collinearity_graph Inc -end \
▷  ▷  -with Gamma -do \
▷  ▷  -graph_theoretic_activity \
▷  ▷  ▷  -automorphism_group \
▷  ▷  -end \
▷  ▷  -with Gamma -do \
▷  ▷  -graph_theoretic_activity \
▷  ▷  ▷  -eigenvalues \
▷  ▷  -end
▷  pdflatex collinearity_graph_eigenvalues.tex
▷  $(OPEN) collinearity_graph_eigenvalues.pdf
```

The group is $\mathrm{PGO}(5,2)$ of order 720. The command creates the group as a permutation group on the 15 vertices of the graph. The group is no longer treated as a matrix group.

16.8 Canonical Forms of Quartic Curves

We wish to study the automorphism groups of the quartic curves of Edge that have been considered in Section 4.10. We start by creating a cheat sheet of the field $\mathbb{F}_{17}$

Example 667

```
F_17_edge:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 17 -end \
▷  ▷  -with F -do -finite_field_activity \
▷  ▷  ▷  -cheat_sheet_GF -end
▷  pdflatex GF_17.tex
▷  $(OPEN) GF_17.pdf
```

Next, we compute the canonical form of the Edge quartic. This command also computes generators for the automorphism group of the curve.

Example 668

```
Edge_curve_17_nauty:
▷  $(ORBITER) -v 3 \
▷  ▷  -define C -combinatorial_object \
▷  ▷  ▷  -label Edge_curve_q17 Edge\_curve\_q17 \
▷  ▷  ▷  -file_of_points Edge_q17.txt \
▷  ▷  -end \
▷  ▷  -define F -finite_field -q 17 -end \
▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
▷  ▷  -with C -do \
▷  ▷  -combinatorial_object_activity \
▷  ▷  ▷  -canonical_form_PG P \
▷  ▷  ▷  ▷  -save_ago \
▷  ▷  ▷  ▷  -save_transversal \
▷  ▷  ▷  ▷  -max_TDO_depth 10 \
▷  ▷  ▷  ▷  -end \
▷  ▷  ▷  -end \
▷  ▷  -with C -do \
▷  ▷  -combinatorial_object_activity \
▷  ▷  ▷  -report \
▷  ▷  ▷  ▷  -export_flag_orbits \
▷  ▷  ▷  ▷  -show_TDO \
▷  ▷  ▷  ▷  -show_TDA \
▷  ▷  ▷  ▷  -dont_show_incidence_matrices \
▷  ▷  ▷  ▷  -export_group_GAP \
▷  ▷  ▷  -end \
▷  ▷  -end
▷  pdflatex Edge_curve_q17_classification.tex
▷  $(OPEN) Edge_curve_q17_classification.pdf
▷  $(ORBITER) -v 2 -draw_matrix \
▷  ▷  -input_csv_file Edge_curve_q17_object0_TDA_flag_orbits.csv \
▷  ▷  -secondary_input_csv_file Edge_curve_q17_object0_TDA.csv \
▷  ▷  -box_width 4 -bit_depth 24 \
```

```
▷  ▷  -end
▷  $(OPEN) Edge_curve_q17_object0_TDA_flag_orbits_draw.bmp
```

Using the generators that have just been computed, we can recreate the group of the quartic curve:

Example 669

```
Edge_curve_17_group:
▷  $(ORBITER) -v 3 \
▷  ▷  -define G -linear_group -PGL 3 17 \
▷  ▷  -subgroup_by_generators "Stab_Edge" "24" 3 \
▷  ▷  ▷  "1,0,0,0,13,0,0,0,4, \
▷  ▷  ▷  1,0,0,0,0,16,0,16,0, \
▷  ▷  ▷  0,1,16,2,4,4,15,4,4" \
▷  ▷  -end \
▷  ▷  -with G -do \
▷  ▷  ▷  -group_theoretic_activity \
▷  ▷  ▷  ▷  -print_elements_tex \
▷  ▷  ▷  ▷  -report_group_table \
▷  ▷  ▷  ▷  -report \
▷  ▷  -end
▷  pdflatex PGL_3_17_Subgroup_Stab_Edge_24_report.tex
▷  $(OPEN) PGL_3_17_Subgroup_Stab_Edge_24_report.pdf
```

Chapter 17

Interfaces

17.1 Graphical Output

Orbiter can produce graphical output in a variety of formats:

1. TikZ / Latex [69],
2. Metapost [37],
3. Bitmap files (.bmp) [72],
4. Povray, see Section 17.2,
5. Gnuplot [33], see Section 17.5.

Bitmaps can be created using the `-draw_matrix` command. The input is an integer-valued matrix in csv format. The matrix entries are translated into colors. The possible commands after `-draw_matrix` are shown in Table 17.1.

Orbiter can draw incidence matrices of geometries, for instance in reports. Table **??** shows the available commands to taylor such drawings.

The poset classification algorithm from Sections 7.3 and 7.4 computes partially ordered sets. The posets are created using the `-draw_poset` option in the poset classification control command package, see Table 7.4. The posets are stored in a file with extension `.layered_graph`. These files can be drawn using the `-draw_layered_graph` command. The commands in Table 17.2 and Table 17.3 show ways in which to customize the drawings. Let us consider an example. Suppose we are interested in the Schreier trees of a permutation group represented in a stabilizer chain. We take $\mathrm{PGL}(4,2)$ in its action on the wedge product. The command

Example 670

```
PGL_6_2_Wedge_4_2_detached_report.tex:
▷ $(ORBITER) -v 4 \
▷ ▷ -define G -linear_group -PGL 4 2 \
▷ ▷ ▷ -wedge_detached \
▷ ▷ -end \
▷ ▷ -with G -do \
▷ ▷ -group_theoretic_activity \
▷ ▷ ▷ -report \
▷ ▷ -end
```

Creating a Bitmap Graphics		
Command	Arguments	Purpose
-input_csv_file	csv-file	Specify the input csv file.
-secondary_input_csv_file	csv-file	Specify a secondary input csv file (optional).
-input_object	I	Specify input to be Orbiter object I.
-partition	w R C	Specify a partition R of rows and C of columns. Use separating lines of with w.
-box_width	w	Use w pixels per matrix entry.
-bit_depth	d	Use color bit depth of d bits ($d = 8$ or $d = 24$).
-invert_colors		Use an inverted color scheme.
-grayscale		Use a grayscale color set.

Table 17.1: Creating a Bitmap Graphics

Options for Drawing Incidence Structures		
Command	Arguments	Purpose
-width	w	Set the width of one box in units of 0.1mm.
-width_10	w	Set the width of one tenth of a box. Should with the value of width divided by 10.
-outline_thin		Draw a thin outline.
-unit_length		Default: 0.065mm.
-thick_lines		Default: 0.5mm.
-thin_lines		Default: 0.15mm.
-geo_line_width		Default: 0.25mm.

Table 17.2: Options for Drawing Incidence Structures

Drawing Options for Layered Graph Files (Part 1)

Command	Arguments	Purpose
-paperheight	h	Set paperheight for the latex document.
-paperwidth	w	Set paperwidth for the latex document.
-xin	a	Assume input x-coordinates are in the interval $[0, a]$. Default value: 10000.
-yin	a	Assume input y-coordinates are in the interval $[0, a]$. Default value: 10000.
-xout	a	Assume output x-coordinates are in the interval $[0, a]$. Default value: 1000000.
-yout	a	Assume output y-coordinates are in the interval $[0, a]$. Default value: 1000000.
-spanning_tree		Place nodes according to a spanning tree. Default value: off.
-circle		Circle all nodes. Default value: on.
-corners		Draw corners at the outside of the drawing. Default value: off.
-radius	r	Use radius r for drawing circles around nodes. Default value: 200.
-embedded		Create latex headers for stand-alone latex files. Default value: off.
-sideways		Create latex figure sideways. Default value: off.
-label_edges		Label the edges in Schreier trees. Default value: off.
-x_stretch	s	Apply x-axis scaling by a factor of s. Default value: $s = 1.0$. This option does not affect the drawing of Schreier trees.
-y_stretch	s	Apply y-axis scaling by a factor of s. Default value: $s = 1.0$. This option does not affect the drawing of Schreier trees.

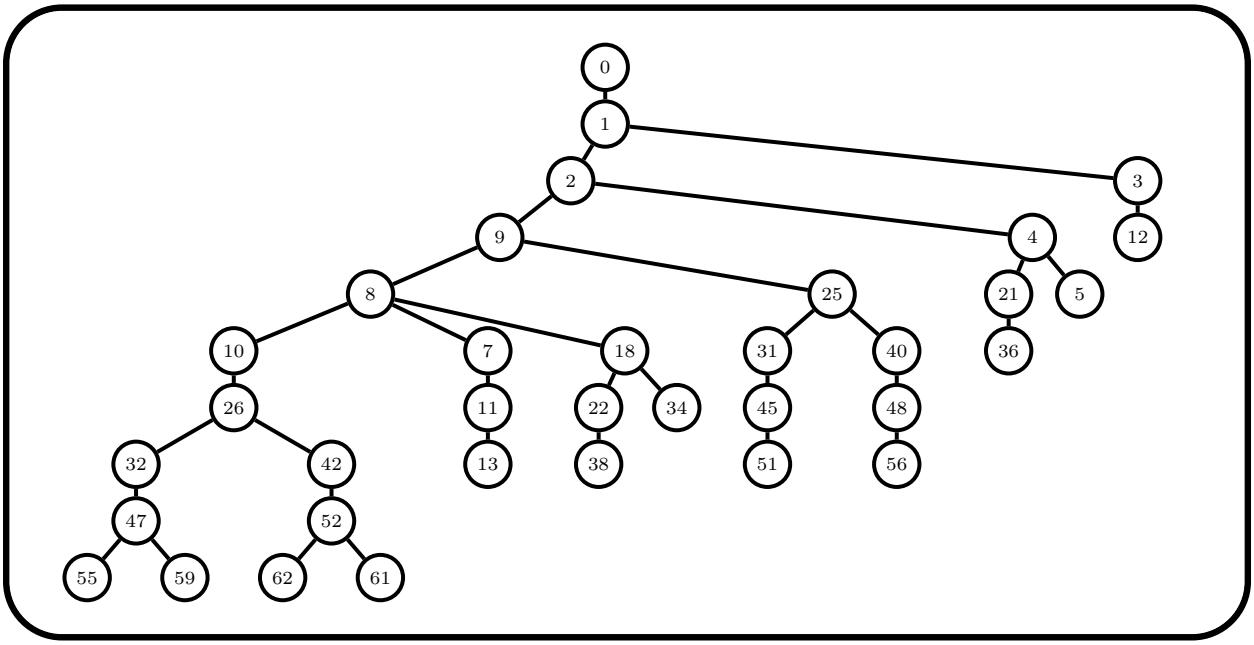
Table 17.3: Drawing Options for Layered Graph Files (Part 1)

Drawing Options for Layered Graph Files (Part 2)		
Command	Arguments	Purpose
-scale	s	Use Tikz global scale-factor of s. Default value: $s = 0.45$.
-line_width	s	Set Tikz line width to s. Default value: $s = 1.5$.
-rotated		Rotate the output.
-nodes		Turn on node drawing.
-nodes_empty		Do not label the nodes. Default value: off.
-select_layers	S	Draw layers whose index is given in the list S only.
-paths_in_between	$l_1\ i_1\ l_2\ i_2$	Draw all paths from node (l_1, i_1) to node (l_2, i_2). Here, (l, i) is the i-th node at layer l (counting from zero). Delete all other edges between layers l_1 and l_2.

Table 17.4: Drawing Options for Layered Graph Files (Part 2)

```
▷ pdflatex PGL_6_2_Wedge_4_2_detached_report.tex
▷ $(OPEN) PGL_6_2_Wedge_4_2_detached_report.pdf
```

produces a report. The following tree is taken from this report. It represents the first basic orbit in the stabilizer chain of the group in that action.

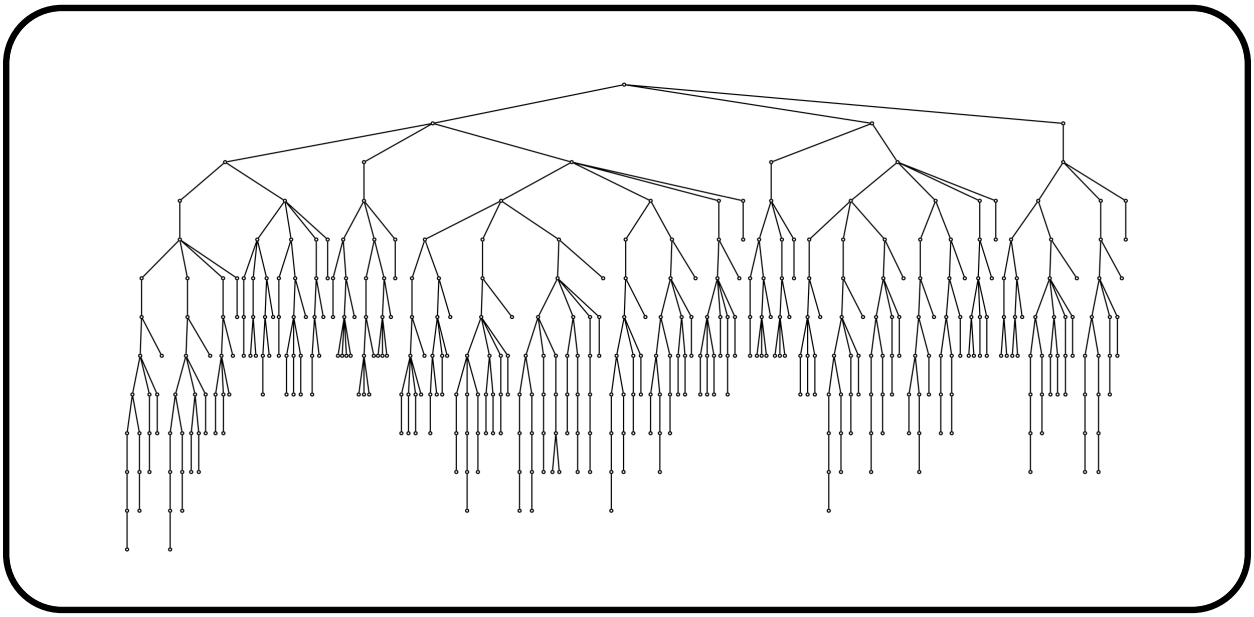


The command

Example 671

```
schreier_tree_graphical_output:
▷  $(ORBITER) -v 4 \
▷  ▷  -draw_options \
▷  ▷  ▷  -yout 500000 \
▷  ▷  ▷  -radius 15 -nodes_empty \
▷  ▷  ▷  -line_width 0.5 -y_stretch 0.25 \
▷  ▷  ▷  -embedded \
▷  ▷  -end \
▷  ▷  -define G -linear_group -PGL 4 2 -end \
▷  ▷  -define Orb -orbits -group G \
▷  ▷  ▷  -on_polynomials 3 \
▷  ▷  -end \
▷  ▷  -with Orb -do -orbits_activity \
▷  ▷  ▷  -export_something "orbit" 6 \
▷  ▷  -end
▷  ▷  -with Orb -do -orbits_activity \
▷  ▷  ▷  -draw_tree 6 \
▷  ▷  -end
▷
```

draws the 6th Schreier tree in the classification of orbits of PGL$(4, 2)$ on homogeneous polynomials of degree 3 in 4 variables, shown below



The orbit has length 420.

Povray Interface		
Command	Arguments	Purpose
-povray	description	Povray job description, see Table 17.5.
-prepare_frames	n	Prepare frames dexription, see Table 17.11.

Table 17.5: Povray Interface

Povray Job Options		
Command	Arguments	Purpose
-video_options		Set options from Tables 17.6 and 17.7.
-round	n	Select round n for rendering only.
-rounds	list	Select a list of rounds for rendering.
-nb_frames_default	f	Select the number of frames per round.
-output_mask	fname-mask	Set mask for output file name generation.
-scene_objects	options	Define Scene objects according to Table 17.10.

Table 17.6: Povray Job Options

17.2 The Povray Interface

Orbiter can be used to create raytracing 3D-graphics. Orbiter serves as a front end for the raytracing software Povray [62]. This is a multi step process: A 3D scene is defined through orbiter commands. Next, Orbiter produces Povray files. After that, the povray files are processed through povray, and turned into graphics files (png), called frames. The frames can be turned into a video by using tools like ffmpeg (see Section 17.3). By default, an rotational animation is produced. The main commands are listed in Table 17.4.

To describe a povray job, the commands in Table 17.5 can be used.

The Orbiter Povray interface requires some general information about the animation, the camera position, the boundary box for clipping, the font size for text and others. Tables 17.6- 17.7 list the commands to control the 3D-povray frontend.

A povray graphics starts with a description of the scene. The scene will be rendered to the final frame. A scene is composed of geometric objects. Various types of objects are available: points, lines, planes, faces, algebraic surfaces, reguli, 3D-text, and others. Some complex objects are predefined, for instance the Eckardt surface. Once the objects are defined, output commands can be invoked to draw them in various colors and with various options. At times, there are many objects in one scene. In order to ease drawing, objects can be grouped together. Only objects of the same type may be grouped. A group of object can be drawn at once. Tables 17.8 and 17.9 summarize the Orbiter commands to build objects of a 3D scene. The commands in Table 17.10 are used to build the scene. Each of these commands applies to a group of objects of the same kind. Groups of objects are created using the commands in Table 17.9 which start with `group_of`. Here is a simple example which combines scene building and graphical output. The example creates a cube with vertices, edges and faces:

Example 672

```
cube:
▷  $(ORBITER) -v 2 -povray \
▷  ▷  -round 0 -nb_frames_default 30 \
```

Povray Interface (Part 1)

Command	Arguments	Purpose
-do_not_rotate		No rotation. By default, the animation consists of a full rotation around a vertical axis.
-rotate_about_z_axis		Rotate around z-axis.
-rotate_about_111		Rotate around $(1,1,1)$-axis (default).
-rotate_about_custom_axis	axis	Rotate around a custom axis. The axis is specified as a vector of length 3.
-boundary_none		Remove the clipping.
-boundary_box		Clip using a box shape.
-boundary_sphere		Clip using a sphere (default).
-font_size	s	Set font size to s.
-stroke_width	s	Set text depth to s.
-omit_bottom_plane		Remove the bottom plane.
-W	w	Set output dimension to w pixels wide.
-H	h	Set output dimension to h pixels height.
-nb_frames	n	Set number of frames to n. One revolution around the axis is split into n frames.
-zoom	$r\ a_s\ a_t\ c_s\ c_t$	Set zoom angle and clipping with in round r to change from a_s to a_t and from c_s to c_t, respectively.
-pan	$r\ F\ T\ C$	In round r, pan the camera from location F to location T in a rotational movement with center at C. Each of F, T, C are three dimensional coordinates.
-pan_reverse	$r\ F\ T\ C$	Same as -pan, but camera movement is in opposite order.
-no_background		Remove background.
-no_bottom_plane	r	Remove bottom plane in round r.
-camera	$r\ S\ C\ L$	In round r, set camera location at C, sky at S and pointing towards L. Each of S, C, L are three-dimensional coordinate vectors.

Table 17.7: Povray Interface (Part 1)

Povray Interface (Part 2)

Command	Arguments	Purpose
-clipping	r c	In round r, set clipping radius to c.
-text	r a text	In round r, produce running text text with sustain value a.
-label	r s a g text	In round r, produce running text text with start value s, sustain s and gravity g.
-latex	r s a praeamble g text l fname	In round r, produce running latex text text with start value s, sustain s and gravity g. Put praeamble in the latex source code. Use fname for the latex file names (no extension).
-global_picture_scale	d	Set scaling factor to d.
-picture	r d fname options	In round r, place picture fname scaled by d using options.
-picture	r d fname options	In round r, place picture fname scaled by d using options.
-look_at	L	Override camera look-at value to L. L is a three-dimensional vector.
-default_angle	a	Set default camera angle to a.
-clipping_radius	f	Set default clipping radius to f.
-scale_factor	s	Set default scale factor to s.
-line_radius	s	Set default line radius to s.

Table 17.8: Povray Interface (Part 2)

```
▷ ▷ -output_mask cube_%d_%03d.pov \
▷ ▷ -video_options -W 1024 -H 768 \
▷ ▷ -global_picture_scale 0.5 \
▷ ▷ -default_angle 75 \
▷ ▷ -clipping_radius 2.7 \
▷ ▷ -end \
▷ ▷ -scene_objects \
▷ ▷ ▷ -obj_file cube_centered.obj \
▷ ▷ ▷ -edge "0, 1" \
▷ ▷ ▷ -edge "0, 2" \
▷ ▷ ▷ -edge "0, 4" \
▷ ▷ ▷ -edge "1, 3" \
▷ ▷ ▷ -edge "1, 5" \
▷ ▷ ▷ -edge "2, 3" \
▷ ▷ ▷ -edge "2, 6" \
▷ ▷ ▷ -edge "3, 7" \
▷ ▷ ▷ -edge "4, 5" \
▷ ▷ ▷ -edge "4, 6" \
▷ ▷ ▷ -edge "5, 7" \
▷ ▷ ▷ -edge "6, 7" \
▷ ▷ ▷ -group_of_things_as_interval 0 8 \
▷ ▷ ▷ -spheres 0 0.3 $(POLISHED_CHROME_WHITE) \
▷ ▷ ▷ -group_of_things_as_interval 0 6 \
```

Scene Elements (Part 1)		
Command	Arguments	Purpose
-cubic_lex	coeffs	Cubic surface given by 20 coefficients in lexicographic ordering
-cubic_orbiter	coeffs	Cubic surface given by 20 coefficients in Orbiter ordering
-cubic_Goursat	$A\ B\ C$	Cubic surface with tetrahedral symmetry given by 3 Goursat coefficients as $Axyz + B(x^2+y^2+z^2)+C=0$
-quadric_lex_10	coeffs	Quadric surface given by 10 coefficients in lexicographic ordering
-quartic_lex_35	coeffs	Quartic surface given by 35 coefficients in lexicographic ordering
-octic_lex_165	coeffs	Octic surface given by 165 coefficients in lexicographic ordering
-point	coeffs	Point given by three coordinates
-point_list_from_csv_file	fname	List of points with coordinates given in a csv file
-line_through_two_points_recentered_from_csv_file	fname	List of lines through two points with point coordinates given in a csv file
-line_through_two_points_from_csv_file	fname	List of lines through two points with point coordinates given in a csv file
-point_as_intersection_of_two_lines	$i_1\ i_2$	Create a point from the intersection of two lines i_1 and i_2
-edge	$i_1\ i_2$	Create an edge (line segment) between points i_1 and i_2
-text	$i_1\ s$	Create a label s located at the point i
-triangular_face_given_by_three_lines	$i_1\ i_2\ i_3$	Create a triangular face give by three lines i_1, i_2, i_3
-face	pts	Create a face through the vertices pts, ordered cyclically
-quadric_through_three_skew_lines	$i_1\ i_2\ i_3$	Create a quadric through three skew lines
-plane_defined_by_three_points	$i_1\ i_2\ i_3$	Create a plane through three noncollinear points
-line_through_two_points_recentered	pt-coords	Create a line through two points given by 6 coordinates, recentered

Table 17.9: Scene Elements (Part 1)

Scene Elements (Part 2)

Command	Arguments	Purpose
-line_through_two_points	pt-coords	Create a line through two points given by 6 coordinates
-line_through_two_existing_points	i_1 i_2	Create a line through two points
-line_through_point_with_direction	x y z u_x u_y u_z	Create a line through a point (x, y, z) with a given direction (u_x, u_y, u_z)
-plane_by_dual_coordinates	a b c d	Create the plane $ax + by + cz + d = 0$ given in dual coordinates
-dodecahedron		Create a Dodecahedron centered at the origin (20 points, 30 edges, 12 faces)
-Hilbert_Cohn_Vossen_surface		Create the Hilbert, Cohn-Vossen surface (1 cubic surface, 45 tritangent planes, 27 lines)
-obj_file	fname	Read points and faces from the given .obj file
-group_of_things	list	Create a group of things from the given list
-group_of_things_with_offset	list offset	Create a group of things from the given list, each value is increase by offset
-group_of_things_as_interval	a b	Create a group of things indexed by the interval $a, \ldots, a + b - 1$
-group_of_things_as_interval_with_exceptions	a b ex	Create a group of things indexed by the interval $a, \ldots, a + b - 1$ with the exceptional elements in the list ex removed
-group_of_all_points		Create a group of things from all points currently defined
-group_of_all_faces		Create a group of things from all faces currently defined
-group_subset_at_random	i f	Create a group of things from the existing group i by picking a random subset with probability f
-create_regulus	i N	Create a regulus for quadric i with N lines

Table 17.10: Scene Elements (Part 2)

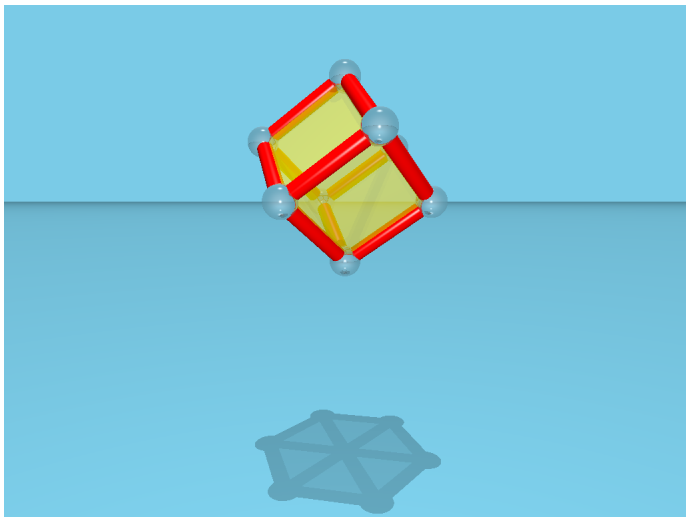
Scene Object Types

Command	Arguments	Purpose
`-spheres`	i r prop	For each element in point group i, create a sphere of radius r with given Povray properties.
`-cylinders`	i r prop	For each element in edge group i, create a cylinder of radius r with given Povray properties.
`-prisms`	i d prop	For each element in face group i, create a prism of half-thickness d with given Povray properties.
`-planes`	i prop	For each element in plane group i, create a plane with given Povray properties.
`-lines`	i r prop	For each element in line group i, create a line of radius r with given Povray properties.
`-cubics`	i prop	For each element in group i of cubics, create a surface with given Povray properties.
`-quadrics`	i prop	For each element in group i of quadrics, create a surface with given Povray properties.
`-quartics`	i prop	For each element in group i of quartics, create a surface with given Povray properties.
`-octics`	i prop	For each element in group i of octics, create a surface with given Povray properties.
`-texts`	i d s prop	For each element in group i of labels, create a text element with half-thickness d and size s with given Povray properties.

Table 17.11: Scene Object Types

```
▷  ▷  ▷  -prisms 1 0.05 $(YELLOW_TRANSPARENT) \
▷  ▷  ▷  -group_of_things_as_interval 0 12 \
▷  ▷  ▷  -cylinders 2 0.15 $(COLOR_RED) \
▷  ▷  -scene_objects_end \
▷  ▷  -povray_end
▷  - rm -rf POV
▷  mkdir POV
▷  mv cube_0_*.pov POV
▷  mv makefile_animation POV
```

This command instructs Orbiter to create 30 povray files (extension .pov), one for each frame of a rotating scene. The scene containes a cube whose vertices are shown in chrome, whose edges are in red, and whose faces are yellow and transparent. The cube turns around a vertical axis of symmetry. Here is the first frame of the result:



The coordinates of the cube are stored in an object file `cube_centered.obj`. The content of this file is:

```
v -1 -1 -1
v 1 -1 -1
v -1 1 -1
v 1 1 -1
v -1 -1 1
v 1 -1 1
v -1 1 1
v 1 1 1
f 1 2 4 3
f 1 2 6 5
f 1 3 7 5
f 2 4 8 6
f 3 4 8 7
```

```
f 5 6 8 7
```

The monkey saddle is a cubic surface, given by the equation

$$z = x^3 - 3xy^2$$

The next example plots the surface knowns as the monkey saddle. The tangent plane at $(0, 0, 0)$ is drawn as well. An animation is created by rotating the scene around the z-axis.

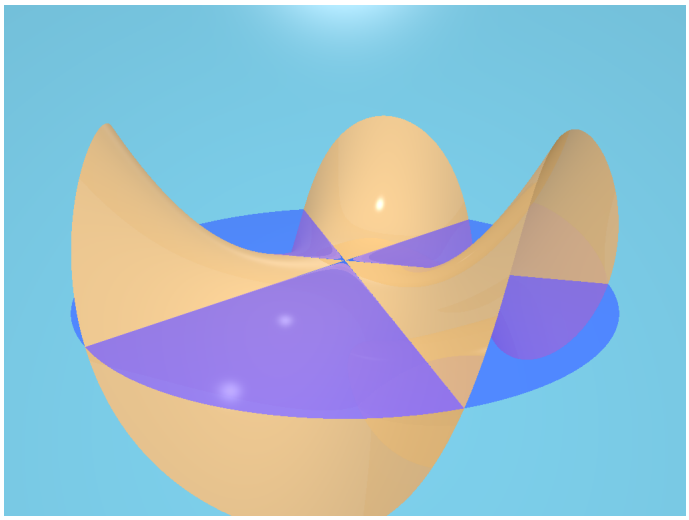
Example 673

```
MONKEY_SADDLE_CUBIC="1,0,0,0,-3,0,0,0,0,0,0,0,0,0,0,0,0,0,-1,0"
```

Example 674

```
monkey:
▷  $(ORBITER) -v 2 -povray \
▷  ▷  -round 0 -nb_frames_default 30 \
▷  ▷  -output_mask monkey_%d_%03d.pov \
▷  ▷  -video_options -W 1024 -H 768 \
▷  ▷  -global_picture_scale 0.8 \
▷  ▷  -default_angle 75 \
▷  ▷  -clipping_radius 0.8 \
▷  ▷  -camera 0 "0,0,1" "1,1,0.5" "0,0,0" \
▷  ▷  -rotate_about_z_axis \
▷  ▷  -end \
▷  ▷  -scene_objects \
▷  ▷  ▷  -cubic_lex $(MONKEY_SADDLE_CUBIC) \
▷  ▷  ▷  -plane_by_dual_coordinates "0,0,1,0" \
▷  ▷  ▷  -group_of_things "0" \
▷  ▷  ▷  -group_of_things "0" \
▷  ▷  ▷  -cubics 0 $(COLOR_GOLD) \
▷  ▷  ▷  -planes 1 $(COLOR_BLUE) \
▷  ▷  -scene_objects_end \
▷  ▷  -povray_end
▷  - rm -rf POV
▷  mkdir POV
▷  mv monkey_0_*.pov POV
▷  mv makefile_animation POV
```

Here is one of the frames that are created:



The Eckardt surface is given by the equation

$$\frac{5}{2}xyz - (x^2 + y^2 + z^2) + 1 = 0.$$

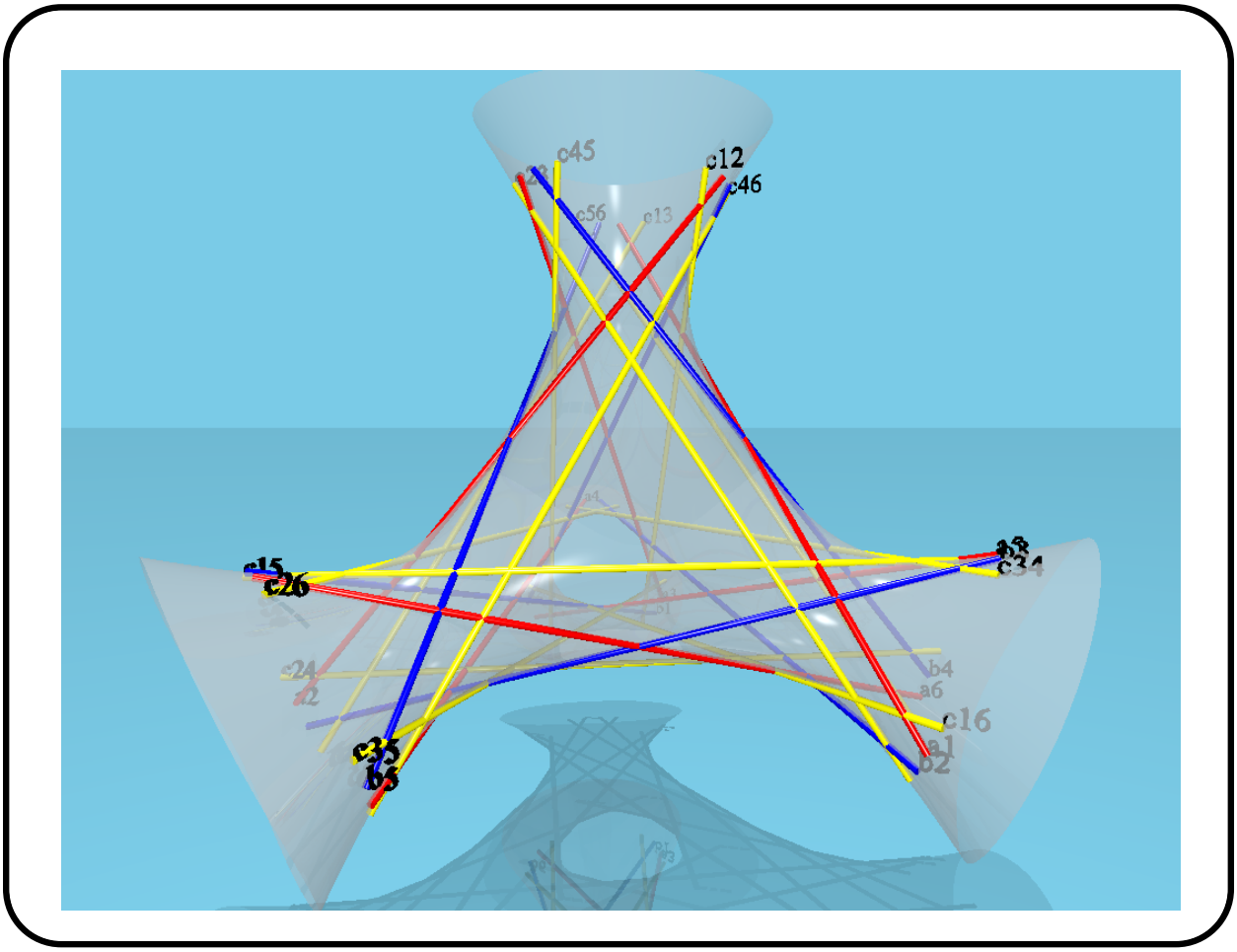
We use the following code to plot the surface and the lines on it. The Schläfli labeling of the lines is indicated.

Example 675

```
Eckardt:
▷  $(ORBITER) -v 2 -povray \
▷  ▷  -round 0 -nb_frames_default 30 \
▷  ▷  -output_mask Eckardt_%d_%03d.pov \
▷  ▷  -video_options -W 1024 -H 768 \
▷  ▷  -global_picture_scale 0.9 \
▷  ▷  -default_angle 75 \
▷  ▷  -clipping_radius 2.4 \
▷  ▷  -camera 0 "1,1,1" "-3,1,3" "0.12,0.12,0.12" \
▷  ▷  -end \
▷  ▷  -scene_objects \
▷  ▷  ▷  -Hilbert_Cohn_Vossen_surface \
▷  ▷  ▷  -group_of_things "0" \
▷  ▷  ▷  -cubics 0 $(SURFACE_COLOR) \
▷  ▷  ▷  -group_of_things_as_interval 0 6 \
▷  ▷  ▷  -group_of_things_as_interval 6 6 \
▷  ▷  ▷  -group_of_things_as_interval_with_exceptions 12 15 \
▷  ▷  ▷  ▷  "14,19,23" \
▷  ▷  ▷  -lines 1 0.02 $(COLOR_RED_SHINY) \
▷  ▷  ▷  -lines 2 0.02 $(COLOR_BLUE_SHINY) \
▷  ▷  ▷  -lines 3 0.02 $(COLOR_YELLOW_SHINY) \
▷  ▷  ▷  -label 0 "a1" \
▷  ▷  ▷  -label 2 "a2" \
▷  ▷  ▷  -label 4 "a3" \
▷  ▷  ▷  -label 6 "a4" \
▷  ▷  ▷  -label 8 "a5" \
▷  ▷  ▷  -label 10 "a6" \
```

```
▷  ▷  ▷  -label 12 "b1" \
▷  ▷  ▷  -label 14 "b2" \
▷  ▷  ▷  -label 16 "b3" \
▷  ▷  ▷  -label 18 "b4" \
▷  ▷  ▷  -label 20 "b5" \
▷  ▷  ▷  -label 22 "b6" \
▷  ▷  ▷  -label 24 "c12" \
▷  ▷  ▷  -label 26 "c13" \
▷  ▷  ▷  -label 30 "c15" \
▷  ▷  ▷  -label 32 "c16" \
▷  ▷  ▷  -label 34 "c23" \
▷  ▷  ▷  -label 36 "c24" \
▷  ▷  ▷  -label 40 "c26" \
▷  ▷  ▷  -label 42 "c34" \
▷  ▷  ▷  -label 44 "c35" \
▷  ▷  ▷  -label 48 "c45" \
▷  ▷  ▷  -label 50 "c46" \
▷  ▷  ▷  -label 52 "c56" \
▷  ▷  ▷  -group_of_things_as_interval 0 6 \
▷  ▷  ▷  -texts 4 0.2 0.15 $(COLOR_BLACK_NO_SHADOW) \
▷  ▷  ▷  -group_of_things_as_interval 6 6 \
▷  ▷  ▷  -texts 5 0.2 0.15 $(COLOR_BLACK_NO_SHADOW) \
▷  ▷  ▷  -group_of_things_as_interval 12 12 \
▷  ▷  ▷  -texts 6 0.2 0.15 $(COLOR_BLACK_NO_SHADOW) \
▷  ▷  -scene_objects_end \
▷  ▷  -povray_end
▷  - rm -rf POV
▷  mkdir POV
▷  mv Eckardt_0_*.pov POV
▷  mv makefile_animation POV
```

The rendering yields:



The Endrass octic [28] is the algebraic surface given by the equation

$$X8 := 64\left(-w^2+x^2\right)\left(-w^2+y^2\right)\left((x+y)^2-2w^2\right)\left((x-y)^2-2w^2\right)-\left(-4\left(1+\sqrt{2}\right)\left(x^2+y^2\right)^2+\left(8\left(2+\sqrt{2}\right)z^2+2\left(2+7\sqrt{2}\right)w^2\right)\left(x^2+y^2\right)-16z^4+8\left(1-2\sqrt{2}\right)z^2w^2-\left(1+12\sqrt{2}\right)w^4\right)^2$$

The following Orbiter command creates a povray graphics of the octic.

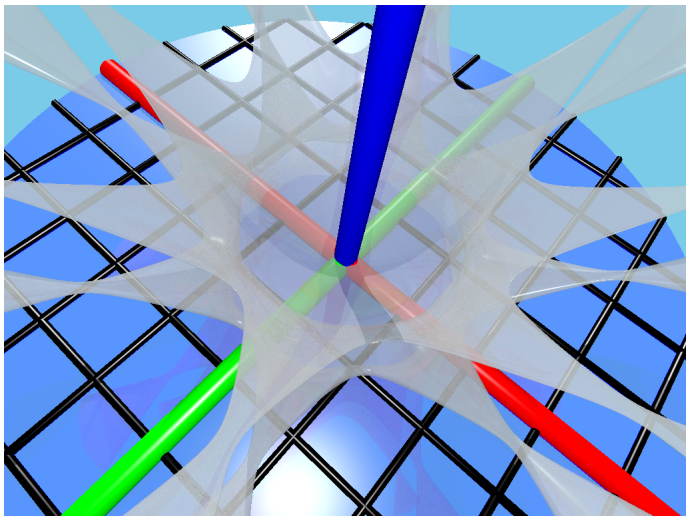
Example 676

```
ENDRASS_OCTIC_LEX_165="-93.2548,0,0,0,-309.019,0,0,527.529,0,395.647,\
0,0,0,0,0,0,0,0,0,0,-687.529,0,0,1582.59,0,1186.94,0,0,0,0,-1055.06,0,\
-1582.59,0,-593.47,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-309.019,\
0,0,1582.59,0,1186.94,0,0,0,0,-2110.12,0,-3165.17,0,-1186.94,0,0,0,0,0,\
0,874.039,0,1560.63,0,1677.92,0,343.362,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,\
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-93.2548,0,0,527.529,0,395.647,\
0,0,0,0,-1055.06,0,-1582.59,0,-593.47,0,0,0,0,0,0,874.039,0,1560.63,0,\
1677.92,0,343.362,0,0,0,0,0,0,0,0,-256,0,-468.077,0,-789.019,0,\
-525.726,0,0.941125"
```

Example 677

```
endrass8:
▷  cp $(ORBITER_PATH)/examples/users_guide/coordinate_grid.csv .
▷  $(ORBITER) -v 2 -povray \
▷  ▷  -round 0 -nb_frames_default 30 \
▷  ▷  -output_mask endrass_octic_%d_%03d.pov \
▷  ▷  -video_options -W 1024 -H 768 \
▷  ▷  -global_picture_scale 0.75 \
▷  ▷  -default_angle 75 \
▷  ▷  -clipping_radius 3.7 \
▷  ▷  -no_bottom_plane \
▷  ▷  -camera 0 "1,1,1" "6,6,3" "0,0,0" \
▷  ▷  -rotate_about_111 \
▷  ▷  -end \
▷  ▷  -scene_objects \
▷  ▷  ▷  -line_through_two_points_recentered_from_csv_file \
▷  ▷  ▷  ▷  coordinate_grid.csv \
▷  ▷  ▷  -group_of_things "0" \
▷  ▷  ▷  -group_of_things "1" \
▷  ▷  ▷  -group_of_things "2" \
▷  ▷  ▷  -group_of_things_as_interval 3 39 \
▷  ▷  ▷  -lines 0 0.15 $(COLOR_RED_SHINY) \
▷  ▷  ▷  -lines 1 0.15 $(COLOR_GREEN_SHINY) \
▷  ▷  ▷  -lines 2 0.15 $(COLOR_BLUE_SHINY) \
▷  ▷  ▷  -lines 3 0.05 $(COLOR_BLACK_SHINY) \
▷  ▷  ▷  -octic_lex_165 $(ENDRASS_OCTIC_LEX_165) \
▷  ▷  ▷  -plane_by_dual_coordinates "0,0,1,0" \
▷  ▷  ▷  -group_of_things "0" \
▷  ▷  ▷  -group_of_things "0" \
▷  ▷  ▷  -octics 4  $(SURFACE_COLOR_SEETHROUGH) \
▷  ▷  ▷  -planes 5 "texture{ pigment{ color Blue transmit 0.5 } \
finish { diffuse 0.9 phong 1}}" \
▷  ▷  -scene_objects_end \
▷  ▷  -povray_end
▷  - rm -rf POV
▷  mkdir POV
▷  mv endrass_octic_0_*.pov POV
▷  mv makefile_animation POV
```

Here is the rendering:



It includes coordinate axes and the x, y-plane.

Preparing Frames for an Animation

Command	Arguments	Purpose
-i	s l mask	Specify the input file names by running a printf command with the given mask applied to the index i where i loops over all integers from s to $s + l - 1$. This option can be repeated.
-step	s	Increment the index in steps of size s.
-o	mask	Create the output file using the given mask.
-output_starts_at	i	Start output file indices at i (default is 0).

Table 17.12: Preparing Frames for an Animation

17.3 Creating Animations

Orbiter can be used to create animations. This relies on the software `ffmpeg`. In a first step, all frames (i.e. individual graphics files) are created using Orbiter's povray interface. After that, the frames are used to create the animation. In order to use `ffmpeg`, the frames should have a uniform file naming scheme, using a consecutive numbering to arrange the files in order. This is achieved by using a printf style mask, with `%d` representing the number of the current frame. In order to do so, Orbiter can be used to copy and rename files. A temporary directory can be used to collect the files. The Orbiter command `prepare_frames` can be used. For a list of commands, see Tables 17.11. For instance, the command

Example 678

```
monkey_video:
▷  - rm -r FRAMES
▷  - mkdir FRAMES
▷  - rm monkey.mp4
▷  $(ORBITER) \
▷  ▷  -prepare_frames \
▷  ▷  ▷  -i 0 30 monkey_0_%03d.png \
▷  ▷  ▷  -output_starts_at 0 \
▷  ▷  ▷  -o FRAMES/frame%04d.png \
▷  ▷  -end
▷  ffmpeg -r 5 -f image2 -i FRAMES/frame%04d.png \
▷  ▷  -f mp4 -q:v 0 -vcodec mpeg4 monkey.mp4
```

creates a video `monkey.mp4` from a set of 30 files. The individual filenames are created using the printf format string `monkey_0_%03d.png`, with an integer index that is drawn from the interval $[0, 29]$. The part that starts with a percent sign and ends with a "d" character defines the way in which the integer is formatted. The number three before the "d" indicates that three characters will be printed. The zero indicates the use of leading zeros. So, the first file would be `monkey_0_000.png` and the very last file is `monkey_0_029.png`. The description of the printf format string can be found in the documentation of the C standard library [43].

17.4 Continuous Function Plotter

Orbiter can plot functions using a built-in function tracker. The functions must be continuous apart from a finite number of poles. The function can have multiple components, each described using an expression. Each expression is specified in Reverse Polish Notation (RPN). Consider an example. A Lissajous curve is defined using coordinate functions of the form

$$x = r\sin(at + c), \quad y = r\sin(bt), \quad a, b, c, r \in \mathbb{R}.$$

The terms

$$r\sin(at + c), \quad r\sin(bt)$$

are the expressions of the two coordinate functions. RPN means that the operator is listed after the operands. A stack data structure is used to hold temporary values. Operators are pushed to the top of the stack using the push commands. A binary operator pops the two elements from the stack, performs the operation, and pushes the resulting value back onto the stack. For a unary operator, only one element is popped and replaced by the result. Here are some examples of expressions rewritten in RPN:

$$\begin{aligned} \sin(x) &\mapsto \text{ push } x \text{ sin}, \\ a + b &\mapsto \text{ push } a \text{ push } b \text{ add}, \\ a \cdot b &\mapsto \text{ push } a \text{ push } b \text{ mult}. \end{aligned}$$

The coordinate functions are enclosed between `-code` and `-code_end` commands. Each coordinate function is described in RPN and terminated using a `return` keyword. By the time the `return` keyword is reached, the RPN expression must have exactly one value on the stack which is considered the value of the expression. Constants are declared between the `-const` and `-const_end` keywords. Likewise, variables are declared between the `-var` and `-var_end` keywords. Picking $a = 3$, $b = 2$, $c = \pi/2$ and $r = 7$, the function is computed using

Example 679

```
lissajous:
▷  $(ORBITER) -v 2 \
▷  ▷  -smooth_curve "lissajous" 0.07 2000 15 0 18.85 \
▷  ▷  -const a 3 b 2 c 1.57 r 7 -const_end \
▷  ▷  -var t -var_end \
▷  ▷  -code \
▷  ▷  ▷  push t push a mult push c add sin push r mult return \
▷  ▷  ▷  push t push b mult sin push r mult return \
▷  ▷  -code_end \
```

The sequence

```
push t push a mult push c add sin push r mult
```

is $r\sin(at + c)$ expressed in RPN. The constants are defined in the line

```
-const a 3 b 2 c 1.57 r 7 -const_end
```

The input variable is defined using the line

```
-var t -var_end
```

The sequence

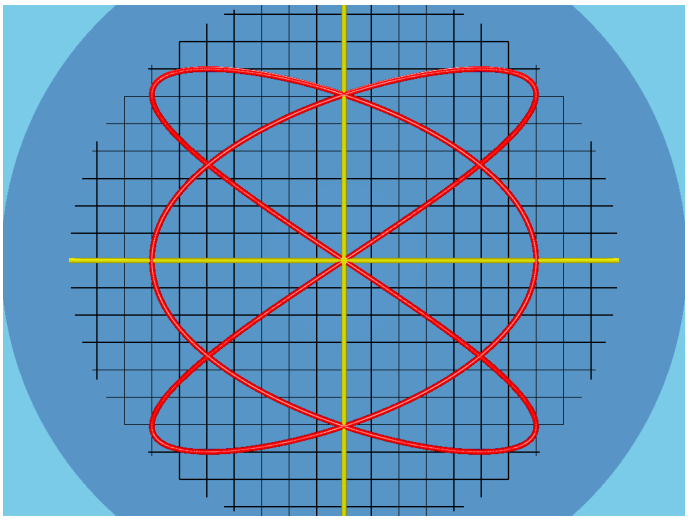
```
-smooth_curve "lissajous" 0.07 2000 15 0 18.85
```

defines the name of the output file, the fact that two consecutive points are never further than $\epsilon = 0.07$ away, the fact that points that are 15 or more away from the origin should be ignored, and the fact that the variable t loops over the range $[0, 18.85]$ with a default of 2000 steps. The evaluator automatically reduces the step-size if consecutive image points are more than ϵ apart. The code to produce the plot is

Example 680

```
lissajous_plot:
▷  cp $(ORBITER_PATH)/examples/users_guide/coordinate_grid.csv .
▷  $(ORBITER) -v 2 -povray \
▷  ▷  -round 0 -nb_frames_default 1 \
▷  ▷  -output_mask lissajous_%d_%03d.pov \
▷  ▷  -video_options -W 1024 -H 768 \
▷  ▷  -global_picture_scale 0.40 \
▷  ▷  -default_angle 45 \
▷  ▷  -clipping_radius 5 \
▷  ▷  -omit_bottom_plane \
▷  ▷  -camera 0 "0,-1,0" "0,0,12" "0,0,0" \
▷  ▷  -rotate_about_z_axis \
▷  ▷  -end \
▷  ▷  -scene_objects \
▷  ▷  ▷  -line_through_two_points_recentered_from_csv_file \
▷  ▷  ▷  coordinate_grid.csv \
▷  ▷  ▷  -group_of_things "0" \
▷  ▷  ▷  -group_of_things "1" \
▷  ▷  ▷  -group_of_things "2" \
▷  ▷  ▷  -lines 0 0.09 "texture{ pigment{ color Yellow } }" \
▷  ▷  ▷  -lines 1 0.09 "texture{ pigment{ color Yellow } }" \
▷  ▷  ▷  -lines 2 0.09 "texture{ pigment{ color Yellow } }" \
▷  ▷  ▷  -group_of_things_as_interval 3 39 \
▷  ▷  ▷  -lines 3 0.02 "texture{ pigment{ color Black } }" \
▷  ▷  ▷  -point_list_from_csv_file \
▷  ▷  ▷  function_lissajous_N2000_points.csv \
▷  ▷  ▷  -group_of_things_as_interval 0 6524\
▷  ▷  ▷  -spheres 4 0.1 "texture{ pigment{ color Red } \
finish { diffuse 0.9 phong 1}}" \
▷  ▷  ▷  -plane_by_dual_coordinates "0,0,1,0" \
▷  ▷  ▷  -group_of_things "0" \
▷  ▷  ▷  -planes 5 "texture{ pigment{ color Blue*0.5 \
transmit 0.5 } }" \
▷  ▷  -scene_objects_end \
▷  ▷  -povray_end
▷  - rm -rf POV
▷  mkdir POV
▷  mv lissajous_0_*.pov POV
▷  mv makefile_animation POV
```

The plot is shown below



We can turn it into a 3D plot by using the t value for the z coordinate. The function is computed using the command

Example 681

```
lissajous_3d:
▷  $(ORBITER) -v 2 \
▷  ▷  -smooth_curve "lissajous_3d" 0.07 2000 50 0 18.85 \
▷  ▷  -const a 3 b 2 c 1.57 r 7 -const_end \
▷  ▷  -var t -var_end \
▷  ▷  -code \
▷  ▷  ▷  push t push a mult push c add sin push r mult return \
▷  ▷  ▷  push t push b mult sin push r mult return \
▷  ▷  ▷  push t return \
▷  ▷  -code_end \
```

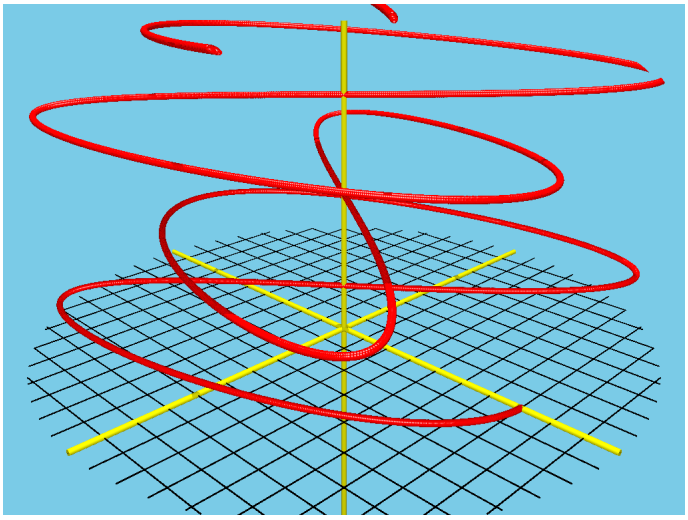
The code to produce the 3D plot is

Example 682

```
lissajous_3d_plot:
▷  cp $(ORBITER_PATH)/examples/users_guide/coordinate_grid.csv .
▷  $(ORBITER) -v 2 -povray \
▷  ▷  -round 0 -nb_frames_default 30 \
▷  ▷  -output_mask lissajous_3d_%d_%03d.pov \
▷  ▷  -video_options -W 1024 -H 768 \
▷  ▷  -global_picture_scale 0.40 \
▷  ▷  -default_angle 45 \
▷  ▷  -clipping_radius 5 \
▷  ▷  -omit_bottom_plane \
▷  ▷  -camera 0 "0,0,1" "7,7,5" "0,0,1" \
▷  ▷  -rotate_about_z_axis \
```

```
▷  ▷  -end \
▷  ▷  -scene_objects \
▷  ▷  ▷  -line_through_two_points_recentered_from_csv_file \
▷  ▷  ▷  coordinate_grid.csv \
▷  ▷  ▷  -group_of_things "0" \
▷  ▷  ▷  -group_of_things "1" \
▷  ▷  ▷  -group_of_things "2" \
▷  ▷  ▷  -lines 0 0.09 "texture{ pigment{ color Yellow } }" \
▷  ▷  ▷  -lines 1 0.09 "texture{ pigment{ color Yellow } }" \
▷  ▷  ▷  -lines 2 0.09 "texture{ pigment{ color Yellow } }" \
▷  ▷  ▷  -group_of_things_as_interval 3 39 \
▷  ▷  ▷  -lines 3 0.02 "texture{ pigment{ color Black } }" \
▷  ▷  ▷  -point_list_from_csv_file \
▷  ▷  ▷  function_lissajous_3d_N2000_points.csv \
▷  ▷  ▷  -group_of_things_as_interval 0 6538\
▷  ▷  ▷  -spheres 4 0.1 "texture{ pigment{ color Red } \
finish { diffuse 0.9 phong 1}}" \
▷  ▷  ▷  -plane_by_dual_coordinates "0,0,1,0" \
▷  ▷  ▷  -group_of_things "0" \
▷  ▷  -scene_objects_end \
▷  ▷  -povray_end
▷  - rm -rf POV
▷  mkdir POV
▷  mv lissajous_3d_0_*.pov POV
▷  mv makefile_animation POV
```

The 3D curve is shown below



17.5 The Gnuplot Interface

Orbiter can interface with gnuplot [33] to draw diagrams. Here is an example. Suppose we have data in a csv file like this:

```
Row,Curve-1,Curve-2
1,2,4
2,6,8
3,9,10
4,13,17
5,17,20
6,19,25
7,23,30
END
```

The following command creates this csv file from a makefile variable

Example 683

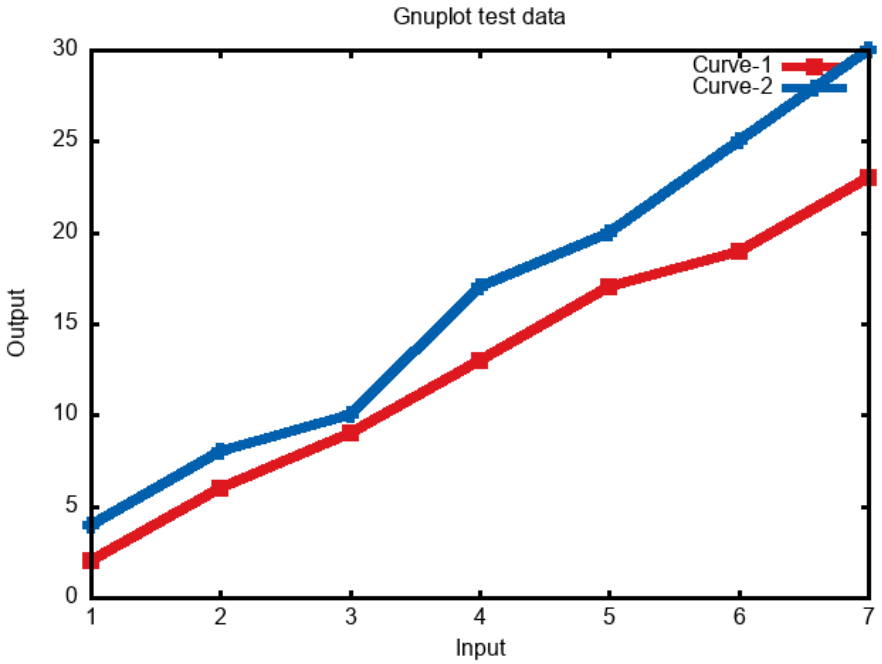
```
GNUPLOT_TEST_INPUT="Row,Curve-1,Curve-2\n1,2,4\n2,6,8\n3,9,10\n4,13,17\n5,17,20\n
6,19,25\n7,23,30\nEND\n"
```

The next command invokes the Orbiter gnuplot interface:

Example 684

```
gnuplot_test_data:
▷  echo $(GNUPLOT_TEST_INPUT) >gnuplot_test_data.csv
▷  $(ORBITER) -v 3 \
▷  ▷  -gnuplot gnuplot_test_data.csv \
▷  ▷  "Gnuplot test data" \
▷  ▷  "Input" \
▷  ▷  "Output"
▷  $(OPEN) gnuplot_test_data_gnuplot.png
```

The resulting plot created by gnuplot is:

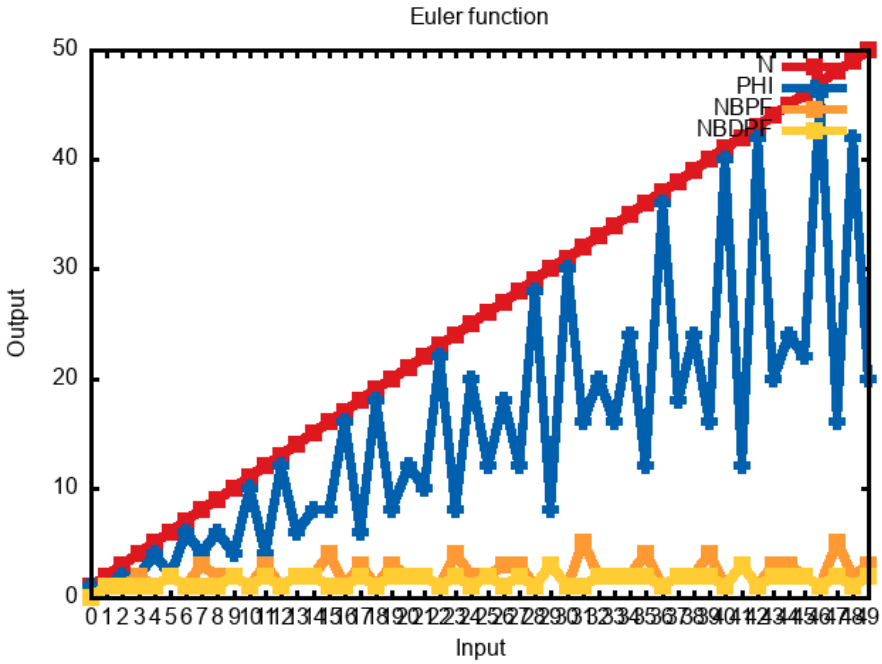


The next command plots the Eulerfunction, the number of prime divisors and the number of distinct prime divisors of all integers n between 1 and 50:

Example 685

```
gnuplot_Eulerfunction_50:
▷  $(ORBITER) -v 1 -eulerfunction_interval 1 50
▷  $(ORBITER) -v 3 \
▷  ▷  -gnuplot table_eulerfunction_1_50.csv \
▷  ▷  "Euler function" \
▷  ▷  "Input" \
▷  ▷  "Output"
```

The resulting plot is:


Euler function
Output
Input
N
PHI
NBPF
NBDPF
0
10
20
30
40
50

Chapter 18

Mathematical Data in Orbiter

18.1 Cubic Surfaces

Orbiter contains a knowledge base of mathematical data. This is like a database, except for the way that the data is stored. The data does not reside in files, like in a database. Rather, the data is compiled into the Orbiter executable. This means that the data is immediately available, without any delay. We will now discuss what data exists in Orbiter and how to access it.

In most cases, the data that Orbiter stores includes the automorphism group of the objects. Individual objects can be accessed by creating them from the catalogue. Other functions exist that can write reports about all objects of a certain type.

The command

Example 686

```
make_table_of_surfaces:
▷  $(ORBITER) -v 3 \
▷  ▷  -make_table_of_surfaces
▷  pdflatex surfaces_report.tex
▷  $(OPEN) surfaces_report.pdf
```

creates a table of all cubic surfaces contained in the Orbiter knowledge base. The output is shown in Appendix A.1.

The command

Example 687

```
make_table_of_quartic_curves:
▷  $(ORBITER) -v 3 \
▷  ▷  -make_table_of_quartic_curves
▷  pdflatex quartic_curves_report.tex
▷  $(OPEN) quartic_curves_report.pdf
```

creates a table of all quartic curves contained in the Orbiter knowledge base.

It is possible to export data on cubic surfaces and quartic curves. Two file types are supported: csv and sql. For instance, the command

Example 688

```
cubic_surfaces_table_q17:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 17 -end \
▷  ▷  -define P -projective_space \
▷  ▷  ▷  -n 3 -field F -v 0 \
▷  ▷  -end \
▷  ▷  -with P -do \
▷  ▷  ▷  -projective_space_activity \
▷  ▷  ▷  ▷  -table_of_cubic_surfaces \
▷  ▷  -end
```

exports the cubic surfaces over the field $\mathbb{F}_{17}$ contained in the Orbiter knowledge base. The files `table_of_cubic_surfaces_q17_info.csv` and `table_of_cubic_surfaces_q17_data.sql` are created. For each isomorphism class of cubic surfaces, one line is generate in the file. The command

Example 689

```
quartic_curves_table_q17:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 17 -end \
▷  ▷  -define P -projective_space \
▷  ▷  ▷  -n 2 -field F -v 0 \
▷  ▷  -end \
▷  ▷  -with P -do \
▷  ▷  ▷  -projective_space_activity \
▷  ▷  ▷  ▷  -table_of_quartic_curves \
▷  ▷  -end
```

exports all quartic curves over the field $\mathbb{F}_{17}$

18.2 BLT-Sets

Orbiter contains a library of BLT-sets. It is possible to export data on BLT-sets to csv-files. For instance, the command

Example 690

```
blt_set_table_q13:
▷ $(ORBITER) -v 3 \
▷ ▷ -define F -finite_field -q 13 -end \
▷ ▷ -define O -orthogonal_space 0 5 F -end \
▷ ▷ -with O -do \
▷ ▷ ▷ -orthogonal_space_activity \
▷ ▷ ▷ ▷ -table_of_blt_sets \
▷ ▷ -end
```

exports the BLT-sets over the field $\mathbb{F}_{13}$ contained in the Orbiter knowledge base.

The command

Example 691

```
blt_set_table:
▷ $(ORBITER) -v 30 \
▷ ▷ -define Q -vector -dense "3,5,7,9,11,13,17,19,23,25,27,29,31" -end \
▷ ▷ -define NB -vector -dense "1,2,2,3,4,3,6,5,9,6,6,9,8" -end \
▷ ▷ -loop_over i Q \
▷ ▷ ▷ -define F%i -finite_field -q "%i[Q]" -end \
▷ ▷ ▷ -define O%i -orthogonal_space 0 5 F%i -end \
▷ ▷ ▷ -with O%i -do \
▷ ▷ ▷ ▷ -orthogonal_space_activity \
▷ ▷ ▷ ▷ ▷ -table_of_blt_sets \
▷ ▷ ▷ ▷ -end \
▷ ▷ ▷ -end \
▷ ▷ -end_loop_over i \
▷ ▷ -print_symbols
```

creates tables of BLT-sets for all orders up to 31.

The command

Example 692

```
BLT_ORDER_Q="3,5,7,9,11,13,17,19,23,25,27,\
29,31,37,41,43,47,49,53,59,61,67,71,73"
```

Example 693

```
BLT_NUMBER_ISO="1,2,2,3,4,3,6,5,9,6,6,\
9,8,7,10,6,10,8,8,9,5,6,8,5"
```

Example 694

```
blt_set_export_gap:
▷  $(ORBITER) -v 30 \
▷  ▷  -define Q -vector -dense \
▷  ▷  ▷  $(BLT_ORDER_Q) \
▷  ▷  -end \
▷  ▷  -define NB -vector -dense \
▷  ▷  ▷  $(BLT_NUMBER_ISO) \
▷  ▷  -end \
▷  ▷  -loop_over i Q \
▷  ▷  ▷  -define F%i -finite_field -q "%i[Q]" -end \
▷  ▷  ▷  -define O%i -orthogonal_space 0 5 F%i -end \
▷  ▷  ▷  -loop j 0 %i[NB] 1 \
▷  ▷  ▷  ▷  -define BLT_%i_%j -BLT_set \
▷  ▷  ▷  ▷  ▷  -space O%i -catalogue %j \
▷  ▷  ▷  ▷  -end \
▷  ▷  ▷  ▷  -with BLT_%i_%j -do -blt_set_activity \
▷  ▷  ▷  ▷  ▷  -export_gap \
▷  ▷  ▷  ▷  -end \
▷  ▷  ▷  -end_loop j \
▷  ▷  -end_loop_over i \
▷  ▷  -print_symbols
```

exports all BLT-sets of order at most 73 to GAP.

Chapter 19

Miscellaneous

19.1 Miscellaneous

Tables 19.1-19.2 list miscellaneous Orbiter commands.
The command `-csv_file_select_rows` can be used to select rows from a csv file. The command `-csv_file_select_cols` can be used to select columns from a csv file. The command `-csv_file_select_rows_and_cols` selects rows and columns. Here is an example. We create the multiplication table of the finite field $\mathbb{F}_7$, ordered according to the powers of a primitive element:

$$\alpha^0, \alpha^1, \alpha^2, \alpha^3, \alpha^4, \alpha^5.$$

The even powers of α create a multiplicative subgroup. We select the rows and columns corresponding to even powers $\alpha^0, \alpha^2, \alpha^4$.

Example 695

```
misc_select:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 7 -end \
▷  ▷  -with F -do -finite_field_activity -cheat_sheet_GF -end
▷  $(ORBITER) -v 4 -csv_file_select_rows_and_cols \
▷  ▷  GF_q7_multiplication_table_reordered.csv \
▷  ▷  "0,2,4" "0,2,4"
```

Here is the file `GF_q7_multiplication_table_reordered.csv`

```
Row,C0,C1,C2,C3,C4,C5
0,1,3,2,6,4,5
1,3,2,6,4,5,1
2,2,6,4,5,1,3
3,6,4,5,1,3,2
4,4,5,1,3,2,6
5,5,1,3,2,6,4
END
```

and next the file that is created by selecting rows and columns $0, 2, 4$:

Miscellaneous Orbiter Commands (Part 1)

Command	Arguments	Purpose
-create_files	descr	Create text files.
-save_matrix_csv	label	Save matrix to a csv file.
-csv_file_tally	fname	Tally a csv file.
-csv_file_select_rows	fname R	Selects rows listed in R from the csv-file fname.
-csv_file_split_rows_modulo	fname n	Splits the rows from the csv-file fname modulo n.
-csv_file_select_cols	fname R	Selects columns listed in R from the csv-file fname.
-csv_file_select_rows_and_cols	fname R C	Selects rows listed in R and columns listed in C from the csv-file fname.
-csv_file_sort_each_row	fname	Sorts each individual row in the given csv file, thinking of the entries as a set.
-csv_file_sort_rows	fname	Sorts the rows in the given csv file.
-csv_file_join	fname col-label	Joins csv file fname according to column with label col-label. This option is given once for each file that should be joined.
-csv_file_concatenate	fname-out f1 f2 ...	Join csv files f1, f2, ... The output is written to fname-out.
-csv_file_concatenate_from_mask	N fname-mask fname-out	Join csv files. The input files arise from fname-mask with all values $i = 0, \ldots, N-1$. The output is written to fname-out.
-csv_file_extract_column_to_txt	fname col-label	
-csv_file_latex	header fname	Produces a latex table from the given csv-file. The binary variable header determines whether a header is produced.
-draw_matrix	descr	Produce a bitmap graphic, see Table 17.1.
-reformat	fname-in fname-out nb-cols	Reformat the data from fname-in in nb-cols columns. Write the output to fname-out.
-split_by_values	fname-in	Split the file fname-in by values. For each value, a separate file is written.

Table 19.1: Miscellaneous Orbiter Commands (Part 1)

Miscellaneous Orbiter Commands (Part 2)

Command	Arguments	Purpose
-change_values	fname-in fname-out map-in map-out	Change the values of the file in fname-in according to a mapping. The output is written to fname-out. The mapping is defined by two corresponding sequences. The input values are in map-in, the corresponding output values are in map-out.
-store_as_csv_file	fname m n L	Stores the data in L to a csv file. The data is an $m \times n$ matrix in row-major ordering.
-mv	fname-in fname-out	Rename file from fname-in to fname-out.
-system	command	Execute the given system command.
-loop	variable val_from val_to val_step	Create a loop with the given loop variable, whose values rangle from loop_from to val_to minus one in steps of val_step.
-loop_over	variable domain	Create a loop with the given loop variable, whose values rangle over the given domain.
-plot_function	function-name	
-draw_projective_curve	options	
-tree_draw	options	Draw a tree from a description. For options, see Table 19.3.
-extract_from_file	fname-in key fname-out	Extract lines from a text file, starting with the given key and continuing until the next empty row.
-extract_from_file_with_tail	fname-in key tail fname-out	Extract lines from a text file, starting with the given key and continuing until the next empty row.
-serialize_file_names	fname output-mask	
-save_4_bit_data_file	fname vector-data	
-gnuplot	fname title label_x label_y	Produce an diagram of data series using gnuplot [33].
-compare_columns	fname col1 col2	
-gcd_worksheet	nb_problems N key	
-draw_layered_graph	options	

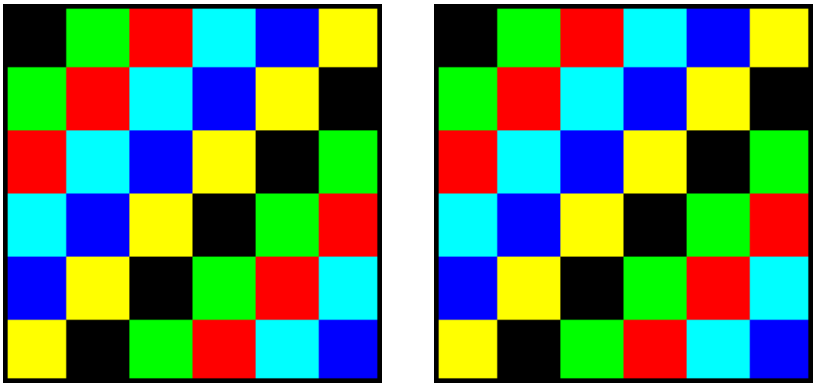
Table 19.2: Miscellaneous Orbiter Commands (Part 2)

Options for Drawing Trees		
Command	Arguments	Purpose
-file	fname	File in .tree format
-restrict	color	
-select_path	path	

Table 19.3: Options for Drawing Trees

```
Row,"C0","C2","C4"
0,"1","2","4"
1,"2","4","1"
2,"4","1","2"
END
```

A graphical output is shown below:



19.2 Limitations

Several limitations exist in Orbiter. Here is a list:

1. Field elements are encoded as `int`. This limits the size of fields that can be handled to 2^{8s-1} where $s =$`sizeof(int)`.
2. The ranks of elements in the permutation domain are encoded as `long int`. This limits the size of permutation domains that can be handled. The degree of a permutation group must be less that 2^{8s-1} where $s =$`sizeof(long int)`.
3. The finite field class builds tables for the addition and multiplication of field elements. This restricts the size of the fields that can be created.
4. The projective geometry class tries to build a bitmatrix for the adjacency matrix if the number of lines is less than `MAX_NUMBER_OF_LINES_FOR_INCIDENCE_MATRIX` which is defined in `src/lib/foundations/geometry/projective_space.cpp`. If the number of lines is too big, the table is not created. In this case, the projective geometry class may behave slower.
5. The projective geometry class tries to build a table for the lines if the number of points is less that `MAX_NUMBER_OF_POINTS_FOR_POINT_TABLE` and the number of lines is less than `MAX_NUMBER_OF_LINES_FOR_LINE_TABLE`, both of which are defined in `src/lib/foundations/geometry/projective_space.cpp`. If the number of points is too big, the table is not created. In this case, the projective geometry class may behave slow.
6. The projective geometry class tries to build a table for the lines through any two points if the number of points is less than `MAX_NB_POINTS_FOR_LINE_THROUGH_TWO_POINTS_TABLE` which is defined in `src/lib/foundations/geometry/projective_space.cpp`. If the number of points is too big, the table is not created. In this case, the projective geometry class may behave slow.
7. The projective geometry class tries to build a table for the intersection points of pairs of lines if the number of points is less than `MAX_NB_POINTS_FOR_LINE_INTERSECTION_TABLE` which is defined in `src/lib/foundations/geometry/projective_space.cpp`. If the number of points or lines is too big, the table is not created. In this case, the projective geometry class may behave slow.
8. For Windows users: Cygwin by default uses 32 bit integers for both `int` and `long int`. Using Cygwin 64 to compile Orbiter recommended.
9. A limited list of primitive polynomials are hard-coded in Orbiter. For large fields, the user must provide their own primitive polynomial. The polynomials encoded in orbiter are not guaranteed to be compatible with the subfield relationship.

The command

Example 696

```
F_15bit:
▷  $(ORBITER) -v 10 \
▷  ▷  -define F -finite_field -q 32749 -without_tables -end \
▷  ▷  -define v -vector -field F -allow_negatives \
▷  ▷  ▷  -dense "-1,-1" -end \
▷  ▷  -with F -do -finite_field_activity -product_of v -end
▷
```

creates the field $\mathbb{F}_{32749}$. Tables are suppressed for speed. Note that

$$2^{15} - 19 = 32749$$

is prime (see https://primes.utm.edu/lists/2small/0bit.html). The command creates two instances of -1 in the field and multiplies them.

The command

Example 697

```
F_31bit:
▷  $(ORBITER) -v 10 \
▷  ▷  -define F -finite_field -q 2147483647 -without_tables -end \
▷  ▷  -define v -vector -field F -allow_negatives \
▷  ▷  ▷  -dense "-1,-1" -end \
▷  ▷  -with F -do -finite_field_activity -product_of v -end
```

creates the field $\mathbb{F}_{2147483647}$. Tables are suppressed for speed. Note that

$$2^{31} - 1 = 2147483647 = p$$

is prime (see https://primes.utm.edu/lists/2small/0bit.html). The command creates two instances of -1 in the field $\mathbb{Z}_p$ and multiplies them. The fact that this is done correctly is remarkable, because the intermediate value exceeds the size of 32 bit signed integers. Recall that -1 is stored as $p-1$, so

$$-1 \equiv 2147483646.$$

But

$$-1 \cdot -1 \equiv 2147483646 \cdot 2147483646 = 4611686009837453316,$$

which exceeds the range of 32 bit signed integers. The reason why this command works is that the multiplication of integers is performed using long integer arithmetic. Long integer arithmetic treats integers as text strings, and hence is not bound by the size of a machine word. Unfortunately, long integer arithmetic is considerably slower than arithmetic in machine words. So, there is a performance drawback with large finite fields in Orbiter.

The command

Example 698

```
F_32bit:
▷  $(ORBITER) -v 10 \
▷  ▷  -define F -finite_field -q 4294967291 -without_tables -end \
```

creates the field $\mathbb{F}_{4294967291}$. Note that

$$2^{32} - 5 = 4294967291$$

is prime (see https://primes.utm.edu/lists/2small/0bit.html). The command will likely not run, and terminate with an error message. The reason is hardware dependent. The command tests the size of a machine word. Most compilers map the C++ data type *signed int* to 32 bit machine words, so this command will fail in such an environment. This is true even with 64 bit hardware. There may be some compiler options to change this.

Machines for which the previous command works correctly, the following command should also be fine:

Example 699

```
F_63bit:
▷  $(ORBITER) -v 10 \
▷  ▷  -define F -finite_field -q 9223372036854775783 -without_tables -end \
▷  ▷  -define v -vector -field F -allow_negatives \
▷  ▷  ▷  -dense "-1,-1" -end \
▷  ▷  -with F -do -finite_field_activity -product_of v -end
```

Note that

$$2^{63} - 25 = 9223372036854775783$$

is prime (see https://primes.utm.edu/lists/2small/0bit.html). However, the next command will almost surely fail:

Example 700

```
F_64bit:
▷  $(ORBITER) -v 10 \
▷  ▷  -define F -finite_field -q 18446744073709551557 -without_tables -end \
▷  ▷  -define v -vector -field F -allow_negatives \
▷  ▷  ▷  -dense "-1,-1" -end \
▷  ▷  -with F -do -finite_field_activity -product_of v -end
```

This is because we are exceeding the range that can be stored in a 64 bit unsigned integer machine word. Note that

$$2^{64} - 59 = 18446744073709551557$$

is prime (see https://primes.utm.edu/lists/2small/0bit.html).

Chapter 20

Orbiter on Windows

20.1 Using Windows Subsystem Linux

The following quote from `https://docs.microsoft.com/en-us/windows/wsl/` summarizes the function of the Windows Subsystem for Linux:

> Windows Subsystem for Linux (WSL) lets developers run a GNU/Linux environment – including most command-line tools, utilities, and applications – directly on Windows, unmodified, without the overhead of a traditional virtual machine or dual-boot setup. You can:
>
> 1. Choose your favorite GNU/Linux distributions from the Microsoft Store.
> 2. Run common command-line tools such as grep, sed, awk, or other ELF-64 binaries.
> 3. Run Bash shell scripts and GNU/Linux command-line applications including:
> 4. Tools: vim, emacs, tmux
> 5. Languages: NodeJS, Javascript, Python, Ruby, C/C++, C# & F#, Rust, Go, etc.
> 6. Services: SSHD, MySQL, Apache, lighttpd, MongoDB, PostgreSQL.
> 7. Install additional software using your own GNU/Linux distribution package manager.
> 8. Invoke Windows applications using a Unix-like command-line shell.
> 9. Invoke GNU/Linux applications on Windows.

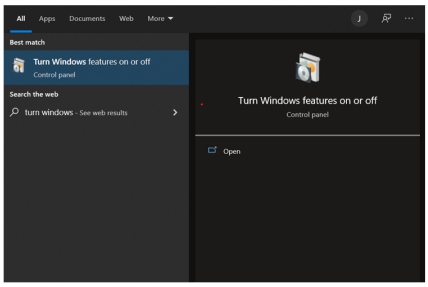
The following set of slides will ilustrate the installation of Orbiter under WSL.

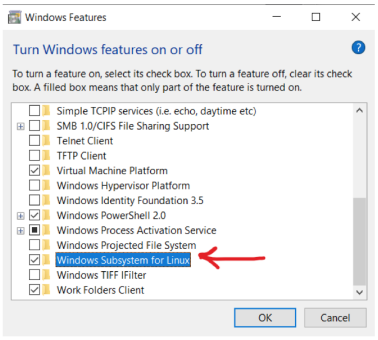
Resources

- Many of the steps will be taken from the following sources:
  - https://okunhardt.github.io/documents/Installing_WSL.pdf
  - https://docs.microsoft.com/en-us/windows/wsl/basic-commands
- Consult the two links for further help and suggestions.

Installing WSL

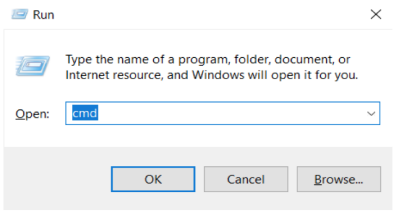
- Search “Turn Windows features on or off” in the Windows search bar
- Search for “Windows Subsystem for Linux”, the box must be checked
- Restart the computer





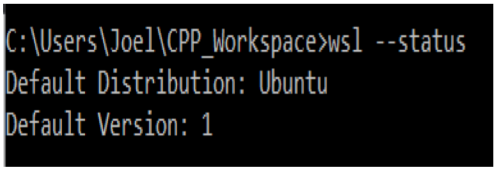
Update

- The Windows Subsystem for Linux kernel does not automatically update due to system settings
- Updates must be done manually
- To update, first you need to command prompt as admin
  - Press Windows + R to open the “Run” box
  - Type “cmd” into the box
  - Press Ctrl + Shift + Enter
  - When the window prompt opens, click “Yes”
  - Command prompt will now open as admin
  - In command prompt
    - Type wsl --update
    - Type wsl --shutdown



WSL1, WSL2

- When using WSL, you can adjust the configurations according to the Linux distribution that you are using
- To run Ubuntu distribution, we need the WSL1 configuration
- To check the status, in the command prompt enter
  - wsl --status
- To change WSL configuration type
  - wsl --set-default-version 1
  - wsl --shutdown

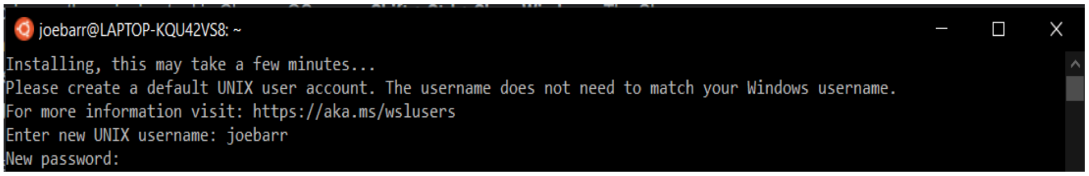


Ubuntu - installation

- Generally, the Ubuntu distribution is installed by default when WSL is installed
  - wsl --status
    - Displays the default distribution
- If you find that Ubuntu was not installed, you can find it in the Microsoft store
- Launch Ubuntu after installation

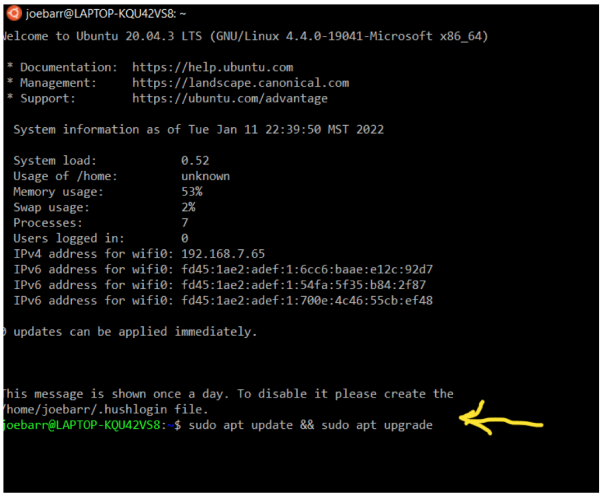
Ubuntu - launching

- After launching Ubuntu, allow the installation to be initiated
- If you receive an error, this could be a result of the configuration
  - Set configuration to WSL1
    - wsl --set-default-version 1
  - Make sure to terminate Ubuntu and reboot
    - wsl --terminate Ubuntu
  - Start Ubuntu again
- Once Ubuntu starts correctly
  - Create Username & Password to complete installation
  - Note: the password will not appear when you type it



Ubuntu - update

- Ubuntu does not update automatically, to update run the command
  - sudo apt update && sudo apt upgrade
- You will be prompted to enter your password
- When update are ready to be installed the message will appear
  - Do you want to continue? [Y/n]
    - Y + enter



Ubuntu – g++ and make

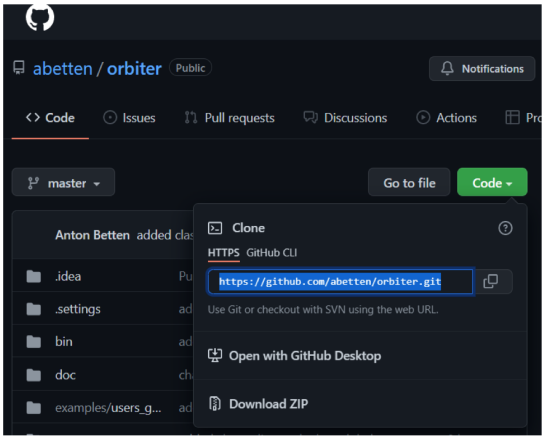
- At this point, you have successfully installed and setup WSL, and now you can use the terminal as you would on Ubuntu
- Terminate and reboot Ubuntu
- Run the command in Ubuntu
  - sudo apt install g++
  - You can now compile C++ in WSL
- Run the command in Ubuntu
  - sudo apt install make
  - You can now use makefiles in WSL

Orbiter - installation

- The easiest way to run make is through the command prompt, not Ubuntu
- To run WSL commands in command prompt, use either
  - wsl <command>
  - wsl.exe <command>
- Open command prompt
- Change directory to Users\username
  - cd C:\Users\"your username"

Orbiter - installation

- In web, go to https://github.com/abetten/orbiter
- Click on the green icon "Code" that opens a drop-down menu
- You want to copy HTTPS URL



Orbiter - installation

- In command prompt, once you are in C:\Users\Joel type the command
  - wsl.exe git clone https://github.com/abetten/orbiter.git
  - Hit enter
- Now, orbiter will begin the cloning process

```
C:\Users\Joel>wsl.exe git clone https://github.com/abetten/orbiter.git
Cloning into 'orbiter'...
remote: Enumerating objects: 87873, done.
remote: Counting objects: 100% (7029/7029), done.
remote: Compressing objects: 100% (5007/5007), done.
remote: Total 87873 (delta 5245), reused 3408 (delta 1713), pack-reused 80844
Receiving objects: 100% (87873/87873), 1.55 GiB | 18.87 MiB/s, done.
Resolving deltas: 100% (47649/47649), done.
Updating files: 100% (1154/1154), done.
```

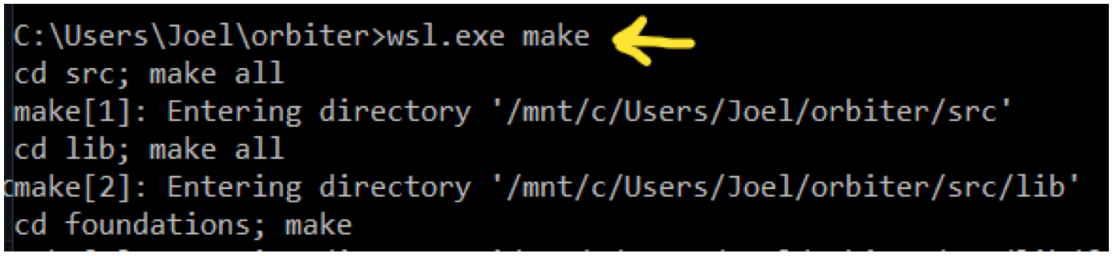
Orbiter - compile

- After cloning orbiter, run the command
  - dir
- You will find a new directory created called “orbiter”
- Change directory to “orbiter”
  - cd orbiter

```
C:\Users\Joel>dir
 Volume in drive C is Windows
 Volume Serial Number is 6C2E-AA7C

 Directory of C:\Users\Joel

01/11/2022  12:00 AM    <DIR>          .
01/11/2022  12:00 AM    <DIR>          ..
12/23/2020  02:51 PM    <DIR>          3D Objects
12/23/2020  02:51 PM    <DIR>          Contacts
12/23/2020  02:51 PM    <DIR>          Desktop
01/09/2022  06:11 PM    <DIR>          Documents
12/23/2020  02:51 PM    <DIR>          Downloads
02/09/2018  03:27 PM    <DIR>          Dropbox
12/23/2020  02:51 PM    <DIR>          Favorites
07/10/2020  10:27 AM    <DIR>          HP
08/09/2019  07:25 AM    <DIR>          Intel
12/23/2020  02:51 PM    <DIR>          Links
12/23/2020  02:51 PM    <DIR>          Music
08/04/2021  09:20 AM    <DIR>          OneDrive
01/11/2022  12:02 AM    <DIR>          orbiter
01/09/2022  09:19 PM    <DIR>          Pictures
05/27/2018  03:38 PM    <DIR>          Roaming
12/23/2020  02:51 PM    <DIR>          Saved Games
12/23/2020  02:51 PM    <DIR>          Searches
01/09/2022  04:36 PM    <DIR>          source
12/23/2020  02:51 PM    <DIR>          Videos
               0 File(s)              0 bytes
              21 Dir(s)  70,471,704,576 bytes free
```

Orbiter - compile

- Now that you are in C:\Users\"your username"\orbiter, run the command
  - wsl.exe make
- The orbiter library will now be compiled, give it some time

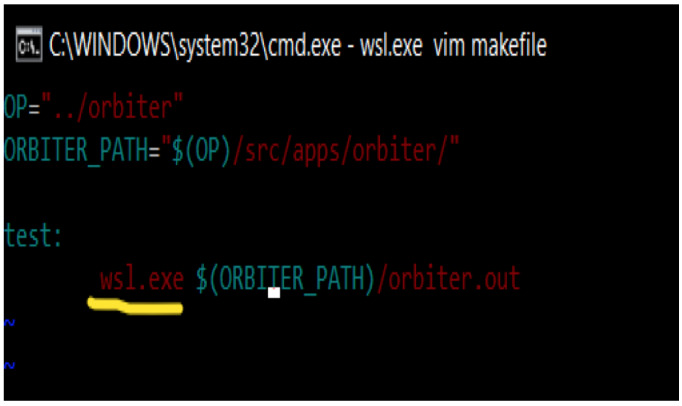


Makefile

- Now that orbiter has been successfully compiled, in the directory C:\Users\"your username"\orbiter
  - Change directory to C:\Users\"your username" and create a new directory
  - Ex: mkdir CPP_Workspace
- Change directory into CPP_Workspace
  - cd CPP_Workspace
- In C:\Users\"your username"\"new directory", run the command
  - wsl.exe vim makefile
- Vim (an IDE) will create the file "makefile"
- For Vim commands, go to https://vim.rtorr.com/
- Remember: all Ubuntu commands must begin with either
  - wsl or wsl.exe

Makefile

- To edit file in vim, click “i”
- You will see --insert-- in the lower left-hand corner
- The example to the right demonstrates a simple test to assure that orbiter is running correctly
- Assuming that orbiter directory is located in C:\Users\”your username” then the variable OP and ORBITER_PATH should work just fine
- Note were wsl.exe is inserted
- Makefile contains Ubuntu commands not windows commands



Running makefile

- Now that you have created the makefile,
  - Click “esc” to finish editing in vim
  - Run the command
    - :wq + enter
  - This saves & closes the makefile in vim
- You will be returned to
  - C:\Users\”your username”\”new directory”
- In the directory run,
  - wsl.exe make test
  - Hit “enter”
- If everything runs correctly, you will see

Orbiter - notes

- Now that everything runs correctly, visit https://www.math.colostate.edu/~betten/orbiter/users_guide.pdf
- This is the Orbiter User's guide
- Remember that you must use "wsl.exe make <target>" or "wsl make <target>" to run make correctly on linux distribution
- Also, note how "wsl.exe" is used inside of the makefile
- Ubuntu commands are used in makefile

Orbiter - update

- To update orbiter, change directories to
  - C:\Users\"your username"\orbiter
- Run the commands
  - wsl.exe make clean ; wsl.exe make
- Good luck!

Chapter 21

Templates

21.1 Templates

h	mon	vector					
0	X_0^2	$(2,0,0,0)$					
1	X_1^2	$(0,2,0,0)$					
2	X_2^2	$(0,0,2,0)$					
3	X_3^2	$(0,0,0,2)$					
4	X_0X_1	$(1,1,0,0)$					
5	X_0X_2	$(1,0,1,0)$					
6	X_0X_3	$(1,0,0,1)$					
7	X_1X_2	$(0,1,1,0)$					
8	X_1X_3	$(0,1,0,1)$					
9	X_2X_3	$(0,0,1,1)$					

h	mon	vector					
0	X_0^3	$(3,0,0,0)$					
1	X_1^3	$(0,3,0,0)$					
2	X_2^3	$(0,0,3,0)$					
3	X_3^3	$(0,0,0,3)$					
4	$X_0^2X_1$	$(2,1,0,0)$					
5	$X_0^2X_2$	$(2,0,1,0)$					
6	$X_0^2X_3$	$(2,0,0,1)$					
7	$X_0X_1^2$	$(1,2,0,0)$					
8	$X_1^2X_2$	$(0,2,1,0)$					
9	$X_1^2X_3$	$(0,2,0,1)$					
10	$X_0X_2^2$	$(1,0,2,0)$					
11	$X_1X_2^2$	$(0,1,2,0)$					
12	$X_2^2X_3$	$(0,0,2,1)$					
13	$X_0X_3^2$	$(1,0,0,2)$					
14	$X_1X_3^2$	$(0,1,0,2)$					
15	$X_2X_3^2$	$(0,0,1,2)$					
16	$X_0X_1X_2$	$(1,1,1,0)$					
17	$X_0X_1X_3$	$(1,1,0,1)$					
18	$X_0X_2X_3$	$(1,0,1,1)$					
19	$X_1X_2X_3$	$(0,1,1,1)$					

Appendix A

Data on Cubic Surfaces

A.1 Data on Cubic Surfaces

Cubic Surfaces by Field Order and Number of Eckardt Points

q	total	0	1	2	3	4	5	6	9	10	13	18	45
4	1	0	0	0	0	0	0	0	0	0	0	0	1
7	1	0	0	0	0	0	0	0	0	0	0	1	0
8	1	0	0	0	0	0	0	0	0	0	1	0	0
9	2	0	0	0	0	0	0	0	1	1	0	0	0
11	2	0	0	0	0	0	0	1	0	1	0	0	0
13	4	0	0	0	0	1	0	1	1	0	0	1	0
16	5	0	0	0	1	0	1	0	1	0	1	0	1
17	7	0	1	0	1	2	0	3	0	0	0	0	0
19	10	0	0	2	2	1	0	2	1	1	0	1	0
23	16	0	2	2	4	3	0	5	0	0	0	0	0
25	18	0	4	3	3	2	0	3	2	0	0	1	0
27	11	0	2	2	2	2	0	2	1	0	0	0	0
29	34	1	6	7	11	3	0	5	0	1	0	0	0
31	43	1	10	9	11	3	0	5	2	1	0	1	0
32	11	1	3	0	4	0	2	0	0	0	1	0	0
37	77	4	25	14	18	5	0	7	3	0	0	1	0
41	107	9	37	19	28	5	0	8	0	1	0	0	0
43	126	11	48	21	27	6	0	9	3	0	0	1	0
47	169	20	67	26	38	7	0	11	0	0	0	0	0
49	121	16	46	19	25	4	0	6	3	1	0	1	0

q	total	0	1	2	3	4	5	6	9	10	13	18	45
53	258	40	110	36	52	8	0	12	0	0	0	0	0
59	376	72	166	48	68	8	0	13	0	1	0	0	0
61	427	85	193	53	69	8	0	12	5	1	0	1	0
64	101	20	51	0	17	0	7	0	2	0	3	0	1
67	595	139	275	65	85	10	0	15	5	0	0	1	0
71	731	189	335	75	105	10	0	16	0	1	0	0	0
73	813	216	378	80	105	11	0	16	6	0	0	1	0
79	1081	321	500	97	127	11	0	17	6	1	0	1	0
81	331	100	149	30	38	4	0	6	3	1	0	0	0
83	1292	411	592	107	149	13	0	20	0	0	0	0	0
89	1673	577	759	127	176	13	0	20	0	1	0	0	0
97	2304	868	1033	154	203	15	0	22	8	0	0	1	0
101	2673	1053	1179	169	233	15	0	23	0	1	0	0	0
103	2877	1152	1268	176	232	16	0	24	8	0	0	1	0
107	3313	1379	1437	191	263	17	0	26	0	0	0	0	0
109	3557	1501	1539	201	265	16	0	24	9	1	0	1	0
113	4067	1776	1733	216	297	18	0	27	0	0	0	0	0
121	2753	1262	1135	140	179	11	0	17	7	1	0	1	0
128	929	443	405	0	60	0	18	0	0	0	3	0	0

Total: 30917

0 Eckardt Points

q	total	Ago
29	1	1
31	1	1
32	1	5
37	4	1^4
41	9	1^9
43	11	1^{11}
47	20	1^{20}
49	16	2^7, 1^9
53	40	1^{40}
59	72	1^{72}
61	85	1^{85}
64	20	2^6, 1^{14}
67	139	1^{139}
71	189	1^{189}
73	216	1^{216}
79	321	1^{321}
81	100	4^2, 2^{15}, 1^{83}
83	411	1^{411}
89	577	1^{577}
97	868	1^{868}
101	1053	1^{1053}
103	1152	1^{1152}
107	1379	1^{1379}
109	1501	1^{1501}
113	1776	1^{1776}
121	1262	2^{95}, 1^{1167}
128	443	1^{443}

Total: 11667

1 Eckardt Points

q	total	Ago
17	1	8
23	2	2^2
25	4	16, 4^3
27	2	2^2
29	6	4, 2^5
31	10	2^{10}
32	3	2^3
37	25	4, 2^{24}
41	37	8, 4, 2^{35}
43	48	2^{48}
47	67	2^{67}
49	46	16, 8, 4^8, 2^{36}
53	110	4^2, 2^{108}
59	166	2^{166}
61	193	4^2, 2^{191}
64	51	12^3, 4^7, 2^{41}
67	275	2^{275}
71	335	2^{335}
73	378	8, 4^2, 2^{375}
79	500	2^{500}
81	149	16, 8^6, 4^{15}, 2^{127}
83	592	2^{592}
89	759	8, 4^3, 2^{755}
97	1033	8, 4^3, 2^{1029}
101	1179	4^4, 2^{1175}
103	1268	2^{1268}
107	1437	2^{1437}
109	1539	4^4, 2^{1535}
113	1733	8, 4^4, 2^{1728}
121	1135	16, 8^2, 4^{77}, 2^{1055}
128	405	2^{405}

Total: 13488

2 Eckardt Points

q	total	Ago
19	2	4^2
23	2	4^2
25	3	8^2, 4
27	2	12, 4
29	7	4^7
31	9	4^9
37	14	4^{14}
41	19	4^{19}
43	21	4^{21}
47	26	4^{26}
49	19	8^7, 4^{12}
53	36	4^{36}
59	48	4^{48}
61	53	4^{53}
67	65	4^{65}
71	75	4^{75}
73	80	4^{80}
79	97	4^{97}
81	30	16^2, 8^6, 4^{22}
83	107	4^{107}
89	127	4^{127}
97	154	4^{154}
101	169	4^{169}
103	176	4^{176}
107	191	4^{191}
109	201	4^{201}
113	216	4^{216}
121	140	8^{27}, 4^{113}

Total: 2089

3 Eckardt Points

q	total	Ago
16	1	12
17	1	6
19	2	6^2
23	4	6^4
25	3	12, 6^2
27	2	6^2
29	11	6^{11}
31	11	6^{11}
32	4	6^4
37	18	6^{18}
41	28	6^{28}
43	27	6^{27}
47	38	6^{38}
49	25	12^{10}, 6^{15}
53	52	6^{52}
59	68	6^{68}
61	69	6^{69}
64	17	12^4, 6^{13}
67	85	6^{85}
71	105	6^{105}
73	105	6^{105}
79	127	6^{127}
81	38	12^7, 6^{31}
83	149	6^{149}
89	176	6^{176}
97	203	6^{203}
101	233	6^{233}
103	232	6^{232}
107	263	6^{263}
109	265	6^{265}
113	297	6^{297}
121	179	12^{24}, 6^{155}
128	60	6^{60}

Total: 2898

4 Eckardt Points

q	total	Ago
13	1	12
17	2	12^2
19	1	12
23	3	12^3
25	2	24, 12
27	2	36, 12
29	3	12^3
31	3	12^3
37	5	12^5
41	5	12^5
43	6	12^6
47	7	12^7
49	4	24^2, 12^2
53	8	12^8
59	8	12^8
61	8	12^8
67	10	12^{10}
71	10	12^{10}
73	11	12^{11}
79	11	12^{11}
81	4	24^2, 12^2
83	13	12^{13}
89	13	12^{13}
97	15	12^{15}
101	15	12^{15}
103	16	12^{16}
107	17	12^{17}
109	16	12^{16}
113	18	12^{18}
121	11	24^4, 12^7

Total: 248

5 Eckardt Points

q	total	Ago
16	1	64
32	2	80, 16
64	7	48, 32^3, 16^3
128	18	16^{18}

Total: 28

6 Eckardt Points

q	total	Ago
11	1	24
13	1	24
17	3	24^3
19	2	24^2
23	5	24^5
25	3	48^2, 24
27	2	24^2
29	5	24^5
31	5	24^5
37	7	24^7
41	8	24^8
43	9	24^9
47	11	24^{11}
49	6	48^3, 24^3
53	12	24^{12}
59	13	24^{13}
61	12	24^{12}
67	15	24^{15}
71	16	24^{16}
73	16	24^{16}
79	17	24^{17}
81	6	48^3, 24^3
83	20	24^{20}
89	20	24^{20}
97	22	24^{22}
101	23	24^{23}
103	24	24^{24}
107	26	24^{26}
109	24	24^{24}
113	27	24^{27}
121	17	48^7, 24^{10}

Total: 378

9 Eckardt Points

q	total	Ago
9	1	432
13	1	108
16	1	216
19	1	54
25	2	216, 108
27	1	162
31	2	54^2
37	3	108, 54^2
43	3	54^3
49	3	216, 108, 54
61	5	108, 54^4
64	2	162, 108
67	5	54^5
73	6	108, 54^5
79	6	54^6
81	3	864, 216, 108
97	8	108, 54^7
103	8	54^8
109	9	108, 54^8
121	7	216, 108^3, 54^3

Total: 77

10 Eckardt Points

q	total	Ago
9	1	240
11	1	120
19	1	120
29	1	120
31	1	120
41	1	120
49	1	240
59	1	120
61	1	120
71	1	120
79	1	120
81	1	480
89	1	120
101	1	120
109	1	120
121	1	240

Total: 16

13 Eckardt Points

q	total	Ago
8	1	576
16	1	384
32	1	192
64	3	1152, 384, 192
128	3	192^3

Total: 9

18 Eckardt Points

q	total	Ago
7	1	648
13	1	648
19	1	648
25	1	1296
31	1	648
37	1	648
43	1	648
49	1	1296
61	1	648
67	1	648
73	1	648
79	1	648
97	1	648
103	1	648
109	1	648
121	1	1296

Total: 16

45 Eckardt Points

q	total	Ago
4	1	51840
16	1	103680
64	1	155520

Total: 3

Even Characteristic

q	total	0	1	3	5	9	13	45
4	1	0	0	0	0	0	0	1
8	1	0	0	0	0	0	1	0
16	5	0	0	1	1	1	1	1
32	11	1	3	4	2	0	1	0
64	101	20	51	17	7	2	3	1
128	929	443	405	60	18	0	3	0

Total: 1048

0 Eckardt Points

q	total	Ago
32	1	5
64	20	2^6, 1^{14}
128	443	1^{443}

Total: 464

1 Eckardt Points

q	total	Ago
32	3	2^3
64	51	12^3, 4^7, 2^{41}
128	405	2^{405}

Total: 459

3 Eckardt Points

q	total	Ago
16	1	12
32	4	6^4
64	17	12^4, 6^{13}
128	60	6^{60}

Total: 82

5 Eckardt Points

q	total	Ago
16	1	64
32	2	80, 16
64	7	48, 32^3, 16^3
128	18	16^{18}

Total: 28

9 Eckardt Points

q	total	Ago
16	1	216
64	2	162, 108

Total: 3

13 Eckardt Points

q	total	Ago
8	1	576
16	1	384
32	1	192
64	3	1152, 384, 192
128	3	192^3

Total: 9

45 Eckardt Points

q	total	Ago
4	1	51840
16	1	103680
64	1	155520

Total: 3

Odd Characteristic

q	total	0	1	2	3	4	6	9	10	18
7	1	0	0	0	0	0	0	0	0	1
9	2	0	0	0	0	0	0	1	1	0
11	2	0	0	0	0	0	1	0	1	0
13	4	0	0	0	0	1	1	1	0	1
17	7	0	1	0	1	2	3	0	0	0
19	10	0	0	2	2	1	2	1	1	1
23	16	0	2	2	4	3	5	0	0	0
25	18	0	4	3	3	2	3	2	0	1
27	11	0	2	2	2	2	2	1	0	0
29	34	1	6	7	11	3	5	0	1	0
31	43	1	10	9	11	3	5	2	1	1
37	77	4	25	14	18	5	7	3	0	1
41	107	9	37	19	28	5	8	0	1	0
43	126	11	48	21	27	6	9	3	0	1
47	169	20	67	26	38	7	11	0	0	0
49	121	16	46	19	25	4	6	3	1	1
53	258	40	110	36	52	8	12	0	0	0
59	376	72	166	48	68	8	13	0	1	0
61	427	85	193	53	69	8	12	5	1	1
67	595	139	275	65	85	10	15	5	0	1
71	731	189	335	75	105	10	16	0	1	0
73	813	216	378	80	105	11	16	6	0	1
79	1081	321	500	97	127	11	17	6	1	1
81	331	100	149	30	38	4	6	3	1	0
83	1292	411	592	107	149	13	20	0	0	0
89	1673	577	759	127	176	13	20	0	1	0
97	2304	868	1033	154	203	15	22	8	0	1
101	2673	1053	1179	169	233	15	23	0	1	0
103	2877	1152	1268	176	232	16	24	8	0	1
107	3313	1379	1437	191	263	17	26	0	0	0
109	3557	1501	1539	201	265	16	24	9	1	1
113	4067	1776	1733	216	297	18	27	0	0	0
121	2753	1262	1135	140	179	11	17	7	1	1

Total: 29869

0 Eckardt Points

q	total	Ago
29	1	1
31	1	1
37	4	1^4
41	9	1^9
43	11	1^{11}
47	20	1^{20}
49	16	2^7, 1^9
53	40	1^{40}
59	72	1^{72}
61	85	1^{85}
67	139	1^{139}
71	189	1^{189}
73	216	1^{216}
79	321	1^{321}
81	100	4^2, 2^{15}, 1^{83}
83	411	1^{411}
89	577	1^{577}
97	868	1^{868}
101	1053	1^{1053}
103	1152	1^{1152}
107	1379	1^{1379}
109	1501	1^{1501}
113	1776	1^{1776}
121	1262	2^{95}, 1^{1167}

Total: 11203

1 Eckardt Points

q	total	Ago
17	1	8
23	2	2^2
25	4	16, 4^3
27	2	2^2
29	6	4, 2^5
31	10	2^{10}
37	25	4, 2^{24}
41	37	8, 4, 2^{35}
43	48	2^{48}
47	67	2^{67}
49	46	16, 8, 4^8, 2^{36}
53	110	4^2, 2^{108}
59	166	2^{166}
61	193	4^2, 2^{191}
67	275	2^{275}
71	335	2^{335}
73	378	8, 4^2, 2^{375}
79	500	2^{500}
81	149	16, 8^6, 4^{15}, 2^{127}
83	592	2^{592}
89	759	8, 4^3, 2^{755}
97	1033	8, 4^3, 2^{1029}
101	1179	4^4, 2^{1175}
103	1268	2^{1268}
107	1437	2^{1437}
109	1539	4^4, 2^{1535}
113	1733	8, 4^4, 2^{1728}
121	1135	16, 8^2, 4^{77}, 2^{1055}

Total: 13029

2 Eckardt Points

q	total	Ago
19	2	4^2
23	2	4^2
25	3	8^2, 4
27	2	12, 4
29	7	4^7
31	9	4^9
37	14	4^{14}
41	19	4^{19}
43	21	4^{21}
47	26	4^{26}
49	19	8^7, 4^{12}
53	36	4^{36}
59	48	4^{48}
61	53	4^{53}
67	65	4^{65}
71	75	4^{75}
73	80	4^{80}
79	97	4^{97}
81	30	16^2, 8^6, 4^{22}
83	107	4^{107}
89	127	4^{127}
97	154	4^{154}
101	169	4^{169}
103	176	4^{176}
107	191	4^{191}
109	201	4^{201}
113	216	4^{216}
121	140	8^{27}, 4^{113}

Total: 2089

3 Eckardt Points

q	total	Ago
17	1	6
19	2	6^2
23	4	6^4
25	3	12, 6^2
27	2	6^2
29	11	6^{11}
31	11	6^{11}
37	18	6^{18}
41	28	6^{28}
43	27	6^{27}
47	38	6^{38}
49	25	12^{10}, 6^{15}
53	52	6^{52}
59	68	6^{68}
61	69	6^{69}
67	85	6^{85}
71	105	6^{105}
73	105	6^{105}
79	127	6^{127}
81	38	12^7, 6^{31}
83	149	6^{149}
89	176	6^{176}
97	203	6^{203}
101	233	6^{233}
103	232	6^{232}
107	263	6^{263}
109	265	6^{265}
113	297	6^{297}
121	179	12^{24}, 6^{155}

Total: 2816

4 Eckardt Points

q	total	Ago
13	1	12
17	2	12^2
19	1	12
23	3	12^3
25	2	24, 12
27	2	36, 12
29	3	12^3
31	3	12^3
37	5	12^5
41	5	12^5
43	6	12^6
47	7	12^7
49	4	24^2, 12^2
53	8	12^8
59	8	12^8
61	8	12^8
67	10	12^{10}
71	10	12^{10}
73	11	12^{11}
79	11	12^{11}
81	4	24^2, 12^2
83	13	12^{13}
89	13	12^{13}
97	15	12^{15}
101	15	12^{15}
103	16	12^{16}
107	17	12^{17}
109	16	12^{16}
113	18	12^{18}
121	11	24^4, 12^7

Total: 248

6 Eckardt Points

q	total	Ago
11	1	24
13	1	24
17	3	24^3
19	2	24^2
23	5	24^5
25	3	48^2, 24
27	2	24^2
29	5	24^5
31	5	24^5
37	7	24^7
41	8	24^8
43	9	24^9
47	11	24^{11}
49	6	48^3, 24^3
53	12	24^{12}
59	13	24^{13}
61	12	24^{12}
67	15	24^{15}
71	16	24^{16}
73	16	24^{16}
79	17	24^{17}
81	6	48^3, 24^3
83	20	24^{20}
89	20	24^{20}
97	22	24^{22}
101	23	24^{23}
103	24	24^{24}
107	26	24^{26}
109	24	24^{24}
113	27	24^{27}
121	17	48^7, 24^{10}

Total: 378

9 Eckardt Points

q	total	Ago
9	1	432
13	1	108
19	1	54
25	2	216, 108
27	1	162
31	2	54^2
37	3	108, 54^2
43	3	54^3
49	3	216, 108, 54
61	5	108, 54^4
67	5	54^5
73	6	108, 54^5
79	6	54^6
81	3	864, 216, 108
97	8	108, 54^7
103	8	54^8
109	9	108, 54^8
121	7	216, 108^3, 54^3

Total: 74

10 Eckardt Points

q	total	Ago
9	1	240
11	1	120
19	1	120
29	1	120
31	1	120
41	1	120
49	1	240
59	1	120
61	1	120
71	1	120
79	1	120
81	1	480
89	1	120
101	1	120
109	1	120
121	1	240

Total: 16

18 Eckardt Points

q	total	Ago
7	1	648
13	1	648
19	1	648
25	1	1296
31	1	648
37	1	648
43	1	648
49	1	1296
61	1	648
67	1	648
73	1	648
79	1	648
97	1	648
103	1	648
109	1	648
121	1	1296

Total: 16

Appendix B

The Class Library

B.1 Classes and Namespaces

0 : orbiter, layer1_foundations, algebra, **a_domain**
1 : orbiter, layer1_foundations, algebra, **algebra_global**
2 : orbiter, layer1_foundations, algebra, **generators_symplectic_group**
3 : orbiter, layer1_foundations, algebra, **group_generators_domain**
4 : orbiter, layer1_foundations, algebra, **heisenberg**
5 : orbiter, layer1_foundations, algebra, **matrix_group**
6 : orbiter, layer1_foundations, algebra, **matrix_group_element**
7 : orbiter, layer1_foundations, algebra, **module**
8 : orbiter, layer1_foundations, algebra, **null_polarity_generator**
9 : orbiter, layer1_foundations, algebra, **rank_checker**
10 : orbiter, layer1_foundations, algebraic_geometry, **algebraic_geometry_global**
11 : orbiter, layer1_foundations, algebraic_geometry, **arc_lifting_with_two_lines**
12 : orbiter, layer1_foundations, algebraic_geometry, **clebsch_map**
13 : orbiter, layer1_foundations, algebraic_geometry, **cubic_curve**
14 : orbiter, layer1_foundations, algebraic_geometry, **del_pezzo_surface_of_degree_two_domain**
15 : orbiter, layer1_foundations, algebraic_geometry, **del_pezzo_surface_of_degree_two_object**
16 : orbiter, layer1_foundations, algebraic_geometry, **eckardt_point**
17 : orbiter, layer1_foundations, algebraic_geometry, **eckardt_point_info**
18 : orbiter, layer1_foundations, algebraic_geometry, **kovalevski_points**
19 : orbiter, layer1_foundations, algebraic_geometry, **quartic_curve_domain**
20 : orbiter, layer1_foundations, algebraic_geometry, **quartic_curve_object**
21 : orbiter, layer1_foundations, algebraic_geometry, **quartic_curve_object_properties**
22 : orbiter, layer1_foundations, algebraic_geometry, **schlaefli**
23 : orbiter, layer1_foundations, algebraic_geometry, **schlaefli_double_six**
24 : orbiter, layer1_foundations, algebraic_geometry, **schlaefli_labels**
25 : orbiter, layer1_foundations, algebraic_geometry, **schlaefli_trihedral_pairs**
26 : orbiter, layer1_foundations, algebraic_geometry, **schlaefli_tritangent_planes**
27 : orbiter, layer1_foundations, algebraic_geometry, **seventytwo_cases**
28 : orbiter, layer1_foundations, algebraic_geometry, **smooth_surface_object_properties**
29 : orbiter, layer1_foundations, algebraic_geometry, **surface_domain**
30 : orbiter, layer1_foundations, algebraic_geometry, **surface_object**
31 : orbiter, layer1_foundations, algebraic_geometry, **surface_object_properties**
32 : orbiter, layer1_foundations, algebraic_geometry, **surface_polynomial_domains**
33 : orbiter, layer1_foundations, algebraic_geometry, **variety_object**
34 : orbiter, layer1_foundations, algebraic_geometry, **web_of_cubic_curves**
35 : orbiter, layer1_foundations, canonical_form_classification, **classification_of_objects**
36 : orbiter, layer1_foundations, canonical_form_classification, **classification_of_objects_description**
Table 11.5
37 : orbiter, layer1_foundations, canonical_form_classification, **classification_of_objects_report_options**
Table 11.6

38 : orbiter, layer1_foundations, canonical_form_classification, **classify_bitvectors**
39 : orbiter, layer1_foundations, canonical_form_classification, **classify_using_canonical_forms**
40 : orbiter, layer1_foundations, canonical_form_classification, **data_input_stream**
41 : orbiter, layer1_foundations, canonical_form_classification, **data_input_stream_description**
Table 11.3
42 : orbiter, layer1_foundations, canonical_form_classification, **data_input_stream_description_element**
43 : orbiter, layer1_foundations, canonical_form_classification, **encoded_combinatorial_object**
44 : orbiter, layer1_foundations, canonical_form_classification, **object_with_canonical_form**
45 : orbiter, layer1_foundations, coding_theory, **code_diagram**
46 : orbiter, layer1_foundations, coding_theory, **coding_theory_domain**
47 : orbiter, layer1_foundations, coding_theory, **crc_code_description Table 10.9**
48 : orbiter, layer1_foundations, coding_theory, **crc_codes**
49 : orbiter, layer1_foundations, coding_theory, **crc_object**
50 : orbiter, layer1_foundations, coding_theory, **crc_options_description Table 10.8**
51 : orbiter, layer1_foundations, coding_theory, **create_BCH_code**
52 : orbiter, layer1_foundations, coding_theory, **create_RS_code**
53 : orbiter, layer1_foundations, coding_theory, **cyclic_codes**
54 : orbiter, layer1_foundations, coding_theory, **error_pattern_generator**
55 : orbiter, layer1_foundations, coding_theory, **error_repository**
56 : orbiter, layer1_foundations, coding_theory, **ttp_codes**
57 : orbiter, layer1_foundations, combinatorics, **apn_functions**
58 : orbiter, layer1_foundations, combinatorics, **boolean_function_domain**
59 : orbiter, layer1_foundations, combinatorics, **brick_domain**
60 : orbiter, layer1_foundations, combinatorics, **combinatorics_domain**
61 : orbiter, layer1_foundations, combinatorics, **decomposition**
62 : orbiter, layer1_foundations, combinatorics, **decomposition_scheme**
63 : orbiter, layer1_foundations, combinatorics, **domino_assignment**
64 : orbiter, layer1_foundations, combinatorics, **domino_change**
65 : orbiter, layer1_foundations, combinatorics, **geo_parameter**
66 : orbiter, layer1_foundations, combinatorics, **pentomino_puzzle**
67 : orbiter, layer1_foundations, combinatorics, **polynomial_function_domain**
68 : orbiter, layer1_foundations, combinatorics, **row_and_col_partition**
69 : orbiter, layer1_foundations, combinatorics, **tdo_data**
70 : orbiter, layer1_foundations, combinatorics, **tdo_refinement**
71 : orbiter, layer1_foundations, combinatorics, **tdo_refinement_description Table 11.8**
72 : orbiter, layer1_foundations, combinatorics, **tdo_scheme_compute**
73 : orbiter, layer1_foundations, combinatorics, **tdo_scheme_synthetic**
74 : orbiter, layer1_foundations, cryptography, **cryptography_domain**
75 : orbiter, layer1_foundations, data_structures, **algorithms**
76 : orbiter, layer1_foundations, data_structures, **ancestry_family**
77 : orbiter, layer1_foundations, data_structures, **ancestry_indi**
78 : orbiter, layer1_foundations, data_structures, **ancestry_tree**
79 : orbiter, layer1_foundations, data_structures, **bitmatrix**
80 : orbiter, layer1_foundations, data_structures, **bitvector**
81 : orbiter, layer1_foundations, data_structures, **data_file**
82 : orbiter, layer1_foundations, data_structures, **data_structures_global**
83 : orbiter, layer1_foundations, data_structures, **fancy_set**
84 : orbiter, layer1_foundations, data_structures, **int_matrix**
85 : orbiter, layer1_foundations, data_structures, **int_vec**
86 : orbiter, layer1_foundations, data_structures, **int_vector**
87 : orbiter, layer1_foundations, data_structures, **lint_vec**
88 : orbiter, layer1_foundations, data_structures, **page_storage**
89 : orbiter, layer1_foundations, data_structures, **partitionstack**
90 : orbiter, layer1_foundations, data_structures, **set_builder**
91 : orbiter, layer1_foundations, data_structures, **set_builder_description Table 2.7**
92 : orbiter, layer1_foundations, data_structures, **set_of_sets**
93 : orbiter, layer1_foundations, data_structures, **set_of_sets_lint**
94 : orbiter, layer1_foundations, data_structures, **sorting**
95 : orbiter, layer1_foundations, data_structures, **spreadsheet**

96 : orbiter, layer1_foundations, data_structures, **string_tools**
97 : orbiter, layer1_foundations, data_structures, **tally**
98 : orbiter, layer1_foundations, data_structures, **tally_lint**
99 : orbiter, layer1_foundations, data_structures, **tally_vector_data**
100 : orbiter, layer1_foundations, data_structures, **vector_builder**
101 : orbiter, layer1_foundations, data_structures, **vector_builder_description Table 2.8**
102 : orbiter, layer1_foundations, data_structures, **vector_hashing**
103 : orbiter, layer1_foundations, expression_parser, **formula**
104 : orbiter, layer1_foundations, expression_parser, **formula_vector**
105 : orbiter, layer1_foundations, expression_parser, **symbolic_object_activity**
106 : orbiter, layer1_foundations, expression_parser, **symbolic_object_activity_description Table 2.12**
107 : orbiter, layer1_foundations, expression_parser, **symbolic_object_builder**
108 : orbiter, layer1_foundations, expression_parser, **symbolic_object_builder_description Table 2.10, Table 2.11**
109 : orbiter, layer1_foundations, expression_parser, **syntax_tree**
110 : orbiter, layer1_foundations, expression_parser, **syntax_tree_latex**
111 : orbiter, layer1_foundations, expression_parser, **syntax_tree_node**
112 : orbiter, layer1_foundations, expression_parser, **syntax_tree_node_terminal**
113 : orbiter, layer1_foundations, field_theory, **finite_field**
114 : orbiter, layer1_foundations, field_theory, **finite_field_activity**
115 : orbiter, layer1_foundations, field_theory, **finite_field_activity_description Table 3.5, Table 3.6, Table 3.8, Table 5.1, Table 3.9, Table 3.10**
116 : orbiter, layer1_foundations, field_theory, **finite_field_description Table 3.4**
117 : orbiter, layer1_foundations, field_theory, **finite_field_implementation_by_tables**
118 : orbiter, layer1_foundations, field_theory, **finite_field_implementation_wo_tables**
119 : orbiter, layer1_foundations, field_theory, **finite_field_io**
120 : orbiter, layer1_foundations, field_theory, **minimum_polynomial**
121 : orbiter, layer1_foundations, field_theory, **norm_tables**
122 : orbiter, layer1_foundations, field_theory, **normal_basis**
123 : orbiter, layer1_foundations, field_theory, **nth_roots**
124 : orbiter, layer1_foundations, field_theory, **related_fields**
125 : orbiter, layer1_foundations, field_theory, **square_nonsquare**
126 : orbiter, layer1_foundations, field_theory, **subfield_structure**
127 : orbiter, layer1_foundations, geometry, **W3q**
128 : orbiter, layer1_foundations, geometry, **andre_construction**
129 : orbiter, layer1_foundations, geometry, **andre_construction_line_element**
130 : orbiter, layer1_foundations, geometry, **andre_construction_point_element**
131 : orbiter, layer1_foundations, geometry, **arc_basic**
132 : orbiter, layer1_foundations, geometry, **arc_in_projective_space**
133 : orbiter, layer1_foundations, geometry, **buekenhout_metz**
134 : orbiter, layer1_foundations, geometry, **desarguesian_spread**
135 : orbiter, layer1_foundations, geometry, **flag**
136 : orbiter, layer1_foundations, geometry, **geometric_object_create**
137 : orbiter, layer1_foundations, geometry, **geometric_object_description Table 4.12, Table 4.13**
138 : orbiter, layer1_foundations, geometry, **geometry_global**
139 : orbiter, layer1_foundations, geometry, **grassmann**
140 : orbiter, layer1_foundations, geometry, **grassmann_embedded**
141 : orbiter, layer1_foundations, geometry, **hermitian**
142 : orbiter, layer1_foundations, geometry, **hjelmslev**
143 : orbiter, layer1_foundations, geometry, **incidence_structure**
144 : orbiter, layer1_foundations, geometry, **intersection_type**
145 : orbiter, layer1_foundations, geometry, **klein_correspondence**
146 : orbiter, layer1_foundations, geometry, **knarr**
147 : orbiter, layer1_foundations, geometry, **point_line**
148 : orbiter, layer1_foundations, geometry, **points_and_lines**
149 : orbiter, layer1_foundations, geometry, **polarity**
150 : orbiter, layer1_foundations, geometry, **projective_space**
151 : orbiter, layer1_foundations, geometry, **projective_space_basic**
152 : orbiter, layer1_foundations, geometry, **projective_space_implementation**

153 : orbiter, layer1_foundations, geometry, **projective_space_of_dimension_three**
154 : orbiter, layer1_foundations, geometry, **projective_space_plane**
155 : orbiter, layer1_foundations, geometry, **projective_space_reporting**
156 : orbiter, layer1_foundations, geometry, **projective_space_subspaces**
157 : orbiter, layer1_foundations, geometry, **spread_domain**
158 : orbiter, layer1_foundations, geometry, **spread_tables**
159 : orbiter, layer1_foundations, geometry, **three_skew_subspaces**
160 : orbiter, layer1_foundations, geometry_builder, **cperm**
161 : orbiter, layer1_foundations, geometry_builder, **decomposition_with_fuse**
162 : orbiter, layer1_foundations, geometry_builder, **gen_geo**
163 : orbiter, layer1_foundations, geometry_builder, **gen_geo_conf**
164 : orbiter, layer1_foundations, geometry_builder, **geometric_backtrack_search**
165 : orbiter, layer1_foundations, geometry_builder, **geometry_builder**
166 : orbiter, layer1_foundations, geometry_builder, **geometry_builder_description Table 11.7**
167 : orbiter, layer1_foundations, geometry_builder, **girth_test**
168 : orbiter, layer1_foundations, geometry_builder, **inc_encoding**
169 : orbiter, layer1_foundations, geometry_builder, **incidence**
170 : orbiter, layer1_foundations, geometry_builder, **iso_type**
171 : orbiter, layer1_foundations, geometry_builder, **test_semicanonical**
172 : orbiter, layer1_foundations, graph_theory, **clique_finder**
173 : orbiter, layer1_foundations, graph_theory, **clique_finder_control Table 15.1**
174 : orbiter, layer1_foundations, graph_theory, **colored_graph**
175 : orbiter, layer1_foundations, graph_theory, **graph_layer**
176 : orbiter, layer1_foundations, graph_theory, **graph_node**
177 : orbiter, layer1_foundations, graph_theory, **graph_theory_domain**
178 : orbiter, layer1_foundations, graph_theory, **layered_graph**
179 : orbiter, layer1_foundations, graph_theory, **rainbow_cliques**
180 : orbiter, layer1_foundations, graphics, **animate**
181 : orbiter, layer1_foundations, graphics, **draw_bitmap_control Table 17.1**
182 : orbiter, layer1_foundations, graphics, **draw_incidence_structure_description**
183 : orbiter, layer1_foundations, graphics, **draw_mod_n_description**
184 : orbiter, layer1_foundations, graphics, **draw_projective_curve_description**
185 : orbiter, layer1_foundations, graphics, **drawable_set_of_objects**
186 : orbiter, layer1_foundations, graphics, **graphical_output**
187 : orbiter, layer1_foundations, graphics, **layered_graph_draw_options Table 17.2, Table 17.3**
188 : orbiter, layer1_foundations, graphics, **mp_graphics**
189 : orbiter, layer1_foundations, graphics, **parametric_curve**
190 : orbiter, layer1_foundations, graphics, **parametric_curve_point**
191 : orbiter, layer1_foundations, graphics, **plot_tools**
192 : orbiter, layer1_foundations, graphics, **povray_job_description Table 17.5**
193 : orbiter, layer1_foundations, graphics, **scene Table 17.10, Table 17.8, Table 17.9**
194 : orbiter, layer1_foundations, graphics, **scene_element_of_type_edge**
195 : orbiter, layer1_foundations, graphics, **scene_element_of_type_face**
196 : orbiter, layer1_foundations, graphics, **scene_element_of_type_line**
197 : orbiter, layer1_foundations, graphics, **scene_element_of_type_plane**
198 : orbiter, layer1_foundations, graphics, **scene_element_of_type_point**
199 : orbiter, layer1_foundations, graphics, **scene_element_of_type_surface**
200 : orbiter, layer1_foundations, graphics, **tree**
201 : orbiter, layer1_foundations, graphics, **tree_draw_options Table 19.3**
202 : orbiter, layer1_foundations, graphics, **tree_node**
203 : orbiter, layer1_foundations, graphics, **video_draw_options Table 17.6, Table 17.7**
204 : orbiter, layer1_foundations, knowledge_base, **knowledge_base**
205 : orbiter, layer1_foundations, l1_interfaces, **easy_BMP_interface**
206 : orbiter, layer1_foundations, l1_interfaces, **expression_parser_sajeeb**
207 : orbiter, layer1_foundations, l1_interfaces, **expression_parser_sajeeb_private_data**
208 : orbiter, layer1_foundations, l1_interfaces, **interface_gap_low**
209 : orbiter, layer1_foundations, l1_interfaces, **interface_magma_low**
210 : orbiter, layer1_foundations, l1_interfaces, **latex_interface**
211 : orbiter, layer1_foundations, l1_interfaces, **nauty_interface**

212 : orbiter, layer1_foundations, l1_interfaces, **nauty_output**
213 : orbiter, layer1_foundations, l1_interfaces, **povray_interface**
214 : orbiter, layer1_foundations, linear_algebra, **gl_class_rep**
215 : orbiter, layer1_foundations, linear_algebra, **gl_classes**
216 : orbiter, layer1_foundations, linear_algebra, **linear_algebra**
217 : orbiter, layer1_foundations, linear_algebra, **linear_algebra_global**
218 : orbiter, layer1_foundations, linear_algebra, **matrix_block_data**
219 : orbiter, layer1_foundations, linear_algebra, **representation_theory_domain**
220 : orbiter, layer1_foundations, linear_algebra, **vector_space**
221 : orbiter, layer1_foundations, number_theory, **cyclotomic_sets**
222 : orbiter, layer1_foundations, number_theory, **elliptic_curve**
223 : orbiter, layer1_foundations, number_theory, **number_theoretic_transform**
224 : orbiter, layer1_foundations, number_theory, **number_theory_domain**
225 : orbiter, layer1_foundations, orbiter_kernel_system, **create_file_description**
226 : orbiter, layer1_foundations, orbiter_kernel_system, **csv_file_support**
227 : orbiter, layer1_foundations, orbiter_kernel_system, **file_io**
228 : orbiter, layer1_foundations, orbiter_kernel_system, **file_output**
229 : orbiter, layer1_foundations, orbiter_kernel_system, **mem_object_registry**
230 : orbiter, layer1_foundations, orbiter_kernel_system, **mem_object_registry_entry**
231 : orbiter, layer1_foundations, orbiter_kernel_system, **memory_object**
232 : orbiter, layer1_foundations, orbiter_kernel_system, **numerics**
233 : orbiter, layer1_foundations, orbiter_kernel_system, **orbiter_data_file**
234 : orbiter, layer1_foundations, orbiter_kernel_system, **orbiter_session Table 2.1**
235 : orbiter, layer1_foundations, orbiter_kernel_system, **orbiter_symbol_table**
236 : orbiter, layer1_foundations, orbiter_kernel_system, **orbiter_symbol_table_entry**
237 : orbiter, layer1_foundations, orbiter_kernel_system, **os_interface**
238 : orbiter, layer1_foundations, orbiter_kernel_system, **override_double**
239 : orbiter, layer1_foundations, orbiter_kernel_system, **prepare_frames Table 17.11**
240 : orbiter, layer1_foundations, orthogonal_geometry, **blt_set_domain**
241 : orbiter, layer1_foundations, orthogonal_geometry, **blt_set_invariants**
242 : orbiter, layer1_foundations, orthogonal_geometry, **hyperbolic_pair**
243 : orbiter, layer1_foundations, orthogonal_geometry, **linear_complex**
244 : orbiter, layer1_foundations, orthogonal_geometry, **orthogonal**
245 : orbiter, layer1_foundations, orthogonal_geometry, **orthogonal_global**
246 : orbiter, layer1_foundations, orthogonal_geometry, **orthogonal_group**
247 : orbiter, layer1_foundations, orthogonal_geometry, **orthogonal_indexing**
248 : orbiter, layer1_foundations, orthogonal_geometry, **orthogonal_plane_invariant**
249 : orbiter, layer1_foundations, orthogonal_geometry, **quadratic_form**
250 : orbiter, layer1_foundations, orthogonal_geometry, **quadratic_form_list_coding**
251 : orbiter, layer1_foundations, orthogonal_geometry, **unusual_model**
252 : orbiter, layer1_foundations, polish, **function_command**
253 : orbiter, layer1_foundations, polish, **function_polish**
254 : orbiter, layer1_foundations, polish, **function_polish_description**
255 : orbiter, layer1_foundations, ring_theory, **finite_ring**
256 : orbiter, layer1_foundations, ring_theory, **homogeneous_polynomial_domain**
257 : orbiter, layer1_foundations, ring_theory, **longinteger_domain**
258 : orbiter, layer1_foundations, ring_theory, **longinteger_object**
259 : orbiter, layer1_foundations, ring_theory, **partial_derivative**
260 : orbiter, layer1_foundations, ring_theory, **polynomial_double**
261 : orbiter, layer1_foundations, ring_theory, **polynomial_double_domain**
262 : orbiter, layer1_foundations, ring_theory, **polynomial_ring_activity_description Table 5.3**
263 : orbiter, layer1_foundations, ring_theory, **polynomial_ring_description Table 5.2**
264 : orbiter, layer1_foundations, ring_theory, **ring_theory_global**
265 : orbiter, layer1_foundations, ring_theory, **table_of_irreducible_polynomials**
266 : orbiter, layer1_foundations, ring_theory, **unipoly_domain**
267 : orbiter, layer1_foundations, solvers, **diophant**
268 : orbiter, layer1_foundations, solvers, **diophant_activity**
269 : orbiter, layer1_foundations, solvers, **diophant_activity_description Table 15.3**
270 : orbiter, layer1_foundations, solvers, **diophant_create**

271 : orbiter, layer1_foundations, solvers, **diophant_description Table 15.2**
272 : orbiter, layer1_foundations, solvers, **dlx_problem_description**
273 : orbiter, layer1_foundations, solvers, **dlx_solver**
274 : orbiter, layer1_foundations, solvers, mckay, **tMCKAY**
275 : orbiter, layer2_discreta, typed_objects, **Vector**
276 : orbiter, layer2_discreta, typed_objects, **bt_key**
277 : orbiter, layer2_discreta, typed_objects, **btree**
278 : orbiter, layer2_discreta, typed_objects, **database**
279 : orbiter, layer2_discreta, typed_objects, **design_parameter**
280 : orbiter, layer2_discreta, typed_objects, **design_parameter_source**
281 : orbiter, layer2_discreta, typed_objects, **discreta_base**
282 : orbiter, layer2_discreta, typed_objects, **discreta_matrix**
283 : orbiter, layer2_discreta, typed_objects, **domain**
284 : orbiter, layer2_discreta, typed_objects, **hollerith**
285 : orbiter, layer2_discreta, typed_objects, **integer**
286 : orbiter, layer2_discreta, typed_objects, **longinteger**
287 : orbiter, layer2_discreta, typed_objects, **matrix_access**
288 : orbiter, layer2_discreta, typed_objects, **memory**
289 : orbiter, layer2_discreta, typed_objects, **number_partition**
290 : orbiter, layer2_discreta, typed_objects, **page_table**
291 : orbiter, layer2_discreta, typed_objects, **permutation**
292 : orbiter, layer2_discreta, typed_objects, **printing_mode**
293 : orbiter, layer2_discreta, typed_objects, **unipoly**
294 : orbiter, layer2_discreta, typed_objects, **with**
295 : orbiter, layer3_group_actions, actions, **action**
296 : orbiter, layer3_group_actions, actions, **action_global**
297 : orbiter, layer3_group_actions, actions, **action_pointer_table**
298 : orbiter, layer3_group_actions, actions, **group_element**
299 : orbiter, layer3_group_actions, actions, **induced_action**
300 : orbiter, layer3_group_actions, actions, **known_groups**
301 : orbiter, layer3_group_actions, actions, **stabilizer_chain_base_data**
302 : orbiter, layer3_group_actions, combinatorics_with_groups, **combinatorics_with_action**
303 : orbiter, layer3_group_actions, combinatorics_with_groups, **flag_orbits_incidence_structure**
304 : orbiter, layer3_group_actions, combinatorics_with_groups, **group_action_on_combinatorial_object**
305 : orbiter, layer3_group_actions, combinatorics_with_groups, **incidence_structure_with_group**
306 : orbiter, layer3_group_actions, combinatorics_with_groups, **orbit_type_repository**
307 : orbiter, layer3_group_actions, combinatorics_with_groups, **translation_plane_via_andre_model**
308 : orbiter, layer3_group_actions, data_structures_groups, **group_container**
309 : orbiter, layer3_group_actions, data_structures_groups, **orbit_rep**
310 : orbiter, layer3_group_actions, data_structures_groups, **orbit_transversal**
311 : orbiter, layer3_group_actions, data_structures_groups, **schreier_vector**
312 : orbiter, layer3_group_actions, data_structures_groups, **schreier_vector_handler**
313 : orbiter, layer3_group_actions, data_structures_groups, **set_and_stabilizer**
314 : orbiter, layer3_group_actions, data_structures_groups, **union_find**
315 : orbiter, layer3_group_actions, data_structures_groups, **union_find_on_k_subsets**
316 : orbiter, layer3_group_actions, data_structures_groups, **vector_ge**
317 : orbiter, layer3_group_actions, data_structures_groups, **vector_ge_description Table 2.9**
318 : orbiter, layer3_group_actions, group_constructions, **direct_product**
319 : orbiter, layer3_group_actions, group_constructions, **linear_group**
320 : orbiter, layer3_group_actions, group_constructions, **linear_group_description Table 6.2, Table 6.3, Table 6.7, Table 6.5**
321 : orbiter, layer3_group_actions, group_constructions, **permutation_group_create**
322 : orbiter, layer3_group_actions, group_constructions, **permutation_group_description Table 6.1**
323 : orbiter, layer3_group_actions, group_constructions, **permutation_representation**
324 : orbiter, layer3_group_actions, group_constructions, **permutation_representation_domain**
325 : orbiter, layer3_group_actions, group_constructions, **polarity_extension**
326 : orbiter, layer3_group_actions, group_constructions, **wreath_product**
327 : orbiter, layer3_group_actions, groups, **conjugacy_class_of_elements**
328 : orbiter, layer3_group_actions, groups, **exceptional_isomorphism_O4**

329 : orbiter, layer3_group_actions, groups, **orbits_on_something**
330 : orbiter, layer3_group_actions, groups, **schreier**
331 : orbiter, layer3_group_actions, groups, **schreier_sims**
332 : orbiter, layer3_group_actions, groups, **sims**
333 : orbiter, layer3_group_actions, groups, **strong_generators**
334 : orbiter, layer3_group_actions, groups, **subgroup**
335 : orbiter, layer3_group_actions, groups, **sylow_structure**
336 : orbiter, layer3_group_actions, induced_actions, **action_by_conjugation**
337 : orbiter, layer3_group_actions, induced_actions, **action_by_representation**
338 : orbiter, layer3_group_actions, induced_actions, **action_by_restriction**
339 : orbiter, layer3_group_actions, induced_actions, **action_by_right_multiplication**
340 : orbiter, layer3_group_actions, induced_actions, **action_by_subfield_structure**
341 : orbiter, layer3_group_actions, induced_actions, **action_on_andre**
342 : orbiter, layer3_group_actions, induced_actions, **action_on_bricks**
343 : orbiter, layer3_group_actions, induced_actions, **action_on_cosets**
344 : orbiter, layer3_group_actions, induced_actions, **action_on_determinant**
345 : orbiter, layer3_group_actions, induced_actions, **action_on_factor_space**
346 : orbiter, layer3_group_actions, induced_actions, **action_on_flags**
347 : orbiter, layer3_group_actions, induced_actions, **action_on_galois_group**
348 : orbiter, layer3_group_actions, induced_actions, **action_on_grassmannian**
349 : orbiter, layer3_group_actions, induced_actions, **action_on_homogeneous_polynomials**
350 : orbiter, layer3_group_actions, induced_actions, **action_on_interior_direct_product**
351 : orbiter, layer3_group_actions, induced_actions, **action_on_k_subsets**
352 : orbiter, layer3_group_actions, induced_actions, **action_on_module**
353 : orbiter, layer3_group_actions, induced_actions, **action_on_orbits**
354 : orbiter, layer3_group_actions, induced_actions, **action_on_orthogonal**
355 : orbiter, layer3_group_actions, induced_actions, **action_on_set_partitions**
356 : orbiter, layer3_group_actions, induced_actions, **action_on_sets**
357 : orbiter, layer3_group_actions, induced_actions, **action_on_sign**
358 : orbiter, layer3_group_actions, induced_actions, **action_on_spread_set**
359 : orbiter, layer3_group_actions, induced_actions, **action_on_subgroups**
360 : orbiter, layer3_group_actions, induced_actions, **action_on_wedge_product**
361 : orbiter, layer3_group_actions, induced_actions, **product_action**
362 : orbiter, layer3_group_actions, interfaces, **conjugacy_classes_and_normalizers**
363 : orbiter, layer3_group_actions, interfaces, **l3_interface_gap**
364 : orbiter, layer3_group_actions, interfaces, **magma_interface**
365 : orbiter, layer3_group_actions, interfaces, **nauty_interface_with_group**
366 : orbiter, layer4_classification, invariant_relations, **classification_step**
367 : orbiter, layer4_classification, invariant_relations, **flag_orbit_node**
368 : orbiter, layer4_classification, invariant_relations, **flag_orbits**
369 : orbiter, layer4_classification, invariant_relations, **orbit_node**
370 : orbiter, layer4_classification, isomorph, **flag_orbit_folding**
371 : orbiter, layer4_classification, isomorph, **isomorph**
372 : orbiter, layer4_classification, isomorph, **isomorph_arguments Table 14.7**
373 : orbiter, layer4_classification, isomorph, **isomorph_global**
374 : orbiter, layer4_classification, isomorph, **isomorph_worker**
375 : orbiter, layer4_classification, isomorph, **representatives**
376 : orbiter, layer4_classification, isomorph, **substructure_classification**
377 : orbiter, layer4_classification, isomorph, **substructure_lifting_data**
378 : orbiter, layer4_classification, orbits_schreier, **orbit_of_equations**
379 : orbiter, layer4_classification, orbits_schreier, **orbit_of_sets**
380 : orbiter, layer4_classification, orbits_schreier, **orbit_of_subspaces**
381 : orbiter, layer4_classification, poset_classification, **classification_base_case**
382 : orbiter, layer4_classification, poset_classification, **extension**
383 : orbiter, layer4_classification, poset_classification, **orbit_based_testing**
384 : orbiter, layer4_classification, poset_classification, **orbit_tracer**
385 : orbiter, layer4_classification, poset_classification, **poset_classification**
386 : orbiter, layer4_classification, poset_classification, **poset_classification_activity**
387 : orbiter, layer4_classification, poset_classification, **poset_classification_activity_description**

Table 7.5
388 : orbiter, layer4_classification, poset_classification, **poset_classification_control Table 7.4**
389 : orbiter, layer4_classification, poset_classification, **poset_classification_report_options Table 7.6**
390 : orbiter, layer4_classification, poset_classification, **poset_of_orbits**
391 : orbiter, layer4_classification, poset_classification, **poset_orbit_node**
392 : orbiter, layer4_classification, poset_classification, **poset_with_group_action**
393 : orbiter, layer4_classification, poset_classification, **upstep_work**
394 : orbiter, layer4_classification, set_stabilizer, **compute_stabilizer**
395 : orbiter, layer4_classification, set_stabilizer, **stabilizer_orbits_and_types**
396 : orbiter, layer4_classification, set_stabilizer, **substructure_classifier**
397 : orbiter, layer4_classification, set_stabilizer, **substructure_stats_and_selection**
398 : orbiter, layer4_classification, solvers_package, **exact_cover**
399 : orbiter, layer4_classification, solvers_package, **exact_cover_arguments**
400 : orbiter, layer5_applications, applications_in_algebraic_geometry, cubic_surfaces_and_arcs, **arc_lifting**
401 : orbiter, layer5_applications, applications_in_algebraic_geometry, cubic_surfaces_and_arcs, **arc_orbits_on_pairs**
402 : orbiter, layer5_applications, applications_in_algebraic_geometry, cubic_surfaces_and_arcs, **arc_partition**
403 : orbiter, layer5_applications, applications_in_algebraic_geometry, cubic_surfaces_and_arcs, **classify_trihedral_pairs**
404 : orbiter, layer5_applications, applications_in_algebraic_geometry, cubic_surfaces_and_arcs, **six_arcs_not_on_a_conic**
405 : orbiter, layer5_applications, applications_in_algebraic_geometry, cubic_surfaces_and_arcs, **surface_classify_using_arc**
406 : orbiter, layer5_applications, applications_in_algebraic_geometry, cubic_surfaces_and_arcs, **surface_create_by_arc_lifting**
407 : orbiter, layer5_applications, applications_in_algebraic_geometry, cubic_surfaces_and_arcs, **surfaces_arc_lifting**
408 : orbiter, layer5_applications, applications_in_algebraic_geometry, cubic_surfaces_and_arcs, **surfaces_arc_lifting_definition_node**
409 : orbiter, layer5_applications, applications_in_algebraic_geometry, cubic_surfaces_and_arcs, **surfaces_arc_lifting_trace**
410 : orbiter, layer5_applications, applications_in_algebraic_geometry, cubic_surfaces_and_arcs, **surfaces_arc_lifting_upstep**
411 : orbiter, layer5_applications, applications_in_algebraic_geometry, cubic_surfaces_and_arcs, **trihedral_pair_with_action**
412 : orbiter, layer5_applications, applications_in_algebraic_geometry, cubic_surfaces_and_double_sixes, **classification_of_cubic_surfaces_with_double_sixes_activity**
413 : orbiter, layer5_applications, applications_in_algebraic_geometry, cubic_surfaces_and_double_sixes, **classification_of_cubic_surfaces_with_double_sixes_activity_description Table 8.6**
414 : orbiter, layer5_applications, applications_in_algebraic_geometry, cubic_surfaces_and_double_sixes, **classify_double_sixes**
415 : orbiter, layer5_applications, applications_in_algebraic_geometry, cubic_surfaces_and_double_sixes, **classify_five_plus_one**
416 : orbiter, layer5_applications, applications_in_algebraic_geometry, cubic_surfaces_and_double_sixes, **identify_cubic_surface**
417 : orbiter, layer5_applications, applications_in_algebraic_geometry, cubic_surfaces_and_double_sixes, **surface_classify_wedge**
418 : orbiter, layer5_applications, applications_in_algebraic_geometry, cubic_surfaces_and_double_sixes, **surface_repository**
419 : orbiter, layer5_applications, applications_in_algebraic_geometry, cubic_surfaces_in_general, **cubic_surface_activity**
420 : orbiter, layer5_applications, applications_in_algebraic_geometry, cubic_surfaces_in_general, **cubic_surface_activity_description Table 8.4**
421 : orbiter, layer5_applications, applications_in_algebraic_geometry, cubic_surfaces_in_general, **surface_clebsch_map**
422 : orbiter, layer5_applications, applications_in_algebraic_geometry, cubic_surfaces_in_general, **surface_create**

423 : orbiter, layer5_applications, applications_in_algebraic_geometry, cubic_surfaces_in_general, **surface_create_description Table 8.1, Table 8.2, Table 8.3**
424 : orbiter, layer5_applications, applications_in_algebraic_geometry, cubic_surfaces_in_general, **surface_domain_high_level**
425 : orbiter, layer5_applications, applications_in_algebraic_geometry, cubic_surfaces_in_general, **surface_object_with_group**
426 : orbiter, layer5_applications, applications_in_algebraic_geometry, cubic_surfaces_in_general, **surface_study**
427 : orbiter, layer5_applications, applications_in_algebraic_geometry, cubic_surfaces_in_general, **surface_with_action**
428 : orbiter, layer5_applications, applications_in_algebraic_geometry, cubic_surfaces_in_general, **table_of_surfaces**
429 : orbiter, layer5_applications, applications_in_algebraic_geometry, quartic_curves, **quartic_curve_activity**
430 : orbiter, layer5_applications, applications_in_algebraic_geometry, quartic_curves, **quartic_curve_activity_description Table 8.9**
431 : orbiter, layer5_applications, applications_in_algebraic_geometry, quartic_curves, **quartic_curve_create**
432 : orbiter, layer5_applications, applications_in_algebraic_geometry, quartic_curves, **quartic_curve_create_description Table 8.8**
433 : orbiter, layer5_applications, applications_in_algebraic_geometry, quartic_curves, **quartic_curve_domain_with_action**
434 : orbiter, layer5_applications, applications_in_algebraic_geometry, quartic_curves, **quartic_curve_from_surface**
435 : orbiter, layer5_applications, applications_in_algebraic_geometry, quartic_curves, **quartic_curve_object_with_group**
436 : orbiter, layer5_applications, apps_algebra, **action_on_forms**
437 : orbiter, layer5_applications, apps_algebra, **action_on_forms_activity**
438 : orbiter, layer5_applications, apps_algebra, **action_on_forms_activity_description**
439 : orbiter, layer5_applications, apps_algebra, **action_on_forms_description**
440 : orbiter, layer5_applications, apps_algebra, **algebra_global_with_action**
441 : orbiter, layer5_applications, apps_algebra, **any_group**
442 : orbiter, layer5_applications, apps_algebra, **character_table_burnside**
443 : orbiter, layer5_applications, apps_algebra, **element_processing_description Table 6.11**
444 : orbiter, layer5_applications, apps_algebra, **group_modification_description Table 6.6**
445 : orbiter, layer5_applications, apps_algebra, **group_theoretic_activity**
446 : orbiter, layer5_applications, apps_algebra, **group_theoretic_activity_description Table 6.8, Table 6.9, Table 6.10, Table 6.13, Table 6.12, Table 9.1**
447 : orbiter, layer5_applications, apps_algebra, **modified_group_create**
448 : orbiter, layer5_applications, apps_algebra, **polynomial_ring_activity**
449 : orbiter, layer5_applications, apps_algebra, **vector_ge_builder**
450 : orbiter, layer5_applications, apps_algebra, **young**
451 : orbiter, layer5_applications, apps_coding_theory, **code_modification_description Table 10.6**
452 : orbiter, layer5_applications, apps_coding_theory, **coding_theoretic_activity**
453 : orbiter, layer5_applications, apps_coding_theory, **coding_theoretic_activity_description Table 10.3, Table 10.4, Table 10.5**
454 : orbiter, layer5_applications, apps_coding_theory, **crc_process**
455 : orbiter, layer5_applications, apps_coding_theory, **crc_process_description Table 10.7**
456 : orbiter, layer5_applications, apps_coding_theory, **create_code**
457 : orbiter, layer5_applications, apps_coding_theory, **create_code_description Table 10.2**
458 : orbiter, layer5_applications, apps_combinatorics, **boolean_function_classify**
459 : orbiter, layer5_applications, apps_combinatorics, **combinatorial_object**
460 : orbiter, layer5_applications, apps_combinatorics, **combinatorial_object_activity**
461 : orbiter, layer5_applications, apps_combinatorics, **combinatorial_object_activity_description Table 11.4**
462 : orbiter, layer5_applications, apps_combinatorics, **combinatorics_global**
463 : orbiter, layer5_applications, apps_combinatorics, **delandtsheer_doyen**
464 : orbiter, layer5_applications, apps_combinatorics, **delandtsheer_doyen_description Table 13.6**
465 : orbiter, layer5_applications, apps_combinatorics, **design_activity**

466 : orbiter, layer5_applications, apps_combinatorics, **design_activity_description Table 13.2**
467 : orbiter, layer5_applications, apps_combinatorics, **design_create**
468 : orbiter, layer5_applications, apps_combinatorics, **design_create_description Table 13.1**
469 : orbiter, layer5_applications, apps_combinatorics, **design_tables**
470 : orbiter, layer5_applications, apps_combinatorics, **difference_set_in_heisenberg_group**
471 : orbiter, layer5_applications, apps_combinatorics, **hadamard_classify**
472 : orbiter, layer5_applications, apps_combinatorics, **hall_system_classify**
473 : orbiter, layer5_applications, apps_combinatorics, **large_set_activity**
474 : orbiter, layer5_applications, apps_combinatorics, **large_set_activity_description**
475 : orbiter, layer5_applications, apps_combinatorics, **large_set_classify**
476 : orbiter, layer5_applications, apps_combinatorics, **large_set_was**
477 : orbiter, layer5_applications, apps_combinatorics, **large_set_was_activity**
478 : orbiter, layer5_applications, apps_combinatorics, **large_set_was_activity_description Table 13.5**
479 : orbiter, layer5_applications, apps_combinatorics, **large_set_was_description**
480 : orbiter, layer5_applications, apps_combinatorics, **regular_linear_space_description**
481 : orbiter, layer5_applications, apps_combinatorics, **regular_ls_classify**
482 : orbiter, layer5_applications, apps_combinatorics, **tactical_decomposition**
483 : orbiter, layer5_applications, apps_geometry, **arc_generator**
484 : orbiter, layer5_applications, apps_geometry, **arc_generator_description Table 7.7**
485 : orbiter, layer5_applications, apps_geometry, **arc_lifting_simeon**
486 : orbiter, layer5_applications, apps_geometry, **choose_points_or_lines**
487 : orbiter, layer5_applications, apps_geometry, **classify_cubic_curves**
488 : orbiter, layer5_applications, apps_geometry, **cubic_curve_with_action**
489 : orbiter, layer5_applications, apps_geometry, **hermitian_spreads_classify**
490 : orbiter, layer5_applications, apps_geometry, **linear_set_classify**
491 : orbiter, layer5_applications, apps_geometry, **mapping**
492 : orbiter, layer5_applications, apps_geometry, **mapping_description Table 5.6**
493 : orbiter, layer5_applications, apps_geometry, **ovoid_classify**
494 : orbiter, layer5_applications, apps_geometry, **ovoid_classify_description**
495 : orbiter, layer5_applications, apps_geometry, **polar**
496 : orbiter, layer5_applications, apps_geometry, **search_blocking_set**
497 : orbiter, layer5_applications, apps_geometry, **singer_cycle**
498 : orbiter, layer5_applications, apps_geometry, **tensor_classify**
499 : orbiter, layer5_applications, apps_geometry, **top_level_geometry_global**
500 : orbiter, layer5_applications, apps_graph_theory, **cayley_graph_search**
501 : orbiter, layer5_applications, apps_graph_theory, **create_graph**
502 : orbiter, layer5_applications, apps_graph_theory, **create_graph_description Table 12.1, Table 12.2**
503 : orbiter, layer5_applications, apps_graph_theory, **graph_classification_activity**
504 : orbiter, layer5_applications, apps_graph_theory, **graph_classification_activity_description Table 12.6**
505 : orbiter, layer5_applications, apps_graph_theory, **graph_classify**
506 : orbiter, layer5_applications, apps_graph_theory, **graph_classify_description Table 12.5**
507 : orbiter, layer5_applications, apps_graph_theory, **graph_modification_description Table 12.3**
508 : orbiter, layer5_applications, apps_graph_theory, **graph_theoretic_activity**
509 : orbiter, layer5_applications, apps_graph_theory, **graph_theoretic_activity_description Table 12.4**
510 : orbiter, layer5_applications, apps_graph_theory, **graph_theory_apps**
511 : orbiter, layer5_applications, canonical_form, **canonical_form_classifier**
512 : orbiter, layer5_applications, canonical_form, **canonical_form_classifier_description Table 16.1**
513 : orbiter, layer5_applications, canonical_form, **canonical_form_nauty**
514 : orbiter, layer5_applications, canonical_form, **canonical_form_of_variety**
515 : orbiter, layer5_applications, canonical_form, **canonical_form_substructure**
516 : orbiter, layer5_applications, canonical_form, **classification_of_combinatorial_objects**
517 : orbiter, layer5_applications, canonical_form, **classification_of_varieties**
518 : orbiter, layer5_applications, canonical_form, **input_objects_of_type_variety**
519 : orbiter, layer5_applications, canonical_form, **object_in_projective_space_with_action**
520 : orbiter, layer5_applications, canonical_form, **object_with_properties**
521 : orbiter, layer5_applications, canonical_form, **quartic_curve_object_with_action**
522 : orbiter, layer5_applications, canonical_form, **variety_object_with_action**
523 : orbiter, layer5_applications, orbits, **orbit_cascade**

524 : orbiter, layer5_applications, orbits, **orbits_activity**
525 : orbiter, layer5_applications, orbits, **orbits_activity_description Table 7.2**
526 : orbiter, layer5_applications, orbits, **orbits_create**
527 : orbiter, layer5_applications, orbits, **orbits_create_description Table 7.1**
528 : orbiter, layer5_applications, orbits, **orbits_global**
529 : orbiter, layer5_applications, orbits, **orbits_on_polynomials**
530 : orbiter, layer5_applications, orbits, **orbits_on_subspaces**
531 : orbiter, layer5_applications, orthogonal_geometry_applications, **BLT_set_create**
532 : orbiter, layer5_applications, orthogonal_geometry_applications, **BLT_set_create_description Table 14.14**
533 : orbiter, layer5_applications, orthogonal_geometry_applications, **blt_set_activity**
534 : orbiter, layer5_applications, orthogonal_geometry_applications, **blt_set_activity_description Table 14.16**
535 : orbiter, layer5_applications, orthogonal_geometry_applications, **blt_set_classify**
536 : orbiter, layer5_applications, orthogonal_geometry_applications, **blt_set_classify_activity**
537 : orbiter, layer5_applications, orthogonal_geometry_applications, **blt_set_classify_activity_description Table 14.18**
538 : orbiter, layer5_applications, orthogonal_geometry_applications, **blt_set_classify_description Table 14.17**
539 : orbiter, layer5_applications, orthogonal_geometry_applications, **blt_set_domain_with_action**
540 : orbiter, layer5_applications, orthogonal_geometry_applications, **blt_set_group_properties**
541 : orbiter, layer5_applications, orthogonal_geometry_applications, **blt_set_with_action**
542 : orbiter, layer5_applications, orthogonal_geometry_applications, **flock_from_blt_set**
543 : orbiter, layer5_applications, orthogonal_geometry_applications, **orthogonal_space_activity**
544 : orbiter, layer5_applications, orthogonal_geometry_applications, **orthogonal_space_activity_description Table 4.6**
545 : orbiter, layer5_applications, orthogonal_geometry_applications, **orthogonal_space_with_action**
546 : orbiter, layer5_applications, orthogonal_geometry_applications, **orthogonal_space_with_action_description Table 4.5**
547 : orbiter, layer5_applications, orthogonal_geometry_applications, **table_of_blt_sets**
548 : orbiter, layer5_applications, packings, **invariants_packing**
549 : orbiter, layer5_applications, packings, **packing_classify**
550 : orbiter, layer5_applications, packings, **packing_invariants**
551 : orbiter, layer5_applications, packings, **packing_long_orbits**
552 : orbiter, layer5_applications, packings, **packing_long_orbits_description Table 14.13**
553 : orbiter, layer5_applications, packings, **packing_was**
554 : orbiter, layer5_applications, packings, **packing_was_activity**
555 : orbiter, layer5_applications, packings, **packing_was_activity_description Table 14.11**
556 : orbiter, layer5_applications, packings, **packing_was_description Table 14.10**
557 : orbiter, layer5_applications, packings, **packing_was_fixpoints**
558 : orbiter, layer5_applications, packings, **packing_was_fixpoints_activity**
559 : orbiter, layer5_applications, packings, **packing_was_fixpoints_activity_description Table 14.12**
560 : orbiter, layer5_applications, packings, **packings_global**
561 : orbiter, layer5_applications, packings, **regular_packing**
562 : orbiter, layer5_applications, projective_geometry, **projective_space_activity**
563 : orbiter, layer5_applications, projective_geometry, **projective_space_activity_description Table 4.7, Table 4.8, Table 4.9, Table 4.10**
564 : orbiter, layer5_applications, projective_geometry, **projective_space_global**
565 : orbiter, layer5_applications, projective_geometry, **projective_space_with_action**
566 : orbiter, layer5_applications, projective_geometry, **projective_space_with_action_description Table 4.1**
567 : orbiter, layer5_applications, projective_geometry, **summary_of_properties_of_objects**
568 : orbiter, layer5_applications, semifields, **semifield_classify**
569 : orbiter, layer5_applications, semifields, **semifield_classify_description**
570 : orbiter, layer5_applications, semifields, **semifield_classify_with_substructure**
571 : orbiter, layer5_applications, semifields, **semifield_downstep_node**
572 : orbiter, layer5_applications, semifields, **semifield_flag_orbit_node**
573 : orbiter, layer5_applications, semifields, **semifield_level_two**
574 : orbiter, layer5_applications, semifields, **semifield_lifting**

575 : orbiter, layer5_applications, semifields, **semifield_substructure**
576 : orbiter, layer5_applications, semifields, **semifield_trace**
577 : orbiter, layer5_applications, semifields, **trace_record**
578 : orbiter, layer5_applications, spreads, **recoordinatize**
579 : orbiter, layer5_applications, spreads, **spread_activity**
580 : orbiter, layer5_applications, spreads, **spread_activity_description Table 14.2**
581 : orbiter, layer5_applications, spreads, **spread_classify**
582 : orbiter, layer5_applications, spreads, **spread_classify_activity**
583 : orbiter, layer5_applications, spreads, **spread_classify_activity_description Table 14.4**
584 : orbiter, layer5_applications, spreads, **spread_classify_description Table 14.3**
585 : orbiter, layer5_applications, spreads, **spread_create**
586 : orbiter, layer5_applications, spreads, **spread_create_description Table 14.1**
587 : orbiter, layer5_applications, spreads, **spread_lifting**
588 : orbiter, layer5_applications, spreads, **spread_table_activity**
589 : orbiter, layer5_applications, spreads, **spread_table_activity_description Table 14.5**
590 : orbiter, layer5_applications, spreads, **spread_table_with_selection**
591 : orbiter, layer5_applications, spreads, **translation_plane_activity**
592 : orbiter, layer5_applications, spreads, **translation_plane_activity_description Table 14.9**
593 : orbiter, layer5_applications, user_interface, **activity_description Table ??**
594 : orbiter, layer5_applications, user_interface, **interface_algebra Table 3.1, Table 3.2**
595 : orbiter, layer5_applications, user_interface, **interface_coding_theory Table 10.1**
596 : orbiter, layer5_applications, user_interface, **interface_combinatorics Table 11.1, Table 11.2**
597 : orbiter, layer5_applications, user_interface, **interface_cryptography Table 9.2, Table 9.3, Table 3.3**
598 : orbiter, layer5_applications, user_interface, **interface_povray Table 17.4**
599 : orbiter, layer5_applications, user_interface, **interface_projective**
600 : orbiter, layer5_applications, user_interface, **interface_symbol_table**
601 : orbiter, layer5_applications, user_interface, **interface_toolkit Table 19.1, Table 19.2**
602 : orbiter, layer5_applications, user_interface, **orbiter_command**
603 : orbiter, layer5_applications, user_interface, **orbiter_top_level_session**
604 : orbiter, layer5_applications, user_interface, **symbol_definition**

Appendix C

The Makefile

C.1 The Makefile

```
1
2
3     ORBITER_PATH=~/orbiter
4
5     #ORBITER_PATH=/s/hipft-logger/b/nobackup/abetten/orbiter
6     #ORBITER_PATH=/scratch/betten/COMPILE/orbiter
7     #ORBITER_PATH=/data/accounts/fac/betten/orbiter
8
9
10    # uncomment exactly one of the following two lines.
11    # uncomment the first if you want to run orbiter through docker.
12    # uncomment the second if you have an installed copy of orbiter and you want to r
      un it directly
13    #ORBITER_PATH=docker run -it --volume ${PWD}:/mnt -w /mnt abetten/orbiter
14
15    ORBITER_EXE_PATH=$(ORBITER_PATH)/src/apps/orbiter/
16
17    ORBITER=$(ORBITER_EXE_PATH)orbiter.out
18
19    SANDBOX=$(MY_PATH)/src/apps/sandbox/sandbox.out
20
21    ###############################################################################
22    # additional configurations for when you want to
23    # compile automatically generated code
24    ###############################################################################
25
26    SRC=$(ORBITER_PATH)/src
27    MY_CPP = g++
28    MY_CC = gcc
29    CPPFLAGS = -Wall -I../../DEV.23/orbiter/src/lib -std=c++14
30    LIB = $(SRC)/lib/liborbiter.a -lpthread
31    LFLAGS = -lm -Wl,-rpath -Wl,/usr/local/gcc-8.2.0/lib64
32
33
34    #MAGMA_PATH=/usr/local/magma
35    MAGMA_PATH=""
36
37
38    #OPEN=open
39    OPEN=echo
```

```
40
41
42
43  ###############################################################################
44  # End of configuration part
45  ###############################################################################
46
47  GINAC_PATH=$(ORBITER_PATH)/src/apps/ginac
48  SANDBOX_PATH=$(ORBITER_PATH)/src/apps/sandbox
49
50  update:
51  ▷  cd $(ORBITER_PATH); make clean;
52  ▷  cd $(MY_PATH); make cleana; git pull; make
53
54
55  update_all:
56  ▷  cd $(MY_PATH); make clean; git pull; make
57
58
59  sandbox:
60  ▷  $(SANDBOX_PATH)/sandbox.out
61
62
63  ###############################################################################
64  # testing:
65  ###############################################################################
66
67  test:
68  ▷  make test_2
69  ▷  make test_3
70  ▷  make test_4
71  ▷  make test_5
72  ▷  make test_6
73  ▷  make test_7
74  ▷  make test_8
75  ▷  make test_9
76  ▷  make test_10
77  ▷  make test_11
78  ▷  make test_12
79  ▷  make test_13
80  ▷  make test_14
81  ▷  make test_15
82  ▷  make test_16
83  ▷  make test_17
84  ▷  make test_18
85  ▷  make test_19
86
87
88
89  test_problem:
90
91
92
93  ###############################################################################
94  # Makefile Variables
95
96
97  # Section 2.7 vectors
98
```

```
99
100
101   # Section 2.8 vector of group elements
102
103
104   HIRSCHFELD_STAB_GENERATORS="\
105   1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1, \
106   1,0,0,0,0,2,0,0,0,0,2,0,0,0,0,1,0, \
107   1,0,0,0,0,3,0,0,0,0,3,0,1,0,0,1,0, \
108   1,0,0,0,0,1,0,0,1,1,1,0,0,1,0,1,1, \
109   1,0,0,0,0,0,1,0,0,1,0,0,0,0,0,1,0, \
110   0,1,0,0,1,0,0,0,0,0,0,1,0,0,1,0,0"
111
112
113
114
115   CLASS_2A_REP="1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1"
116
117   FILE_NAME_CLASS_2A="PGGL_4_4_Subgroup_Hirschfeld_Stab_51840_class_of_2A.csv"
118
119
120
121   # Section 2.9 Symbolic Objects
122
123   TEST_FORMULA="(-a+b*b)*x*x+a*b*x"
124
125   BCH_POLYNOMIAL_ALFA_F256="X^2 + 167*X + 214"
126
127   BCH_POLYNOMIAL_BRAVO_F256="X^4+213*X^3+27*X^2+23*X+1"
128
129   VANDERMONDE_4X4_FORMULA="1,x0,x0^2,x0^3,1,x1,x1^2,x1^3,\
130   1,x2,x2^2,x2^3,1,x3,x3^2,x3^3"
131
132   ECKARDT_SURFACE_EQN="a*X3^3-a*b^2*(X0^2+X1^2+X2^2)*X3+b^3*(a^2+1)*X0*X1*X2"
133
134   # general normal form for cubic surfaces with 27 rational lines:
135
136   F_abcd_eqn="-(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*(b - d)*X0^2*X2 \
137   + (a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*(a + b - c - d)*X0*X1*X2 \
138   + (a^2*c - a^2*d - a*c^2 + b*c^2 + a*d - b*c)*(b - d)*X0*X1*X3 \
139   - (a*d - b*c)*(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*X0*X2^2 \
140   - (a^2*c*d - a*b*c^2 - a^2*d + a*b*d + b*c^2 - b*c*d)*(b - d)*X0*X2*X3 \
141   - (a - c)*(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*X1^2*X2 \
142   - (a - c)*(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*X1^2*X3 \
143   + (a*d - b*c)*(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*X1*X2^2 \
144   + ((1+1)*a^2*b*c*d - a^2*b*d^2 - (1+1)*a^2*c*d^2 \
145   - (1+1)*a*b^2*c^2 + a*b^2*c*d + (1+1)*a*b*c^2*d + a*b*c*d^2 \
146   - b^2*c^2*d - a^2*b*c + a^2*c*d + a^2*d^2 + a*b^2*c + a*b*c^2 \
147   - (1+1+1+1)*a*b*c*d - a*c^2*d + a*c*d^2 + b^2*c^2)*X1*X2*X3 \
148   + c*a*(a*d - b*c - a + b + c - d)*(b - d)*X1*X3^2"
149
150
151   # Chapter 3 Basic algebra
152
153
154
155   # Section 3.4 Linear algebra over finite fields
156
157
```

```
158
159
160   V7_VANDERMONDE_EXTENDED="\
161   1,0,0,0,0,0,0,1,0,0,0,0,0,0, \
162   1,1,1,1,1,1,1,0,1,0,0,0,0,0, \
163   1,2,4,1,2,4,1,0,0,1,0,0,0,0, \
164   1,3,2,6,4,5,1,0,0,0,1,0,0,0, \
165   1,4,2,1,4,2,1,0,0,0,0,1,0,0, \
166   1,5,4,6,2,3,1,0,0,0,0,0,1,0, \
167   1,6,1,6,1,6,1,0,0,0,0,0,0,1"
168
169
170
171   CODE_RS_6_4_7="\
172   621000 \
173   062100 \
174   006210 \
175   000621"
176
177
178
179   # Chapter 4 Geometry
180
181
182   # Section 4.5 Algebraic sets
183
184
185   SURFACE_F9_1_EQN="4*X0*X1^2+4*X1^2*X2+8*X0*X2^2+\
186   3*X1*X2^2+3*X0*X1*X2+X0^2*X3+6*X0*X3^2+\
187   5*X0*X1*X3+X0*X2*X3+5*X1*X2*X3"
188
189
190   EC_11_EQUATION="1,0,3,0,0,0,10,1,0,0"
191
192
193
194   HIRSCHFELD_SURFACE_EQUATION="0,0,0,0,0,0,1,0,1,0, 0,1,0,1,0,0,0,0,0,0"
195
196   HIRSCHFELD_SURFACE_EQUATION_ALGEBRAIC_FORM="X0^2*X3 + X1^2*X2 + X1*X2^2 + X0*X3^2
      "
197
198   #X_0^2*X_3 + X_1^2*X_2 + X_1*X_2^2 + X_0*X_3^2
199   #X^2*W+X*W^2+Y^2*Z+Y*Z^2
200
201
202
203   # Section 4.9 Advanced topics
204
205   H1_LABEL="H1"
206
207   H1_GENS=-PGGL 4 4 \
208   ▷  -subgroup_by_generators "H1" 4 2 \
209   ▷  "1,0,0,0,1,1,0,0,0,0,1,0,0,0,1,1,0, \
210   ▷  1,0,0,0,0,1,0,0,1,0,1,0,0,1,0,1,0" \
211   ▷  -end
212
213
214
215
```

```
# Section 4.10 Geometric objects


DOILY="Row,C0,C1,C2\
\n0,0,12,13\
\n1,1,12,14\
\n2,8,9,12\
\n3,4,6,8\
\n4,6,10,14\
\n5,3,7,8\
\n6,7,10,13\
\n7,4,11,13\
\n8,3,11,14\
\n9,0,5,6\
\n10,1,5,7\
\n11,5,9,11\
\n12,0,2,3\
\n13,1,2,4\
\n14,2,9,10\
\nEND"


# q=17:
# 3 is p.e. mod 17.
# so we pick f=3.
# then, 2f^2=18=1
# 4f = 12

# X^4 -Y^4 -Z^4 +2f^2Y^2Z^2 +4fX^2YZ

#(1,-1,-1,0,0,0,0,0,0,0,0,0,2f^2,4f,0,0)

EDGE_CURVE_Q17_EQUATION="1,16,16,0,0,0,0,0,0,0,0,1,12,0,0"

EDGE_CURVE_Q17_AS_POINTS="4, 7, 16, 19, 20, 23, 32, 35, 89, 100, 244, 251"


FILE_Q17="orbit,curve,pts_on_curve,bitangents,go\
\n0,\"$(EDGE_CURVE_Q17_EQUATION)\",\"$(EDGE_CURVE_Q17_AS_POINTS)\",\"\",-1\
\nEND"


EDGE_CURVE_Q23_AS_POINTS="4, 25, 26, 47, 48, 71, 92, 95, 114, 119, \
136, 143, 158, 167, 180, 191, 202, 215, 224, 239, 246, 263, 268, \
287, 290, 311, 312, 334, 335, 356, 359, 378, 383, 400, 407, 422, \
431, 444, 455, 466, 479, 488, 503, 510, 527, 530, 532, 551"







# Chapter 5 Ring Theory

# Section 5.1 polynomials over finite fields



CRC32_SPARSE="1,32,1,26,1,23,1,22,1,16,1,12,1,11,\
1,10,1,8,1,7,1,5,1,4,1,2,1,1,1,0"
```

```
275
276
277   TWO_TO_THE_32_MINUS_2=4294967294
278
279   INVERSE_SPARSE="1,31,1,25,1,22,1,21,1,15,\
280   1,11,1,10,1,9,1,7,1,6,1,4,1,3,1,1,1,0"
281
282
283
284   # Section 5.2 multivariate polynomial rings
285
286
287
288   CREMONA_MAP_Y0="3*y0^5*y2+4*y0^3*y1^2*y2\
289   +2*y0^3*y2^3+y0*y1^4*y2\
290   +6*y0*y1^2*y2^3+9*y0*y2^5"
291
292   CREMONA_MAP_Y1="y0^5*y1+5*y0^3*y1^3\
293   +12*y0^3*y1*y2^2+3*y0*y1^5\
294   +5*y0*y1^3*y2^2+y0*y1*y2^4"
295
296   CREMONA_MAP_Y2="10*y0^6+11*y0^4*y1^2\
297   +11*y0^4*y2^2+4*y0^2*y1^4\
298   +9*y0^2*y1^2*y2^2+4*y0^2*y2^4"
299
300   CREMONA_MAP_Y3="0"
301
302
303
304
305
306   PTS_OF_SURFACE_ORBIT211_Q3_L9_E4="\
307   0,1,2,5,7,8,10,14,9,12, \
308   15,3,16,37,31,34,20,19,17,32,36,33"
309
310   SURFACE_F_9="x0*x0*x1 - x0*x1*x1 -x0*x1*x3 -x2*x2*x3 - x2*x3*x3"
311
312
313   ENDRASS_SPARSE="\
314   6,0,4,4,2,7,5,9,6,20,6,23,1,25,3,30,1,32,3,34,4,56,6,59,1,61,6,66, \
315   2,68,6,70,3,77,2,79,6,83,6,120,2,123,5,125,3,130,1,132,3,134,3,141, \
316   2,143,6,147,3,156"
317
318
319
320
321   # Section 5.3 Algebraic geometry
322
323   FILE_HERMITIAN_CURVE="Line,equation\n0,\"X^4+Y^4+Z^4\"\nEND\n\n"
324
325   FILE_HERMITIAN_SURFACE="Line,equation\n0,\"X^4+Y^4+Z^4+W^4\"\nEND\n\n"
326
327
328   # Chapter 6 Group Theory
329
330
331
332   # Section 6.1 generic permutation groups
333
```

```
334
335
336   GENERATORS_M12="3,1,9,0,10,11,6,7,8,2,4,5, 7,2,3,1,11,9,5,8,0,6,4,10"
337
338   # Section 6.2 Matrix groups
339
340
341   # Section 6.3 Subgroups
342
343   GEN_C13="1,2,3,4,5,6,7,8,9,10,11,12,0"
344   # (0,1,2,3,4,5,6,7,8,9,10,11,12)
345
346
347
348
349
350
351   GENERATORS_HESSE_GROUP="\
352   3000300030 \
353   2000201230 \
354   1000100111 \
355   1000220200 \
356   1002312010 \
357   0331003211 \
358   2200011331"
359
360
361
362
363   GENERATORS_WEYL_GROUP_E8="\
364   -1,-1,-1,-1,0,0,0,0, \
365   0,0,0,1,0,0,0,0, \
366   1,0,0,0,0,0,0,0, \
367   0,0,1,0,0,0,0,0, \
368   0,1,0,1,1,0,0,0, \
369   0,0,0,0,0,1,0,0, \
370   0,0,0,0,0,0,1,0, \
371   0,0,0,0,0,0,0,1, \
372   -1,0,-1,-1,-1,-1,-1,-1, \
373   0,1,0,1,1,1,1,1, \
374   1,0,0,0,0,0,0,0, \
375   0,0,1,0,0,0,0,0, \
376   0,0,0,1,0,0,0,0, \
377   0,0,0,0,1,0,0,0, \
378   0,0,0,0,0,1,0,0, \
379   0,0,0,0,0,0,1,0"
380
381
382
383   PGOp_6_2_GENS="\
384   1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,1,0, \
385   1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,1,1,1,1,0,0,1,0,0,1,0,0,1,0,1,0, \
386   1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,1,0,0,0,1,0,0,1,0,0,0,0,1,0, \
387   1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,0,0,1,0,0,1,0,0,0,1,0,0, \
388   1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,1,0,1,0,0,0,0,0,1,0,0,0,0,1,0,1, \
389   1,0,0,0,0,0,0,1,1,1,1,1,1,0,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,0,1,0,0,1,0,0, \
390   1,0,0,0,0,0,0,1,0,1,0,0,1,0,1,0,0,1,0,0,0,1,0,0,0,0,0,1,1,0,0,0,0,0,0,1, \
391   0,1,0,1,1,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,1,0,1,0,0,0,0,0,1,0,0,1,1,0,0,1, \
392   0,0,1,1,1,1,1,0,1,0,0,0,0,1,1,1,1,1,1,1,1,1,0,0,0,1,0,1,0,1,0,1,0,1,1,0, \
```

```
393   0,0,0,0,1,0,1,0,1,1,1,1,1,0,1,0,1,0,1,0,0,1,1,0,1,0,0,0,0,0,1,1,1,1,1,0,"
394
395
396
397
398
399
400   # Section 6.4 New groups from old
401
402
403   SURFACE_Q13_STAB_GENS="1,0,0,0,9,12,0,0,10,0,12,0,9,0,0,12, \
404   1,0,0,0,0,12,12,6,6,0,0,7,1,0,2,0, \
405   0,1,1,7,3,9,9,11,2,10,10,3,9,9,1,11"
406
407
408
409
410   # Section 6.6 Group theoretic activities based on Magma
411
412   # large sets of PG(2,3):
413
414   GENERATORS_H5="1,2,3,4,0,6,7,8,9,5,10,11,12"
415   # (0, 1, 2, 3, 4)(5, 6, 7, 8, 9)
416
417   GENERATORS_N5="\
418   0,1,2,3,4,9,5,6,7,8,10,11,12, \
419   0,1,2,3,4,5,6,7,8,9,10,12,11, \
420   0,4,3,2,1,5,9,8,7,6,10,11,12, \
421   0,2,4,1,3,5,7,9,6,8,10,11,12, \
422   0,1,2,3,4,5,6,7,8,9,11,10,12, \
423   1,2,3,4,0,6,7,8,9,5,10,11,12, \
424   5,9,8,7,6,0,4,3,2,1,10,11,12"
425
426
427   PGOM_6_2_CLASS_2A_REP="1,0,0,0,0,0,0,1,0,0,\
428   0,0,0,0,1,0,0,0,0,0,1,1,1,1,0,0,1,0,0,1,0,0,1,0,1,0"
429
430   FILE_NAME_PGOM_6_2_CLASS_2A="PGOm_6_2_class_of_2A.csv"
431
432   PGOM_6_2_CLASS_2A_REP_CLEAN="1,0,0,0,0,0,0,1,0,0,\
433   0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1"
434
435   PGOM_6_2_CLASS_2A_CENTRALIZER_GENS="\
436   1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,1, \
437   1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,1,0, \
438   1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1, \
439   1,0,0,0,0,0,1,1,0,0,0,1,0,0,1,0,0,0,0,0,0,1,0,0,1,0,0,0,1,1,0,0,0,0,0,1, \
440   1,0,0,0,0,0,1,1,0,0,1,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,0,0,0,1,1, \
441   1,0,0,0,0,0,1,1,1,1,0,0,1,0,0,1,0,0,1,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1, \
442   0,1,1,1,1,0,1,0,1,1,1,0,1,1,0,1,1,0,1,1,1,0,1,0,0,0,0,0,1,0,1,1,1,1,1,1, \
443   1,1,0,0,0,1,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,1,0,0,1,1,0,0,0,0,0,1, \
444   0,1,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,1"
445
446
447
448   # Section 6.7 Group theoretic activities based on GAP
449
450
451
```

```
452  # Section 6.8 Linear groups, advanced topics
453
454
455
456
457  #Co3 from Conway et al., 1985 (ATLAS)
458  #order = 495766656000
459  #Co3 from the paper by Suleiman and Wilson 1997
460
461
462  CONWAY_GEN1="\
463  11011100010000010010000\
464  11110101111010000101011\
465  00000010000001000010101\
466  11111001101100010011110\
467  01010100000000100111101\
468  00000100000001000010101\
469  00100000000001000010101\
470  00010000110000001111111\
471  11101001001101000100011\
472  00000000000001100101010\
473  00000000001001000010101\
474  01101111110100111011111\
475  00000000000011000010101\
476  00000000000001000000101\
477  00000000010001000010101\
478  00000000000001000111010\
479  00010001100000100110110\
480  00000000000000000010101\
481  00000000000000101010101\
482  00000000000001000010100\
483  00000000000001000010111\
484  00000000000001000010001"
485
486  CONWAY_GEN2="\
487  01010000101110101111111\
488  01100101000111101100000\
489  00110100001111110101111\
490  00011011100010110100011\
491  10100100001000010111110\
492  11010000000010101000011\
493  11001010100011110101010\
494  10001101001101010101010\
495  01001100010100000000111\
496  11000000101001010100010\
497  01011101100111000001010\
498  01011111010100111111001\
499  10000101010101010100010\
500  00010100001111001001111\
501  00110100101110110011111\
502  01001100101100111111010\
503  11010110011111011000011\
504  01001010010010001000010\
505  11001011000010011100011\
506  01011101100101000000001\
507  00000011011110001011110\
508  11011010101011100001010"
509
510
```

```
511
512
513
514   Ree_gen1="21,5,1,6,17,1,1, 3,13,5,21,6,6,18, 21,3,21,21,22,6,14, \
515   14,18,1,5,13,6,7, 3,3,2,1,24,16,3, 17,3,22,10,16,24,26, \
516   21,21,6,18,20,2,5"
517
518   Ree_gen2="16,3,11,5,16,22,20, 24,6,18,24,7,1,26, 9,23,17,18,23,20,13, \
519   9,7,2,15,17,5,11, 3,3,6,21,4,24,16, 25,8,6,24,21,12,7, \
520   24,15,2,13,11,14,24"
521
522
523
524   # Chapter 7 Orbit Algorithms
525
526
527   # Section 7.7
528
529   SURFACE_F9_1_EQN_RESTRICTED="4*X0*X1^2+4*X1^2*X2+8*X0*X2^2+3*X1*X2^2+3*X0*X1*X2"
530
531
532   # Chapter 8 Cubic Surfaces
533
534   # Section 8.2 Generalities
535
536
537   # Section 8.2 Creation
538
539   # Section 8.3 Activities
540
541   # points:
542   #0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,23,26,27,30,31,34,35,38,39,42,47,48,51,52,53,
      54,59,60,61,62,67,68,69,70,75,76,79,80,81,82
543
544   # points off:
545   #15,16,17,18,19,20,21,22,24,25,28,29,32,33,36,37,40,41,43,44,45,46,49,50,55,56,57
      ,58,63,64,65,66,71,72,73,74,77,78,83,84
546
547   HIRSCHFELD_SURFACE_POINTS_OFF="15,16,17,18,19,20,21,22,24,\
548   25,28,29,32,33,36,37,40,41,43,44,45,46,49,50,55,56,57,58,\
549   63,64,65,66,71,72,73,74,77,78,83,84"
550
551   #Hesse planes:
552   #7,8,11,13,14,16,17,19,28,29,32,34,35,37,38,40,42,43,44,45,47,48,52,54,56,57,60,6
      1,63,64,65,66,68,69,73,75,77,78,81,82
553
554   HIRSCHFELD_SURFACE_HESSE_PLANES="7,8,11,13,14,16,17,19,28,\
555   29,32,34,35,37,38,40,42,43,44,45,47,48,52,54,56,57,60,61,\
556   63,64,65,66,68,69,73,75,77,78,81,82"
557
558
559
560
561   # Section 8.4 Classification
562
563
564   # Section 8.5 Quartic Curves
565
566
```

```
567
568   NB_CUBIC_SURFACES_Q7=1
569
570
571   NB_CUBIC_SURFACES_Q13=4
572
573
574   NB_CUBIC_SURFACES_Q19=10
575
576   NB_QUARTIC_CURVES_Q19=14
577
578
579
580   # Section 8.6 Interface to GAP
581
582
583
584   # Chapter 9 Applications
585
586
587   # Section 9.1 Number Theory
588
589   # Section 9.2 Representation Theory
590
591
592   # Section 9.3 Cryptography
593
594
595   NTRU_N=7
596
597   NTRU_P=3
598
599   NTRU_Q=41
600
601   NTRU_D=2
602
603   NTRU_XN1="-1,0,0,0,0,0,0,1,"
604
605   # D + 1 plus ones and D minus ones
606
607   ALICE_PRIVATE_F="-1,0,1,1,-1,0,1"
608
609   # D plus ones and D minus ones
610
611   ALICE_PRIVATE_G="0,-1,-1,0,1,0,1"
612
613   #F_q(x) = 8X^{6} + 26X^{5} + 31X^{4} + 21X^{3} + 40X^{2} + 2X + 37
614
615   ALICE_PRIVATE_FQ="37,2,40,21,31,26,8"
616
617
618   #F_p(x) = X^{6} + 2X^{5} + X^{3} + X^{2} + X + 1
619   ALICE_PRIVATE_FP="1,1,1,1,0,2,1"
620
621   #C(X)=20X^{6} + 40X^{5} + 2X^{4} + 38X^{3} + 8X^{2} + 26X + 30
622   ALICE_PUBLIC_KEY="30,26,8,38,2,40,20"
623
624   BOB_MESSAGE="1,-1,1,1,0,-1"
625
```

```
626 BOB_ONE_TIME_KEY="-1,1,0,0,0,-1,1"
627
628 # E(X) = 31X^{6} + 19X^{5} + 4X^{4} + 2X^{3} + 40X^{2} + 3X + 25
629 BOB_ENCRYPT= "25,3,40,2,4,19,31"
630
631 #C(X)=X^{6} + 10X^{5} + 33X^{4} + 40X^{3} + 40X^{2} + X + 40
632 ALICE_C1="40,1,40,40,33,10,1"
633
634 #A(X)=X^{6} + 10X^{5} - 8X^{4} - X^{3} - X^{2} + X - 1
635 ALICE_C2="-1,1,-1,-1,-8,10,1"
636
637 #A(X)=X^{6} + X^{5} + X^{4} + 2X^{3} + 2X^{2} + X + 2
638 ALICE_C3="2,1,2,2,1,1,1"
639
640
641 #C(X)=2X^{5} + X^{3} + X^{2} + 2X + 1
642 ALICE_C4="1,2,1,1,0,2"
643
644
645
646 # Chapter 10 Coding Theory
647
648 # Section 10.1 Introduction
649
650 CODE_5_2_3_CODEWORDS="0,7,25,30"
651
652
653 # Section 10.2 Linear codes
654
655
656 SIMPLEX_CODE_GENERATOR="\
657 1,0,1,0,1,0,1, \
658 0,1,1,0,0,1,1, \
659 0,0,0,1,1,1,1"
660
661
662
663 CODE_RM_3_1_GENMA="\
664 11111111\
665 01010101\
666 00110011\
667 00001111"
668
669 CODE_RM_4_1_GENMA="\
670 1111111111111111\
671 0101010101010101\
672 0011001100110011\
673 0000111100001111\
674 0000000011111111"
675
676
677 HAMMING_CODE_ROWS_IN_BINARY_RANKS="67,37,22,15"
678
679 HAMMING_CODE_CODEWORDS="0, 67, 37, 102, 22, 85, \
680 51, 112, 15, 76, 42, 105, 25, 90, 60, 127"
681
682
683 HAMMING_CODE_GENERATOR="\
684 1,0,0,0,0,1,1, \
```

```
685   0,1,0,0,1,0,1, \
686   0,0,1,0,1,1,0, \
687   0,0,0,1,1,1,1"
688
689
690   GOLAY23_CODE_GENERATOR="\
691   1,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,0,0,1,0,\
692   0,1,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,1,1,1,0,0,1,\
693   0,0,1,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,0,0,0,1,1,\
694   0,0,0,1,0,0,0,0,0,0,0,0,1,1,0,0,1,0,0,1,1,1,1,\
695   0,0,0,0,1,0,0,0,0,0,0,0,1,0,1,1,0,1,0,1,1,1,0,\
696   0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,0,1,0,1,1,1,0,0,\
697   0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,1,1,1,1,0,1,0,1,\
698   0,0,0,0,0,0,0,1,0,0,0,0,0,1,1,1,0,1,1,0,1,0,0,\
699   0,0,0,0,0,0,0,0,1,0,0,0,0,1,1,0,1,1,0,1,0,1,0,\
700   0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,1,1,0,1,1,0,0,1,\
701   0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,0,0,0,1,1,1,\
702   0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,1,1,1,1,1"
703
704
705
706   # Section 10.3 Golay codes
707
708
709
710
711
712   GOLAY_23_COLUMN_RANKS_PROJECTIVELY="0, 1, 2, 3, 4, 5, 6, 7, \
713   8, 9, 10, 11, 132, 913, 1460, 1750, 1898, 2518, 2787, 2874, \
714   3320, 3357, 3662"
715
716   #[23,12,8]
717   #0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 132, 913, 1460, 1750, 1898, 2518, 2787, 28
      74, 3320, 3357, 3662
718
719   #[24,12,8]
720   #0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 132, 913, 1460, 1750, 1898, 2518, 2787, 28
      74, 3320, 3357, 3662, 4004
721
722   GOLAY_24_CODE_PROJECTIVE="0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 132, 913, 1460, 1
      750, 1898, 2518, 2787, 2874, 3320, 3357, 3662, 4004"
723
724   GOLAY_24_CODE_AUT_GENS="\
725   1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0
      ,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,
      0,0,0,1,1,0,0,1,1,0,1,1,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,1,0,1,1,0,1,1,1,0,0,0
      ,0,0,0,0,0,0,1,0,0,0,1,0,1,1,1,0,0,0,1,0,1,1,\
726   1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0
      ,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,
      0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,1,0,1,1,0,1,0,1,0,1,0,1,0,1,1,0,1,1,0,0,0
      ,0,0,0,0,0,1,0,0,0,0,0,0,0,1,1,1,1,1,0,0,1,1,\
727   1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0
      ,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,
      0,0,0,0,1,0,1,1,1,0,0,1,1,0,1,1,1,0,0,0,1,1,1,0,1,0,1,0,1,1,0,1,0,1,1,1,0,0,1,0,0
      ,1,0,0,1,0,1,1,1,1,1,1,1,1,1,1,0,0,0,1,1,1,0,0,\
728   1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0
      ,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,
      0,0,0,0,1,1,0,1,0,1,1,1,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,1,0,1,1,1,0,0,1,1,0,1,1,0
      ,1,1,1,0,0,0,1,0,1,1,0,0,0,0,0,0,0,0,1,0,0,0,\
```

```
729 1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0
    ,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,1,0,0,0,
    0,0,0,0,1,0,1,1,1,0,0,1,1,0,1,0,1,1,0,1,0,1,1,1,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0
    ,0,0,0,0,0,1,0,0,0,0,1,1,0,0,1,1,0,1,1,0,1,1,0,1,0,\
730 1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0
    ,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,1,0,1,1,0,1,1,1,0,1,1,0,0,1,1,0,1,1,
    0,1,0,0,1,0,1,1,1,0,0,1,1,0,1,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,1,1,1,0,1,0,1,0,0
    ,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,\
731 1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0
    ,0,0,0,0,0,0,0,1,0,0,1,0,1,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,1,0,1,0,1,1,0,
    1,1,0,1,1,0,0,0,1,1,1,0,1,0,1,0,1,1,0,1,0,1,1,1,0,0,1,0,1,0,1,1,1,0,0,1,1,0,1,0,0
    ,0,0,0,1,0,0,0,0,0,0,1,1,1,1,0,0,0,1,1,1,0,0,\
732 1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,1,1,1,1
    ,1,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,
    0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,1,1,0,1,1,1,0,0,0,1,0,1,1,1,1,0,0,1,1,0,1,1,0,1,0,0,0
    ,0,1,1,1,1,1,0,0,1,1,0,0,0,0,0,0,0,0,0,1,0,0,\
733 1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,1
    ,1,1,1,0,0,1,1,0,1,1,0,1,0,1,1,1,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,
    1,0,0,1,1,0,0,0,1,1,1,1,0,1,0,1,1,1,1,0,0,1,1,0,1,1,1,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0
    ,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,\
734 1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,0,0,1,1,1,0,0,0
    ,1,1,1,0,1,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,1,1,1,1,0,0,1,1,0,0,0,0,0,0,0,0,1,
    0,0,0,0,1,0,1,1,1,0,0,1,1,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0
    ,0,1,0,1,1,0,1,1,1,0,1,0,1,0,1,0,1,1,0,1,1,0,\
735 1,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,1,0,0,1,1,0,1,0,1,1,0,1,0,1,1,1,0,0,1,0,0,0,0,0
    ,1,0,0,0,0,0,0,1,0,1,1,1,0,0,0,1,0,1,1,0,0,0,0,0,0,0,1,0,0,0,0,1,1,0,0,0,1,1,1,0,
    1,0,1,0,0,0,1,0,0,0,0,0,0,0,0,1,0,1,0,1,0,1,1,0,1,1,0,0,0,0,0,1,0,0,0,0,0,0,0,1,1
    ,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,\
736 0,1,1,1,0,0,1,0,0,1,1,1,1,1,0,0,0,1,1,1,0,1,0,1,1,1,0,0,1,1,0,1,1,0,1,0,0,0,0,1,1
    ,1,1,1,0,0,1,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,1,0,1,0,1,1,1,
    0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,1,0,0,1,1,0,1,0,0
    ,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0"
737
738
739 M24_ELEMENT_2A="0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,1,1,1,0,0,0,1
    ,0,1,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,
    1,1,1,0,0,1,0,0,1,1,1,1,1,1,1,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0
    ,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,"
740
741 M24_ELEMENT_2B="0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0
    ,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,
    0,0,1,0,1,1,0,1,1,1,0,1,1,0,0,1,1,0,1,1,0,1,0,0,0,0,1,0,0,0,0,0,0,0,1,0,1,1,1
    ,0,0,1,1,0,1,1,0,1,1,1,0,0,0,1,0,1,1,0,0,0,0,1,0,0,0,0,0,0,0,"
742
743
744 # 2A:
745 # Class 1 / 26
746 # order of element = 2
747 # class size = 11385
748 # centralizer order = 21504
749 # element has 255 fixed points in PG(11,2)
750
751 # 2B:
752 # Class 2 / 26
753 # order of element = 2
754 # class size = 31878
755 # centralizer order = 7680
756 # element has 63 fixed points in PG(11,2)
757
```

```
758   M24_2A_PERM="10, 1, 21, 4, 3, 5, 22, 13, 11, 9, 0, 8, \
759   16, 7, 23, 15, 12, 17, 18, 19, 20, 2, 6, 14"
760
761
762   M24_2B_PERM="2, 5, 0, 8, 11, 1, 16, 14, 3, 19, 21, 4, \
763   22, 23, 7, 20, 6, 18, 17, 9, 15, 10, 12, 13"
764
765   GOLAY24_2A_FIX_SUBCODE="1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, \
766   0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, \
767   0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, \
768   0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, \
769   0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, \
770   0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, \
771   0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, \
772   0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, \
773   0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, \
774   0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, \
775   1, 1, 0, 0, 0, 1, 1, 0, 0, 1"
776
777   GOLAY24_2B_FIX_SUBCODE="1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, \
778   0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, \
779   0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, \
780   0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, \
781   0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, \
782   1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, \
783   0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, \
784   0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1"
785
786
787
788
789   # Section 10.4 CRC codes
790
791
792   CRC4="1,4,1,2,1,1,1,0"
793
794   CRC7="1,7,1,3,1,0"
795
796   CRC8_ATM="1,8,1,2,1,1,1,0"
797
798   CRC16_CCITT="1,16,1,12,1,5,1,0"
799
800   CRC32_ETHERNET="1,32,1,26,1,23,1,22,1,16,1,12,1,11,1,10,1,8,1,7,\
801   1,5,1,4,1,2,1,1,1,0"
802
803   CRC32_ETHERNET_POLY="X^32+X^26+X^23+X^22+X^16+\
804   X^12+X^11+X^10+X^8+X^7+X^5+X^4+X^2+X^1+1"
805
806   CRC32_CASTAGNOLI="1,32,1,28,1,27,1,26,1,25,1,23,1,22,1,20,1,19,1,\
807   18,1,14,1,13,1,11,1,10,1,9,1,8,1,6,1,0"
808
809   CRC64_ECMA182="1,64,1,62,1,57,1,55,1,54,1,53,1,52,1,47,1,46,1,45,\
810   1,40,1,39,1,38,1,37,1,35,1,33,1,32,1,31,1,29,1,27,1,24,1,23,1,22,\
811   1,21,1,19,1,17,1,13,1,12,1,10,1,9,1,7,1,4,1,1,1,0"
812
813   CRC64_ROCKSOFT="1,64,1,63,1,61,1,59,1,58,1,56,1,55,1,52,1,49,1,48,\
814   1,47,1,46,1,44,1,41,1,37,1,36,1,34,1,32,1,31,1,28,1,26,1,23,1,22,1,\
815   19,1,16,1,13,1,12,1,10,1,9,1,6,1,4,1,3,1,0"
816
```

```
817
818
819   CRC_POLY_Q256_DEG2_DENSE="214,167,1"
820
821
822
823   CRC_POLY_BRAVO_DENSE="1,23,27,213,1"
824
825
826   CRC_POLY_CHARLIE_DENSE="1,126,25,1,196,209,244,3,121,126,35,65,1"
827
828
829   POLY_Q256_DEG30_SPARSE="1,0,26,1,210,2,24,3,\
830   138,4,148,5,160,6,58,7,108,8,199,9,95,10,56,\
831   11,9,12,205,13,194,14,193,15,3,16,248,17,110,\
832   18,150,19,24,20,169,21,192,22,212,23,112,24,\
833   144,25,97,26,109,27,174,28,253,29,1,30"
834
835   POLY_Q256_DEG30_DENSE="1,26,210,24,138,148,\
836   160,58,108,199,95,56,9,205,194,193,3,248,110,\
837   150,24,169,192,212,112,144,97,109,174,253,1"
838
839   CRC_FILE=allen_Gates
840
841   CRC_FILE_EXTENSION=bmp
842
843   TEN_MILLION=10000000
844
845
846   CRC_BCH_Q16_N51_D3_PROJECTIVE_SET="1900, 2703, 2347, 1653, 232, 3681, 752, 511, 3
      916, 1222, 632, 2014, 3951, 4006, 2198, 3258, 2804, 3771, 3120, 2132, 3754, 2640,
       1995, 943, 3319, 1748, 36, 546, 1198, 4296, 4013, 2118, 2234, 3034, 2441, 4085,
      4326, 318, 1180, 2488, 4229, 765, 1470, 2306, 666, 3454, 1763, 0, 1, 2, 3 "
847
848   CRC_POLY_DELTA_DENSE="1, 2, 13, 5, 1"
849   #create_BCH_code::init Q=X^{4} + 5X^{3} + 13X^{2} + 2X + 1
850   #create_BCH_code::init BCH_POLYNOMIAL_F16="X^4+5*X^3+13*X^2+2*X+1"
851
852   ERROR_POLY_BRAVO_COEFF="737, 417, 171, 634, 158, 30, 480, 688, 94, 739, 147, 694,
       724, 741, 51, 411, 753, 622, 592, 667, 25, 709, 55, 450, 255, 744, 28, 718, 82,
      740, 372, 144, 435, 505, 478, 233, 325, 231, 27, 518, 492, 350, 621, 468, 297, 14
      1, 735, 338, 227, 434, 54, 180, 390, 245, 301, 617, 423, 408, 521, 42, 697, 719,
      668, 618, 107, 746, 441, 376, 705, 353, 368, 254, 393, 291, 762, 188, 506, 580, 4
      19, 525, 500, 652, 433, 130, 52, 212, 511, 191, 182, 100, 676, 706, 248, 268, 365
      , 626, 303, 5, 490, 524, 237, 210, 326, 619, 485, 767, 432, 721, 391, 650, 88, 62
      7, 177, 700, 354, 682, 38, 360, 70, 636, 609, 766, 203, 7, 369, 584, 475, 84, 87,
       251, 680, 479, 256, 121, 473, 604, 614, 265, 585, 40, 458, 551, 163, 661, 358, 6
      10, 349, 748, 371, 310, 31, 136, 153, 649, 662, 547, 120, 324, 162, 671, 62, 725,
       603, 645, 670, 554, 364, 127, 312, 708, 386, 286, 532, 58, 658, 366, 74, 442, 69
      9, 707, 164, 102, 593, 657, 293, 638, 370, 716, 454, 126, 204, 232, 637, 409, 491
      , 666, 402, 209, 639, 560, 578, 467, 24, 629, 93, 453, 352, 425, 406, 37, 684, 18
      9, 648, 186, 345, 691, 673, 193, 264, 247, 714, 267, 540, 215, 134, 729, 588, 295
      , 294, 241, 577, 398, 722, 46, 214, 60, 640, 770, 477, 106, 97, 381, 280, 675, 41
      8, 29, 385, 269, 151, 665, 109, 514, 305, 377, 95, 544, 570, 552, 128, 152, 599,
      397, 415, 149, 451, 738, 743, 302, 140, 273, 103, 284, 396, 726, 327, 221, 288, 2
      00, 344, 285, 112, 542, 22, 313, 263, 218, 61, 157, 362, 559, 507, 243, 664, 206,
       550, 242, 244, 230, 145, 761, 413, 679, 392, 447, 17, 330, 156, 279, 752, 487, 7
      31, 651, 388, 535, 154, 502, 108, 452, 363, 240, 569, 564, 713, 704, 495, 589, 83
      , 15, 483, 261, 537, 595, 219, 573, 516, 252, 199, 337, 616, 339, 486, 555, 653,
```

```
      728, 137, 216, 56, 211, 734, 449, 44, 26, 348, 426, 91, 139, 238, 539, 444, 754,
      672, 695, 543, 659, 601, 654, 742, 300, 69, 115, 756, 122, 582, 374, 236, 656, 53
      3, 755, 512, 594, 404, 65, 611, 316, 703, 361, 318, 80, 92, 12, 720, 378, 763, 22
      8, 429, 623, 608, 712, 33, 612, 185, 747, 11, 620, 78, 234, 202, 165, 498, 173, 1
      60, 494, 768, 75, 306, 482, 641, 53, 222, 379, 460, 258, 567, 50, 531, 548, 207,
      424, 320, 383, 124, 757, 249, 10, 405, 575, 597, 587, 341, 328, 336, 513, 317, 17
      2, 681, 169, 187, 430, 351, 523, 81, 553, 389, 496, 689, 461, 607, 359, 436, 105,
       229, 8, 101, 176, 663, 43, 701, 356, 630, 562, 678, 272, 224, 387, 271, 311, 456
      , 530, 401, 278, 239, 520, 529, 287, 343, 59, 367, 342, 21, 503, 420, 466, 322, 1
      46, 504, 563, 196, 545, 558, 161, 394, 257, 49, 99, 598, 6, 77, 143, 68, 118, 133
      , 690, 410, 235, 536, 48, 205, 270, 445, 283, 198, 660, 90, 47, 57, 727, 41, 463,
       76, 581, 576, 192, 250, 686, 509, 34, 175, 246, 400, 168, 111, 334, 403, 517, 67
      7, 373, 497, 431, 470, 501, 174, 225"
853
854   ERROR_POLY_BRAVO_PLACE="100, 202, 178, 171, 102, 243, 43, 238, 132, 200, 126, 107
      , 61, 252, 150, 5, 18, 51, 161, 248, 230, 166, 207, 48, 138, 132, 221, 248, 228,
      42, 126, 117, 253, 85, 133, 165, 56, 125, 224, 92, 223, 129, 108, 179, 197, 224,
      163, 241, 196, 207, 53, 73, 183, 98, 172, 7, 65, 90, 35, 15, 59, 219, 95, 41, 93,
       52, 51, 19, 205, 78, 66, 6, 218, 47, 109, 228, 45, 134, 141, 173, 225, 93, 194,
      5, 164, 100, 199, 10, 118, 30, 163, 110, 152, 247, 212, 214, 146, 104, 164, 7, 11
      7, 87, 129, 159, 235, 129, 233, 224, 72, 250, 167, 128, 208, 220, 150, 195, 40, 1
      68, 73, 21, 98, 185, 128, 27, 30, 74, 251, 134, 186, 40, 109, 138, 181, 202, 137,
       86, 72, 230, 117, 18, 123, 180, 10, 23, 122, 47, 241, 16, 193, 71, 189, 203, 212
      , 68, 241, 143, 191, 62, 228, 255, 82, 12, 178, 134, 161, 178, 251, 119, 66, 209,
       33, 32, 17, 124, 47, 96, 191, 20, 10, 4, 82, 36, 65, 36, 164, 113, 230, 30, 33,
      176, 75, 54, 130, 25, 171, 227, 1, 41, 182, 164, 87, 232, 210, 74, 161, 84, 24, 1
      87, 106, 106, 52, 153, 247, 152, 252, 245, 182, 253, 213, 203, 45, 168, 193, 190,
       42, 105, 201, 1, 211, 224, 153, 189, 89, 187, 59, 238, 53, 95, 105, 103, 24, 2,
      62, 250, 57, 70, 68, 110, 45, 45, 226, 254, 194, 103, 186, 140, 209, 1, 37, 87, 1
      5, 200, 26, 22, 216, 251, 175, 137, 33, 228, 132, 152, 86, 179, 106, 68, 69, 49,
      202, 146, 67, 168, 130, 223, 55, 234, 177, 88, 146, 195, 100, 176, 210, 177, 61,
      161, 1, 58, 5, 191, 30, 49, 183, 83, 125, 13, 170, 211, 93, 110, 219, 93, 246, 12
      0, 169, 97, 206, 86, 241, 208, 195, 199, 53, 31, 133, 253, 181, 247, 192, 209, 19
      , 159, 175, 39, 247, 154, 12, 225, 43, 231, 198, 83, 125, 33, 175, 79, 242, 224,
      150, 251, 213, 125, 155, 230, 61, 109, 32, 239, 227, 40, 8, 133, 186, 118, 232, 1
      57, 218, 41, 24, 168, 32, 205, 172, 216, 36, 94, 179, 37, 212, 13, 35, 171, 16, 1
      71, 39, 69, 241, 165, 62, 91, 136, 4, 227, 168, 53, 6, 226, 251, 58, 251, 38, 234
      , 245, 255, 168, 119, 169, 47, 179, 223, 28, 32, 17, 185, 49, 44, 196, 122, 89, 1
      16, 157, 223, 252, 204, 40, 119, 89, 8, 7, 178, 235, 52, 12, 51, 217, 60, 104, 10
      0, 59, 111, 14, 244, 49, 223, 225, 245, 187, 61, 242, 174, 29, 145, 172, 177, 131
      , 188, 157, 76, 104, 244, 94, 39, 96, 246, 131, 77, 255, 70, 211, 201, 166, 105,
      68, 129, 4, 80, 20, 89, 58, 222, 159, 208, 82, 255, 245, 121, 210, 149, 202, 105,
       212, 42, 35, 188, 3, 121, 23, 139, 50, 24, 224, 20, 140, 119, 55, 118, 53, 155,
      19, 70, 138, 178, 202, 153, 78, 44, 90, 109, 32, 253, 64, 130, 238, 243, 87, 12,
      204, 63, 116, 163, 82, 25, 14, 31, 197, 193, 203, 235, 79, 86, 118, 118, 88, 190,
       26, 55, 179, 156, 153, 89, 227, 221"
855
856   CRC_ERROR_POLY_BRAVO="104*X^5+118*X^6+27*X^7+39*X^8+51*X^10+245*X^11+91*X^12+247*
      X^15+125*X^17+202*X^21+130*X^22+210*X^24+230*X^25+125*X^26+224*X^27+221*X^28+70*X
      ^29+243*X^30+189*X^31+58*X^33+193*X^34+106*X^37+40*X^38+18*X^40+204*X^41+15*X^42+
      77*X^43+213*X^44+187*X^46+243*X^47+44*X^48+140*X^49+204*X^50+150*X^51+164*X^52+12
      2*X^53+53*X^54+207*X^55+242*X^56+87*X^57+124*X^58+121*X^59+238*X^60+177*X^61+82*X
      ^62+171*X^65+19*X^68+24*X^69+73*X^70+191*X^74+185*X^75+116*X^76+53*X^77+168*X^78+
      165*X^80+29*X^81+228*X^82+181*X^83+134*X^84+186*X^87+167*X^88+238*X^90+61*X^91+62
      *X^92+161*X^93+132*X^94+186*X^95+24*X^97+119*X^99+30*X^100+96*X^101+36*X^102+132*
      X^103+244*X^105+103*X^106+93*X^107+206*X^108+226*X^109+118*X^111+67*X^112+168*X^1
      15+70*X^118+191*X^120+202*X^121+205*X^122+235*X^124+176*X^126+119*X^127+37*X^128+
      5*X^130+138*X^133+42*X^134+203*X^136+175*X^137+109*X^139+33*X^140+224*X^141+155*X
      ^143+117*X^144+5*X^145+188*X^146+126*X^147+22*X^149+45*X^151+87*X^152+212*X^153+1
```

```
    69*X^154+170*X^156+88*X^157+102*X^158+28*X^160+24*X^161+228*X^162+10*X^163+82*X^1
    64+47*X^165+86*X^168+245*X^169+178*X^171+223*X^172+223*X^173+227*X^174+203*X^175+
    246*X^176+208*X^177+73*X^180+118*X^182+38*X^185+152*X^186+187*X^187+228*X^188+153
    *X^189+10*X^191+25*X^192+253*X^193+23*X^196+64*X^198+12*X^199+49*X^200+169*X^202+
    128*X^203+75*X^204+90*X^205+177*X^206+89*X^207+41*X^209+87*X^210+224*X^211+100*X^
    212+59*X^214+190*X^215+79*X^216+234*X^218+175*X^219+68*X^221+89*X^222+68*X^224+22
    1*X^225+196*X^227+168*X^228+94*X^229+58*X^230+125*X^231+54*X^232+165*X^233+119*X^
    234+153*X^235+36*X^236+117*X^237+32*X^238+159*X^239+208*X^240+224*X^241+161*X^242
    +176*X^243+1*X^244+98*X^245+235*X^246+203*X^247+152*X^248+12*X^249+14*X^250+40*X^
    251+154*X^252+6*X^254+138*X^255+181*X^256+20*X^257+223*X^258+209*X^261+55*X^263+2
    13*X^264+230*X^265+168*X^267+247*X^268+110*X^269+109*X^270+4*X^271+105*X^272+228*
    X^273+222*X^278+211*X^279+62*X^280+253*X^283+152*X^284+146*X^285+32*X^286+255*X^2
    87+69*X^288+47*X^291+164*X^293+211*X^294+1*X^295+197*X^297+41*X^300+172*X^301+137
    *X^302+146*X^303+194*X^305+49*X^306+71*X^310+80*X^311+66*X^312+223*X^313+171*X^31
    6+49*X^317+241*X^318+7*X^320+35*X^322+62*X^324+56*X^325+129*X^326+106*X^327+111*X
    ^328+13*X^330+118*X^334+14*X^336+225*X^337+241*X^338+231*X^339+59*X^341+149*X^342
    +245*X^343+202*X^344+252*X^345+155*X^348+241*X^349+129*X^350+242*X^351+24*X^352+7
    8*X^353+150*X^354+70*X^356+122*X^358+76*X^359+168*X^360+69*X^361+146*X^362+241*X^
    363+251*X^364+212*X^365+96*X^366+210*X^367+66*X^368+30*X^369+230*X^370+193*X^371+
    126*X^372+55*X^373+216*X^374+19*X^376+103*X^377+4*X^378+116*X^379+2*X^381+178*X^3
    83+68*X^385+33*X^386+129*X^387+246*X^388+172*X^389+183*X^390+72*X^391+183*X^392+2
    18*X^393+224*X^394+86*X^396+200*X^397+189*X^398+79*X^400+58*X^401+1*X^402+88*X^40
    3+35*X^404+217*X^405+106*X^406+90*X^408+25*X^409+202*X^410+5*X^411+30*X^413+26*X^
    415+202*X^417+57*X^418+141*X^419+212*X^420+65*X^423+8*X^424+187*X^425+230*X^426+5
    3*X^429+61*X^430+156*X^431+233*X^432+194*X^433+207*X^434+253*X^435+104*X^436+51*X
    ^441+20*X^442+227*X^444+32*X^445+83*X^447+251*X^449+48*X^450+216*X^451+86*X^452+8
    4*X^453+33*X^454+20*X^456+123*X^458+157*X^460+188*X^461+63*X^463+42*X^466+232*X^4
    67+179*X^468+153*X^470+137*X^473+251*X^475+105*X^477+133*X^478+138*X^479+43*X^480
    +44*X^482+192*X^483+235*X^485+198*X^486+110*X^487+164*X^490+171*X^491+223*X^492+3
    2*X^494+133*X^495+177*X^496+179*X^497+179*X^498+225*X^500+89*X^501+97*X^502+105*X
    ^503+3*X^504+85*X^505+45*X^506+100*X^507+197*X^509+199*X^511+212*X^512+244*X^513+
    254*X^514+247*X^516+190*X^517+92*X^518+208*X^520+35*X^521+174*X^523+7*X^524+173*X
    ^525+82*X^529+89*X^530+40*X^531+17*X^532+179*X^533+120*X^535+78*X^536+19*X^537+23
    9*X^539+193*X^540+168*X^542+186*X^543+140*X^544+139*X^545+143*X^547+119*X^548+61*
    X^550+180*X^551+1*X^552+145*X^553+178*X^554+83*X^555+50*X^558+195*X^559+164*X^560
    +201*X^562+121*X^563+199*X^564+252*X^567+195*X^569+209*X^570+39*X^573+60*X^575+82
    *X^576+153*X^577+87*X^578+134*X^580+163*X^581+172*X^582+74*X^584+117*X^585+100*X^
    587+201*X^588+253*X^589+161*X^592+65*X^593+13*X^594+159*X^595+104*X^597+55*X^598+
    15*X^599+232*X^601+178*X^603+86*X^604+157*X^607+226*X^608+98*X^609+47*X^610+16*X^
    611+251*X^612+72*X^614+43*X^616+7*X^617+41*X^618+159*X^619+255*X^620+108*X^621+51
    *X^622+6*X^623+214*X^626+128*X^627+74*X^629+211*X^630+171*X^634+21*X^636+130*X^63
    7+113*X^638+182*X^639+53*X^640+196*X^641+134*X^645+247*X^648+68*X^649+250*X^650+9
    3*X^651+93*X^652+125*X^653+157*X^654+94*X^656+36*X^657+47*X^658+118*X^659+130*X^6
    60+23*X^661+241*X^662+131*X^663+210*X^664+45*X^665+227*X^666+248*X^667+95*X^668+1
    61*X^670+255*X^671+8*X^672+182*X^673+250*X^675+163*X^676+26*X^677+166*X^678+49*X^
    679+109*X^680+225*X^681+195*X^682+52*X^684+31*X^686+238*X^688+131*X^689+178*X^690
    +245*X^691+107*X^694+133*X^695+59*X^697+10*X^699+220*X^700+255*X^701+39*X^703+31*
    X^704+205*X^705+110*X^706+4*X^707+209*X^708+166*X^709+251*X^712+53*X^713+45*X^714
    +30*X^716+248*X^718+219*X^719+136*X^720+224*X^721+89*X^722+61*X^724+12*X^725+179*
    X^726+12*X^727+33*X^728+105*X^729+219*X^731+150*X^734+163*X^735+100*X^737+251*X^7
    38+200*X^739+42*X^740+252*X^741+218*X^742+175*X^743+132*X^744+52*X^746+234*X^747+
    16*X^748+93*X^752+18*X^753+40*X^754+37*X^755+32*X^756+52*X^757+191*X^761+109*X^76
    2+227*X^763+185*X^766+129*X^767+17*X^768+95*X^770"
857
858
859
860 # Section 10.5 Reed-Muller codes
861
862
```

```
863
864
865   RM_6_GENERATOR_1="0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,\
866   22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,\
867   46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63"
868
869   RM_6_GENERATOR_2="1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,\
870   41,43,45,47,49,51,53,55,57,59,61,63"
871
872   RM_6_GENERATOR_3="2,3,6,7,18,19,22,23,10,11,14,15,26,27,30,31,34,35,38,\
873   39,42,43,46,47,50,51,54,55,58,59,62,63"
874
875   RM_6_GENERATOR_4="4,6,12,14,36,38,52,54,5,7,13,15,37,39,53,55,20,22,28,\
876   30,44,46,60,62,21,23,29,31,45,47,61,63"
877
878   RM_6_GENERATOR_5="8,9,12,13,24,25,28,29,10,11,14,15,26,27,30,31,40,41,\
879   44,45,56,57,60,61,42,43,46,47,58,59,62,63"
880
881   RM_6_GENERATOR_6="16,18,24,26,48,50,56,58,17,19,25,27,49,51,57,59,20,22,\
882   28,30,52,54,60,62,21,23,29,31,53,55,61,63"
883
884   RM_6_GENERATOR_7="32,34,48,50,33,35,49,51,36,38,52,54,37,39,53,55,40,42,\
885   56,58,41,43,57,59,44,46,60,62,45,47,61,63"
886
887
888
889
890   # Section 10.6 BCH-codes
891
892
893   #CODE_BCH_F8_N21_D5_GENMA
894
895   CODE_BCH_F8_N21_D5_GENMA_OVERRIDE_POLYNOMIAL11="\
896   2,1,2,1,2,3,3,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,\
897   0,2,1,2,1,2,3,3,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,\
898   0,0,2,1,2,1,2,3,3,1,0,0,0,0,0,0,0,0,0,0,0,0,0,\
899   0,0,0,2,1,2,1,2,3,3,1,0,0,0,0,0,0,0,0,0,0,0,0,\
900   0,0,0,0,2,1,2,1,2,3,3,1,0,0,0,0,0,0,0,0,0,0,0,\
901   0,0,0,0,0,2,1,2,1,2,3,3,1,0,0,0,0,0,0,0,0,0,0,\
902   0,0,0,0,0,0,2,1,2,1,2,3,3,1,0,0,0,0,0,0,0,0,0,\
903   0,0,0,0,0,0,0,2,1,2,1,2,3,3,1,0,0,0,0,0,0,0,0,\
904   0,0,0,0,0,0,0,0,2,1,2,1,2,3,3,1,0,0,0,0,0,0,0,\
905   0,0,0,0,0,0,0,0,0,2,1,2,1,2,3,3,1,0,0,0,0,0,0,\
906   0,0,0,0,0,0,0,0,0,0,2,1,2,1,2,3,3,1,0,0,0,0,0,\
907   0,0,0,0,0,0,0,0,0,0,0,2,1,2,1,2,3,3,1,0,0,0,0,\
908   0,0,0,0,0,0,0,0,0,0,0,0,2,1,2,1,2,3,3,1,0,0,0,\
909   0,0,0,0,0,0,0,0,0,0,0,0,0,2,1,2,1,2,3,3,1"
910
911   CODE_BCH_F8_N21_K14_D5_GENMA="\
912   6,1,5,5,4,4,3,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,\
913   0,6,1,5,5,4,4,3,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,\
914   0,0,6,1,5,5,4,4,3,1,0,0,0,0,0,0,0,0,0,0,0,0,0,\
915   0,0,0,6,1,5,5,4,4,3,1,0,0,0,0,0,0,0,0,0,0,0,0,\
916   0,0,0,0,6,1,5,5,4,4,3,1,0,0,0,0,0,0,0,0,0,0,0,\
917   0,0,0,0,0,6,1,5,5,4,4,3,1,0,0,0,0,0,0,0,0,0,0,\
918   0,0,0,0,0,0,6,1,5,5,4,4,3,1,0,0,0,0,0,0,0,0,0,\
919   0,0,0,0,0,0,0,6,1,5,5,4,4,3,1,0,0,0,0,0,0,0,0,\
920   0,0,0,0,0,0,0,0,6,1,5,5,4,4,3,1,0,0,0,0,0,0,0,\
921   0,0,0,0,0,0,0,0,0,6,1,5,5,4,4,3,1,0,0,0,0,0,\
```

```
922   0,0,0,0,0,0,0,0,0,0,6,1,5,5,4,4,3,1,0,0,0,\
923   0,0,0,0,0,0,0,0,0,0,0,6,1,5,5,4,4,3,1,0,0,\
924   0,0,0,0,0,0,0,0,0,0,0,0,6,1,5,5,4,4,3,1,0,\
925   0,0,0,0,0,0,0,0,0,0,0,0,0,6,1,5,5,4,4,3,1"
926
927   CODE_BCH_F8_N63_K9_D43_GENMA="\
928   4,5,2,2,4,5,6,2,4,2,6,0,0,7,1,5,3,7,1,0,0,5,6,4,7,6,7,7,1,2,6,3,1,6,0,0,6,6,6,4,7
      ,7,0,0,3,4,7,5,6,2,5,1,4,4,1,0,0,0,0,0,0,0,0,\
929   0,4,5,2,2,4,5,6,2,4,2,6,0,0,7,1,5,3,7,1,0,0,5,6,4,7,6,7,7,1,2,6,3,1,6,0,0,6,6,6,4
      ,7,7,0,0,3,4,7,5,6,2,5,1,4,4,1,0,0,0,0,0,0,0,\
930   0,0,4,5,2,2,4,5,6,2,4,2,6,0,0,7,1,5,3,7,1,0,0,5,6,4,7,6,7,7,1,2,6,3,1,6,0,0,6,6,6
      ,4,7,7,0,0,3,4,7,5,6,2,5,1,4,4,1,0,0,0,0,0,0,\
931   0,0,0,4,5,2,2,4,5,6,2,4,2,6,0,0,7,1,5,3,7,1,0,0,5,6,4,7,6,7,7,1,2,6,3,1,6,0,0,6,6
      ,6,4,7,7,0,0,3,4,7,5,6,2,5,1,4,4,1,0,0,0,0,0,\
932   0,0,0,0,4,5,2,2,4,5,6,2,4,2,6,0,0,7,1,5,3,7,1,0,0,5,6,4,7,6,7,7,1,2,6,3,1,6,0,0,6
      ,6,6,4,7,7,0,0,3,4,7,5,6,2,5,1,4,4,1,0,0,0,0,\
933   0,0,0,0,0,4,5,2,2,4,5,6,2,4,2,6,0,0,7,1,5,3,7,1,0,0,5,6,4,7,6,7,7,1,2,6,3,1,6,0,0
      ,6,6,6,4,7,7,0,0,3,4,7,5,6,2,5,1,4,4,1,0,0,0,\
934   0,0,0,0,0,0,4,5,2,2,4,5,6,2,4,2,6,0,0,7,1,5,3,7,1,0,0,5,6,4,7,6,7,7,1,2,6,3,1,6,0
      ,0,6,6,6,4,7,7,0,0,3,4,7,5,6,2,5,1,4,4,1,0,0,\
935   0,0,0,0,0,0,0,4,5,2,2,4,5,6,2,4,2,6,0,0,7,1,5,3,7,1,0,0,5,6,4,7,6,7,7,1,2,6,3,1,6
      ,0,0,6,6,6,4,7,7,0,0,3,4,7,5,6,2,5,1,4,4,1,0,\
936   0,0,0,0,0,0,0,0,4,5,2,2,4,5,6,2,4,2,6,0,0,7,1,5,3,7,1,0,0,5,6,4,7,6,7,7,1,2,6,3,1
      ,6,0,0,6,6,6,4,7,7,0,0,3,4,7,5,6,2,5,1,4,4,1"
937
938
939   BCH_21_15_GENERATOR_MATRIX="1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0,
       1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1"

940
941
942   BCH_21_6_GENERATOR_MATRIX=" 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0,
       0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1,
       0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1,
       1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1,
       1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1
      "
943
944   # Section 10.7 Reed-Solomon codes
945
946
947
948
949   CODE_RS_F8_N7_K5_D3_GENMA="\
950   5,6,1,0,0,0,0,\
951   0,5,6,1,0,0,0,\
952   0,0,5,6,1,0,0,\
953   0,0,0,5,6,1,0,\
954   0,0,0,0,5,6,1"
```

```
955
956
957   RS_8_reduced="\
958   0100011000000000000000\
959   0011100100000000000000\
960   1100111001000000000000\
961   0000100011000000000000\
962   0000011100100000000000\
963   0001100110010000000000\
964   0000000100011000000000\
965   0000000011100100000000\
966   0000001100110010000000\
967   0000000000100011000000\
968   0000000000011100100000\
969   0000000001100110010000\
970   0000000000000100011000\
971   0000000000000011100100\
972   0000000000000110011001"
973
974
975   CODE_21_15_4="\
976   1110001000000000000000 \
977   1101000100000000000000 \
978   1011000010000000000000 \
979   0111000001000000000000 \
980   1100100000100000000000 \
981   1010100000010000000000 \
982   0110100000001000000000 \
983   1001100000000100000000 \
984   0101100000000010000000 \
985   0011100000000001000000 \
986   1111100000000000100000 \
987   1100010000000000010000 \
988   1010010000000000001000 \
989   0110010000000000000100 \
990   1001010000000000000010 \
991   0100101000000000000001"
992
993   # there are 5 [15,6,6]
994
995   # ago=12
996   CODE_15_6_6_A="\
997   111111111100000 \
998   111110000010000 \
999   111001100001000 \
1000  110101010000100 \
1001  101010110000010 \
1002  101101001000001"
1003
1004  # ago=12
1005  CODE_15_6_6_B="\
1006  111111111100000 \
1007  111110000010000 \
1008  111001100001000 \
1009  110101010000100 \
1010  101010110000010 \
1011  011011001000001"
1012
1013  #ago=720:
```

```
1014  CODE_15_6_6_C="\
1015  111111111100000 \
1016  111110000010000 \
1017  111001100001000 \
1018  110101010000100 \
1019  101101001000010 \
1020  100010111000001"
1021
1022  #ago=96:
1023  CODE_15_6_6_D="\
1024  111111111100000 \
1025  111110000010000 \
1026  111001100001000 \
1027  110101010000100 \
1028  101011001000010 \
1029  011001011000001"
1030
1031  #ago=360
1032  CODE_15_6_6_E="\
1033  111111111100000 \
1034  111110000010000 \
1035  111001100001000 \
1036  100111010000100 \
1037  010101110000010 \
1038  010110101000001"
1039
1040
1041
1042  # Section 10.9 Bounds
1043
1044
1045  CODE_GV_N15_K6="\
1046  111111111100000\
1047  111110000010000\
1048  111001100001000\
1049  110101010000100\
1050  101010110000010\
1051  101101001000001"
1052
1053
1054  CODE_GV_N15_K6_CHECK="\
1055  100000000111111\
1056  010000000111100\
1057  001000000111011\
1058  000100000110101\
1059  000010000110010\
1060  000001000101101\
1061  000000100101010\
1062  000000010100110\
1063  000000001100001"
1064
1065
1066
1067  # Chapter 11 Combinatorics
1068
1069
1070
1071  # Section 11.2 Combinatorial Objects
1072
```

```
1073  HIRSCHFELD_SURFACE_Q4_SET_OF_POINTS="0,1,2,3,4,5,6,7,8,9,\
1074  10,11,12,13,14,23,26,27,30,31,34,35,38,39,42,47,48,51,52,\
1075  53,54,59,60,61,62,67,68,69,70,75,76,79,80,81,82"
1076
1077
1078  HYPEROVAL_16_144="0, 1, 2, 3, 52, 67, 89, 106, 126, \
1079  141, 159, 176, 184, 199, 220, 235, 245, 262"
1080
1081  HYPEROVAL_16_16320="0, 1, 2, 3, 52, 70, 83, 109, 127, \
1082  139, 156, 174, 186, 199, 217, 229, 256, 264"
1083
1084
1085  # Section 11.4 Classification of Configurations and Geometries
1086
1087
1088  DESARGUES_PATH_CANONICAL_ANCESTOR="10 10 3\n0\n1 0\n2 112 119\n3 89 112 119\n4 11
      4 118 89 82\n5 106 114 69 107 111\n6 85 105 112 99 83 61\n7 94 105 113 85 35 83 6
      0\n8 26 119 55 105 92 79 74 48\n9 119 93 106 15 26 79 55 73 47\n10 0 119 93 106 1
      5 26 79 55 73 47\n-1\n"
1089
1090  GEO_BLOCKS_600="Row,C0,C1,C2\n\
1091  0, 0, 5, 14, \n\
1092  1, 1, 6, 10, \n\
1093  2, 2, 7, 11, \n\
1094  3, 3, 8, 12, \n\
1095  4, 4, 9, 13, \n\
1096  5, 0, 6, 13, \n\
1097  6, 1, 7, 14, \n\
1098  7, 2, 8, 10, \n\
1099  8, 3, 9, 11, \n\
1100  9, 4, 5, 12, \n\
1101  10, 0, 7, 12, \n\
1102  11, 1, 8, 13, \n\
1103  12, 2, 9, 14, \n\
1104  13, 3, 5, 10, \n\
1105  14, 4, 6, 11, \n\
1106  15, 0, 8, 11, \n\
1107  16, 1, 9, 12, \n\
1108  17, 2, 5, 13, \n\
1109  18, 3, 6, 14, \n\
1110  19, 4, 7, 10, \n\
1111  20, 0, 9, 10, \n\
1112  21, 1, 5, 11, \n\
1113  22, 2, 6, 12, \n\
1114  23, 3, 7, 13, \n\
1115  24, 4, 8, 14, \n\
1116  END\n"
1117
1118
1119  LSQ_5A_TABLE="0,1,2,3,4,1,2,3,4,0,2,3,4,0,1,3,4,0,1,2,4,0,1,2,3"
1120
1121  LSQ_5B_TABLE="0,1,2,3,4,1,0,4,2,3,2,4,3,1,0,3,2,0,4,1,4,3,1,0,2"
1122
1123
1124
1125
1126  # Chapter 12 Graph theory
1127
1128  # Section 12.1 Creating graphs
```

```
1129
1130  TRIANGLE_GRAPH="0,1,1\n1,0,1\n1,1,0\n"
1131
1132
1133
1134
1135
1136
1137  # Chapter 13 Design theory
1138
1139
1140
1141
1142  # Section 13.2 Assumed symmetry
1143
1144
1145  GEN_C5="1,2,3,4,0,6,7,8,9,5,11,12,13,14,10"
1146
1147
1148
1149  # Section 13.3 Design Theory - Large sets
1150
1151
1152
1153
1154  AG_2_3_BLOCKS="0,13,22,27,35,41,47,53,55,59,71,76"
1155
1156  LARGE_SET_AG_2_3_NEIGHBOR_SET="129,130,133,134,136,\
1157  139,141,142,153,154,156,160,165,166,178,179,183,184,\
1158  185,190,192,194,197,203,204,206,218,221,222,225,227,\
1159  231,248,251,252,255,256,259,261,262,272,277,279,283,\
1160  285,287,299,301,303,305,306,309,313,315,319,320,323,\
1161  325,341,342,343,344,345,349,368,371,375,378,381,383,\
1162  392,393,397,402,403,405,416,419,421,422,425,426,429,\
1163  430,440,443,447,449,453,454,464,467,468,473,474,479,\
1164  490,493,494,497,500,503,513,517,518,520,523,527,536,\
1165  539,541,542,544,547,548,551,563,566,567,571,572,573,\
1166  585,589,590,593,595,596,600,601,603,611,614,615,625,\
1167  629,631,635,637,638,657,659,661,667,668,671,681,683,\
1168  686,689,691,693,705,706,709,710,712,715,717,718,720,\
1169  723,724,729,733,735,747,748,750,752,754,757,777,780,\
1170  781,784,790,791,802,804,807,808,811,814,824,827,828,\
1171  831,832,835,837,838"
1172
1173
1174
1175  # Section 13.4: Design Theory - Delandtsheer Doyen
1176
1177
1178
1179  PP4=-d1 1 -q1 3 -d2 1 -q2 7 -K 5 -problem_label PP4
1180
1181  PP4_GROUP1=-subgroup "1,1,1,1, " "21" -group_label "cyclic21"
1182
1183  PP4_MASK1=\
1184  ▷  -nb_orbits_on_blocks 1 \
1185  ▷  -mask_label "no_mask"
1186
1187
```

```
1188
1189  DELANDTSHEER_DOYEN_PROBLEM_COLBOURN_COLBOURN_7_13= -d1 1 -q1 7 -d2 1 -q2 13 -K 6
      -search_control -W -end -problem_label DD_CC_7_13
1190
1191  DELANDTSHEER_DOYEN_PROBLEM_COLBOURN_COLBOURN_7_13_GROUP1=-subgroup "1,1,1,1, " "9
      1" -group_label "cyclic91"
1192
1193  DELANDTSHEER_DOYEN_PROBLEM_COLBOURN_COLBOURN_7_13_MASK1=\
1194  ▷  -nb_orbits_on_blocks 3 \
1195  ▷  -depth 6 \
1196  ▷  -mask_label "no_mask"
1197
1198
1199
1200  DELANDTSHEER_DOYEN_PROBLEM_27_53= -d1 1 -q1 27 -d2 1 -q2 53 -K 11 -DDx 2 -DDy 1 -
      search_control -W -end
1201
1202  DELANDTSHEER_DOYEN_PROBLEM_27_53_GROUP1=-subgroup \
1203  ▷  "1,1,1,0, 1,3,1,0, 1,9,1,0, 1,0,1,1, -2,0,-4,0" "18603" -group_label "group1"
1204  # mask 1:
1205  #  XX.
1206  #  X.X+
1207
1208  DELANDTSHEER_DOYEN_PROBLEM_27_53_MASK1=\
1209  ▷  -masktest 3 x ge 1 \
1210  ▷  -masktest 4 x+y ge 3 \
1211  ▷  -depth 4 \
1212  ▷  -mask_label "mask1"
1213
1214  DELANDTSHEER_DOYEN_PROBLEM_3_7= -d1 1 -q1 3 -d2 1 -q2 7 -K 5 -DDx 3 -DDy 1 -searc
      h_control -W -end
1215
1216  DELANDTSHEER_DOYEN_PROBLEM_3_7_GROUP1=-subgroup \
1217  ▷  "1,1,1,0, 1,0,1,1 " "21" -group_label "group_cyclic"
1218
1219  DELANDTSHEER_DOYEN_PROBLEM_3_7_MASK1= -mask_label "mask1" -depth 5
1220
1221
1222
1223
1224
1225
1226  # Chapter 14 Finite geometry
1227
1228  # Section 14.1 Spreads
1229
1230
1231  SPREAD_SET_27_RAO_RAO="\
1232  0,0,0,0,0,0,0,0,0, \
1233  1,1,0,2,1,1,0,0,2, \
1234  1,0,1,1,2,2,0,1,0, \
1235  1,2,2,1,2,0,2,2,2, \
1236  0,0,2,2,2,0,1,2,0, \
1237  1,1,2,0,2,1,2,1,0, \
1238  0,1,0,1,0,1,0,2,1, \
1239  2,0,2,0,0,2,1,1,0, \
1240  2,2,2,0,1,1,0,1,2, \
1241  2,0,0,1,0,2,1,2,1, \
1242  0,2,2,2,2,2,2,0,2, \
```

```
2,1,2,0,2,0,2,0,1, \
0,1,2,2,0,1,0,1,1, \
1,0,0,0,1,0,0,0,1, \
2,1,0,1,2,1,0,2,0, \
0,2,0,0,2,2,1,1,2, \
0,0,1,0,1,2,2,2,1, \
2,0,1,2,2,1,1,0,1, \
0,1,1,1,1,0,1,2,2, \
2,2,0,2,0,0,0,2,2, \
2,1,1,1,1,2,2,1,2, \
2,2,1,2,1,0,2,0,0, \
1,2,0,2,0,2,1,0,0, \
1,2,1,1,0,0,1,1,1, \
0,2,1,1,1,1,2,2,0, \
1,1,1,0,0,1,1,0,2, \
1,0,2,2,1,2,2,1,1 \
"

SPREAD_S27_RAO_RAO="\
0, 33879, 5418, 13103, 30556, 22107, 27225, 4045, 24924, 31961, \
3196, 30100, 28081, 25862, 1339, 6696, 8242, 11747, 14000, 14705, \
9784, 17843, 20772, 9271, 19413, 18678, 16109, 23924"




SPREADS_27_ISO_0="\
0, 33879, 1339, 2678, 3994, 7671, 10180, 5862, 9524, 6852, 22243, \
12745, 24295, 11062, 13615, 23894, 15056, 29367, 16429, 31521, 17726, \
31103, 18887, 26333, 19566, 28400, 21531, 27228"

SPREADS_27_ISO_1="\
0, 33879, 1339, 2678, 3994, 7671, 10182, 5761, 6796, 9327, 15339, \
31914, 24415, 12713, 22748, 11666, 13353, 23555, 30103, 16395, 17827, \
30790, 18254, 26422, 20046, 28112, 20900, 26801"

SPREADS_27_ISO_2="\
0, 33879, 1339, 2678, 3994, 7671, 10182, 5817, 6796, 9276, 23891, \
15368, 11666, 22124, 12713, 24415, 13353, 29619, 15910, 31914, 17030, \
30931, 19213, 26422, 19905, 28112, 21217, 27545"

SPREADS_27_ISO_3="\
0, 33879, 1339, 2678, 3994, 7671, 10625, 6590, 9476, 5576, 24688, \
23043, 10996, 22124, 12723, 13522, 15421, 29894, 16532, 32442, 17997, \
31015, 18311, 26109, 19807, 28113, 21220, 27195"

SPREADS_27_ISO_4="\
0, 33879, 1339, 2678, 3994, 7674, 7051, 10666, 9327, 5419, 19806, \
21332, 22124, 13353, 24415, 12401, 11062, 23717, 15056, 29660, 16395, \
31950, 17873, 31153, 19212, 26221, 28515, 26708"

SPREADS_27_ISO_5="\
0, 33879, 1339, 2678, 3994, 7988, 5333, 6672, 10666, 9327, 11062, \
22124, 13353, 15056, 16395, 17347, 31055, 27061, 24415, 12401, 23076, \
30103, 32394, 19041, 26109, 20380, 28400, 21332"

SPREADS_27_ISO_6="\
0, 33879, 1339, 2679, 3992, 7672, 31140, 6913, 13513, 23167, 5653, \
```

```
10607, 9131, 11225, 22548, 13074, 24645, 15124, 29345, 16226, 31506, \
17684, 18732, 26116, 19458, 28361, 21571, 27680"



# Section 14.3 Packings







# nice generators, from Michael Epstein:

PGL_4_5_SUBGROUP_3B_ME=-PGL 4 5 \
▷  -subgroup_by_generators "3B" 3 1 \
▷  "1,0,0,0, 0,1,0,0, 0,0,2,2, 0,0,4,2"

PGL_4_5_SUBGROUP_3B_ME_NORMALIZER=-PGL 4 5 \
▷  -subgroup_by_generators "normalizer_3B" "5760" 8 \
▷  "1,0,0,0,0,4,0,0,0,0,1,0,0,0,0,1, \
▷  1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,4, \
▷  1,0,0,0,0,4,0,0,0,0,4,0,0,0,0,4, \
▷  1,0,0,0,0,1,0,0,0,0,3,0,0,0,0,3, \
▷  1,0,0,0,0,3,0,0,0,0,1,0,0,0,0,1, \
▷  1,0,0,0,0,1,0,0,0,0,2,4,0,0,2,3, \
▷  1,0,0,0,4,4,0,0,0,0,1,0,0,0,0,1, \
▷  0,1,0,0,1,0,0,0,0,0,4,0,0,0,0,4,"

PGL_4_5_SUBGROUP_31_ME=-PGL 4 5 \
▷  -subgroup_by_generators "31" 31 1 \
▷  "1,0,0,0, 0,3,4,3, 0,3,3,4, 0,3,2,3"

PGL_4_5_SUBGROUP_31_ME_NORMALIZER=-PGL 4 5 \
▷  -subgroup_by_generators "normalizer_31" "372" 4 \
▷  "1,0,0,0,0,4,0,0,0,0,4,0,0,0,0,4, \
▷  1,0,0,0,0,3,0,0,0,0,3,0,0,0,0,3, \
▷  1,0,0,0,0,4,0,0,0,0,2,1,0,3,2,4, \
▷  1,0,0,0,0,0,1,0,0,0,0,1,0,1,1,3,"







# Section 14.4 BLT-Sets


BLT_ORDER_Q="3,5,7,9,11,13,17,19,23,25,27,\
29,31,37,41,43,47,49,53,59,61,67,71,73"

BLT_NUMBER_ISO="1,2,2,3,4,3,6,5,9,6,6,\
9,8,7,10,6,10,8,8,9,5,6,8,5"



# Chapter 15 Computer Science Primitives

# Section 15.2 Diophantine systems

```

```
1361
1362  TEST_SYSTEM="\
1363  0,1,0,1,0,0, \
1364  0,0,1,0,1,0, \
1365  1,0,1,0,0,0, \
1366  0,1,0,1,0,1, \
1367  1,0,0,0,0,1, \
1368  1,0,1,0,0,0, \
1369  0,1,0,0,1,1"
1370
1371
1372
1373
1374  # Chapter 16 Canonical Forms with Nauty
1375
1376  # Section 16.2 Canonical forms of objects in projective space
1377
1378
1379
1380
1381  SUZUKI_8_GENERATORS="\
1382  1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1, \
1383  1,0,0,0,0,6,0,0,0,0,2,0,0,0,0,3,0, \
1384  1,0,0,0,1,1,1,0,0,0,1,0,1,0,0,1,0, \
1385  1,0,0,0,3,6,2,2,5,0,2,0,3,0,6,3,2, \
1386  0,1,0,0,1,0,0,0,0,0,0,1,0,0,1,0,2"
1387  # group order 87360 = 3 * 29120
1388
1389
1390
1391  # Section 16.3 Canonical Forms of Incidence Geometries
1392
1393
1394
1395  FILE_24_3_TFC_INC="24 24 72\
1396  \n0 1 2 24 27 28 48 53 54 73 79 80 97 105 106 122 131 \
1397  132 146 157 158 171 175 183 195 203 208 220 225 233 244 \
1398  258 259 269 272 282 293 300 308 318 325 333 342 352 358 \
1399  367 379 381 392 398 400 417 428 429 442 443 450 466 471 \
1400  479 492 497 502 517 519 521 542 548 551 571 574 575 \
1401               48\
1402  \n0 1 2 24 27 28 48 53 54 73 79 80 97 105 106 122 131 \
1403  132 146 157 158 171 175 183 195 203 208 220 225 233 244 \
1404  258 259 269 272 281 293 301 308 318 324 327 342 354 357 \
1405  367 373 378 392 400 403 417 419 430 442 446 447 466 472 \
1406  479 492 500 503 518 525 526 545 549 551 571 572 574 \
1407               48\
1408  \n0 1 2 24 27 28 48 53 54 73 79 80 97 105 106 122 131 \
1409  132 146 157 158 171 175 179 195 201 207 220 226 232 244 \
1410  257 258 269 274 277 293 300 307 318 323 329 342 352 356 \
1411  367 374 381 392 397 406 416 423 431 441 450 454 468 476 \
1412  477 494 499 503 519 521 525 544 547 550 570 572 575 \
1413              144\
1414  \n-1 3"
1415
1416
1417  # Section 16.5 Canonical forms of linear codes
1418
1419  # consider the binary code with generator matrix:
```

```
# 1 0 1
# 0 1 1

CODE_N3_K2_Q2_GENMA="1,0,1, 0,1,1"

CODE_N6_K3_Q2_GENMA="\
111100\
110010\
101001"



# Chapter 17 Interfaces


# Section 17.2 Poyvray interface

# povray colors:
POLISHED_CHROME_WHITE=\
▷  "texture{ Polished_Chrome pigment{quick_color White} }"

YELLOW_TRANSPARENT=\
▷  "texture{ pigment{ color Yellow transmit 0.7 } \
▷  finish {diffuse 0.9 phong 0.6} }"

COLOR_RED=\
▷  "texture{ pigment{ color Red } \
▷  finish {diffuse 0.9 phong 0.6} }"

COLOR_RED_SHINY=\
▷  "texture{ pigment{ color Red } \
▷  finish { diffuse 0.9 phong 1}}"

COLOR_GREEN_SHINY=\
▷  "texture{ pigment{ color Green } \
▷  finish { diffuse 0.9 phong 1}}"

COLOR_BLUE_SHINY=\
▷  "texture{ pigment{ color Blue } \
▷  finish { diffuse 0.9 phong 1}}"

COLOR_YELLOW_SHINY=\
▷  "texture{ pigment{ color Yellow } \
▷  finish { diffuse 0.9 phong 1}}"

COLOR_BLACK_SHINY=\
▷  "texture{ pigment{ color Black } \
▷  finish { diffuse 0.9 phong 1}}"

COLOR_RED_SEE_THROUGH=\
▷  "texture{ pigment{ color Red transmit 0.5 } \
▷  finish { diffuse 0.9 phong 1}}"

COLOR_GREEN_SEE_THROUGH=\
▷  "texture{ pigment{ color Green transmit 0.5 } \
▷  finish { diffuse 0.9 phong 1}}"

COLOR_BLUE_SEE_THROUGH=\
▷  "texture{ pigment{ color Blue transmit 0.5 } \
```

```
1479  ▷  finish { diffuse 0.9 phong 1}}"
1480
1481  COLOR_YELLOW_SEE_THROUGH=\
1482  ▷  "texture{ pigment{ color Yellow transmit 0.5 } \
1483  ▷  finish { diffuse 0.9 phong 1}}"
1484
1485  COLOR_YELLOW_THICK=\
1486  ▷  "texture{ pigment{ color Yellow } \
1487  ▷  finish { diffuse 0.9 phong 1}}"
1488
1489  COLOR_BLACK_NO_SHADOW=\
1490  ▷  "texture{ pigment{Black} } no_shadow"
1491
1492  SURFACE_COLOR=\
1493  ▷  "texture{ pigment{ White*0.5  } \
1494  ▷  finish {ambient 0.4 diffuse 0.5 roughness 0.001 \
1495  ▷  reflection 0.1 specular .8} }"
1496
1497  SURFACE_COLOR_SEETHROUGH=\
1498  ▷  "texture{ pigment{ White*0.5 transmit 0.5 } \
1499  ▷  finish {ambient 0.4 diffuse 0.5 roughness 0.001 \
1500  ▷  reflection 0.1 specular .8} }"
1501
1502  COLOR_GOLD=\
1503  ▷  "texture{ pigment{ Gold } finish \
1504  ▷  {ambient 0.4 diffuse 0.5 roughness 0.001 \
1505  ▷  reflection 0.1 specular .8} }"
1506
1507  COLOR_TURQUOISE=\
1508  ▷  "texture{ pigment{Cyan*1.3} \
1509  ▷  finish {ambient 0.4 diffuse 0.6 roughness 0.001 \
1510  ▷  reflection 0 specular .8} }"
1511
1512  MONKEY_SADDLE_CUBIC="1,0,0,0,-3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-1,0"
1513
1514  ECKARDT_CUBIC_DEFORM1_LEX="0, 10, 0, -8, 10, \
1515  25, 2, 0, -20, -8, -20, -10, -24, 10, -2, 12, \
1516  0, -8, 8, 16"
1517
1518  ECKARDT_CUBIC_DEFORM2_LEX="0, -5, 0, -5, -5, \
1519  10, -1, 0, 10, 4, 10, 5, 3, -5, 1, -6, 0, -5, \
1520  -4, 1"
1521
1522  KUMMER_QUARTIC_LEX_35="-2,0,0,0,2,0,0,2,0,2,0,0,\
1523  0,0,0,0,0,0,0,0,-2,0,0,2,0,2,0,0,0,0,-2,0,2,0,-2"
1524
1525
1526
1527
1528  BEAUVILLE_QUINTIC_LEX_56="-44, 228, 400, 315, -396, -852, \
1529  -512, -553, -1050, -354, 284, 504, -62, -707, -1390, -1010, \
1530  281, -167, -1644, -1024, -72, -196, 192, 373, 322, 78, 150, \
1531  966, 1540, 348, -475, -492, 1063, 1550, 390, 0, 96, 3, -337, \
1532  -426, -66, 425, 673, -156, -216, -223, -60, 1543, 1998, 618, \
1533  263, -250, -919, 557, 1800, 741"
1534
1535  ENDRASS_OCTIC_LEX_165="-93.2548,0,0,0,-309.019,0,0,527.529,0,395.647,\
1536  0,0,0,0,0,0,0,0,0,0,-687.529,0,0,1582.59,0,1186.94,0,0,0,0,-1055.06,0,\
1537  -1582.59,0,-593.47,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-309.019,\
```

```
1538  0,0,1582.59,0,1186.94,0,0,0,0,-2110.12,0,-3165.17,0,-1186.94,0,0,0,0,0,\
1539  0,874.039,0,1560.63,0,1677.92,0,343.362,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,\
1540  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-93.2548,0,0,527.529,0,395.647,\
1541  0,0,0,0,-1055.06,0,-1582.59,0,-593.47,0,0,0,0,0,0,874.039,0,1560.63,0,\
1542  1677.92,0,343.362,0,0,0,0,0,0,0,0,-256,0,-468.077,0,-789.019,0,\
1543  -525.726,0,0.941125"
1544
1545
1546  # Clebsch map up for surface created using arc lifting
1547  # We take a circle of radius r centered at the origin in the affine real plane
1548  # and map it up on the surface.
1549  # The Clebsch surface has
1550  # a = d = 2.618033988 = (3+sqrt(5))/2
1551  # b = c = 1.618033988 = (1+sqrt(5))/2
1552  #
1553  CLEBSCH_A=2.618033988
1554
1555  CLEBSCH_D=2.618033988
1556
1557  CLEBSCH_B=1.618033988
1558
1559  CLEBSCH_C=1.618033988
1560
1561  TWO_PI=6.283185308
1562
1563
1564  # to go from the arclifting surface to the defining equation:
1565  #Matrix(4, 4, [[-0.44721360215312733, 1.1708204000530853, 1.1708204000530853, -0.
      44721359579991158], [-1.1708204000530853, 0.4472136021531272, 1.4472136021531272,
      0.44721359579991158], [4.2360680044124255, -4.2360680044124255, -4.236068004412425
      5, 0.], [1.6180340022062127, -2.6180340022062127, -1.6180340022062127, 0.]])
1566  #-0.44721360215312733, 1.1708204000530853, 1.1708204000530853, -0.447213595799915
      8
1567  #-1.1708204000530853, 0.4472136021531272, 1.4472136021531272, 0.44721359579991158
1568  #4.2360680044124255, -4.2360680044124255, -4.2360680044124255, 0.
1569  #1.6180340022062127, -2.6180340022062127, -1.6180340022062127, 0.
1570
1571
1572  T00=-0.44721360215312733
1573
1574  T01=1.1708204000530853
1575
1576  T02=1.1708204000530853
1577
1578  T03=-0.44721359579991158
1579
1580  T10=-1.1708204000530853
1581
1582  T11=0.4472136021531272
1583
1584  T12=1.4472136021531272
1585
1586  T13=0.44721359579991158
1587
1588  T20=4.2360680044124255
1589
1590  T21=-4.2360680044124255
1591
1592  T22=-4.2360680044124255
```

```
1593
1594  T23=0.
1595
1596  T30=1.6180340022062127
1597
1598  T31=-2.6180340022062127
1599
1600  T32=-1.6180340022062127
1601
1602  T33=0.
1603
1604
1605  CLEBSCH_CUBICS=\
1606  ▷  push b push b mult push d push c push m mult add mult \
1607  ▷  push b push c push d push d push m mult mult add mult \
1608  ▷  push a push d push d push m add mult mult add add \
1609  ▷  push a push c push m mult add mult \
1610  ▷  store c001 \
1611  ▷  push b push d mult \
1612  ▷  push b push 1 push m push c mult add mult \
1613  ▷  push d push a push 1 push m mult add mult add \
1614  ▷  push m push a mult add push c add \
1615  ▷  push c push m push a mult add \
1616  ▷  mult mult \
1617  ▷  store c002 \
1618  ▷  push b \
1619  ▷  push d push c push a push m mult add mult \
1620  ▷  push c push a push m push 1 mult add mult add mult \
1621  ▷  push a push d mult push c push 1 push m mult add mult \
1622  ▷  push m mult add \
1623  ▷  push a push c push m mult add mult \
1624  ▷  store c011 \
1625  ▷  push b push b push c mult mult \
1626  ▷  push 1 push d push m mult add mult \
1627  ▷  push a push b mult push c push d push d push m mult mult add mult \
1628  ▷  push m mult add \
1629  ▷  push a push d mult push c push d push m mult add mult add \
1630  ▷  push a push c push m mult add mult \
1631  ▷  store c012 \
1632  ▷  push m \
1633  ▷  push b push d push m mult add push c mult \
1634  ▷  push d push b push 1 push m mult add mult push m mult add push a mult \
1635  ▷  push b push c mult push d push 1 push m mult add mult add mult \
1636  ▷  push b push d push m mult add mult \
1637  ▷  store d001 \
1638  ▷  push m \
1639  ▷  push d push c push m mult add push a push a mult mult \
1640  ▷  push c push c mult push d push m mult add push a mult add \
1641  ▷  push m push b push c mult mult push c push 1 push m mult add mult add mult \
1642  ▷  push b push d push m mult add mult \
1643  ▷  store d011 \
1644  ▷  push m \
1645  ▷  push c push d mult push d push m mult add push a push a mult mult \
1646  ▷  push c push c mult push d push m mult add push a push b push m mult mult mult a
      dd \
1647  ▷  push b push d push c push m mult add mult push c push m mult mult add \
1648  ▷  push b push d push m mult add mult mult \
1649  ▷  store d012 \
1650  ▷  push d push 1 push m mult add push a mult push m push b mult push 1 add push c
```

```
      mult add \
1651  ▷  push b add push m push d mult add \
1652  ▷  push a push c mult mult \
1653  ▷  push b push d push m mult add mult \
1654  ▷  store d112 \
1655  ▷  push m \
1656  ▷  push b push d push m mult add push c mult push d push b push 1 push m mult add
      mult \
1657  ▷  push m mult add push a mult push b push c mult push d push 1 push m mult add mu
      lt add \
1658  ▷  push b push d push m mult add mult mult \
1659  ▷  store m002 \
1660  ▷  push m \
1661  ▷  push d push c push m mult add push a push a mult mult \
1662  ▷  push c push c mult push d push m mult add push a mult add \
1663  ▷  push b push c push m mult mult push c push 1 push m mult add mult add \
1664  ▷  push b push d push m mult add mult mult \
1665  ▷  store m012 \
1666  ▷  push m\
1667  ▷  push c push d mult push d push m mult add push a push a mult mult \
1668  ▷  push m push c push c mult push d push m mult add push a push b mult mult mult a
      dd \
1669  ▷  push m push b push d push c push m mult add push c mult mult mult add \
1670  ▷  push b push d push m mult add mult mult \
1671  ▷  store m022 \
1672  ▷  push d push 1 push m mult add push a mult \
1673  ▷  push m push b mult push 1 add push c mult add \
1674  ▷  push b add push m push d mult add \
1675  ▷  push a push c mult mult \
1676  ▷  push b push d push m mult add mult \
1677  ▷  store m122 \
1678  ▷  push m push a mult push c add push d mult push c push a push 1 push m mult add
      mult add \
1679  ▷  push b mult \
1680  ▷  push m push a push d mult mult push c push 1 push m mult add mult add \
1681  ▷  push b push d push m mult add mult \
1682  ▷  store n002 \
1683  ▷  push m \
1684  ▷  push c push d push m mult add push b mult push m push d push c push 1 push m mu
      lt add mult mult add \
1685  ▷  push a mult \
1686  ▷  push b push c mult push d push 1 push m mult add mult add mult \
1687  ▷  push a push b push c push m mult push d push m mult add add add mult \
1688  ▷  store n012 \
1689  ▷  push c push d push m mult add push b mult \
1690  ▷  push m push d push c push 1 push m mult add mult mult add \
1691  ▷  push a mult \
1692  ▷  push b push c mult push d push 1 push m mult add mult add \
1693  ▷  push a push d mult push m push b push c mult mult add mult \
1694  ▷  store n022 \
1695  ▷  push m \
1696  ▷  push c push d push m mult add push b mult \
1697  ▷  push m push d push c push 1 push m mult add mult mult add \
1698  ▷  push a mult \
1699  ▷  push b push c mult push d push 1 push m mult add mult add \
1700  ▷  push m push a mult push c add mult mult \
1701  ▷  store n112 \
1702  ▷  push m \
1703  ▷  push c push d push m mult add push b mult \
```

```
1704  ▷  push m push d push c push 1 push m mult add mult mult add \
1705  ▷  push a mult \
1706  ▷  push b push c mult push d push 1 push m mult add mult add \
1707  ▷  push a push d mult push m push b push c mult mult add mult mult \
1708  ▷  store n122
1709
1710
1711
1712  # Section 17.5 The Gnuplot interface
1713
1714  GNUPLOT_TEST_INPUT="Row,Curve-1,Curve-2\n1,2,4\n2,6,8\n3,9,10\n4,13,17\n5,17,20\n
      6,19,25\n7,23,30\nEND\n"
1715
1716  # Chapter 18 Mathematical Data in Orbiter
1717
1718
1719  # Section 18.1 Mathematical on Cubic Surfaces
1720
1721  # Section 18.2 Mathematical on BLT-sets
1722
1723  BLT_ORDER_Q="3,5,7,9,11,13,17,19,23,25,27,29,31,37,41,43,47,49,53,59,61,67,71,73"

1724
1725  BLT_NUMBER_ISO="1,2,2,3,4,3,6,5,9,6,6,9,8,7,10,6,10,8,8,9,5,6,8,5"
1726
1727
1728
1729
1730  ##############################################################################
1731  # Chapter 2 - Getting Started
1732  ##############################################################################
1733
1734  test_2:
1735  ▷  make test_2_2
1736  ▷  make test_2_3
1737  ▷  make test_2_4
1738  ▷  make test_2_5
1739  ▷  make test_2_6
1740  ▷  make test_2_7
1741  ▷  make test_2_8
1742  ▷  make test_2_9
1743
1744
1745
1746  ##############################################################################
1747  # Section 2.2: Orbiter Session
1748  ▷
1749  ▷
1750  SECTION_ORBITER_SESSION:
1751
1752  test_2_2:
1753  ▷  make test_orbiter_session
1754
1755
1756  test_orbiter_session:
1757  ▷  $(ORBITER)
1758
1759
1760
```

```
1761  ##############################################################################
1762  # Section 2.3: Makefiles and Shell Scripts
1763
1764  SECTION_MAKEFILES_AND_SHELL_SCRIPTS:
1765
1766  test_2_3:
1767
1768
1769  ##############################################################################
1770  # Section 2.4: Objects and Activities
1771
1772  SECTION_OBJECTS_AND_ACTIVITIES:
1773
1774  test_2_4:
1775  ▷  make example_set
1776  ▷  make object_F_2
1777  ▷  make object_PG_3_2
1778  ▷  make vector_ex
1779
1780
1781  example_set:
1782  ▷  $(ORBITER) -v 2 \
1783  ▷  ▷  -define S -set -here "2,3,5,7,11,13" -end \
1784  ▷  ▷  -print_symbols
1785
1786
1787
1788  object_F_2:
1789  ▷  $(ORBITER) -v 3 -define F -finite_field -q 2 -end
1790
1791
1792  object_PG_3_2:
1793  ▷  $(ORBITER) \
1794  ▷  ▷  -define F -finite_field -q 2 -end \
1795  ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end
1796
1797
1798  vector_ex:
1799  ▷  $(ORBITER) -v 2 \
1800  ▷  ▷  -define F -finite_field -q 5 -end \
1801  ▷  ▷  -define v -vector -field F -dense "0,1,2,3,4" -end \
1802  ▷  ▷  -print_symbols
1803
1804
1805
1806
1807
1808
1809
1810  ##############################################################################
1811  # Section 2.5: Mathematical Data
1812
1813  SECTION_MATHEMATICAL_DATA:
1814
1815
1816  test_2_5:
1817  ▷  #make create_BLT_5_1
1818  ▷  make create_surface_4_0
1819
```

```
1820  create_BLT_5_1:
1821  ▷  $(ORBITER) -v 2 \
1822  ▷  ▷  -define F -finite_field -q 5 -end \
1823  ▷  ▷  -define O -orthogonal_space 0 5 F -end \
1824  ▷  ▷  -define BLT -BLT_set \
1825  ▷  ▷  ▷  -space O -catalogue 1 \
1826  ▷  ▷  -end \
1827  ▷  ▷  -with BLT -do -blt_set_activity \
1828  ▷  ▷  ▷  -report \
1829  ▷  ▷  -end
1830  ▷  pdflatex BLT_catalogue_q5_iso1.tex
1831  ▷  $(OPEN) BLT_catalogue_q5_iso1.pdf
1832
1833
1834
1835  create_surface_4_0:
1836  ▷  $(ORBITER) -v 3 \
1837  ▷  ▷  -define F -finite_field -q 4 -end \
1838  ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
1839  ▷  ▷  -define S4_0 -cubic_surface -space P -catalogue 0 -end \
1840  ▷  ▷  -with S4_0 -do \
1841  ▷  ▷  ▷  -cubic_surface_activity \
1842  ▷  ▷  ▷  -report \
1843  ▷  ▷  -end
1844  ▷  pdflatex surface_catalogue_q4_iso0_report.tex
1845  ▷  $(OPEN) surface_catalogue_q4_iso0_report.pdf
1846
1847
1848
1849  ################################################################################
1850  # Section 2.6: Set Builder
1851
1852  SECTION_SET_BUILDER:
1853
1854
1855  test_2_6:
1856  ▷  make set_of_primes
1857  ▷  make set_interval
1858
1859  set_of_primes:
1860  ▷  $(ORBITER) -v 2 \
1861  ▷  ▷  -define S -set -here "2,3,5,7,11,13" -end \
1862  ▷  ▷  -print_symbols
1863
1864  set_interval:
1865  ▷  $(ORBITER) -v 2 -define S -set -loop 0 64 1 -end \
1866  ▷  ▷  -print_symbols
1867
1868
1869
1870
1871  ################################################################################
1872  # Section 2.7: Vector Builder
1873
1874  SECTION_VECTOR_BUILDER:
1875
1876
1877  test_2_7:
1878  ▷  make vector_example1
```

```
1879  ▷  make vector_example2
1880  ▷  make vector_example_repeat
1881  ▷  make vector_example_all_one_11
1882  ▷  make vector_loop_8
1883  ▷  make vector_loop_odd_16
1884  ▷  make vector_concatenate
1885  ▷  make matrix_example1
1886  ▷  make vector_example_sparse
1887  ▷  make matrix_example2
1888  ▷  make matrix_example_co_1
1889
1890
1891  vector_example1:
1892  ▷  $(ORBITER) -v 2 \
1893  ▷  ▷  -define F -finite_field -q 5 -end \
1894  ▷  ▷  -define v -vector -field F -dense "0,1,2,3,4" -end \
1895  ▷  ▷  -print_symbols
1896
1897
1898  vector_example2:
1899  ▷  $(ORBITER) -v 2 \
1900  ▷  ▷  -define F -finite_field -q 5 -end \
1901  ▷  ▷  -define v -vector -field F -format 2 -dense "0,1,2,3,4,0" -end \
1902  ▷  ▷  -print_symbols
1903
1904  vector_example_repeat:
1905  ▷  $(ORBITER) -v 2 \
1906  ▷  ▷  -define v -vector -repeat "0,1,2,3" 11 -end \
1907  ▷  ▷  -print_symbols
1908
1909
1910  vector_example_all_one_11:
1911  ▷  $(ORBITER) -v 2 \
1912  ▷  ▷  -define v -vector -repeat 1 11 -end \
1913  ▷  ▷  -print_symbols
1914
1915  vector_loop_8:
1916  ▷  $(ORBITER) -v 2 \
1917  ▷  ▷  -define v -vector -loop 0 8 1 -end \
1918
1919  vector_loop_odd_16:
1920  ▷  $(ORBITER) -v 2 \
1921  ▷  ▷  -define v -vector -loop 1 16 2 -end \
1922
1923
1924  vector_concatenate:
1925  ▷  $(ORBITER) -v 2 \
1926  ▷  ▷  -define a -vector -compact "1,2,3" -end \
1927  ▷  ▷  -define b -vector -compact "4,5,6" -end \
1928  ▷  ▷  -define c -vector -compact "7,8,9" -end \
1929  ▷  ▷  -define abc -vector -concatenate a,b,c -end
1930
1931
1932  matrix_example1:
1933  ▷  $(ORBITER) -v 2 \
1934  ▷  ▷  -define v -vector -format 2 \
1935  ▷  ▷  ▷  -dense "1,2,3,4,5,6" -end \
1936  ▷  ▷  -print_symbols
1937
```

```
1938
1939  vector_example_sparse:
1940  ▷  $(ORBITER) -v 2 \
1941  ▷  ▷  -define F -finite_field -q 5 -end \
1942  ▷  ▷  -define v -vector -field F -format 4 -sparse 20 "1,0,1,19" -end \
1943  ▷  ▷  -print_symbols
1944
1945
1946
1947  matrix_example2:
1948  ▷  $(ORBITER) -v 2 \
1949  ▷  ▷  -define F -finite_field -q 2 -end \
1950  ▷  ▷  -define v -vector -field F -format 4 \
1951  ▷  ▷  ▷  -dense $(HAMMING_CODE_GENERATOR) -end \
1952  ▷  ▷  -print_symbols
1953
1954
1955
1956
1957  matrix_example_co_1:
1958  ▷  $(ORBITER) -v 2 \
1959  ▷  ▷  -define F -finite_field -q 2 -end \
1960  ▷  ▷  -define v -vector -field F -format 22 \
1961  ▷  ▷  ▷  -compact $(CONWAY_GEN1) -end \
1962  ▷  ▷  -print_symbols
1963
1964
1965
1966
1967
1968  ###############################################################################
1969  # Section 2.8: Vector of Group Elements Builder
1970
1971  SECTION_VECTOR_OF_GROUP_ELEMENTS_BUILDER:
1972
1973
1974  test_2_8:
1975  ▷  make Hirschfeld_Class_2A
1976  ▷  make surface_4_0_export_action
1977
1978
1979  Hirschfeld_Class_2A:
1980  ▷  $(ORBITER) -v 9 \
1981  ▷  ▷  -orbiter_path $(ORBITER_PATH) \
1982  ▷  ▷  -define G -linear_group -PGGL 4 4 \
1983  ▷  ▷  ▷  -subgroup_by_generators "Hirschfeld_Stab" \
1984  ▷  ▷  ▷  51840 6 $(HIRSCHFELD_STAB_GENERATORS) \
1985  ▷  ▷  ▷  -end \
1986  ▷  ▷  -with G -do \
1987  ▷  ▷  -group_theoretic_activity \
1988  ▷  ▷  ▷  -conjugacy_class_of "2A" $(CLASS_2A_REP) \
1989  ▷  ▷  -end
1990
1991
1992
1993
1994
1995  surface_4_0_export_action:
1996  ▷  $(ORBITER) -v 3 \
```

```
1997  ▷  ▷  -define Class2A -vector -file $(FILE_NAME_CLASS_2A) -end \
1998  ▷  ▷  -define F -finite_field -q 4 -end \
1999  ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
2000  ▷  ▷  -define S -cubic_surface -space P -catalogue 0 -end \
2001  ▷  ▷  -define G -linear_group -PGGL 4 4 -end \
2002  ▷  ▷  -define gens_ge -vector_ge \
2003  ▷  ▷  ▷  -action G \
2004  ▷  ▷  ▷  -vector_data Class2A \
2005  ▷  ▷  -end \
2006  ▷  ▷  -with S -do \
2007  ▷  ▷  -cubic_surface_activity \
2008  ▷  ▷  ▷  -export_something_with_group_element "action_on_tritangent_planes" gens_ge
      \
2009  ▷  ▷  -end \
2010  ▷  ▷  -with S -do \
2011  ▷  ▷  -cubic_surface_activity \
2012  ▷  ▷  ▷  -export_something_with_group_element "action_on_double_sixes" gens_ge \
2013  ▷  ▷  -end \
2014  ▷  ▷  -with S -do \
2015  ▷  ▷  -cubic_surface_activity \
2016  ▷  ▷  ▷  -export_something_with_group_element "action_on_lines" gens_ge \
2017  ▷  ▷  -end \
2018
2019
2020  ##############################################################################
2021  # Section 2.9: Symbolic Algebra
2022
2023  SECTION_FORMULA_BUILDER:
2024
2025  test_2_9:
2026  ▷  make symbolic_poly
2027  ▷  make symbolic_CRC32
2028  ▷  make symbolic_CRC_alfa
2029  ▷  make symbolic_CRC_alfa_draw_trees
2030  ▷  make symbolic_CRC_bravo
2031  ▷  make symbolic_CRC_bravo_draw_trees
2032  ▷  make symbolic_test_4_expand
2033  ▷  make symbolic_test_4_expand_draw
2034  ▷  make symbolic_test_5_expand
2035  ▷  make test_expand_1
2036  ▷  make test_expand_2
2037  ▷  make test_expand_3
2038  ▷  make test_expand_4
2039  ▷  make symbolic_objects_building_on_earlier_objects
2040  ▷  make symbolic_objects_self_referential_1
2041  ▷  make symbolic_objects_self_referential_2
2042  ▷  make symbolic_test_6
2043  ▷  make symbolic_matrix1
2044  ▷  make symbolic_matrix_2x2
2045  ▷  make symbolic_matrix_3x3
2046  ▷  make symbolic_determinant_2x2
2047  ▷  make symbolic_determinant_3x3
2048  ▷  make symbolic_determinant_vandermonde
2049  ▷  make symbolic_characteristic_polynomial_2x2
2050  ▷  make symbolic_characteristic_polynomial_2x2_draw
2051  ▷  make symbolic_characteristic_polynomial_3x3
2052  ▷  make symbolic_object1
2053  ▷  make symbolic_object1_expand
2054  ▷  make symbolic_object1_draw
```

```
2055  ▷  make symbolic_object1_evaluate
2056  ▷  make symbolic_Eckardt_surface
2057  ▷  make symbolic_Eckardt_surface_q13_a3_b1
2058  ▷  make symbolic_Fabcd
2059  ▷  make symbolic_Fabcd_expand
2060  ▷  make symbolic_Fabcd_q31_E18
2061  ▷  make symbolic_Fabcd_q31_E18_with_implicit_substitution
2062  ▷  make symbolic_Fabcd_q31_E18_with_implicit_substitution_do_not_simplify
2063  ▷  make symbolic_matrix_minor
2064  ▷  make symbolic_matrix_nullspace_1
2065  ▷  make symbolic_matrix_nullspace_2
2066  ▷  make symbolic_Clebsch
2067  ▷  make Clebsch_map_up_Fabcd
2068
2069
2070
2071
2072
2073
2074  symbolic_poly:
2075  ▷  $(ORBITER) -v 3 \
2076  ▷  ▷  -define F -finite_field -q 7 -end \
2077  ▷  ▷  -define M -symbolic_object \
2078  ▷  ▷  ▷  -field F \
2079  ▷  ▷  ▷  -text "-X^2-X+1"\
2080  ▷  ▷  -end \
2081
2082
2083
2084  symbolic_CRC32:
2085  ▷  $(ORBITER) -v 3 \
2086  ▷  ▷  -define F -finite_field -q 2 -end \
2087  ▷  ▷  -define M -symbolic_object \
2088  ▷  ▷  ▷  -field F \
2089  ▷  ▷  ▷  -text $(CRC32_ETHERNET_POLY) \
2090  ▷  ▷  -end \
2091
2092
2093  symbolic_CRC_alfa:
2094  ▷  $(ORBITER) -v 3 \
2095  ▷  ▷  -define F -finite_field -q 256 -end \
2096  ▷  ▷  -define M -symbolic_object \
2097  ▷  ▷  ▷  -field F \
2098  ▷  ▷  ▷  -text $(BCH_POLYNOMIAL_ALFA_F256) \
2099  ▷  ▷  -end \
2100
2101
2102
2103
2104  symbolic_CRC_alfa_draw_trees:
2105  ▷  $(ORBITER) -v 3 \
2106  ▷  ▷  -define F -finite_field -q 256 -end \
2107  ▷  ▷  -define M -symbolic_object \
2108  ▷  ▷  ▷  -field F \
2109  ▷  ▷  ▷  -text $(BCH_POLYNOMIAL_ALFA_F256) \
2110  ▷  ▷  -end \
2111  ▷  ▷  -define Mx -symbolic_object \
2112  ▷  ▷  ▷  -field F \
2113  ▷  ▷  ▷  -expand M \
```

```
2114  ▷  ▷  ▷  -write_trees_during_expand \
2115  ▷  ▷  -end
2116  ▷  ▷  pdflatex Mx_0_efsxsfscsc.tex
2117  ▷  ▷  $(OPEN) Mx_0_efsxsfscsc.pdf
2118
2119
2120  symbolic_CRC_bravo:
2121  ▷  $(ORBITER) -v 3 \
2122  ▷  ▷  -define F -finite_field -q 256 -end \
2123  ▷  ▷  -define M -symbolic_object \
2124  ▷  ▷  ▷  -field F \
2125  ▷  ▷  ▷  -text $(BCH_POLYNOMIAL_BRAVO_F256) \
2126  ▷  ▷  -end \
2127
2128  symbolic_CRC_bravo_draw_trees:
2129  ▷  $(ORBITER) -v 3 \
2130  ▷  ▷  -define F -finite_field -q 256 -end \
2131  ▷  ▷  -define M -symbolic_object \
2132  ▷  ▷  ▷  -field F \
2133  ▷  ▷  ▷  -text $(BCH_POLYNOMIAL_BRAVO_F256) \
2134  ▷  ▷  -end \
2135  ▷  ▷  -define Mx -symbolic_object \
2136  ▷  ▷  ▷  -field F \
2137  ▷  ▷  ▷  -expand M \
2138  ▷  ▷  ▷  -write_trees_during_expand \
2139  ▷  ▷  -end
2140  ▷  ▷  pdflatex Mx_0_efsxsfscsc.tex
2141  ▷  ▷  $(OPEN) Mx_0_efsxsfscsc.pdf
2142
2143
2144  symbolic_test_4_expand:
2145  ▷  $(ORBITER) -v 3 \
2146  ▷  ▷  -define F -finite_field -q 13 -end \
2147  ▷  ▷  -define M -symbolic_object \
2148  ▷  ▷  ▷  -field F \
2149  ▷  ▷  ▷  -matrix 1 \
2150  ▷  ▷  ▷  -text "(a+b)^2" \
2151  ▷  ▷  -end \
2152  ▷  ▷  -define M1 -symbolic_object \
2153  ▷  ▷  ▷  -field F \
2154  ▷  ▷  ▷  -expand M \
2155  ▷  ▷  ▷  -write_trees_during_expand \
2156  ▷  ▷  -end
2157
2158  symbolic_test_4_expand_draw:
2159  ▷  pdflatex M.tex
2160  ▷  $(OPEN) M.pdf
2161  ▷  pdflatex M1.tex
2162  ▷  $(OPEN) M1.pdf
2163  ▷  pdflatex M1_0.tex
2164  ▷  $(OPEN) M1_0.pdf
2165  ▷  pdflatex M1_0_e.tex
2166  ▷  $(OPEN) M1_0_e.pdf
2167  ▷  pdflatex M1_0_ef.tex
2168  ▷  $(OPEN) M1_0_ef.pdf
2169  ▷  pdflatex M1_0_efs.tex
2170  ▷  $(OPEN) M1_0_efs.pdf
2171  ▷  pdflatex M1_0_efsx.tex
2172  ▷  $(OPEN) M1_0_efsx.pdf
```

```
2173  ▷  pdflatex M1_0_efsxs.tex
2174  ▷  $(OPEN) M1_0_efsxs.pdf
2175  ▷  pdflatex M1_0_efsxsf.tex
2176  ▷  $(OPEN) M1_0_efsxsf.pdf
2177  ▷  pdflatex M1_0_efsxsfs.tex
2178  ▷  $(OPEN) M1_0_efsxsfs.pdf
2179  ▷  pdflatex M1_0_efsxsfsc.tex
2180  ▷  $(OPEN) M1_0_efsxsfsc.pdf
2181  ▷  pdflatex M1_0_efsxsfscs.tex
2182  ▷  $(OPEN) M1_0_efsxsfscs.pdf
2183  ▷  pdflatex M1_0_efsxsfscsc.tex
2184  ▷  $(OPEN) M1_0_efsxsfscsc.pdf
2185
2186
2187  symbolic_test_5_expand:
2188  ▷  $(ORBITER) -v 3 \
2189  ▷  ▷  -define F -finite_field -q 13 -end \
2190  ▷  ▷  -define M -symbolic_object \
2191  ▷  ▷  ▷  -field F \
2192  ▷  ▷  ▷  -text "(a+b)^3" \
2193  ▷  ▷  -end \
2194  ▷  ▷  -define M1 -symbolic_object \
2195  ▷  ▷  ▷  -field F \
2196  ▷  ▷  ▷  -expand M \
2197  ▷  ▷  -end
2198  ▷  pdflatex M.tex
2199  ▷  $(OPEN) M.pdf
2200  ▷  pdflatex M1.tex
2201  ▷  $(OPEN) M1.pdf
2202
2203
2204
2205  test_expand_1:
2206  ▷  $(ORBITER) -v 3 \
2207  ▷  ▷  -define Fq -finite_field -q 31 -end \
2208  ▷  ▷  -define f -symbolic_object \
2209  ▷  ▷  ▷  -field Fq \
2210  ▷  ▷  ▷  -text "a*(x+y)" \
2211  ▷  ▷  -end \
2212  ▷  ▷  -define fe -symbolic_object \
2213  ▷  ▷  ▷  -field Fq \
2214  ▷  ▷  ▷  -expand f \
2215  ▷  ▷  ▷  -write_trees_during_expand \
2216  ▷  ▷  -end
2217  ▷  pdflatex fe_0_efsxsfscsc.tex
2218  ▷  $(OPEN) fe_0_efsxsfscsc.pdf
2219
2220  test_expand_2:
2221  ▷  $(ORBITER) -v 3 \
2222  ▷  ▷  -define Fq -finite_field -q 31 -end \
2223  ▷  ▷  -define f -symbolic_object \
2224  ▷  ▷  ▷  -field Fq \
2225  ▷  ▷  ▷  -text "a*(-(x-y))" \
2226  ▷  ▷  -end \
2227  ▷  ▷  -define fe -symbolic_object \
2228  ▷  ▷  ▷  -field Fq \
2229  ▷  ▷  ▷  -expand f \
2230  ▷  ▷  ▷  -write_trees_during_expand \
2231  ▷  ▷  -end
```

```
2232  ▷  pdflatex fe_0_efsxsfscsc.tex
2233  ▷  $(OPEN) fe_0_efsxsfscsc.pdf
2234
2235  test_expand_3:
2236  ▷  $(ORBITER) -v 3 \
2237  ▷  ▷  -define Fq -finite_field -q 31 -end \
2238  ▷  ▷  -define f -symbolic_object \
2239  ▷  ▷  ▷  -field Fq \
2240  ▷  ▷  ▷  -text "a*(-(x-y)+(x+y))" \
2241  ▷  ▷  -end \
2242  ▷  ▷  -define fe -symbolic_object \
2243  ▷  ▷  ▷  -field Fq \
2244  ▷  ▷  ▷  -expand f \
2245  ▷  ▷  ▷  -write_trees_during_expand \
2246  ▷  ▷  -end
2247  ▷  pdflatex fe_0_efs.tex
2248  ▷  $(OPEN) fe_0_efs.pdf
2249  ▷  pdflatex fe_0_efsx.tex
2250  ▷  $(OPEN) fe_0_efsx.pdf
2251  ▷  pdflatex fe_0_efsxsf.tex
2252  ▷  $(OPEN) fe_0_efsxsf.pdf
2253  ▷  pdflatex fe_0_efsxsfs.tex
2254  ▷  $(OPEN) fe_0_efsxsfs.pdf
2255  ▷  pdflatex fe_0_efsxsfsc.tex
2256  ▷  $(OPEN) fe_0_efsxsfsc.pdf
2257  ▷  pdflatex fe_0_efsxsfscsefsx.tex
2258  ▷  $(OPEN) fe_0_efsxsfscsefsx.pdf
2259  ▷  pdflatex fe_0_efsxsfscsefsxsf.tex
2260  ▷  $(OPEN) fe_0_efsxsfscsefsxsf.pdf
2261  ▷  pdflatex fe_0_efsxsfscsefsxsfscsc.tex
2262  ▷  $(OPEN) fe_0_efsxsfscsefsxsfscsc.pdf
2263
2264
2265  test_expand_4:
2266  ▷  $(ORBITER) -v 3 \
2267  ▷  ▷  -define Fq -finite_field -q 31 -end \
2268  ▷  ▷  -define f -symbolic_object \
2269  ▷  ▷  ▷  -field Fq \
2270  ▷  ▷  ▷  -text "x*(a+(x*(a+(x*(a+x*(a+x*(a+x*(a+x)))))))" \
2271  ▷  ▷  -end \
2272  ▷  ▷  -define fx -symbolic_object \
2273  ▷  ▷  ▷  -field Fq \
2274  ▷  ▷  ▷  -expand f \
2275  ▷  ▷  ▷  -write_trees_during_expand \
2276  ▷  ▷  -end
2277  ▷  pdflatex fx_0_efsxsfscsefsxsfscsefsxsfscsefsxsfscsefsxsfscsefsxsfscsc.tex
2278  ▷  $(OPEN) fx_0_efsxsfscsefsxsfscsefsxsfscsefsxsfscsefsxsfscsefsxsfscsc.pdf
2279
2280
2281  symbolic_objects_building_on_earlier_objects:
2282  ▷  $(ORBITER) -v 3 \
2283  ▷  ▷  -define F -finite_field -q 2 -end \
2284  ▷  ▷  -define a -symbolic_object \
2285  ▷  ▷  ▷  -field F \
2286  ▷  ▷  ▷  -text "x+y"\
2287  ▷  ▷  -end \
2288  ▷  ▷  -define b -symbolic_object \
2289  ▷  ▷  ▷  -field F \
2290  ▷  ▷  ▷  -text "a+z"\
```

```
2291  ▷  ▷  -end \
2292
2293
2294  symbolic_objects_self_referential_1:
2295  ▷  $(ORBITER) -v 3 \
2296  ▷  ▷  -define F -finite_field -q 2 -end \
2297  ▷  ▷  -define x -symbolic_object \
2298  ▷  ▷  ▷  -field F \
2299  ▷  ▷  ▷  -text "x"\
2300  ▷  ▷  -end \
2301
2302  symbolic_objects_self_referential_2:
2303  ▷  $(ORBITER) -v 3 \
2304  ▷  ▷  -define F -finite_field -q 2 -end \
2305  ▷  ▷  -define x -symbolic_object \
2306  ▷  ▷  ▷  -field F \
2307  ▷  ▷  ▷  -text "x"\
2308  ▷  ▷  -end \
2309  ▷  ▷  -define a -symbolic_object \
2310  ▷  ▷  ▷  -field F \
2311  ▷  ▷  ▷  -text "x+a"\
2312  ▷  ▷  -end \
2313
2314
2315  symbolic_test_6:
2316  ▷  $(ORBITER) -v 3 \
2317  ▷  ▷  -define F -finite_field -q 13 -end \
2318  ▷  ▷  -define a -symbolic_object \
2319  ▷  ▷  ▷  -field F \
2320  ▷  ▷  ▷  -text "x" \
2321  ▷  ▷  -end \
2322  ▷  ▷  -define a -symbolic_object \
2323  ▷  ▷  ▷  -field F \
2324  ▷  ▷  ▷  -text "a^2" \
2325  ▷  ▷  -end \
2326  ▷  ▷  -define a -symbolic_object \
2327  ▷  ▷  ▷  -field F \
2328  ▷  ▷  ▷  -text "a^2" \
2329  ▷  ▷  -end \
2330  ▷  ▷  -define a -symbolic_object \
2331  ▷  ▷  ▷  -field F \
2332  ▷  ▷  ▷  -text "a^2" \
2333  ▷  ▷  -end
2334  ▷  pdflatex a.tex
2335  ▷  $(OPEN) a.pdf
2336
2337
2338
2339  symbolic_test_7:
2340  ▷  $(ORBITER) -v 10 \
2341  ▷  ▷  -define Fq -finite_field -q 13 -end \
2342  ▷  ▷  -define M -symbolic_object \
2343  ▷  ▷  ▷  -field Fq \
2344  ▷  ▷  ▷  -text "a*b^-1" \
2345  ▷  ▷  -end \
2346  ▷  ▷  -define values -symbolic_object \
2347  ▷  ▷  ▷  -field Fq \
2348  ▷  ▷  ▷  -text "1,3" \
2349  ▷  ▷  -end \
```

```
2350  ▷  ▷  -define eval -symbolic_object \
2351  ▷  ▷  ▷  -field Fq \
2352  ▷  ▷  ▷  -substitute "a,b" M values \
2353  ▷  ▷  -end
2354  ▷  #pdflatex M.tex
2355  ▷  #$(OPEN) M.pdf
2356
2357
2358
2359  symbolic_matrix1:
2360  ▷  $(ORBITER) -v 3 \
2361  ▷  ▷  -define F -finite_field -q 13 -end \
2362  ▷  ▷  -define M -symbolic_object \
2363  ▷  ▷  ▷  -field F \
2364  ▷  ▷  ▷  -matrix 4 \
2365  ▷  ▷  ▷  -text $(VANDERMONDE_4X4_FORMULA) \
2366  ▷  ▷  -end
2367
2368
2369  symbolic_matrix_2x2:
2370  ▷  $(ORBITER) -v 3 \
2371  ▷  ▷  -define F -finite_field -q 13 -end \
2372  ▷  ▷  -define M -symbolic_object \
2373  ▷  ▷  ▷  -field F \
2374  ▷  ▷  ▷  -matrix 2 \
2375  ▷  ▷  ▷  -text "a,b,c,d" \
2376  ▷  ▷  -end
2377
2378
2379  symbolic_matrix_3x3:
2380  ▷  $(ORBITER) -v 3 \
2381  ▷  ▷  -define F -finite_field -q 13 -end \
2382  ▷  ▷  -define M -symbolic_object \
2383  ▷  ▷  ▷  -field F \
2384  ▷  ▷  ▷  -matrix 3 \
2385  ▷  ▷  ▷  -text "a,b,c,d,e,f,g,h,i" \
2386  ▷  ▷  -end
2387
2388
2389  symbolic_determinant_2x2:
2390  ▷  $(ORBITER) -v 3 \
2391  ▷  ▷  -define F -finite_field -q 13 -end \
2392  ▷  ▷  -define M -symbolic_object \
2393  ▷  ▷  ▷  -field F \
2394  ▷  ▷  ▷  -matrix 2 \
2395  ▷  ▷  ▷  -text "a,b,c,d" \
2396  ▷  ▷  -end \
2397  ▷  ▷  -define det -symbolic_object \
2398  ▷  ▷  ▷  -field F \
2399  ▷  ▷  ▷  -determinant M \
2400  ▷  ▷  -end \
2401  ▷  ▷  -define det1 -symbolic_object \
2402  ▷  ▷  ▷  -field F \
2403  ▷  ▷  ▷  -expand det \
2404  ▷  ▷  -end
2405  ▷  pdflatex det1.tex
2406  ▷  $(OPEN) det1.pdf
2407
2408
```

```
2409  symbolic_determinant_3x3:
2410  ▷  $(ORBITER) -v 3 \
2411  ▷  ▷  -define F -finite_field -q 13 -end \
2412  ▷  ▷  -define M -symbolic_object \
2413  ▷  ▷  ▷  -field F \
2414  ▷  ▷  ▷  -matrix 3 \
2415  ▷  ▷  ▷  -text "a,b,c,d,e,f,g,h,i" \
2416  ▷  ▷  -end \
2417  ▷  ▷  -define det -symbolic_object \
2418  ▷  ▷  ▷  -field F \
2419  ▷  ▷  ▷  -determinant M \
2420  ▷  ▷  -end \
2421  ▷  ▷  -define det1 -symbolic_object \
2422  ▷  ▷  ▷  -field F \
2423  ▷  ▷  ▷  -expand det \
2424  ▷  ▷  -end
2425  ▷  pdflatex det1.tex
2426  ▷  $(OPEN) det1.pdf
2427
2428
2429  symbolic_determinant_vandermonde:
2430  ▷  $(ORBITER) -v 3 \
2431  ▷  ▷  -define F -finite_field -q 13 -end \
2432  ▷  ▷  -define M -symbolic_object \
2433  ▷  ▷  ▷  -field F \
2434  ▷  ▷  ▷  -matrix 4 \
2435  ▷  ▷  ▷  -text $(VANDERMONDE_4X4_FORMULA) \
2436  ▷  ▷  -end \
2437  ▷  ▷  -define det -symbolic_object \
2438  ▷  ▷  ▷  -field F \
2439  ▷  ▷  ▷  -determinant M \
2440  ▷  ▷  -end \
2441  ▷  ▷  -define det1 -symbolic_object \
2442  ▷  ▷  ▷  -field F \
2443  ▷  ▷  ▷  -expand det \
2444  ▷  ▷  -end
2445
2446
2447  symbolic_characteristic_polynomial_2x2:
2448  ▷  $(ORBITER) -v 3 \
2449  ▷  ▷  -define F -finite_field -q 13 -end \
2450  ▷  ▷  -define M -symbolic_object \
2451  ▷  ▷  ▷  -field F \
2452  ▷  ▷  ▷  -matrix 2 \
2453  ▷  ▷  ▷  -text "a,b,c,d" \
2454  ▷  ▷  -end \
2455  ▷  ▷  -define p -symbolic_object \
2456  ▷  ▷  ▷  -field F \
2457  ▷  ▷  ▷  -characteristic_polynomial lambda M \
2458  ▷  ▷  -end \
2459  ▷  ▷  -define p1 -symbolic_object \
2460  ▷  ▷  ▷  -field F \
2461  ▷  ▷  ▷  -expand p \
2462  ▷  ▷  ▷  -write_trees_during_expand \
2463  ▷  ▷  -end \
2464  ▷  ▷  -define coeffs -symbolic_object \
2465  ▷  ▷  ▷  -field F \
2466  ▷  ▷  ▷  -collect p1 lambda \
2467  ▷  ▷  -end
```

```
2468
2469  symbolic_characteristic_polynomial_2x2_draw:
2470  ▷  pdflatex p1.tex
2471  ▷  $(OPEN) p1.pdf
2472  ▷  pdflatex p1_0.tex
2473  ▷  $(OPEN) p1_0.pdf
2474  ▷  pdflatex p1_0_e.tex
2475  ▷  $(OPEN) p1_0_e.pdf
2476  ▷  pdflatex p1_0_ef.tex
2477  ▷  $(OPEN) p1_0_ef.pdf
2478  ▷  pdflatex p1_0_efs.tex
2479  ▷  $(OPEN) p1_0_efs.pdf
2480  ▷  pdflatex p1_0_efsx.tex
2481  ▷  $(OPEN) p1_0_efsx.pdf
2482  ▷  pdflatex p1_0_efsxs.tex
2483  ▷  $(OPEN) p1_0_efsxs.pdf
2484  ▷  pdflatex p1_0_efsxsf.tex
2485  ▷  $(OPEN) p1_0_efsxsf.pdf
2486  ▷  pdflatex p1_0_efsxsfs.tex
2487  ▷  $(OPEN) p1_0_efsxsfs.pdf
2488  ▷  pdflatex p1_0_efsxsfsc.tex
2489  ▷  $(OPEN) p1_0_efsxsfsc.pdf
2490  ▷  pdflatex p1_0_efsxsfscs.tex
2491  ▷  $(OPEN) p1_0_efsxsfscs.pdf
2492  ▷  pdflatex p1_0_efsxsfscsc.tex
2493  ▷  $(OPEN) p1_0_efsxsfscsc.pdf
2494
2495  symbolic_characteristic_polynomial_3x3:
2496  ▷  $(ORBITER) -v 3 \
2497  ▷  ▷  -define F -finite_field -q 13 -end \
2498  ▷  ▷  -define M -symbolic_object \
2499  ▷  ▷  ▷  -field F \
2500  ▷  ▷  ▷  -matrix 3 \
2501  ▷  ▷  ▷  -text "a,b,c,d,e,f,g,h,i" \
2502  ▷  ▷  -end \
2503  ▷  ▷  -define p -symbolic_object \
2504  ▷  ▷  ▷  -field F \
2505  ▷  ▷  ▷  -characteristic_polynomial lambda M \
2506  ▷  ▷  -end \
2507  ▷  ▷  -define p1 -symbolic_object \
2508  ▷  ▷  ▷  -field F \
2509  ▷  ▷  ▷  -expand p \
2510  ▷  ▷  -end \
2511  ▷  ▷  -define coeffs -symbolic_object \
2512  ▷  ▷  ▷  -field F \
2513  ▷  ▷  ▷  -collect p1 lambda \
2514  ▷  ▷  -end
2515  ▷  #pdflatex p1.tex
2516  ▷  #$(OPEN) p1.pdf
2517
2518
2519
2520  symbolic_object1:
2521  ▷  $(ORBITER) -v 3 \
2522  ▷  ▷  -define F -finite_field -q 13 -end \
2523  ▷  ▷  -define M -symbolic_object \
2524  ▷  ▷  ▷  -field F \
2525  ▷  ▷  ▷  -text "a+2*b^2*(x-y)+3*-(2*c^4+3*d-(a*b*c)^2)^3" \
2526  ▷  ▷  -end
```

```
2527
2528
2529  symbolic_object1_expand:
2530  ▷  $(ORBITER) -v 3 \
2531  ▷  ▷  -define F -finite_field -q 13 -end \
2532  ▷  ▷  -define M -symbolic_object \
2533  ▷  ▷  ▷  -field F \
2534  ▷  ▷  ▷  -text "a+2*b^2*(x-y)+3*-(2*c^4+3*d-(a*b*c)^2)^3" \
2535  ▷  ▷  -end \
2536  ▷  ▷  -define M1 -symbolic_object \
2537  ▷  ▷  ▷  -field F \
2538  ▷  ▷  ▷  -expand M \
2539  ▷  ▷  -end
2540  ▷  pdflatex M1.tex
2541  ▷  $(OPEN) M1.pdf
2542
2543
2544
2545  symbolic_object1_draw:
2546  ▷  $(ORBITER) -v 3 \
2547  ▷  ▷  -define F -finite_field -q 13 -end \
2548  ▷  ▷  -define M -symbolic_object \
2549  ▷  ▷  ▷  -field F \
2550  ▷  ▷  ▷  -text "a+2*b^2*(x-y)+3*-(2*c^4+3*d-(a*b*c)^2)^3" \
2551  ▷  ▷  -end
2552  ▷  pdflatex sajeeb_tree0.tex
2553  ▷  $(OPEN) sajeeb_tree0.pdf
2554  ▷  pdflatex M.tex
2555  ▷  $(OPEN) M.pdf
2556
2557
2558
2559  symbolic_object1_evaluate:
2560  ▷  $(ORBITER) -v 3 \
2561  ▷  ▷  -define Fq -finite_field -q 13 -end \
2562  ▷  ▷  -define f -symbolic_object \
2563  ▷  ▷  ▷  -field Fq \
2564  ▷  ▷  ▷  -text "a+2*b^2*(x-y)+3*-(2*c^4+3*d-(a*b*c)^2)^3" \
2565  ▷  ▷  -end \
2566  ▷  ▷  -define assignment -symbolic_object \
2567  ▷  ▷  ▷  -field Fq \
2568  ▷  ▷  ▷  -text "1,2,3,4,5,6" \
2569  ▷  ▷  -end \
2570  ▷  ▷  -define eval -symbolic_object \
2571  ▷  ▷  ▷  -field Fq \
2572  ▷  ▷  ▷  -substitute "a,b,c,d,x,y" f assignment \
2573  ▷  ▷  -end \
2574
2575
2576
2577
2578
2579  symbolic_Eckardt_surface:
2580  ▷  $(ORBITER) -v 3 \
2581  ▷  ▷  -define F -finite_field -q 13 -end \
2582  ▷  ▷  -define M -symbolic_object \
2583  ▷  ▷  ▷  -field F \
2584  ▷  ▷  ▷  -managed_variables "X0,X1,X2,X3" \
2585  ▷  ▷  ▷  -text $(ECKARDT_SURFACE_EQN) \
```

```
2586  ▷  ▷  -end
2587
2588
2589
2590  symbolic_Eckardt_surface_q13_a3_b1:
2591  ▷  $(ORBITER) -v 3 \
2592  ▷  ▷  -define F -finite_field -q 13 -end \
2593  ▷  ▷  -define M -symbolic_object \
2594  ▷  ▷  ▷  -field F \
2595  ▷  ▷  ▷  -managed_variables "X0,X1,X2,X3" \
2596  ▷  ▷  ▷  -text $(ECKARDT_SURFACE_EQN) \
2597  ▷  ▷  -end \
2598  ▷  ▷  -define L -symbolic_object \
2599  ▷  ▷  ▷  -field F \
2600  ▷  ▷  ▷  -text "3,1" \
2601  ▷  ▷  -end \
2602  ▷  ▷  -define M1 -symbolic_object \
2603  ▷  ▷  ▷  -field F \
2604  ▷  ▷  ▷  -substitute "a,b" M L \
2605  ▷  ▷  -end \
2606  ▷  ▷  -define M2 -symbolic_object \
2607  ▷  ▷  ▷  -field F \
2608  ▷  ▷  ▷  -expand M1 \
2609  ▷  ▷  ▷  -write_trees_during_expand \
2610  ▷  ▷  -end
2611  ▷  pdflatex M2.tex
2612  ▷  $(OPEN) M2.pdf
2613
2614
2615  symbolic_Fabcd:
2616  ▷  $(ORBITER) -v 3 \
2617  ▷  ▷  -define F -finite_field -q 13 -end \
2618  ▷  ▷  -define M -symbolic_object \
2619  ▷  ▷  ▷  -field F \
2620  ▷  ▷  ▷  -managed_variables "X0,X1,X2,X3" \
2621  ▷  ▷  ▷  -text $(F_abcd_eqn) \
2622  ▷  ▷  -end
2623
2624
2625  symbolic_Fabcd_expand:
2626  ▷  $(ORBITER) -v 3 \
2627  ▷  ▷  -define Fq -finite_field -q 31 -end \
2628  ▷  ▷  -define Fabcd -symbolic_object \
2629  ▷  ▷  ▷  -field Fq \
2630  ▷  ▷  ▷  -managed_variables "X0,X1,X2,X3" \
2631  ▷  ▷  ▷  -text $(F_abcd_eqn) \
2632  ▷  ▷  -end \
2633  ▷  ▷  -define Fabcd_e -symbolic_object \
2634  ▷  ▷  ▷  -field Fq \
2635  ▷  ▷  ▷  -managed_variables "X0,X1,X2,X3" \
2636  ▷  ▷  ▷  -expand Fabcd \
2637  ▷  ▷  ▷  -write_trees_during_expand \
2638  ▷  ▷  -end
2639
2640
2641  # q=31
2642  #E=18 (a,b,c,d) = (25,5,5,25)
2643  #E=10 (a,b,c,d) = (14,13,13,14)
2644  #E=9 (a,b,c,d) = (5,3,12,5)
```

```
#E=6 (a,b,c,d) = (2,30,30,2)
#E=4 (a,b,c,d) = (4,2,2,4)
#E=3 (a,b,c,d) = (6,2,3,6)
#E=2 (a,b,c,d) = (2,3,3,2)
#E=1 (a,b,c,d) = (2,3,5,2)
#E=0 (a,b,c,d) = (2,5,9,7)


symbolic_Fabcd_q31_E18:
	$(ORBITER) -v 3 \
	  -define F -finite_field -q 31 -end \
	  -define M -symbolic_object \
	    -field F \
	    -managed_variables "X0,X1,X2,X3" \
	    -text $(F_abcd_eqn) \
	  -end \
	  -define L -symbolic_object \
	    -field F \
	    -text "25,5,5,25" \
	  -end \
	  -define M1 -symbolic_object \
	    -field F \
	    -substitute "a,b,c,d" M L \
	  -end \
	  -define M2 -symbolic_object \
	    -field F \
	    -expand M1 \
	    -write_trees_during_expand \
	  -end
	pdflatex M2.tex
	$(OPEN) M2.pdf


#(13 X1 X2 X3) + (22 X1 X2^{2}) + (22 X1^{2} X3) + (22 X1^{2} X2) + (9 X0 X2^{2})
 + (9 X0^{2} X2) + (22 X1 X3^{2})\\


symbolic_Fabcd_q31_E18_with_implicit_substitution:
	$(ORBITER) -v 3 \
	  -define F -finite_field -q 31 -end \
	  -define a -symbolic_object \
	    -field F \
	    -text "25" \
	  -end \
	  -define b -symbolic_object \
	    -field F \
	    -text "5" \
	  -end \
	  -define c -symbolic_object \
	    -field F \
	    -text "5" \
	  -end \
	  -define d -symbolic_object \
	    -field F \
	    -text "25" \
	  -end \
	  -define M -symbolic_object \
	    -field F \
```

```
2703  ▷  ▷  ▷  -managed_variables "X0,X1,X2,X3" \
2704  ▷  ▷  ▷  -text $(F_abcd_eqn) \
2705  ▷  ▷  -end \
2706  ▷  ▷  -define M2 -symbolic_object \
2707  ▷  ▷  ▷  -field F \
2708  ▷  ▷  ▷  -expand M \
2709  ▷  ▷  -end
2710  ▷  pdflatex M2.tex
2711  ▷  $(OPEN) M2.pdf
2712
2713
2714
2715  symbolic_Fabcd_q31_E18_with_implicit_substitution_do_not_simplify:
2716  ▷  $(ORBITER) -v 3 \
2717  ▷  ▷  -define F -finite_field -q 31 -end \
2718  ▷  ▷  -define a -symbolic_object \
2719  ▷  ▷  ▷  -field F \
2720  ▷  ▷  ▷  -text "25" \
2721  ▷  ▷  -end \
2722  ▷  ▷  -define b -symbolic_object \
2723  ▷  ▷  ▷  -field F \
2724  ▷  ▷  ▷  -text "5" \
2725  ▷  ▷  -end \
2726  ▷  ▷  -define c -symbolic_object \
2727  ▷  ▷  ▷  -field F \
2728  ▷  ▷  ▷  -text "5" \
2729  ▷  ▷  -end \
2730  ▷  ▷  -define d -symbolic_object \
2731  ▷  ▷  ▷  -field F \
2732  ▷  ▷  ▷  -text "25" \
2733  ▷  ▷  -end \
2734  ▷  ▷  -define M -symbolic_object \
2735  ▷  ▷  ▷  -field F \
2736  ▷  ▷  ▷  -managed_variables "X0,X1,X2,X3" \
2737  ▷  ▷  ▷  -text $(F_abcd_eqn) \
2738  ▷  ▷  ▷  -do_not_simplify \
2739  ▷  ▷  -end
2740
2741
2742
2743
2744  symbolic_matrix_minor:
2745  ▷  $(ORBITER) -v 3 \
2746  ▷  ▷  -define F -finite_field -q 13 -end \
2747  ▷  ▷  -define A -symbolic_object \
2748  ▷  ▷  ▷  -field F \
2749  ▷  ▷  ▷  -matrix 4 \
2750  ▷  ▷  ▷  -text "1,0,0,0,0,0,0,1,x0,x1,x2,x3,a,b,c,d" \
2751  ▷  ▷  -end \
2752  ▷  ▷  -define M -symbolic_object \
2753  ▷  ▷  ▷  -field F \
2754  ▷  ▷  ▷  -minor A 3 1  \
2755  ▷  ▷  -end \
2756
2757
2758  symbolic_matrix_nullspace_1:
2759  ▷  $(ORBITER) -v 3 \
2760  ▷  ▷  -define F -finite_field -q 13 -end \
2761  ▷  ▷  -define A1 -symbolic_object \
```

```
2762  ▷  ▷  ▷  -field F \
2763  ▷  ▷  ▷  -matrix 3 \
2764  ▷  ▷  ▷  -text "1,0,0,0,0,0,0,1,y0,y1,y2,0" \
2765  ▷  ▷  -end \
2766  ▷  ▷  -define M1 -symbolic_object \
2767  ▷  ▷  ▷  -field F \
2768  ▷  ▷  ▷  -symbolic_nullspace A1 \
2769  ▷  ▷  -end \
2770
2771  #0, ((12) * (y2)), (y1), 0
2772
2773
2774  symbolic_matrix_nullspace_2:
2775  ▷  $(ORBITER) -v 3 \
2776  ▷  ▷  -define F -finite_field -q 13 -end \
2777  ▷  ▷  -define A2 -symbolic_object \
2778  ▷  ▷  ▷  -field F \
2779  ▷  ▷  ▷  -matrix 3 \
2780  ▷  ▷  ▷  -text "0,1,0,0,0,0,-1,1,y0,y1,y2,0" \
2781  ▷  ▷  -end \
2782  ▷  ▷  -define M2 -symbolic_object \
2783  ▷  ▷  ▷  -field F \
2784  ▷  ▷  ▷  -symbolic_nullspace A2 \
2785  ▷  ▷  -end \
2786
2787
2788  #(y2), 0, (12) * (y0), ((12) * (y0))
2789
2790  symbolic_Clebsch:
2791  ▷  $(ORBITER) -v 3 \
2792  ▷  ▷  -define F -finite_field -q 13 -end \
2793  ▷  ▷  -define A1 -symbolic_object \
2794  ▷  ▷  ▷  -field F \
2795  ▷  ▷  ▷  -matrix 3 \
2796  ▷  ▷  ▷  -text "1,0,0,0,0,0,0,1,y0,y1,y2,0" \
2797  ▷  ▷  -end \
2798  ▷  ▷  -define A2 -symbolic_object \
2799  ▷  ▷  ▷  -field F \
2800  ▷  ▷  ▷  -matrix 3 \
2801  ▷  ▷  ▷  -text "0,1,0,0,0,0,-1,1,y0,y1,y2,0" \
2802  ▷  ▷  -end \
2803  ▷  ▷  -define A3 -symbolic_object \
2804  ▷  ▷  ▷  -field F \
2805  ▷  ▷  ▷  -matrix 2 \
2806  ▷  ▷  ▷  -text "0,0,1,0,0,0,0,1" \
2807  ▷  ▷  -end \
2808  ▷  ▷  -define M1 -symbolic_object \
2809  ▷  ▷  ▷  -field F \
2810  ▷  ▷  ▷  -symbolic_nullspace A1 \
2811  ▷  ▷  -end \
2812  ▷  ▷  -define M2 -symbolic_object \
2813  ▷  ▷  ▷  -field F \
2814  ▷  ▷  ▷  -symbolic_nullspace A2 \
2815  ▷  ▷  -end \
2816  ▷  ▷  -define L -symbolic_object \
2817  ▷  ▷  ▷  -field F \
2818  ▷  ▷  ▷  -stack_matrices_vertically M1,M2,A3 \
2819  ▷  ▷  -end \
2820
```

```
2821
2822
2823
2824
2825  Clebsch_map_up_Fabcd:
2826  ▷  $(ORBITER) -v 3 \
2827  ▷  ▷  -define F -finite_field -q 13 -end \
2828  ▷  ▷  -define M -symbolic_object \
2829  ▷  ▷  ▷  -field F \
2830  ▷  ▷  ▷  -managed_variables "X0,X1,X2,X3" \
2831  ▷  ▷  ▷  -text $(F_abcd_eqn) \
2832  ▷  ▷  -end \
2833  ▷  ▷  -define L -symbolic_object \
2834  ▷  ▷  ▷  -field F \
2835  ▷  ▷  ▷  -text "y0+t*y0,t*y1,t*y2,y2" \
2836  ▷  ▷  -end \
2837  ▷  ▷  -define C1 -symbolic_object \
2838  ▷  ▷  ▷  -field F \
2839  ▷  ▷  ▷  -managed_variables "y0,y1,y2" \
2840  ▷  ▷  ▷  -substitute "X0,X1,X2,X3" M L \
2841  ▷  ▷  -end \
2842  ▷  ▷  -define M -symbolic_object \
2843  ▷  ▷  ▷  -field F \
2844  ▷  ▷  ▷  -expand C1 \
2845  ▷  ▷  ▷  -managed_variables "y0,y1,y2" \
2846  ▷  ▷  ▷  -write_trees_during_expand \
2847  ▷  ▷  -end \
2848  ▷  ▷  -define coeff_t -symbolic_object \
2849  ▷  ▷  ▷  -field F \
2850  ▷  ▷  ▷  -managed_variables "y0,y1,y2" \
2851  ▷  ▷  ▷  -collect M t \
2852  ▷  ▷  -end
2853
2854
2855
2856
2857
2858  ################################################################################
2859  # Chapter 3 - Basic Algebra
2860  ################################################################################
2861
2862
2863  test_3:
2864  ▷  make test_3_1
2865  ▷  make test_3_2
2866  ▷  make test_3_3
2867  ▷  make test_3_4
2868  ▷  make test_3_5
2869  ▷  make test_3_6
2870
2871
2872  ################################################################################
2873  # Section 3.1: Basic Number Theory
2874
2875  SECTION_BASIC_NUMBER_THEORY:
2876
2877  test_3_1:
2878  ▷  make PR101
2879  ▷  make PR101_smallest
```

```
2880  ▷  make PR29
2881  ▷  make PR31
2882  ▷  make PR37
2883  ▷  make PR_2_100
2884  ▷  make PR_2_1000
2885  ▷  make PE_number_2_1000
2886  ▷  make number_of_primitive_roots_10000
2887  ▷  make IM_3_19
2888  ▷  make IM_723
2889  ▷  make IM
2890  ▷  make IM_gcd
2891  ▷  make PM3a
2892  ▷  make PR_915839
2893  ▷  make PR_915839_check
2894  ▷  make DL_915839
2895  ▷  make sqrt_mod
2896  ▷  make sqrt_5_mod_11
2897  ▷  make sqrt_5_mod_19
2898  ▷  make sqrt_mod_20_31
2899  ▷  make order_of_2_mod_n
2900  ▷  make Eulerfunction_150
2901  ▷  make Eulerfunction_900
2902  ▷  make Eulerfunction_10000
2903  ▷  make power_function_2_mod_11
2904  ▷  make draw_mod_13
2905  ▷  make inverse_mod_a
2906  ▷  make smith_normal_form_1
2907  ▷  make smith_normal_form_2
2908  ▷  make jacobi_35_41
2909  ▷  make jacobi_33_41
2910  ▷  make jacobi_a
2911  ▷  make jacobi_5_19
2912  ▷  make sqrt_mod_7817
2913  ▷  make Chinese_remainders_A
2914  ▷  make Chinese_remainders_B
2915  ▷  make Chinese_remainders_C2
2916  ▷  make Chinese_remainders_C3
2917
2918
2919
2920  PR101:
2921  ▷  $(ORBITER) -v 5 -primitive_root 101
2922
2923
2924  PR101_smallest:
2925  ▷  $(ORBITER) -v 5 -smallest_primitive_root 101
2926
2927
2928  PR29:
2929  ▷  $(ORBITER) -v 1 -smallest_primitive_root 29
2930
2931  PR31:
2932  ▷  $(ORBITER) -v 1 -smallest_primitive_root 31
2933
2934  PR37:
2935  ▷  $(ORBITER) -v 1 -smallest_primitive_root 37
2936
2937  PR_2_100:
2938  ▷  $(ORBITER) -v 1 -smallest_primitive_root_interval 2 100
```

```
2939
2940
2941
2942
2943  PR_2_1000:
2944  ▷  $(ORBITER) -v 1 -smallest_primitive_root_interval 2 1000
2945  ▷  $(ORBITER) -v 1 -csv_file_latex 1 \
2946  ▷  ▷  primitive_element_table_2_1000.csv
2947  ▷  pdflatex primitive_element_table_2_1000.tex
2948  ▷  $(OPEN) primitive_element_table_2_1000.pdf
2949
2950  PE_number_2_1000:
2951  ▷  $(ORBITER) -v 1 -number_of_primitive_roots_interval 2 1000
2952  ▷  $(ORBITER) -v 1 -csv_file_latex 1 table_number_of_pe_2_1000.csv
2953  ▷  pdflatex table_number_of_pe_2_1000.tex
2954  ▷  $(OPEN) table_number_of_pe_2_1000.pdf
2955
2956  number_of_primitive_roots_10000:
2957  ▷  $(ORBITER) -v 1 -number_of_primitive_roots_interval 10000 10001
2958
2959
2960
2961  IM_3_19:
2962  ▷  $(ORBITER) -v 5 -inverse_mod 3 19
2963
2964
2965  IM_723:
2966  ▷  $(ORBITER) -v 5 -inverse_mod 723 4060
2967
2968
2969
2970  IM:
2971  ▷  $(ORBITER) -v 5 -inverse_mod 1865025205 2147483647
2972
2973
2974  IM_gcd:
2975  ▷  $(ORBITER) -v 5 -extended_gcd 1865025205 2147483647
2976
2977
2978  PM3a:
2979  ▷  $(ORBITER) -v 5 -power_mod 16807 1073741823 2147483647
2980
2981  # randomized algo:
2982
2983  PR_915839:
2984  ▷  $(ORBITER) -v 5 -primitive_root 915839
2985
2986  #a primitive root modulo 915839 is 43085
2987
2988  PR_915839_check:
2989  ▷  $(ORBITER) -v 5 -power_mod 43085 49842 915839
2990
2991  #the power of 43085 to the 49842 mod 915839 is 487320
2992
2993  DL_915839:
2994  ▷  $(ORBITER) -v 5 -discrete_log 487320 43085 915839
2995
2996
2997  #The discrete log is 49842 since 487320 = 43085^49842 mod 915839, time: 0:22
```

```
2998
2999  sqrt_mod:
3000  ▷  $(ORBITER) -v 2 -square_root_mod 33 41
3001
3002
3003  sqrt_5_mod_11:
3004  ▷  $(ORBITER) -v 2 -square_root_mod 5 11
3005
3006
3007  sqrt_5_mod_19:
3008  ▷  $(ORBITER) -v 2 -square_root_mod 5 19
3009
3010
3011  sqrt_mod_20_31:
3012  ▷  $(ORBITER) -v 2 -square_root_mod 20 31
3013
3014  order_of_2_mod_n:
3015  ▷  $(ORBITER) -v 3 -order_of_q_mod_n 2 3 51
3016  ▷  $(ORBITER) -v 1 -csv_file_latex 1 \
3017  ▷  ▷  order_of_q_mod_n_q2_3_51.csv
3018  ▷  pdflatex order_of_q_mod_n_q2_3_51.tex
3019  ▷  $(OPEN) order_of_q_mod_n_q2_3_51.pdf
3020
3021
3022
3023
3024
3025
3026  Eulerfunction_150:
3027  ▷  $(ORBITER) -v 1 -eulerfunction_interval 1 150
3028  ▷  $(ORBITER) -v 1 -csv_file_latex 1 \
3029  ▷  ▷  table_eulerfunction_1_150.csv
3030  ▷  pdflatex table_eulerfunction_1_150.tex
3031  ▷  $(OPEN) table_eulerfunction_1_150.pdf
3032
3033  Eulerfunction_900:
3034  ▷  $(ORBITER) -v 1 -eulerfunction_interval 900 1150
3035  ▷  $(ORBITER) -v 1 -csv_file_latex 1 \
3036  ▷  ▷  table_eulerfunction_900_1150.csv
3037  ▷  pdflatex table_eulerfunction_900_1150.tex
3038  ▷  $(OPEN) table_eulerfunction_900_1150.pdf
3039
3040  Eulerfunction_10000:
3041  ▷  $(ORBITER) -v 1 -eulerfunction_interval 10000 10150
3042  ▷  $(ORBITER) -v 1 -csv_file_latex 1 \
3043  ▷  ▷  table_eulerfunction_10000_10150.csv
3044  ▷  pdflatex table_eulerfunction_10000_10150.tex
3045  ▷  $(OPEN) table_eulerfunction_10000_10150.pdf
3046
3047
3048
3049  power_function_2_mod_11:
3050  ▷  $(ORBITER) -v 5 -power_function_mod_n 2 11
3051  ▷  $(ORBITER) -v 1 -csv_file_latex 1 power_function_k2_n11.csv
3052  ▷  pdflatex power_function_k2_n11.tex
3053  ▷  $(OPEN) power_function_k2_n11.pdf
3054
3055
3056  draw_mod_13:
```

```
3057  ▷  $(ORBITER) -v 2 \
3058  ▷  ▷  -draw_options -embedded -end \
3059  ▷  ▷  -draw_mod_n \
3060  ▷  ▷  ▷  -n 13 \
3061  ▷  ▷  ▷  -file mod_13 \
3062  ▷  ▷  ▷  -power_cycle 2 \
3063  ▷  ▷  -end
3064  ▷  pdflatex mod_13_draw.tex
3065  ▷  $(OPEN) mod_13_draw.pdf
3066
3067
3068
3069
3070  inverse_mod_26_99:
3071  ▷  $(ORBITER) -v 2 -inverse_mod 26 99
3072
3073
3074  inverse_mod_a:
3075  ▷  $(ORBITER) -v 2 -inverse_mod 18059241 58014043
3076
3077
3078
3079  smith_normal_form_1:
3080  ▷  $(ORBITER) -v 3 \
3081  ▷  ▷  -define M -vector \
3082  ▷  ▷  ▷  -dense "1,2,3,4,5,6,7,8,9" \
3083  ▷  ▷  ▷  -format 3 \
3084  ▷  ▷  -end \
3085  ▷  ▷  -smith_normal_form M
3086
3087
3088  smith_normal_form_2:
3089  ▷  $(ORBITER) -v 3 \
3090  ▷  ▷  -define M -vector \
3091  ▷  ▷  ▷  -dense "9,8,7,6,5,4,3,2,1" \
3092  ▷  ▷  ▷  -format 3 \
3093  ▷  ▷  -end \
3094  ▷  ▷  -smith_normal_form M
3095
3096
3097
3098  jacobi_35_41:
3099  ▷  $(ORBITER) -v 5 -jacobi 35 41
3100  ▷  pdflatex jacobi_35_41.tex
3101  ▷  $(OPEN) jacobi_35_41.pdf
3102
3103
3104  jacobi_33_41:
3105  ▷  $(ORBITER) -v 5 -jacobi 33 41
3106  ▷  pdflatex jacobi_33_41.tex
3107  ▷  $(OPEN) jacobi_33_41.pdf
3108
3109
3110  jacobi_a:
3111  ▷  $(ORBITER) -v 5 -jacobi 2221 7817
3112
3113  jacobi_5_19:
3114  ▷  $(ORBITER) -v 5 -jacobi 5 19
3115
```

```
3116  sqrt_mod_7817:
3117  ▷  ▷  $(ORBITER) -v 2 -square_root_mod 2221 7817
3118
3119
3120
3121
3122
3123  Chinese_remainders_A:
3124  ▷  $(ORBITER) -v 2 \
3125  ▷  ▷  -define R -vector -dense "2,2,5" -end \
3126  ▷  ▷  -define M -vector -dense "5,6,7" -end \
3127  ▷  ▷  -Chinese_remainders R M
3128
3129  Chinese_remainders_B:
3130  ▷  $(ORBITER) -v 2 \
3131  ▷  ▷  -define R -vector -dense "38,2" -end \
3132  ▷  ▷  -define M -vector -dense "74,27" -end \
3133  ▷  ▷  -Chinese_remainders R M
3134
3135  Chinese_remainders_C2:
3136  ▷  $(ORBITER) -v 2 \
3137  ▷  ▷  -define R -vector -dense "2,3" -end \
3138  ▷  ▷  -define M -vector -dense "2147483647,5915587277" -end \
3139  ▷  ▷  -Chinese_remainders R M
3140
3141
3142  #The solution is 5684294357108828365 modulo 12703626939758759219 (computed in lon
      ginteger)
3143
3144  # checking with Maple:
3145  #5684294357108828365 mod 2147483647;
3146  #                               2
3147
3148  #5684294357108828365 mod 5915587277;
3149  #                               3
3150
3151
3152
3153  Chinese_remainders_C3:
3154  ▷  $(ORBITER) -v 2 \
3155  ▷  ▷  -define R -vector -dense "2,3,4" -end \
3156  ▷  ▷  -define M -vector -dense "2147483647,5915587277,3267000013" -end \
3157  ▷  ▷  -Chinese_remainders R M
3158
3159
3160  #The solution is 3143154175932447732745157253 9 modulo 41502749377339016585336869
      847 (computed in longinteger)
3161
3162  # checking with Maple:
3163  #3143154175932447732745157253 9 mod 2147483647;
3164  #                               2
3165
3166  #3143154175932447732745157253 9 mod 5915587277;
3167  #                               3
3168
3169  #3143154175932447732745157253 9 mod 3267000013;
3170  #                               4
3171
3172
```

```
3173
3174  ##############################################################################
3175  # Section 3.2: Finite Fields
3176
3177  SECTION_FINITE_FIELDS:
3178
3179
3180  test_3_2:
3181  ▷  make F_2
3182  ▷  make F_3
3183  ▷  make F_5
3184  ▷  make F_5_add_table
3185  ▷  make F_7
3186  ▷  make F_7_tables
3187  ▷  make F_127
3188  ▷  make F_11_product_of_all_nonzero_elements
3189  ▷  make F_7_vandermonde
3190  ▷  make F_101_wo
3191  ▷  make F_1021_wo
3192
3193
3194
3195  F_2:
3196  ▷  $(ORBITER) -v 3 -list_arguments \
3197  ▷  ▷  -define F -finite_field -q 2 -end \
3198  ▷  ▷  -with F -do -finite_field_activity -cheat_sheet_GF -end
3199  ▷  pdflatex GF_2.tex
3200  ▷  $(OPEN) GF_2.pdf
3201
3202
3203  F_3:
3204  ▷  $(ORBITER) -v 3 \
3205  ▷  ▷  -define F -finite_field -q 3 -end \
3206  ▷  ▷  -with F -do -finite_field_activity -cheat_sheet_GF -end
3207  ▷  #pdflatex GF_3.tex
3208  ▷  #$(OPEN) GF_3.pdf
3209
3210  F_5:
3211  ▷  $(ORBITER) -v 3 \
3212  ▷  ▷  -define F -finite_field -q 5 -end \
3213  ▷  ▷  -with F -do -finite_field_activity -cheat_sheet_GF -end
3214  ▷  pdflatex GF_5.tex
3215  ▷  $(OPEN) GF_5.pdf
3216
3217
3218  F_5_add_table:
3219  ▷  $(ORBITER) -v 3 \
3220  ▷  ▷  -define F -finite_field -q 5 -end \
3221  ▷  ▷  -with F -do -finite_field_activity -export_tables -end \
3222  ▷  ▷  -draw_matrix -input_csv_file GF_q5_table_add.csv \
3223  ▷  ▷  ▷  -box_width 40 -bit_depth 24 -partition 3 5 5 -end \
3224
3225
3226  F_7:
3227  ▷  $(ORBITER) -v 3 \
3228  ▷  ▷  -define F -finite_field -q 7 -end \
3229  ▷  ▷  -with F -do -finite_field_activity -cheat_sheet_GF -end
3230  ▷  pdflatex GF_7.tex
3231  ▷  $(OPEN) GF_7.pdf
```

```
3232
3233
3234  F_7_tables:
3235  ▷  $(ORBITER) -v 3 \
3236  ▷  ▷  -define F -finite_field -q 7  -end \
3237  ▷  ▷  -with F -do -finite_field_activity -export_tables -end
3238  ▷  $(ORBITER) -v 3 \
3239  ▷  ▷  -define all_one -vector -repeat 1 7 -end \
3240  ▷  ▷  -draw_matrix -input_csv_file GF_q7_table_add.csv \
3241  ▷  ▷  ▷  -box_width 40 -bit_depth 24 \
3242  ▷  ▷  ▷  -partition 3 all_one all_one \
3243  ▷  ▷  -end \
3244  ▷  ▷  -draw_matrix -input_csv_file GF_q7_table_mul.csv \
3245  ▷  ▷  ▷  -box_width 40 -bit_depth 24 \
3246  ▷  ▷  ▷  -partition 3 all_one all_one \
3247  ▷  ▷  -end
3248  ▷  convert GF_q7_table_add_draw.bmp GF_q7_table_add_draw.png
3249  ▷  convert GF_q7_table_mul_draw.bmp GF_q7_table_mul_draw.png
3250
3251
3252  F_127:
3253  ▷  $(ORBITER) -v 3 \
3254  ▷  ▷  -define F -finite_field -q 127 -end \
3255  ▷  ▷  -with F -do -finite_field_activity -cheat_sheet_GF -end
3256
3257
3258  F_11_product_of_all_nonzero_elements:
3259  ▷  $(ORBITER) -v 3 \
3260  ▷  ▷  -define F -finite_field -q 11 -end \
3261  ▷  ▷  -define S -vector -field F -loop 1 11 1 -end \
3262  ▷  ▷  -with F -do -finite_field_activity -product_of S -end
3263
3264
3265  F_7_vandermonde:
3266  ▷  $(ORBITER) -v 3 \
3267  ▷  ▷  -define F -finite_field -q 7 -end \
3268  ▷  ▷  -with F -do -finite_field_activity \
3269  ▷  ▷  ▷  -Vandermonde_matrix \
3270  ▷  ▷  -end
3271
3272
3273  F_101_wo:
3274  ▷  $(ORBITER) -v 3 \
3275  ▷  ▷  -define F -finite_field -q 101 -without_tables -end \
3276  ▷  ▷  -with F -do -finite_field_activity -cheat_sheet_GF -end
3277  ▷  pdflatex GF_101.tex
3278  ▷  $(OPEN) GF_101.pdf
3279
3280
3281  F_1021_wo:
3282  ▷  $(ORBITER) -v 3 \
3283  ▷  ▷  -define F -finite_field -q 1021 -without_tables -end \
3284  ▷  ▷  -with F -do -finite_field_activity -cheat_sheet_GF -end
3285
3286
3287
3288
3289  ##############################################################################
3290  # Section 3.3: Extension Fields
```

```
3291
3292  SECTION_EXTENSION_FIELDS:
3293  ▷
3294
3295  test_3_3:
3296  ▷  make F_4
3297  ▷  make F_4_tables
3298  ▷  make F_8
3299  ▷  make F_8_tables
3300  ▷  make F_8b
3301  ▷  make F_9
3302  ▷  make F_16
3303  ▷  make F_16_tables
3304  ▷  make F_64
3305  ▷  make nth_roots_63
3306  ▷  make F8_F2_field_reduction
3307  ▷  make F_4096
3308  ▷  #make F_4489
3309  ▷  #make F_2_power_24
3310
3311
3312  F_4:
3313  ▷  $(ORBITER) -v 3 \
3314  ▷  ▷  -define F -finite_field -q 4 -end \
3315  ▷  ▷  -with F -do -finite_field_activity -cheat_sheet_GF -end
3316  ▷  pdflatex GF_4.tex
3317  ▷  $(OPEN) GF_4.pdf
3318
3319
3320
3321
3322  F_4_tables:
3323  ▷  $(ORBITER) -v 3 \
3324  ▷  ▷  -define F -finite_field -q 4 -end \
3325  ▷  ▷  -with F -do -finite_field_activity -export_tables -end
3326  ▷  $(ORBITER) -v 3 \
3327  ▷  ▷  -define all_one -vector -repeat 1 4 -end \
3328  ▷  ▷  -draw_matrix -input_csv_file GF_q4_table_add.csv \
3329  ▷  ▷  ▷  -box_width 40 -bit_depth 24 \
3330  ▷  ▷  ▷  -partition 3 all_one all_one \
3331  ▷  ▷  -end \
3332  ▷  ▷  -draw_matrix -input_csv_file GF_q4_table_mul.csv \
3333  ▷  ▷  ▷  -box_width 40 -bit_depth 24 \
3334  ▷  ▷  ▷  -partition 3 all_one all_one \
3335  ▷  ▷  -end
3336  ▷  convert GF_q4_table_add_draw.bmp GF_q4_table_add_draw.png
3337  ▷  convert GF_q4_table_mul_draw.bmp GF_q4_table_mul_draw.png
3338
3339
3340
3341  F_8:
3342  ▷  $(ORBITER) -v 3 \
3343  ▷  ▷  -define F -finite_field -q 8 -end \
3344  ▷  ▷  -with F -do -finite_field_activity -cheat_sheet_GF -end
3345  ▷  pdflatex GF_8.tex
3346  ▷  $(OPEN) GF_8.pdf
3347
3348
3349  F_8_tables:
```

```
3350  ▷  $(ORBITER) -v 3 \
3351  ▷  ▷  -define all_one -vector -repeat 1 8 -end \
3352  ▷  ▷  -define F -finite_field -q 8 -end \
3353  ▷  ▷  -with F -do -finite_field_activity -export_tables -end \
3354  ▷  ▷  -with F -do -finite_field_activity -cheat_sheet_GF -end \
3355  ▷  ▷  -draw_matrix -input_csv_file GF_q8_table_add.csv \
3356  ▷  ▷  ▷  -box_width 40 -bit_depth 24 -partition 3 all_one all_one -end \
3357  ▷  ▷  -draw_matrix -input_csv_file GF_q8_table_mul.csv \
3358  ▷  ▷  ▷  -box_width 40 -bit_depth 24 -partition 3 all_one all_one -end
3359
3360
3361  F_8b:
3362  ▷  $(ORBITER) -v 3 \
3363  ▷  ▷  -define F -finite_field -q 8 -override_polynomial 11 -end \
3364  ▷  ▷  -with F -do -finite_field_activity -cheat_sheet_GF -end
3365  ▷  pdflatex GF_8.tex
3366  ▷  $(OPEN) GF_8.pdf
3367
3368
3369
3370  F_9:
3371  ▷  $(ORBITER) -v 3 \
3372  ▷  ▷  -define F -finite_field -q 9 -end \
3373  ▷  ▷  -with F -do -finite_field_activity -cheat_sheet_GF -end
3374  ▷  pdflatex GF_9.tex
3375  ▷  $(OPEN) GF_9.pdf
3376
3377
3378
3379  F_16:
3380  ▷  $(ORBITER) -v 10 \
3381  ▷  ▷  -define F -finite_field -q 16 -compute_related_fields -end \
3382  ▷  ▷  -with F -do -finite_field_activity -cheat_sheet_GF -end
3383  ▷  pdflatex GF_16.tex
3384  ▷  $(OPEN) GF_16.pdf
3385
3386
3387
3388
3389  F_16_tables:
3390  ▷  $(ORBITER) -v 3 \
3391  ▷  ▷  -define F -finite_field -q 16 -end \
3392  ▷  ▷  -with F -do -finite_field_activity -export_tables -end
3393  ▷  $(ORBITER) -v 3 \
3394  ▷  ▷  -define all_one -vector -repeat 1 16 -end \
3395  ▷  ▷  -draw_matrix -input_csv_file GF_q16_table_add.csv \
3396  ▷  ▷  ▷  -box_width 40 -bit_depth 24 \
3397  ▷  ▷  ▷  -partition 3 all_one all_one \
3398  ▷  ▷  -end \
3399  ▷  ▷  -draw_matrix -input_csv_file GF_q16_table_mul.csv \
3400  ▷  ▷  ▷  -box_width 40 -bit_depth 24 \
3401  ▷  ▷  ▷  -partition 3 all_one all_one \
3402  ▷  ▷  -end
3403  ▷  convert GF_q16_table_add_draw.bmp GF_q16_table_add_draw.png
3404  ▷  convert GF_q16_table_mul_draw.bmp GF_q16_table_mul_draw.png
3405
3406
3407
3408  F_64:
```

```
3409  ▷  $(ORBITER) -v 3 \
3410  ▷  ▷  -define F -finite_field -q 64 -end \
3411  ▷  ▷  -with F -do -finite_field_activity -cheat_sheet_GF -end
3412  ▷  pdflatex GF_64.tex
3413  ▷  $(OPEN) GF_64.pdf
3414
3415
3416
3417  nth_roots_63:
3418  ▷  $(ORBITER) -v 3 \
3419  ▷  ▷  -define F -finite_field -q 2 -end \
3420  ▷  ▷  -with F -do -finite_field_activity \
3421  ▷  ▷  ▷  -nth_roots 63 \
3422  ▷  ▷  -end
3423  ▷  pdflatex Nth_roots_q2_n63.tex
3424  ▷  $(OPEN) Nth_roots_q2_n63.pdf
3425
3426
3427  F8_F2_field_reduction:
3428  ▷  $(ORBITER) -v 2 \
3429  ▷  ▷  -define F -finite_field -q 8 -end \
3430  ▷  ▷  -define v -vector -loop 0 8 1 -end \
3431  ▷  ▷  -define C -code -field F -Reed_Solomon 7 3 -end \
3432  ▷  ▷  -with F -do \
3433  ▷  ▷  -finite_field_activity \
3434  ▷  ▷  -field_reduction "F8_to_F2" \
3435  ▷  ▷  ▷  2 1 8 v \
3436  ▷  ▷  -end
3437  ▷  $(ORBITER) -v 2 \
3438  ▷  ▷  -draw_matrix -input_csv_file F8_to_F2.csv \
3439  ▷  ▷  -box_width 40 -bit_depth 24 \
3440  ▷  ▷  -partition 4 "3" "3,3,3,3,3,3,3,3" -end
3441  ▷  convert F8_to_F2_draw.bmp F8_to_F2_draw.png
3442  ▷  $(OPEN) F8_to_F2_draw.png
3443
3444
3445  F_4096:
3446  ▷  $(ORBITER) -v 4 \
3447  ▷  ▷  -define F -finite_field -q 4096 \
3448  ▷  ▷  ▷  -compute_related_fields \
3449  ▷  ▷  -end \
3450  ▷  ▷  -with F -do -finite_field_activity -cheat_sheet_GF -end
3451  ▷  pdflatex GF_4096.tex
3452  ▷  $(OPEN) GF_4096.pdf
3453
3454
3455
3456  # 67^2 = 4489
3457
3458  F_4489:
3459  ▷  $(ORBITER) -v 4 \
3460  ▷  ▷  -define F -finite_field -q 4489 -without_tables -end \
3461
3462
3463  # ToDo error message
3464  #finite_field::alpha_power !f_has_table
3465
3466  F_2_power_24:
3467  ▷  $(ORBITER) -v 4 \
```

```
3468  ▷  ▷  -define F -finite_field -q 16777216 \
3469  ▷  ▷  ▷  -compute_related_fields \
3470  ▷  ▷  ▷  -without_tables \
3471  ▷  ▷  -end \
3472  ▷  ▷  -with F -do -finite_field_activity \
3473  ▷  ▷  ▷  -cheat_sheet_GF \
3474  ▷  ▷  -end
3475  ▷  pdflatex GF_16777216.tex
3476  ▷  $(OPEN) GF_16777216.pdf
3477
3478
3479  ################################################################################
3480  # Section 3.4: Linear Algebra over Finite Fields
3481
3482
3483  SECTION_LINEAR_ALGEBRA:
3484
3485
3486  test_3_4:
3487  ▷  make RREF
3488  ▷  make RREF_V7
3489  ▷  make nullspace
3490  ▷  make RREF_RS_6_4_7
3491  ▷  make eigenstuff
3492  ▷  make classes_GL_3_2
3493  ▷  make classes_GL_4_2
3494
3495
3496  RREF:
3497  ▷  $(ORBITER) -v 2 \
3498  ▷  ▷  -define F -finite_field -q 2 -end \
3499  ▷  ▷  -define v -vector -field F -format 2 \
3500  ▷  ▷  ▷  -dense "1,1,1,1,0,1,1,0,0,1" \
3501  ▷  ▷  -end \
3502  ▷  ▷  -with F -do -finite_field_activity \
3503  ▷  ▷  ▷  -RREF v \
3504  ▷  ▷  -end
3505  ▷  pdflatex v_rref.tex
3506  ▷  $(OPEN) v_rref.pdf
3507
3508
3509  RREF_V7:
3510  ▷  $(ORBITER) -v 2 \
3511  ▷  ▷  -define F -finite_field -q 7 -end \
3512  ▷  ▷  -define V7 -vector -format 7 \
3513  ▷  ▷  ▷  -dense $(V7_VANDERMONDE_EXTENDED) \
3514  ▷  ▷  -end \
3515  ▷  ▷  -with F -do -finite_field_activity \
3516  ▷  ▷  ▷  -RREF V7 \
3517  ▷  ▷  -end
3518  ▷  pdflatex V7_rref.tex
3519  ▷  $(OPEN) V7_rref.pdf
3520
3521
3522
3523  nullspace:
3524  ▷  $(ORBITER) -v 2 \
3525  ▷  ▷  -define F2 -finite_field -q 2 -end \
3526  ▷  ▷  -define v -vector -field F2 -format 2 \
```

```
3527  ▷  ▷  ▷  -dense "1,1,1,1,0,1,1,0,0,1" \
3528  ▷  ▷  -end \
3529  ▷  ▷  -with F2 -do \
3530  ▷  ▷  -finite_field_activity \
3531  ▷  ▷  ▷  -nullspace v \
3532  ▷  ▷  -end
3533  ▷  pdflatex v_nullspace.tex
3534  ▷  $(OPEN) v_nullspace.pdf
3535
3536  RREF_RS_6_4_7:
3537  ▷  $(ORBITER) -v 2 \
3538  ▷  ▷  -define F -finite_field -q 7 -end \
3539  ▷  ▷  -define v -vector -field F -format 4 \
3540  ▷  ▷  ▷  -compact $(CODE_RS_6_4_7) \
3541  ▷  ▷  -end \
3542  ▷  ▷  -with F -do \
3543  ▷  ▷  -finite_field_activity -RREF v -end
3544  ▷  pdflatex v_rref.tex
3545  ▷  $(OPEN) v_rref.pdf
3546
3547
3548  ## these are global algebra functions:
3549
3550
3551  eigenstuff:
3552  ▷  $(ORBITER) -v 6 \
3553  ▷  ▷  -define F -finite_field -q 5 -end \
3554  ▷  ▷  -eigenstuff F 4 "0,1,0,2,0,1,2,1,4,2,3,1,2,0,4,3"
3555
3556  classes_GL_3_2:
3557  ▷  $(ORBITER) -v 7 \
3558  ▷  ▷  -define F -finite_field -q 2 -end \
3559  ▷  ▷  -all_rational_normal_forms F 3
3560  ▷  pdflatex Class_reps_GL_3_2.tex
3561  ▷  $(OPEN) Class_reps_GL_3_2.pdf
3562
3563
3564  classes_GL_4_2:
3565  ▷  $(ORBITER) -v 7 \
3566  ▷  ▷  -define F -finite_field -q 2 -end \
3567  ▷  ▷  -all_rational_normal_forms F 4
3568  ▷  pdflatex Class_reps_GL_4_2.tex
3569  ▷  $(OPEN) Class_reps_GL_4_2.pdf
3570
3571
3572  ##############################################################################
3573  # Section 3.5: Advanced Topics in finite fields
3574
3575
3576
3577  SECTION_ADVANCED_TOPICS_IN_FINITE_FIELDS:
3578
3579
3580  test_3_5:
3581  ▷  make normal_basis_2_3
3582  ▷  make normal_basis_2_4
3583  ▷  make normal_basis_2_4_poly19
3584  ▷  make normal_basis_2_6
3585  ▷  make normal_basis_2_8
```

```
3586  ▷  make F8_over_F2_field_reduction
3587  ▷  make F_64_over_F8_field_reduction
3588  ▷  make F_64_over_F4_field_reduction
3589  ▷  make F_64_over_F2_field_reduction
3590  ▷  make F_8_Nth_roots_21
3591  ▷  make F_8_vandermonde
3592  ▷  make F_1024_vandermonde
3593  ▷  make NTT_k4_q17.cpp
3594  ▷  make F_17_NTT_compile
3595
3596
3597
3598  normal_basis_2_3:
3599  ▷  $(ORBITER) -v 2 \
3600  ▷  ▷  -define F -finite_field -q 2 -end \
3601  ▷  ▷  -with F -do -finite_field_activity \
3602  ▷  ▷  -normal_basis 3 -end
3603  ▷  pdflatex normal_basis_q2_d3.tex
3604  ▷  $(OPEN) normal_basis_q2_d3.pdf
3605
3606
3607  normal_basis_2_4:
3608  ▷  $(ORBITER) -v 2 \
3609  ▷  ▷  -define F -finite_field -q 2 -end \
3610  ▷  ▷  -with F -do -finite_field_activity \
3611  ▷  ▷  -normal_basis 4 -end
3612
3613
3614  normal_basis_2_4_poly19:
3615  ▷  $(ORBITER) -v 2 \
3616  ▷  ▷  -define F -finite_field -q 2 -end \
3617  ▷  ▷  -with F -do -finite_field_activity \
3618  ▷  ▷  -normal_basis_with_given_polynomial 19 4 -end
3619  ▷  pdflatex normal_basis_q2_d4_poly19.tex
3620  ▷  $(OPEN) normal_basis_q2_d4_poly19.pdf
3621
3622
3623  normal_basis_2_6:
3624  ▷  $(ORBITER) -v 2 \
3625  ▷  ▷  -define F -finite_field -q 2 -end \
3626  ▷  ▷  -with F -do -finite_field_activity \
3627  ▷  ▷  -normal_basis 6 -end
3628
3629
3630
3631  normal_basis_2_8:
3632  ▷  $(ORBITER) -v 2 \
3633  ▷  ▷  -define F -finite_field -q 2 -end \
3634  ▷  ▷  -with F -do -finite_field_activity \
3635  ▷  ▷  -normal_basis 8 -end
3636  ▷  pdflatex normal_basis_q2_d8.tex
3637  ▷  $(OPEN) normal_basis_q2_d8.pdf
3638
3639
3640
3641  F8_over_F2_field_reduction:
3642  ▷  $(ORBITER) -v 2 \
3643  ▷  ▷  -define F -finite_field -q 8 -end \
3644  ▷  ▷  -loop L 0 8 1 \
```

```
3645  ▷  ▷  ▷  -with F -do \
3646  ▷  ▷  ▷  -finite_field_activity \
3647  ▷  ▷  ▷  ▷  -field_reduction "F8_red_%L" 2 1 1 "%L" \
3648  ▷  ▷  ▷  -end \
3649  ▷  ▷  -end_loop L
3650  ▷  $(ORBITER) -v 2 \
3651  ▷  ▷  -loop L 0 8 1 \
3652  ▷  ▷  ▷  -draw_matrix \
3653  ▷  ▷  ▷  ▷  -input_csv_file F8_red_%L.csv \
3654  ▷  ▷  ▷  ▷  -box_width 40 -bit_depth 24 \
3655  ▷  ▷  ▷  ▷  -partition 4 3 3 \
3656  ▷  ▷  ▷  -end \
3657  ▷  ▷  -end_loop L
3658  ▷  #pdflatex field_reduction_Q8_q2_5_7.tex
3659
3660
3661
3662
3663  F_64_over_F8_field_reduction:
3664  ▷  $(ORBITER) -v 2 \
3665  ▷  ▷  -define F -finite_field -q 64 -end \
3666  ▷  ▷  -define elts -vector -field F -loop 0 64 1 -end \
3667  ▷  ▷  -with F -do \
3668  ▷  ▷  -finite_field_activity -field_reduction "F64_over_F8" 8 8 8 \
3669  ▷  ▷  ▷  elts -end
3670  ▷  $(ORBITER) -v 2 -draw_matrix \
3671  ▷  ▷  -input_csv_file F64_over_F8.csv \
3672  ▷  ▷  -box_width 40 -bit_depth 24 \
3673  ▷  ▷  -partition 4 "2,2,2,2,2,2,2,2" "2,2,2,2,2,2,2,2" -end
3674  ▷  $(OPEN) F64_over_F8_draw.bmp
3675  ▷  #pdflatex field_reduction_Q64_q8_8_8.tex
3676  ▷  #$(OPEN) field_reduction_Q64_q8_8_8.pdf
3677
3678
3679  F_64_over_F4_field_reduction:
3680  ▷  $(ORBITER) -v 2 \
3681  ▷  ▷  -define F -finite_field -q 64 -end \
3682  ▷  ▷  -define elts -vector -field F -loop 0 64 1 -end \
3683  ▷  ▷  -with F -do \
3684  ▷  ▷  -finite_field_activity \
3685  ▷  ▷  ▷  -field_reduction "F64_over_F4" 4 8 8 elts -end
3686  ▷  $(ORBITER) -v 2 -draw_matrix \
3687  ▷  ▷  -input_csv_file F64_over_F4.csv \
3688  ▷  ▷  -box_width 40 -bit_depth 24 \
3689  ▷  ▷  -partition 4 "3,3,3,3,3,3,3,3" "3,3,3,3,3,3,3,3" -end
3690  ▷  $(OPEN) F64_over_F4_draw.bmp
3691  ▷  #pdflatex field_reduction_Q64_q4_8_8.tex
3692  ▷  #$(OPEN) field_reduction_Q64_q4_8_8.pdf
3693
3694
3695  F_64_over_F2_field_reduction:
3696  ▷  $(ORBITER) -v 2 \
3697  ▷  ▷  -define F -finite_field -q 64 -end \
3698  ▷  ▷  -define elts -vector -field F -loop 0 64 1 -end \
3699  ▷  ▷  -with F -do \
3700  ▷  ▷  -finite_field_activity \
3701  ▷  ▷  ▷  -field_reduction "F64_over_F2" 2 8 8 elts -end
3702  ▷  $(ORBITER) -v 2 -draw_matrix \
3703  ▷  ▷  -input_csv_file F64_over_F2.csv \
```

```
3704  ▷  ▷  -box_width 40 -bit_depth 24 \
3705  ▷  ▷  -partition 4 "6,6,6,6,6,6,6,6" "6,6,6,6,6,6,6,6" -end
3706  ▷  $(OPEN) F64_over_F2_draw.bmp
3707  ▷  #pdflatex field_reduction_Q64_q2_8_8.tex
3708  ▷  #$(OPEN) field_reduction_Q64_q2_8_8.pdf
3709
3710
3711
3712
3713
3714  F_8_Nth_roots_21:
3715  ▷  $(ORBITER) -v 3 \
3716  ▷  ▷  -define F -finite_field -q 8 -override_polynomial 11 -end \
3717  ▷  ▷  -with F -do -finite_field_activity \
3718  ▷  ▷  ▷  -nth_roots 21 \
3719  ▷  ▷  -end
3720  ▷  pdflatex Nth_roots_q8_n21.tex
3721  ▷  $(OPEN) Nth_roots_q8_n21.pdf
3722
3723
3724
3725
3726  F_8_vandermonde:
3727  ▷  $(ORBITER) -v 3 \
3728  ▷  ▷  -define F -finite_field -q 8 -end \
3729  ▷  ▷  -with F -do -finite_field_activity \
3730  ▷  ▷  ▷  -Vandermonde_matrix \
3731  ▷  ▷  -end
3732
3733
3734
3735  F_1024_vandermonde:
3736  ▷  $(ORBITER) -v 3 \
3737  ▷  ▷  -define F -finite_field -q 1024 -end \
3738  ▷  ▷  -with F -do -finite_field_activity \
3739  ▷  ▷  ▷  -Vandermonde_matrix \
3740  ▷  ▷  -end
3741  ▷  #rm Vandermonde_1024.csv
3742  ▷  #rm Vandermonde_inv_1024.csv
3743
3744  #User time: 0:46
3745
3746
3747
3748
3749  NTT_k4_q17.cpp:
3750  ▷  $(ORBITER) -v 3 \
3751  ▷  ▷  -define F -finite_field -q 17 -end \
3752  ▷  ▷  -with F -do -coding_theoretic_activity \
3753  ▷  ▷  ▷  -NTT 4 17 \
3754  ▷  ▷  -end
3755
3756  F_17_NTT_compile: NTT_k4_q17.cpp
3757  ▷  $(MY_CPP)  NTT_k4_q17.cpp $(CPPFLAGS) \
3758  ▷  ▷  $(LIB) $(LFLAGS) -o NTT_k4_q17.out
3759  ▷  ./NTT_k4_q17.out
3760
3761  F64_1:
3762  ▷  $(ORBITER) -v 3 \
```

```
3763  ▷  ▷  -define F -finite_field -q 64 -override_polynomial 97 -end \
3764  ▷  ▷  -with F -do -finite_field_activity \
3765  ▷  ▷  ▷  -nth_roots 7 \
3766  ▷  ▷  -end
3767  ▷  pdflatex Nth_roots_q64_n7.tex
3768  ▷  $(OPEN) Nth_roots_q64_n7.pdf
3769
3770
3771  ################################################################################
3772  # Section 3.6: Basic Ring Theory
3773
3774
3775
3776  SECTION_BASIC_RING_THEORY:
3777
3778
3779  test_3_6:
3780  ▷  make Polynomial_ring
3781
3782
3783
3784  Polynomial_ring:
3785  ▷  $(ORBITER) -v 3 \
3786  ▷  ▷  -define F -finite_field -q 4 -end \
3787  ▷  ▷  -define R -polynomial_ring -field F \
3788  ▷  ▷  ▷  -number_of_variables 4 \
3789  ▷  ▷  ▷  -homogeneous_of_degree 3 \
3790  ▷  ▷  ▷  -variables "x0,x1,x2,x3" "x_0,x_1,x_2,x_3" \
3791  ▷  ▷  -end
3792
3793
3794
3795
3796  ################################################################################
3797  # Chapter 4 - Geometry
3798  ################################################################################
3799
3800
3801  test_4:
3802  ▷  make test_4_1
3803  ▷  make test_4_2
3804  ▷  make test_4_3
3805  ▷  make test_4_4
3806  ▷  make test_4_5
3807  ▷  make test_4_6
3808  ▷  make test_4_7
3809  ▷  make test_4_8
3810  ▷  make test_4_9
3811  ▷  make test_4_10
3812
3813
3814
3815  ################################################################################
3816  # Section 4.1: Finite Projective Spaces
3817
3818  SECTION_FINITE_PROJECTIVE_SPACES:
3819
3820
3821  test_4_1:
```

```
3822  ▷  make PG_3_2_easy
3823  ▷  make PG_1_16
3824  ▷  make PG_2_2
3825  ▷  make PG_2_3
3826  ▷  make PG_2_4
3827  ▷  make PG_2_13
3828  ▷  make PG_2_64
3829  ▷  make PG_3_2
3830  ▷  make PG_3_4
3831  ▷  make PG_3_5
3832  ▷  make PG_3_7
3833  ▷  make PG_3_8
3834  ▷  make PG_3_16
3835  ▷  make PG_3_25
3836  ▷  make PG_4_3
3837  ▷  make PG_8_2
3838
3839
3840  PG_3_2_easy:
3841  ▷  $(ORBITER) -v 2 \
3842  ▷  ▷  -define F -finite_field -q 2 -end \
3843  ▷  ▷  -define P -projective_space -n 3 -field F -end
3844
3845
3846
3847
3848  PG_1_16:
3849  ▷  $(ORBITER) -v 2 \
3850  ▷  ▷  -define F -finite_field -q 16 -end \
3851  ▷  ▷  -define P -projective_space -n 1 -field F -v 0 -end \
3852  ▷  ▷  -with P -do -projective_space_activity \
3853  ▷  ▷  ▷  -cheat_sheet \
3854  ▷  ▷  -end
3855  ▷  pdflatex PG_1_16.tex
3856  ▷  $(OPEN) PG_1_16.pdf
3857
3858
3859  PG_2_2:
3860  ▷  $(ORBITER) -v 2 \
3861  ▷  ▷  -define F -finite_field -q 2 -end \
3862  ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
3863  ▷  ▷  -with P -do -projective_space_activity \
3864  ▷  ▷  ▷  -cheat_sheet \
3865  ▷  ▷  -end
3866  ▷  pdflatex PG_2_2.tex
3867  ▷  $(OPEN) PG_2_2.pdf
3868
3869  PG_2_3:
3870  ▷  $(ORBITER) -v 2 \
3871  ▷  ▷  -define F -finite_field -q 3 -end \
3872  ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
3873  ▷  ▷  -with P -do -projective_space_activity \
3874  ▷  ▷  ▷  -cheat_sheet \
3875  ▷  ▷  -end
3876  ▷  pdflatex PG_2_3.tex
3877  ▷  $(OPEN) PG_2_3.pdf
3878
3879
3880  PG_2_4:
```

```
3881  ▷  $(ORBITER) -v 2 \
3882  ▷  ▷  -define F -finite_field -q 4 -end \
3883  ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
3884  ▷  ▷  -with P -do -projective_space_activity \
3885  ▷  ▷  ▷  -cheat_sheet \
3886  ▷  ▷  -end
3887  ▷  pdflatex PG_2_4.tex
3888  ▷  $(OPEN) PG_2_4.pdf
3889
3890
3891
3892
3893  PG_2_13:
3894  ▷  $(ORBITER) -v 2 \
3895  ▷  ▷  -define F -finite_field -q 13 -end \
3896  ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
3897  ▷  ▷  -with P -do -projective_space_activity \
3898  ▷  ▷  ▷  -cheat_sheet \
3899  ▷  ▷  -end
3900  ▷  pdflatex PG_2_13.tex
3901  ▷  $(OPEN) PG_2_13.pdf
3902
3903
3904
3905
3906  PG_2_64:
3907  ▷  $(ORBITER) -v 2 \
3908  ▷  ▷  -define F -finite_field -q 64 -end \
3909  ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
3910  ▷  ▷  -with P -do -projective_space_activity \
3911  ▷  ▷  ▷  -cheat_sheet \
3912  ▷  ▷  -end
3913  ▷  pdflatex PG_2_64.tex
3914  ▷  $(OPEN) PG_2_64.pdf
3915
3916
3917
3918  PG_3_2:
3919  ▷  $(ORBITER) -v 2 \
3920  ▷  ▷  -define F -finite_field -q 2 -end \
3921  ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
3922  ▷  ▷  -with P -do -projective_space_activity \
3923  ▷  ▷  ▷  -cheat_sheet \
3924  ▷  ▷  -end
3925  ▷  pdflatex PG_3_2.tex
3926  ▷  $(OPEN) PG_3_2.pdf
3927
3928
3929  PG_3_4:
3930  ▷  $(ORBITER) -v 2 \
3931  ▷  ▷  -define F -finite_field -q 4 -end \
3932  ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
3933  ▷  ▷  -with P -do -projective_space_activity \
3934  ▷  ▷  ▷  -cheat_sheet \
3935  ▷  ▷  -end
3936  ▷  pdflatex PG_3_4.tex
3937  ▷  $(OPEN) PG_3_4.pdf
3938
3939  PG_3_5:
```

```
3940  ▷  $(ORBITER) -v 2 \
3941  ▷  ▷  -define F -finite_field -q 5 -end \
3942  ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
3943  ▷  ▷  -with P -do -projective_space_activity \
3944  ▷  ▷  ▷  -cheat_sheet \
3945  ▷  ▷  -end
3946  ▷  pdflatex PG_3_5.tex
3947  ▷  $(OPEN) PG_3_5.pdf
3948
3949
3950  PG_3_7:
3951  ▷  $(ORBITER) -v 2 \
3952  ▷  ▷  -define F -finite_field -q 7 -end \
3953  ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
3954  ▷  ▷  -with P -do -projective_space_activity \
3955  ▷  ▷  ▷  -cheat_sheet \
3956  ▷  ▷  -end
3957  ▷  pdflatex PG_3_7.tex
3958  ▷  $(OPEN) PG_3_7.pdf
3959
3960
3961
3962
3963  PG_3_8:
3964  ▷  $(ORBITER) -v 2 \
3965  ▷  ▷  -define F -finite_field -q 8 -end \
3966  ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
3967  ▷  ▷  -with P -do -projective_space_activity \
3968  ▷  ▷  ▷  -cheat_sheet \
3969  ▷  ▷  -end
3970  ▷  pdflatex PG_3_8.tex
3971  ▷  $(OPEN) PG_3_8.pdf
3972
3973
3974  PG_3_16:
3975  ▷  $(ORBITER) -v 2 \
3976  ▷  ▷  -define F -finite_field -q 16 -end \
3977  ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
3978  ▷  ▷  -with P -do -projective_space_activity \
3979  ▷  ▷  ▷  -cheat_sheet \
3980  ▷  ▷  -end
3981  ▷  pdflatex PG_3_16.tex
3982  ▷  $(OPEN) PG_3_16.pdf
3983
3984
3985
3986
3987  PG_3_25:
3988  ▷  $(ORBITER) -v 2 \
3989  ▷  ▷  -define F -finite_field -q 25 -end \
3990  ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
3991  ▷  ▷  -with P -do -projective_space_activity \
3992  ▷  ▷  ▷  -cheat_sheet \
3993  ▷  ▷  -end
3994  ▷  pdflatex PG_3_25.tex
3995  ▷  $(OPEN) PG_3_25.pdf
3996
3997
3998
```

```
3999
4000  PG_4_3:
4001  ▷  $(ORBITER) -v 2 \
4002  ▷  ▷  -define F -finite_field -q 3 -end \
4003  ▷  ▷  -define P -projective_space -n 4 -field F -v 0 -end \
4004  ▷  ▷  -with P -do -projective_space_activity \
4005  ▷  ▷  ▷  -cheat_sheet \
4006  ▷  ▷  -end
4007  ▷  pdflatex PG_4_3.tex
4008  ▷  $(OPEN) PG_4_3.pdf
4009
4010
4011  PG_8_2:
4012  ▷  $(ORBITER) -v 2 \
4013  ▷  ▷  -define F -finite_field -q 2 -end \
4014  ▷  ▷  -define P -projective_space -n 8 -field F -v 0 -end \
4015  ▷  ▷  -with P -do -projective_space_activity \
4016  ▷  ▷  ▷  -cheat_sheet \
4017  ▷  ▷  -end
4018  ▷  pdflatex PG_8_2.tex
4019  ▷  $(OPEN) PG_8_2.pdf
4020
4021
4022
4023
4024  ###############################################################################
4025  # Section 4.2: Indexing Points and Lines
4026
4027  SECTION_INDEXING_POINTS_AND_LINES:
4028
4029  test_4_2:
4030  ▷  make PG_2_4_rank_point
4031  ▷  make PG_2_4_unrank_point
4032  ▷  make PG_2_2_rank_lines
4033  ▷  make PG_2_2_unrank_lines
4034  ▷  make PG_3_3_rank_lines
4035  ▷  make PG_3_5_unrank_lines
4036
4037
4038  PG_2_4_rank_point:
4039  ▷  $(ORBITER) -v 2 \
4040  ▷  ▷  -define v -vector -dense "3,3,1" -format 1 -end \
4041  ▷  ▷  -define F -finite_field -q 4 -end \
4042  ▷  ▷  -with F -do -finite_field_activity \
4043  ▷  ▷  ▷  -rank_point_in_PG v -end
4044
4045  # geometry_global::do_rank_point_in_PG coeff: ( 3, 3, 1 ) has rank 20
4046
4047  PG_2_4_unrank_point:
4048  ▷  $(ORBITER) -v 2 \
4049  ▷  ▷  -define v -vector -dense "20" -end \
4050  ▷  ▷  -define F -finite_field -q 4 -end \
4051  ▷  ▷  -with F -do -finite_field_activity \
4052  ▷  ▷  ▷  -unrank_point_in_PG 2 v -end
4053
4054
4055
4056  PG_2_2_rank_lines:
4057  ▷  $(ORBITER) -v 2 \
```

```
4058  ▷  ▷  -define v -vector -format 4 \
4059  ▷  ▷  -dense "1,0,0, 0,1,0, 1,0,0, 0,0,1, 1,1,1, 0,1,0, 1,1,1, 0,0,1" \
4060  ▷  ▷  -end \
4061  ▷  ▷  -define F -finite_field -q 2 -end \
4062  ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
4063  ▷  ▷  -with P -do \
4064  ▷  ▷  -projective_space_activity \
4065  ▷  ▷  ▷  -rank_lines_in_PG v \
4066  ▷  ▷  -end \
4067
4068
4069  PG_2_2_unrank_lines:
4070  ▷  $(ORBITER) -v 2 \
4071  ▷  ▷  -define v -vector -format 4 \
4072  ▷  ▷  -dense "0,1,2,3,4,5,6" \
4073  ▷  ▷  -end \
4074  ▷  ▷  -define F -finite_field -q 2 -end \
4075  ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
4076  ▷  ▷  -with P -do \
4077  ▷  ▷  -projective_space_activity \
4078  ▷  ▷  ▷  -unrank_lines_in_PG v \
4079  ▷  ▷  -end \
4080
4081
4082
4083  PG_3_3_rank_lines:
4084  ▷  $(ORBITER) -v 2 \
4085  ▷  ▷  -define v1 -vector -format 3 \
4086  ▷  ▷  -dense "1,0,2,2,0,1,1,2, 1,0,2,0,0,1,1,2, 1,0,2,2,0,1,2,1" \
4087  ▷  ▷  -end \
4088  ▷  ▷  -define v2 -vector -format 3 \
4089  ▷  ▷  -dense "1,0,0,0,0,1,0,0, 1,0,0,0,0,0,0,1, 0,1,0,0,0,0,2,1" \
4090  ▷  ▷  -end \
4091  ▷  ▷  -define F -finite_field -q 3 -end \
4092  ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
4093  ▷  ▷  -with P -do \
4094  ▷  ▷  -projective_space_activity \
4095  ▷  ▷  ▷  -rank_lines_in_PG v1 \
4096  ▷  ▷  -end \
4097  ▷  ▷  -with P -do \
4098  ▷  ▷  -projective_space_activity \
4099  ▷  ▷  ▷  -rank_lines_in_PG v2 \
4100  ▷  ▷  -end
4101
4102
4103  PG_3_5_unrank_lines:
4104  ▷  $(ORBITER) -v 2 \
4105  ▷  ▷  -define v -vector \
4106  ▷  ▷  ▷  -dense "0,36,72,108,144,805" \
4107  ▷  ▷  -end \
4108  ▷  ▷  -define F -finite_field -q 5 -end \
4109  ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
4110  ▷  ▷  -with P -do \
4111  ▷  ▷  -projective_space_activity \
4112  ▷  ▷  ▷  -unrank_lines_in_PG v \
4113  ▷  ▷  -end
4114
4115
4116
```

```
4117
4118
4119
4120  ###############################################################################
4121  # Section 4.3: Incidence Matrices
4122
4123
4124  SECTION_INCIDENCE_MATRICES:
4125
4126
4127  test_4_3:
4128  ▷  make PG_2_16
4129  ▷  make PG_2_2_incidence_matrix
4130  ▷  make PG_2_4_incidence_matrix
4131  ▷  make PG_2_8_incidence_matrix
4132  ▷  make PG_2_4_with_decomposition
4133  ▷  make PG_2_4_incma_cyclic
4134  ▷  make PG_2_4_incma_singer_sub_3
4135  ▷  make PG_2_4_incma_singer_sub_7
4136
4137
4138  PG_2_16:
4139  ▷  $(ORBITER) -v 2 \
4140  ▷  ▷  -draw_options -xin 20000 -yin 20000 \
4141  ▷  ▷  -radius 200 -line_width 0.3 -nodes_empty -end \
4142  ▷  ▷  -define F -finite_field -q 16 -end \
4143  ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
4144  ▷  ▷  -with P -do -projective_space_activity \
4145  ▷  ▷  ▷  -cheat_sheet \
4146  ▷  ▷  -end
4147  ▷  pdflatex PG_2_16.tex
4148  ▷  $(OPEN) PG_2_16.pdf
4149
4150
4151
4152  PG_2_2_incidence_matrix:
4153  ▷  $(ORBITER) -v 2 \
4154  ▷  ▷  -define F -finite_field -q 2 -end \
4155  ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
4156  ▷  ▷  -with P -do -projective_space_activity \
4157  ▷  ▷  ▷  -export_point_line_incidence_matrix \
4158  ▷  ▷  -end
4159  ▷  $(ORBITER) -v 2 \
4160  ▷  ▷  -define all_one -vector -repeat 1 7 -end \
4161  ▷  ▷  -draw_matrix \
4162  ▷  ▷  ▷  -input_csv_file PG_n2_q2_incidence_matrix.csv \
4163  ▷  ▷  ▷  -box_width 20 -bit_depth 8 \
4164  ▷  ▷  ▷  -partition 3 \
4165  ▷  ▷  ▷  ▷  all_one all_one \
4166  ▷  ▷  -end
4167  ▷  $(OPEN) PG_n2_q2_incidence_matrix_draw.bmp
4168
4169
4170  PG_2_4_incidence_matrix:
4171  ▷  $(ORBITER) -v 2 \
4172  ▷  ▷  -define F -finite_field -q 4 -end \
4173  ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
4174  ▷  ▷  -with P -do -projective_space_activity \
4175  ▷  ▷  ▷  -export_point_line_incidence_matrix \
```

```
4176  ▷  ▷  -end
4177  ▷  $(ORBITER) -v 2 \
4178  ▷  ▷  -define all_one -vector -repeat 1 21 -end \
4179  ▷  ▷  -draw_matrix \
4180  ▷  ▷  ▷  -input_csv_file PG_n2_q4_incidence_matrix.csv \
4181  ▷  ▷  ▷  -box_width 20 -bit_depth 8 \
4182  ▷  ▷  ▷  -partition 3 \
4183  ▷  ▷  ▷  ▷  all_one all_one \
4184  ▷  ▷  -end
4185  ▷  $(OPEN) PG_n2_q4_incidence_matrix_draw.bmp
4186
4187  # writes PG_n2_q4_incidence_matrix.csv
4188
4189
4190
4191  PG_2_8_incidence_matrix:
4192  ▷  $(ORBITER) -v 2 \
4193  ▷  ▷  -define F -finite_field -q 8 -end \
4194  ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
4195  ▷  ▷  -with P -do -projective_space_activity \
4196  ▷  ▷  ▷  -export_point_line_incidence_matrix \
4197  ▷  ▷  -end
4198  ▷  $(ORBITER) -v 2 \
4199  ▷  ▷  -define all_one -vector -repeat 1 73 -end \
4200  ▷  ▷  -draw_matrix \
4201  ▷  ▷  ▷  -input_csv_file PG_n2_q8_incidence_matrix.csv \
4202  ▷  ▷  ▷  -box_width 20 -bit_depth 8 \
4203  ▷  ▷  ▷  -partition 3 \
4204  ▷  ▷  ▷  ▷  all_one all_one \
4205  ▷  ▷  -end
4206  ▷  $(OPEN) PG_n2_q8_incidence_matrix_draw.bmp
4207
4208
4209
4210  # ToDo:
4211
4212  PG_2_4_with_decomposition:
4213  ▷  $(ORBITER) -v 2 \
4214  ▷  ▷  -define F -finite_field -q 4 -end \
4215  ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
4216  ▷  ▷  -with P -do -projective_space_activity \
4217  ▷  ▷  ▷  -decomposition_by_element_PG \
4218  ▷  ▷  ▷  1 "0,1,0, 0,0,1, 2,1,1, 0" \
4219  ▷  ▷  ▷  "PG_2_4_singer" \
4220  ▷  ▷  -end
4221  ▷  pdflatex PG_2_4_singer.tex
4222  ▷  $(OPEN) PG_2_4_singer.pdf
4223
4224
4225  #PG_2_4_singer_incma_cyclic.csv
4226  #PG_2_4_singer_incma_subgroup_index_3.csv
4227  #PG_2_4_singer_incma_subgroup_index_7.csv
4228
4229
4230
4231  PG_2_4_incma_cyclic:
4232  ▷  $(ORBITER) -v 2 \
4233  ▷  ▷  -list_arguments \
4234  ▷  ▷  -define R -vector -repeat 1 21 -end \
```

```
4235  ▷  ▷  -define C -vector -repeat 1 21 -end \
4236  ▷  ▷  -draw_matrix \
4237  ▷  ▷  -input_csv_file PG_2_4_singer_incma_cyclic.csv \
4238  ▷  ▷  -box_width 40 -bit_depth 24 \
4239  ▷  ▷  -partition 3 R C \
4240  ▷  ▷  -end
4241  ▷  $(OPEN) PG_2_4_singer_incma_cyclic_draw.bmp
4242
4243
4244  PG_2_4_incma_singer_sub_3:
4245  ▷  $(ORBITER) -v 2 \
4246  ▷  ▷  -list_arguments \
4247  ▷  ▷  -define R -vector -repeat 3 7 -end \
4248  ▷  ▷  -define C -vector -repeat 3 7 -end \
4249  ▷  ▷  -draw_matrix \
4250  ▷  ▷  -input_csv_file PG_2_4_singer_incma_subgroup_index_3.csv \
4251  ▷  ▷  -box_width 40 -bit_depth 24 \
4252  ▷  ▷  -partition 3 R C \
4253  ▷  ▷  -end
4254  ▷  $(OPEN) PG_2_4_singer_incma_subgroup_index_3_draw.bmp
4255
4256  PG_2_4_incma_singer_sub_7:
4257  ▷  $(ORBITER) -v 2 \
4258  ▷  ▷  -draw_matrix \
4259  ▷  ▷  -input_csv_file PG_2_4_singer_incma_subgroup_index_7.csv \
4260  ▷  ▷  -box_width 20 -bit_depth 24 \
4261  ▷  ▷  -partition 3 3,3,3,3,3,3,3 3,3,3,3,3,3,3 -end
4262  ▷  $(OPEN) PG_2_4_singer_incma_subgroup_index_7_draw.bmp
4263
4264
4265
4266
4267
4268
4269
4270
4271
4272
4273  ################################################################################
4274  # Section 4.4: The Grassmannian
4275
4276
4277  SECTION_GRASSMANNIAN:
4278
4279
4280
4281  test_4_4:
4282  ▷  make GR_3_2_2
4283  ▷  make planes_in_pencil
4284
4285
4286
4287
4288
4289  GR_3_2_2:
4290  ▷  $(ORBITER) -v 2 \
4291  ▷  ▷  -define F -finite_field -q 2 -end \
4292  ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
4293  ▷  ▷  -with P -do \
```

```
4294  ▷  ▷  -projective_space_activity \
4295  ▷  ▷  ▷  -report_Grassmannian 2 \
4296  ▷  ▷  -end
4297  ▷  pdflatex Gr_3_2_2.tex
4298  ▷  $(OPEN) Gr_3_2_2.pdf
4299
4300
4301
4302
4303
4304
4305  planes_in_pencil:
4306  ▷  $(ORBITER) -v 2 \
4307  ▷  ▷  -define F -finite_field -q 8 -end \
4308  ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
4309  ▷  ▷  -with P -do \
4310  ▷  ▷  -projective_space_activity \
4311  ▷  ▷  ▷  -planes_through_line 0 \
4312  ▷  ▷  -end
4313
4314
4315
4316  ################################################################################
4317  # Section 4.5: Algebraic Sets
4318
4319
4320  SECTION_ALGEBRAIC_SETS:
4321
4322
4323  test_4_5:
4324  ▷  make Fermat_curve_F7
4325  ▷  make Hesse_pencil_F7
4326  ▷  make surface_F9
4327  ▷  make elliptic_curve_b1_c3_q11.txt
4328  ▷  make EC_11.txt
4329  ▷  make Hirschfeld_surface_q4.txt
4330  ▷  make Hirschfeld_surface_q16.txt
4331
4332
4333
4334  Fermat_curve_F7:
4335  ▷  $(ORBITER) -v 6 \
4336  ▷  ▷  -define F -finite_field -q 7 -end \
4337  ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
4338  ▷  ▷  -define R -polynomial_ring \
4339  ▷  ▷  ▷  -field F \
4340  ▷  ▷  ▷  -number_of_variables 3 \
4341  ▷  ▷  ▷  -homogeneous_of_degree 3 \
4342  ▷  ▷  ▷  -monomial_ordering_lex \
4343  ▷  ▷  ▷  -variables "y0,y1,y2" "y_0,y_1,y_2" \
4344  ▷  ▷  ▷  -end \
4345  ▷  ▷  -define Fermat -symbolic_object \
4346  ▷  ▷  ▷  -field F \
4347  ▷  ▷  ▷  -text "y0^3+y1^3+y2^3" \
4348  ▷  ▷  ▷  -end \
4349  ▷  ▷  -with P -do \
4350  ▷  ▷  -projective_space_activity \
4351  ▷  ▷  ▷  -projective_variety R Fermat "" \
4352  ▷  ▷  -end
```

```
4353
4354
4355  Hesse_pencil_F7:
4356  ▷  $(ORBITER) -v 6 \
4357  ▷  ▷  -define F -finite_field -q 7 -end \
4358  ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
4359  ▷  ▷  -define R -polynomial_ring \
4360  ▷  ▷  ▷  -field F \
4361  ▷  ▷  ▷  -number_of_variables 3 \
4362  ▷  ▷  ▷  -homogeneous_of_degree 3 \
4363  ▷  ▷  ▷  -monomial_ordering_lex \
4364  ▷  ▷  ▷  -variables "y0,y1,y2" "y_0,y_1,y_2" \
4365  ▷  ▷  ▷  -end \
4366  ▷  ▷  -define Hesse_pencil -symbolic_object \
4367  ▷  ▷  ▷  -field F \
4368  ▷  ▷  ▷  -text "y0^3+y1^3+y2^3+y0*y1*y2" \
4369  ▷  ▷  ▷  -end \
4370  ▷  ▷  -with P -do \
4371  ▷  ▷  -projective_space_activity \
4372  ▷  ▷  ▷  -projective_variety R Hesse_pencil "" \
4373  ▷  ▷  -end
4374
4375
4376  surface_F9:
4377  ▷  $(ORBITER) -v 6 \
4378  ▷  ▷  -define F -finite_field -q 9 -print_numerically -end \
4379  ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
4380  ▷  ▷  -define R -polynomial_ring \
4381  ▷  ▷  ▷  -field F \
4382  ▷  ▷  ▷  -number_of_variables 4 \
4383  ▷  ▷  ▷  -homogeneous_of_degree 3 \
4384  ▷  ▷  ▷  -monomial_ordering_lex \
4385  ▷  ▷  ▷  -variables "X0,X1,X2,X3" "X_0,X_1,X_2,X_3" \
4386  ▷  ▷  ▷  -end \
4387  ▷  ▷  -define surface_F9 -symbolic_object \
4388  ▷  ▷  ▷  -field F \
4389  ▷  ▷  ▷  -text $(SURFACE_F9_1_EQN) \
4390  ▷  ▷  ▷  -end \
4391  ▷  ▷  -with P -do \
4392  ▷  ▷  -projective_space_activity \
4393  ▷  ▷  ▷  -projective_variety R surface_F9 "" \
4394  ▷  ▷  -end
4395
4396
4397  elliptic_curve_b1_c3_q11.txt:
4398  ▷  $(ORBITER) -v 2 \
4399  ▷  ▷  -define F -finite_field -q 11 -end \
4400  ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
4401  ▷  ▷  -define EC -geometric_object P \
4402  ▷  ▷  ▷  -elliptic_curve 1 3 \
4403  ▷  ▷  -end \
4404  ▷  ▷  -with EC -do -combinatorial_object_activity -save \
4405  ▷  ▷  -end
4406
4407
4408  EC_11.txt:
4409  ▷  $(ORBITER) -v 2 \
4410  ▷  ▷  -define F -finite_field -q 11 -end \
4411  ▷  ▷  -define R -polynomial_ring -field F \
```

```
4412  ▷  ▷  ▷  -number_of_variables 3 \
4413  ▷  ▷  ▷  -homogeneous_of_degree 3 \
4414  ▷  ▷  ▷  -end \
4415  ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
4416  ▷  ▷  -define EC -geometric_object P \
4417  ▷  ▷  ▷  -projective_variety R \
4418  ▷  ▷  ▷  ▷  "EC_11" "EC\_11" \
4419  ▷  ▷  ▷  ▷  $(EC_11_EQUATION) \
4420  ▷  ▷  ▷  -end \
4421  ▷  ▷  -with EC -do -combinatorial_object_activity -save \
4422  ▷  ▷  -end
4423
4424
4425
4426
4427  Hirschfeld_surface_q4.txt:
4428  ▷  $(ORBITER) -v 2 \
4429  ▷  ▷  -define F -finite_field -q 4 -end \
4430  ▷  ▷  -define R -polynomial_ring -field F \
4431  ▷  ▷  ▷  -number_of_variables 4 \
4432  ▷  ▷  ▷  -homogeneous_of_degree 3 \
4433  ▷  ▷  ▷  -end \
4434  ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
4435  ▷  ▷  -define H4 -geometric_object P \
4436  ▷  ▷  ▷  -projective_variety R \
4437  ▷  ▷  ▷  ▷  "Hirschfeld_surface_q4" \
4438  ▷  ▷  ▷  ▷  "Hirschfeld\_surface\_q4" \
4439  ▷  ▷  ▷  ▷  $(HIRSCHFELD_SURFACE_EQUATION) \
4440  ▷  ▷  ▷  -end \
4441  ▷  ▷  -with H4 -do -combinatorial_object_activity -save \
4442  ▷  ▷  -end
4443
4444  # creates Hirschfeld_surface_q4.txt
4445
4446  Hirschfeld_surface_q16.txt:
4447  ▷  $(ORBITER) -v 2 \
4448  ▷  ▷  -define F -finite_field -q 16 -end \
4449  ▷  ▷  -define R -polynomial_ring -field F \
4450  ▷  ▷  ▷  -number_of_variables 4 \
4451  ▷  ▷  ▷  -homogeneous_of_degree 3 \
4452  ▷  ▷  ▷  -end \
4453  ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
4454  ▷  ▷  -define H16 -geometric_object P \
4455  ▷  ▷  ▷  -projective_variety R \
4456  ▷  ▷  ▷  ▷  "Hirschfeld_surface_q16" \
4457  ▷  ▷  ▷  ▷  "Hirschfeld\_surface\_q16" \
4458  ▷  ▷  ▷  ▷  $(HIRSCHFELD_SURFACE_EQUATION) \
4459  ▷  ▷  ▷  -end \
4460  ▷  ▷  -with H16 -do -combinatorial_object_activity -save \
4461  ▷  ▷  -end
4462
4463
4464  # the coefficient vector is given as a list of pairs.
4465  # 165 = binomial(11,3)
4466
4467
4468
4469  ###############################################################################
4470  # Section 4.6: The Klein Quadric and Pluecker coordinates
```

```
4471
4472
4473  SECTION_KLEIN_QUADRIC_AND_PLUECKER_COORDINATES:
4474
4475
4476  test_4_6:
4477  ▷  make GR_4_2_2
4478
4479
4480  GR_4_2_2:
4481  ▷  $(ORBITER) -v 2 \
4482  ▷  ▷  -define F -finite_field -q 2 -end \
4483  ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
4484  ▷  ▷  -with P -do \
4485  ▷  ▷  -projective_space_activity \
4486  ▷  ▷  ▷  -report_Grassmannian 2 \
4487  ▷  ▷  -end
4488  ▷  pdflatex Gr_4_2_2.tex
4489  ▷  $(OPEN) Gr_4_2_2.pdf
4490
4491
4492  ################################################################################
4493  # Section 4.7: Orthogonal spaces
4494
4495
4496  SECTION_ORTHOGONAL_SPACES:
4497
4498
4499  test_4_7:
4500  ▷  make Op_4_2
4501  ▷  make O_5_2_incidence_matrix.csv
4502  ▷  make Op_6_2
4503  ▷  make Op_6_2_incidence_matrix.csv
4504  ▷  make Op_6_2_with_group
4505  ▷  make Op_6_8
4506  ▷  make Op_8_2
4507  ▷  make Op_6_64
4508  ▷  #make Op_6_64_line_rank_problem
4509  ▷  make Op_6_64_line_rank
4510  ▷  make Op_6_64_report
4511  ▷  #make elliptic_quadric_subspace  fails
4512  ▷  make BLT_database_7_1
4513  ▷  make BLT_database_7_1_print
4514  ▷  make Doily_W_2
4515
4516
4517
4518  Op_4_2:
4519  ▷  $(ORBITER) -v 2 \
4520  ▷  ▷  -define F -finite_field -q 2 -end \
4521  ▷  ▷  -define O -orthogonal_space 1 4 F -without_group -end \
4522  ▷  ▷  -with O -do -orthogonal_space_activity \
4523  ▷  ▷  ▷  -cheat_sheet_orthogonal -end
4524  ▷  pdflatex O_1_4_2_report.tex
4525  ▷  $(OPEN) O_1_4_2_report.pdf
4526
4527
4528  O_5_2_incidence_matrix.csv:
4529  ▷  $(ORBITER) -v 2 \
```

```
4530  ▷  ▷  -define F -finite_field -q 2 -end \
4531  ▷  ▷  -define O -orthogonal_space 0 5 F -without_group -end \
4532  ▷  ▷  -with O -do -orthogonal_space_activity \
4533  ▷  ▷  ▷  -export_point_line_incidence_matrix \
4534  ▷  ▷  -end
4535  ▷  $(ORBITER) -v 2 \
4536  ▷  ▷  -define all_one_r -vector -repeat 1 15 -end \
4537  ▷  ▷  -define all_one_c -vector -repeat 1 15 -end \
4538  ▷  ▷  -draw_matrix \
4539  ▷  ▷  ▷  -input_csv_file O_5_2_incidence_matrix.csv \
4540  ▷  ▷  ▷  -box_width 20 -bit_depth 8 \
4541  ▷  ▷  ▷  -partition 2 \
4542  ▷  ▷  ▷  ▷  all_one_r all_one_c \
4543  ▷  ▷  -end
4544  ▷  $(OPEN) O_5_2_incidence_matrix_draw.bmp
4545
4546  #O(5,2) projectively = Q(4,2) = (dual of) W(3,2) = W(3,2)
4547  # recall that W(3,2) and Q(4,q) are self dual if q is even
4548
4549
4550  Op_6_2:
4551  ▷  $(ORBITER) -v 2 \
4552  ▷  ▷  -define F -finite_field -q 2 -end \
4553  ▷  ▷  -define O -orthogonal_space 1 6 F -without_group -end \
4554  ▷  ▷  -with O -do -orthogonal_space_activity \
4555  ▷  ▷  ▷  -cheat_sheet_orthogonal -end
4556  ▷  pdflatex O_1_6_2_report.tex
4557  ▷  $(OPEN) O_1_6_2_report.pdf
4558
4559
4560  Op_6_2_incidence_matrix.csv:
4561  ▷  $(ORBITER) -v 2 \
4562  ▷  ▷  -define F -finite_field -q 2 -end \
4563  ▷  ▷  -define O -orthogonal_space 1 6 F -without_group -end \
4564  ▷  ▷  -with O -do -orthogonal_space_activity \
4565  ▷  ▷  ▷  -export_point_line_incidence_matrix \
4566  ▷  ▷  -end
4567  ▷  $(ORBITER) -v 2 \
4568  ▷  ▷  -define all_one_r -vector -repeat 1 35 -end \
4569  ▷  ▷  -define all_one_c -vector -repeat 1 105 -end \
4570  ▷  ▷  -draw_matrix \
4571  ▷  ▷  ▷  -input_csv_file Op_6_2_incidence_matrix.csv \
4572  ▷  ▷  ▷  -box_width 20 -bit_depth 8 \
4573  ▷  ▷  ▷  -partition 2 \
4574  ▷  ▷  ▷  ▷  all_one_r all_one_c \
4575  ▷  ▷  -end
4576  ▷  $(OPEN) Op_6_2_incidence_matrix_draw.bmp
4577
4578
4579  Op_6_2_with_group:
4580  ▷  $(ORBITER) -v 2 \
4581  ▷  ▷  -define F -finite_field -q 2 -end \
4582  ▷  ▷  -define O -orthogonal_space 1 6 F -end \
4583  ▷  ▷  -with O -do -orthogonal_space_activity \
4584  ▷  ▷  ▷  -cheat_sheet_orthogonal -end
4585  ▷  pdflatex O_1_6_2_report.tex
4586  ▷  $(OPEN) O_1_6_2_report.pdf
4587
4588  # problem:
```

```
4589  # error message:
4590  #stabilizer_chain_base_data::allocate_base_data degree is too large
4591
4592
4593  Op_6_8:
4594  ▷  $(ORBITER) -v 2 \
4595  ▷  ▷  -define F -finite_field -q 8 -end \
4596  ▷  ▷  -define O -orthogonal_space 1 6 F -without_group -end \
4597  ▷  ▷  -with O -do -orthogonal_space_activity \
4598  ▷  ▷  ▷  -cheat_sheet_orthogonal \
4599  ▷  ▷  -end
4600  ▷  pdflatex O_1_6_8_report.tex
4601  ▷  $(OPEN) O_1_6_8_report.pdf
4602
4603
4604  Op_8_2:
4605  ▷  $(ORBITER) -v 2 \
4606  ▷  ▷  -define F -finite_field -q 2 -end \
4607  ▷  ▷  -define O -orthogonal_space 1 8 F -without_group -end \
4608  ▷  ▷  -with O -do -orthogonal_space_activity \
4609  ▷  ▷  ▷  -cheat_sheet_orthogonal -end
4610  ▷  pdflatex O_1_8_2_report.tex
4611  ▷  $(OPEN) O_1_8_2_report.pdf
4612
4613
4614  Op_6_64:
4615  ▷  $(ORBITER) -v 2 \
4616  ▷  ▷  -define F -finite_field -q 64 -end \
4617  ▷  ▷  -define O -orthogonal_space 1 6 F -without_group -end \
4618  ▷  ▷  -with O -do -orthogonal_space_activity \
4619  ▷  ▷  ▷  -cheat_sheet_orthogonal -end
4620  ▷  pdflatex O_1_6_64_report.tex
4621  ▷  $(OPEN) O_1_6_64_report.pdf
4622
4623
4624  # problem, because we are trying to create PGL(6,64):
4625
4626  Op_6_64_line_rank_problem:
4627  ▷  $(ORBITER) -v 4 \
4628  ▷  ▷  -define F -finite_field -q 64 -end \
4629  ▷  ▷  -define O -orthogonal_space 1 6 F -end \
4630  ▷  ▷  -with O -do -orthogonal_space_activity \
4631  ▷  ▷  ▷  -unrank_line_through_two_points 15447347 15225451 \
4632  ▷  ▷  -end
4633
4634  # use option -without_group to skip the group. This will work:
4635
4636  Op_6_64_line_rank:
4637  ▷  $(ORBITER) -v 4 \
4638  ▷  ▷  -define F -finite_field -q 64 -end \
4639  ▷  ▷  -define O -orthogonal_space 1 6 F -without_group -end \
4640  ▷  ▷  -with O -do -orthogonal_space_activity \
4641  ▷  ▷  ▷  -unrank_line_through_two_points 15447347 15225451 \
4642  ▷  ▷  -end
4643
4644  # this will create a basic report without the group:
4645
4646  Op_6_64_report:
4647  ▷  $(ORBITER) -v 4 \
```

```
4648  ▷  ▷  -define F -finite_field -q 64 -end \
4649  ▷  ▷  -define O -orthogonal_space 1 6 F -without_group -end \
4650  ▷  ▷  -with O -do -orthogonal_space_activity \
4651  ▷  ▷  ▷  -cheat_sheet_orthogonal \
4652  ▷  ▷  -end
4653  ▷  pdflatex O_1_6_64_report.tex
4654  ▷  $(OPEN) O_1_6_64_report.pdf
4655
4656
4657
4658  # ToDo: this command fails:
4659
4660  elliptic_quadric_subspace:
4661  ▷  $(ORBITER) -v 3 \
4662  ▷  ▷  -define F -finite_field -q 5 -end \
4663  ▷  ▷  -define v -vector -format 4 \
4664  ▷  ▷  ▷  -dense "1,3,0,0,0,0, 0,0,1,0,0,0, 0,0,0,1,0,0, 0,0,0,0,1,1" \
4665  ▷  ▷  -end \
4666  ▷  ▷  -define O -orthogonal_space 1 6 F -end \
4667  ▷  ▷  -with O -do -orthogonal_space_activity \
4668  ▷  ▷  ▷  -intersect_with_subspace v \
4669  ▷  ▷  -end
4670
4671  #We found that the intersection contains 26 points:
4672  #( 2, 3, 1572, 100, 98, 116, 132, 1570, 1588, 1604, 1631, 1688, 1704, 1706, 1747,
       1745, 1847, 1845, 1856, 1888, 1904, 1931, 1988, 2004, 2022, 2020 )
4673
4674
4675
4676  BLT_database_7_1:
4677  ▷  $(ORBITER) -v 2 \
4678  ▷  ▷  -define F -finite_field -q 7 -end \
4679  ▷  ▷  -define P -projective_space -n 4 -field F -v 0 -end \
4680  ▷  ▷  -define S -geometric_object P \
4681  ▷  ▷  ▷  -BLT_database 1 \
4682  ▷  ▷  -end \
4683  ▷  ▷  -with S -do -combinatorial_object_activity -save \
4684  ▷  ▷  -end
4685
4686  # writes BLT_7_1.txt
4687
4688  BLT_database_7_1_print:
4689  ▷  $(ORBITER) -v 2 \
4690  ▷  ▷  -define F -finite_field -q 7 -end \
4691  ▷  ▷  -define O -orthogonal_space 0 5 F -without_group -end \
4692  ▷  ▷  -define S -set -file_orbiter_format BLT_7_1.txt -end \
4693  ▷  ▷  -with O -do -orthogonal_space_activity \
4694  ▷  ▷  ▷  -print_points S -end
4695  ▷  pdflatex S_set_report.tex
4696  ▷  $(OPEN) S_set_report.pdf
4697
4698  Doily_W_2:
4699  ▷  $(ORBITER) -v 2 \
4700  ▷  ▷  -define F -finite_field -q 2 -end \
4701  ▷  ▷  -define O -orthogonal_space 0 5 F -without_group -end \
4702  ▷  ▷  -define W2_points -set -loop 0 15 1 -end \
4703  ▷  ▷  -define W2_lines -set -loop 0 15 1 -end \
4704  ▷  ▷  -with O -do \
4705  ▷  ▷  -orthogonal_space_activity \
```

```
4706  ▷  ▷  ▷  -print_points W2_points \
4707  ▷  ▷  -end \
4708  ▷  ▷  -with O -do \
4709  ▷  ▷  -orthogonal_space_activity \
4710  ▷  ▷  ▷  -print_lines W2_lines \
4711  ▷  ▷  -end
4712  ▷  pdflatex W2_points_set_report.tex
4713  ▷  $(OPEN) W2_points_set_report.pdf
4714  ▷  pdflatex W2_lines_set_of_lines_report.tex
4715  ▷  $(OPEN) W2_lines_set_of_lines_report.pdf
4716
4717
4718  # the output defines doily.csv
4719
4720
4721
4722
4723
4724  ##############################################################################
4725  # Section 4.8: Hermitian varieties
4726
4727
4728  SECTION_HERMITIAN_VARIETIES:
4729
4730  test_4_8:
4731  ▷  make H_2_4
4732  ▷  make H_2_9
4733  ▷  make H_3_4
4734
4735
4736  H_2_4:
4737  ▷  $(ORBITER) -v 2 \
4738  ▷  ▷  -define F -finite_field -q 4 -end \
4739  ▷  ▷  -with F -do -finite_field_activity \
4740  ▷  ▷  ▷  -cheat_sheet_hermitian 2 -end
4741  ▷  pdflatex H_2_4.tex
4742  ▷  $(OPEN) H_2_4.pdf
4743
4744
4745  H_2_9:
4746  ▷  $(ORBITER) -v 2 \
4747  ▷  ▷  -define F -finite_field -q 9 -end \
4748  ▷  ▷  -with F -do -finite_field_activity \
4749  ▷  ▷  ▷  -cheat_sheet_hermitian 2 -end
4750  ▷  pdflatex H_2_9.tex
4751  ▷  $(OPEN) H_2_9.pdf
4752
4753
4754  # 28 points: 6, 11, 9, 7, 14, 19, 17, 15, 80, 75, 78, 74, 35, 30, 33, 29, 62, 57,
       60, 56, 26, 21, 24, 3, 37, 82, 64, 46
4755
4756
4757  H_3_4:
4758  ▷  $(ORBITER) -v 2 \
4759  ▷  ▷  -define F -finite_field -q 4 -end \
4760  ▷  ▷  -with F -do -finite_field_activity \
4761  ▷  ▷  ▷  -cheat_sheet_hermitian 3 -end
4762  ▷  pdflatex H_3_4.tex
4763  ▷  $(OPEN) H_3_4.pdf
```

```
4764
4765
4766  # H_3_4 = the Hirschfeld surface
4767
4768
4769  ################################################################################
4770  # Section 4.9: Projective Space Advanced Topics
4771
4772
4773  SECTION_PROJECTIVE_SPACE_ADVANCED_TOPICS:
4774
4775
4776  test_4_9:
4777  ▷  make fixed_objects_2A
4778  ▷  make fixed_objects_2B
4779  ▷  make fixed_objects_2C
4780  ▷  make fix_structure_G1
4781  ▷  make trans
4782  ▷  make PG_2_3_plane_pencil
4783
4784
4785
4786  fixed_objects_2A:
4787  ▷  $(ORBITER) -v 2 \
4788  ▷  ▷  -define F -finite_field -q 4 -end \
4789  ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
4790  ▷  ▷  -with P -do \
4791  ▷  ▷  -projective_space_activity \
4792  ▷  ▷  ▷  -report_fixed_objects \
4793  ▷  ▷  ▷  ▷  "1,0,0,0, 0,1,0,0, 0,0,1,0, 0,0,0,1, 1" \
4794  ▷  ▷  ▷  ▷  fix_structure_2A.tex \
4795  ▷  ▷  -end
4796  ▷  pdflatex fix_structure_2A.tex
4797  ▷  $(OPEN) fix_structure_2A.pdf
4798
4799
4800  fixed_objects_2B:
4801  ▷  $(ORBITER) -v 2 \
4802  ▷  ▷  -define F -finite_field -q 4 -end \
4803  ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
4804  ▷  ▷  -with P -do \
4805  ▷  ▷  -projective_space_activity \
4806  ▷  ▷  ▷  -report_fixed_objects \
4807  ▷  ▷  ▷  ▷  "1,0,0,0, 1,1,0,0, 0,0,1,0, 0,0,0,1, 0" \
4808  ▷  ▷  ▷  ▷  fix_structure_2B.tex \
4809  ▷  ▷  -end
4810  ▷  pdflatex fix_structure_2B.tex
4811  ▷  $(OPEN) fix_structure_2B.pdf
4812
4813
4814  fixed_objects_2C:
4815  ▷  $(ORBITER) -v 2 \
4816  ▷  ▷  -define F -finite_field -q 4 -end \
4817  ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
4818  ▷  ▷  -with P -do \
4819  ▷  ▷  -projective_space_activity \
4820  ▷  ▷  ▷  -report_fixed_objects \
4821  ▷  ▷  ▷  ▷  "1,0,0,0, 1,1,0,0, 0,0,1,0, 0,0,1,1, 0" \
4822  ▷  ▷  ▷  ▷  fix_structure_2C.tex \
```

```
4823  ▷  ▷  -end
4824  ▷  pdflatex fix_structure_2C.tex
4825  ▷  $(OPEN) fix_structure_2C.pdf
4826
4827
4828
4829
4830  fix_structure_G1:
4831  ▷  $(ORBITER) -v 2 \
4832  ▷  ▷  -define F -finite_field -q 4 -end \
4833  ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
4834  ▷  ▷  -with P -do \
4835  ▷  ▷  -projective_space_activity \
4836  ▷  ▷  ▷  -decomposition_by_subgroup \
4837  ▷  ▷  ▷  ▷  $(H1_LABEL) $(H1_GENS)  \
4838  ▷  ▷  -end
4839  ▷  pdflatex PGGL_4_4_Subgroup_$(H1_LABEL)_4_decomp.tex
4840  ▷  $(OPEN) PGGL_4_4_Subgroup_$(H1_LABEL)_4_decomp.pdf
4841
4842
4843
4844
4845  trans:
4846  ▷  $(ORBITER) -v 5 \
4847  ▷  ▷  -define F -finite_field -q 16 -end \
4848  ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
4849  ▷  ▷  -with P -do \
4850  ▷  ▷  -projective_space_activity \
4851  ▷  ▷  ▷  -move_two_lines_in_hyperplane_stabilizer_text \
4852  ▷  ▷  ▷  ▷  "1,0,0,0, 0,0,0,1" "1,1,0,2, 0,0,1,0" \
4853  ▷  ▷  ▷  ▷  "1,0,0,0, 0,0,0,1" "0,1,0,1, 0,0,1,0" \
4854  ▷  ▷  -end
4855
4856
4857  PG_2_3_plane_pencil:
4858  ▷  $(ORBITER) -v 2 \
4859  ▷  ▷  -define F -finite_field -q 2 -end \
4860  ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
4861  ▷  ▷  -with P -do \
4862  ▷  ▷  -projective_space_activity \
4863  ▷  ▷  ▷  -planes_through_line 0 \
4864  ▷  ▷  -end \
4865
4866
4867
4868
4869  ##############################################################################
4870  # Section 4.10: Geometric Objects
4871
4872
4873  SECTION_GEOMETRIC_OBJECTS:
4874
4875
4876
4877  test_4_10:
4878  ▷  make elliptic_quadric_ovoid_q8
4879  ▷  make ovoid_ST_q8
4880  ▷  make Baer_PG_2_4
4881  ▷  make Baer_PG_3_4
```

```
4882  ▷  make BLT_database_5_0
4883  ▷  make BLT_database_7_0
4884  ▷  make BLT_database_7_0_print
4885  ▷  make BLT_database_67_4
4886  ▷  make Doily_disjoint_sets_graph_cliques_3
4887  ▷  make Doily_disjoint_sets_graph_cliques_5
4888  ▷  make PG_3_2_test
4889  ▷  make Edge_curve_17
4890  ▷  make Edge_curve_17_line_type
4891  ▷  make Edge_curve_q23_line_type
4892
4893
4894
4895  elliptic_quadric_ovoid_q8:
4896  ▷  $(ORBITER) -v 2 \
4897  ▷  ▷  -define F -finite_field -q 8 -end \
4898  ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
4899  ▷  ▷  -define O -geometric_object P \
4900  ▷  ▷  ▷  -elliptic_quadric_ovoid \
4901  ▷  ▷  -end \
4902  ▷  ▷  -with O -do -combinatorial_object_activity -save \
4903  ▷  ▷  -end
4904
4905  #ovoid_q8.txt
4906  # 65 points
4907
4908  ovoid_ST_q8:
4909  ▷  $(ORBITER) -v 2 \
4910  ▷  ▷  -define F -finite_field -q 8 -end \
4911  ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
4912  ▷  ▷  -define O -geometric_object P \
4913  ▷  ▷  ▷  -ovoid_ST \
4914  ▷  ▷  -end \
4915  ▷  ▷  -with O -do -combinatorial_object_activity -save \
4916  ▷  ▷  -end
4917
4918  #ovoid_ST_q8.txt
4919
4920
4921  Baer_PG_2_4:
4922  ▷  $(ORBITER) -v 2 \
4923  ▷  ▷  -define F -finite_field -q 4 -end \
4924  ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
4925  ▷  ▷  -define O -geometric_object P \
4926  ▷  ▷  ▷  -Baer_substructure \
4927  ▷  ▷  -end \
4928  ▷  ▷  -with O -do -combinatorial_object_activity -save \
4929  ▷  ▷  -end
4930
4931  Baer_PG_3_4:
4932  ▷  $(ORBITER) -v 2 \
4933  ▷  ▷  -define F -finite_field -q 4 -end \
4934  ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
4935  ▷  ▷  -define O -geometric_object P \
4936  ▷  ▷  ▷  -Baer_substructure \
4937  ▷  ▷  -end \
4938  ▷  ▷  -with O -do -combinatorial_object_activity -save \
4939  ▷  ▷  -end
4940
```

```
4941 BLT_database_5_0:
4942 ▷  $(ORBITER) -v 2 \
4943 ▷  ▷  -define F -finite_field -q 5 -end \
4944 ▷  ▷  -define P -projective_space -n 4 -field F -v 0 -end \
4945 ▷  ▷  -define O -geometric_object P \
4946 ▷  ▷  ▷  -BLT_database 0 \
4947 ▷  ▷  -end \
4948 ▷  ▷  -with O -do -combinatorial_object_activity -save \
4949 ▷  ▷  -end
4950
4951 # writes BLT_5_0.txt
4952
4953
4954 BLT_database_7_0:
4955 ▷  $(ORBITER) -v 2 \
4956 ▷  ▷  -define F -finite_field -q 7 -end \
4957 ▷  ▷  -define P -projective_space -n 4 -field F -v 0 -end \
4958 ▷  ▷  -define O -geometric_object P \
4959 ▷  ▷  ▷  -BLT_database 0 \
4960 ▷  ▷  -end \
4961 ▷  ▷  -with O -do -combinatorial_object_activity -save \
4962 ▷  ▷  -end
4963
4964 # writes BLT_7_0.txt
4965
4966
4967 BLT_database_7_0_print:
4968 ▷  $(ORBITER) -v 2 \
4969 ▷  ▷  -define F -finite_field -q 7 -end \
4970 ▷  ▷  -define P -projective_space -n 4 -field F -v 0 -end \
4971 ▷  ▷  -define Obj -geometric_object P \
4972 ▷  ▷  ▷  -BLT_database 0 \
4973 ▷  ▷  -end \
4974 ▷  ▷  -with Obj -do -combinatorial_object_activity -save \
4975 ▷  ▷  -end \
4976 ▷  ▷  -define O -orthogonal_space 0 5 F -without_group -end \
4977 ▷  ▷  -define BLT_7_0 -set -file_orbiter_format BLT_7_0.txt -end \
4978 ▷  ▷  -with O -do -orthogonal_space_activity \
4979 ▷  ▷  ▷  -print_points BLT_7_0 -end
4980 ▷  pdflatex BLT_7_0_set_report.tex
4981 ▷  $(OPEN) BLT_7_0_set_report.pdf
4982
4983
4984 BLT_database_67_4:
4985 ▷  $(ORBITER) -v 2 \
4986 ▷  ▷  -define F -finite_field -q 67 -end \
4987 ▷  ▷  -define P -projective_space -n 4 -field F -v 0 -end \
4988 ▷  ▷  -define Obj -geometric_object P \
4989 ▷  ▷  ▷  -BLT_database 4 \
4990 ▷  ▷  -end \
4991 ▷  ▷  -with Obj -do -combinatorial_object_activity -save \
4992 ▷  ▷  -end \
4993 ▷  ▷  -define O -orthogonal_space 0 5 F -without_group -end \
4994 ▷  ▷  -define BLT_67_4 -set -file_orbiter_format BLT_67_4.txt -end \
4995 ▷  ▷  -with O -do -orthogonal_space_activity \
4996 ▷  ▷  ▷  -print_points BLT_67_4 -end
4997 ▷  pdflatex BLT_67_4_set_report.tex
4998 ▷  $(OPEN) BLT_67_4_set_report.pdf
4999
```

```
5000
5001
5002
5003
5004
5005
5006
5007  Doily_disjoint_sets_graph_cliques_3:
5008  ▷  echo $(DOILY) >doily.csv
5009  ▷  $(ORBITER) -v 2 \
5010  ▷  ▷  -define G -graph -disjoint_sets_graph \
5011  ▷  ▷  ▷  doily.csv \
5012  ▷  ▷  -end \
5013  ▷  ▷  -with G -do \
5014  ▷  ▷  ▷  -graph_theoretic_activity \
5015  ▷  ▷  ▷  -find_cliques \
5016  ▷  ▷  ▷  ▷  -target_size 3 \
5017  ▷  ▷  ▷  ▷  -output_file doily_cliques \
5018  ▷  ▷  ▷  -end \
5019  ▷  ▷  -end \
5020  ▷  ▷  -print_symbols
5021  ▷  $(ORBITER) -v 2 \
5022  ▷  ▷  -union doily.csv doily_cliques.txt doily_cliques_union.csv
5023
5024  # 80 cliques
5025
5026  Doily_disjoint_sets_graph_cliques_5:
5027  ▷  echo $(DOILY) >doily.csv
5028  ▷  $(ORBITER) -v 2 \
5029  ▷  ▷  -define G -graph -disjoint_sets_graph \
5030  ▷  ▷  ▷  doily.csv \
5031  ▷  ▷  -end \
5032  ▷  ▷  -with G -do \
5033  ▷  ▷  ▷  -graph_theoretic_activity \
5034  ▷  ▷  ▷  -find_cliques \
5035  ▷  ▷  ▷  ▷  -target_size 5 \
5036  ▷  ▷  ▷  ▷  -output_file doily_cliques_5 \
5037  ▷  ▷  ▷  -end \
5038  ▷  ▷  -end \
5039  ▷  ▷  -print_symbols
5040
5041  # 6 cliques
5042  # doily_cliques_5.csv
5043
5044
5045  PG_3_2_test:
5046  ▷  $(ORBITER) -v 2 \
5047  ▷  ▷  -define F -finite_field -q 2 -end \
5048  ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
5049  ▷  ▷  -with P -do -projective_space_activity \
5050  ▷  ▷  ▷  -cheat_sheet \
5051  ▷  ▷  -end
5052  ▷  pdflatex PG_3_2.tex
5053  ▷  $(OPEN) PG_3_2.pdf
5054
5055
5056  Edge_curve_17:
5057  ▷  $(ORBITER) -v 2 \
5058  ▷  ▷  -define F -finite_field -q 17 -end \
```

```
5059  ▷  ▷  -define R -polynomial_ring -field F \
5060  ▷  ▷  ▷  -number_of_variables 3 \
5061  ▷  ▷  ▷  -homogeneous_of_degree 4 \
5062  ▷  ▷  ▷  -end \
5063  ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
5064  ▷  ▷  -define C -geometric_object P \
5065  ▷  ▷  ▷  -projective_variety R \
5066  ▷  ▷  ▷  ▷  "Edge_q17" "Edge\_q17" \
5067  ▷  ▷  ▷  ▷  $(EDGE_CURVE_Q17_EQUATION) \
5068  ▷  ▷  -end \
5069  ▷  ▷  -with C -do -combinatorial_object_activity -save \
5070  ▷  ▷  -end
5071
5072  #Edge_q17.txt
5073  #combinatorial_object_create::init created a set of size 12
5074
5075
5076
5077
5078
5079
5080
5081
5082  Edge_curve_17_line_type:
5083  ▷  echo $(FILE_Q17) >edge_q17.csv
5084  ▷  $(ORBITER) -v 2 \
5085  ▷  ▷  -define F -finite_field -q 17 -end \
5086  ▷  ▷  -define R -polynomial_ring -field F \
5087  ▷  ▷  ▷  -number_of_variables 3 \
5088  ▷  ▷  ▷  -homogeneous_of_degree 4 \
5089  ▷  ▷  ▷  -end \
5090  ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
5091  ▷  ▷  -define C -geometric_object P \
5092  ▷  ▷  ▷  -projective_variety R \
5093  ▷  ▷  ▷  ▷  "Edge_q17" "Edge\_q17" \
5094  ▷  ▷  ▷  ▷  $(EDGE_CURVE_Q17_EQUATION) \
5095  ▷  ▷  ▷  -end \
5096  ▷  ▷  -with C -do \
5097  ▷  ▷  -combinatorial_object_activity \
5098  ▷  ▷  ▷  -line_type_old \
5099  ▷  ▷  -end \
5100  ▷  ▷  -print_symbols
5101
5102  #( 4^6, 2^30, 1^132, 0^139 )
5103
5104
5105  Edge_curve_q23_line_type:
5106  ▷  $(ORBITER) -v 2 \
5107  ▷  ▷  -define F -finite_field -q 23 -end \
5108  ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
5109  ▷  ▷  -define E -geometric_object P \
5110  ▷  ▷  ▷  -set "edge_curve_q23" "edge\_curve\_q23" \
5111  ▷  ▷  ▷  $(EDGE_CURVE_Q23_AS_POINTS) \
5112  ▷  ▷  -end \
5113  ▷  ▷  -with E -do \
5114  ▷  ▷  -combinatorial_object_activity \
5115  ▷  ▷  ▷  -save \
5116  ▷  ▷  -end \
5117  ▷  ▷  -with E -do \
```

```
5118  ▷  ▷  -combinatorial_object_activity \
5119  ▷  ▷  ▷  -line_type_old \
5120  ▷  ▷  -end \
5121  ▷  ▷  -print_symbols
5122
5123
5124
5125
5126
5127  ###############################################################################
5128  # Chapter 5 - Ring Theory
5129  ###############################################################################
5130
5131
5132  test_5:
5133  ▷  make test_5_1
5134  ▷  make test_5_2
5135  ▷  make test_5_3
5136
5137
5138
5139
5140  ###############################################################################
5141  # Section 5.1: Polynomials over Finite Fields
5142
5143  SECTION_POLYNOMIALS:
5144
5145
5146  test_5_1:
5147  ▷  make sift_polynomials_deg3_q2
5148  ▷  make sift_polynomials_deg4_q2
5149  ▷  make poly_division
5150  ▷  make poly_division2
5151  ▷  make poly_gcd
5152  ▷  make poly_mult_mod1
5153  ▷  make poly_mult_mod2
5154  ▷  make poly_mult_mod_F4
5155  ▷  make mult_polynomials_2_5_7
5156  ▷  make polynomial_division_ranked_2_27_13
5157  ▷  make mult_polynomials_2_8_15
5158  ▷  make polynomial_division_ranked_2_120_25
5159  ▷  make mult_polynomials_2_7_7
5160  ▷  make mult_polynomials_2_4_6
5161  ▷  make polynomial_division_ranked_2_24_13
5162  ▷  make mult_polynomials_1024_999_997
5163  ▷  make polynomial_division_ranked_2_349147_1033
5164  ▷  make mult_polynomials_17_12
5165  ▷  make polynomial_division_ranked_2_204_37
5166  ▷  make Berlekamp_matrix_crc32
5167  ▷  make power_mod_inverse
5168  ▷  make mult_mod_to_get_one
5169  ▷  make Berlekamp_matrix_2_3
5170  ▷  make Berlekamp_matrix_2_4
5171  ▷  make Berlekamp_matrix_4_3a
5172  ▷  make Berlekamp_matrix_4_3b
5173  ▷  make find_roots_a
5174  ▷  make find_roots_b
5175  ▷  make find_roots_c
```

```
5177  ▷  make find_roots_d
5178  ▷  make find_roots_e
5179  ▷  make roots_over_F2
5180  ▷  make roots_over_F8
5181  ▷  make irred_3_2
5182  ▷  make irred_4_2
5183  ▷  make irred_5_2
5184  ▷  make irred_6_2
5185  ▷  make irred_7_2
5186  ▷  make irred_8_2
5187  ▷  make irred_9_2
5188  ▷  make irred_10_2
5189  ▷  make irred_2_4
5190  ▷  make irred_3_4
5191  ▷  make F2_get_primitive_poly_of_deg_10
5192  ▷  make F2_get_primitive_poly_range_2_10
5193
5194
5195  # check which polynomials are irreducible and which are primitive:
5196
5197  sift_polynomials_deg3_q2:
5198  ▷  $(ORBITER) -v 2 \
5199  ▷  ▷  -define F -finite_field -q 2 -end \
5200  ▷  ▷  -with F -do \
5201  ▷  ▷  -finite_field_activity -sift_polynomials 8 16 -end
5202
5203
5204
5205  sift_polynomials_deg4_q2:
5206  ▷  $(ORBITER) -v 2 \
5207  ▷  ▷  -define F -finite_field -q 2 -end \
5208  ▷  ▷  -with F -do \
5209  ▷  ▷  -finite_field_activity -sift_polynomials 16 32 -end
5210
5211
5212
5213
5214  poly_division:
5215  ▷  $(ORBITER) -v 2 \
5216  ▷  ▷  -define F -finite_field -q 2 -end \
5217  ▷  ▷  -with F -do \
5218  ▷  ▷  -finite_field_activity \
5219  ▷  ▷  -polynomial_division "1,0,0,0,0,0,0,0,0,0,1" "1,0,1,1" -end
5220
5221  poly_division2:
5222  ▷  $(ORBITER) -v 2 \
5223  ▷  ▷  -define F -finite_field -q 2 -end \
5224  ▷  ▷  -define A -vector -field F -sparse 11 "1,0,1,10" -end \
5225  ▷  ▷  -define B -vector -field F -dense "1,0,1,1" -end \
5226  ▷  ▷  -with F -do \
5227  ▷  ▷  -finite_field_activity \
5228  ▷  ▷  -polynomial_division A B -end
5229
5230
5231  poly_gcd:
5232  ▷  $(ORBITER) -v 2 \
5233  ▷  ▷  -define F -finite_field -q 2 -end \
5234  ▷  ▷  -with F -do \
5235  ▷  ▷  -finite_field_activity \
```

```
5236 ▷ ▷ -extended_gcd_for_polynomials "1,0,0,0,0,0,0,0,0,0,1" "1,0,1,1" -end
5237
5238 poly_mult_mod1:
5239 ▷ $(ORBITER) -v 2 \
5240 ▷ ▷ -define F -finite_field -q 7 -end \
5241 ▷ ▷ -with F -do \
5242 ▷ ▷ -finite_field_activity \
5243 ▷ ▷ -polynomial_mult_mod "1,2,3" "3,4,5" "6,0,0,1" -end
5244
5245 poly_mult_mod2:
5246 ▷ $(ORBITER) -v 2 \
5247 ▷ ▷ -define F -finite_field -q 7 -end \
5248 ▷ ▷ -with F -do \
5249 ▷ ▷ -finite_field_activity \
5250 ▷ ▷ -polynomial_mult_mod "3,1,2" "5,3,4" "6,0,0,1" -end
5251
5252
5253
5254 poly_mult_mod_F4:
5255 ▷ $(ORBITER) -v 2 \
5256 ▷ ▷ -define F -finite_field -q 2 -end \
5257 ▷ ▷ -with F -do \
5258 ▷ ▷ -finite_field_activity \
5259 ▷ ▷ -polynomial_mult_mod "1,1" "1,1" "1,1,1" -end
5260 ▷ $(ORBITER) -v 2 \
5261 ▷ ▷ -define F -finite_field -q 2 -end \
5262 ▷ ▷ -with F -do \
5263 ▷ ▷ -finite_field_activity \
5264 ▷ ▷ -polynomial_mult_mod "0,1" "1,1" "1,1,1" -end
5265 ▷ $(ORBITER) -v 2 \
5266 ▷ ▷ -define F -finite_field -q 2 -end \
5267 ▷ ▷ -with F -do \
5268 ▷ ▷ -finite_field_activity \
5269 ▷ ▷ -polynomial_mult_mod "0,1" "0,1" "1,1,1" -end
5270
5271
5272
5273
5274
5275 mult_polynomials_2_5_7:
5276 ▷ $(ORBITER) -v 2 \
5277 ▷ ▷ -define F -finite_field -q 2 -end \
5278 ▷ ▷ -with F -do \
5279 ▷ ▷ -finite_field_activity -mult_polynomials 5 7 -end
5280 ▷ pdflatex polynomial_mult_5_7.tex
5281 ▷ $(OPEN) polynomial_mult_5_7.pdf
5282
5283
5284 polynomial_division_ranked_2_27_13:
5285 ▷ $(ORBITER) -v 2 \
5286 ▷ ▷ -define F -finite_field -q 2 -end \
5287 ▷ ▷ -with F -do \
5288 ▷ ▷ -finite_field_activity \
5289 ▷ ▷ ▷ -polynomial_division_ranked 27 13 \
5290 ▷ ▷ -end
5291 ▷ pdflatex polynomial_division_27_13.tex
5292 ▷ $(OPEN) polynomial_division_27_13.pdf
5293
5294
```

```
5295
5296
5297  mult_polynomials_2_8_15:
5298  ▷  $(ORBITER) -v 2 \
5299  ▷  ▷  -define F -finite_field -q 2 -end \
5300  ▷  ▷  -with F -do \
5301  ▷  ▷  -finite_field_activity -mult_polynomials 8 15 -end
5302  ▷  pdflatex polynomial_mult_8_15.tex
5303  ▷  $(OPEN) polynomial_mult_8_15.pdf
5304
5305  polynomial_division_ranked_2_120_25:
5306  ▷  $(ORBITER) -v 2 \
5307  ▷  ▷  -define F -finite_field -q 2 -end \
5308  ▷  ▷  -with F -do \
5309  ▷  ▷  -finite_field_activity \
5310  ▷  ▷  ▷  -polynomial_division_ranked 120 25 \
5311  ▷  ▷  -end
5312  ▷  pdflatex polynomial_division_120_25.tex
5313  ▷  $(OPEN) polynomial_division_120_25.pdf
5314
5315  # the answer is 5
5316
5317
5318  mult_polynomials_2_7_7:
5319  ▷  $(ORBITER) -v 2 \
5320  ▷  ▷  -define F -finite_field -q 2 -end \
5321  ▷  ▷  -with F -do \
5322  ▷  ▷  -finite_field_activity \
5323  ▷  ▷  -mult_polynomials 7 7 -end
5324  ▷  pdflatex polynomial_mult_7_7.tex
5325  ▷  $(OPEN) polynomial_mult_7_7.pdf
5326
5327
5328
5329  mult_polynomials_2_4_6:
5330  ▷  $(ORBITER) -v 2 \
5331  ▷  ▷  -define F -finite_field -q 2 -end \
5332  ▷  ▷  -with F -do \
5333  ▷  ▷  -finite_field_activity \
5334  ▷  ▷  -mult_polynomials 4 6 -end
5335  ▷  pdflatex polynomial_mult_4_6.tex
5336  ▷  $(OPEN) polynomial_mult_4_6.pdf
5337
5338  polynomial_division_ranked_2_24_13:
5339  ▷  $(ORBITER) -v 2 \
5340  ▷  ▷  -define F -finite_field -q 2 -end \
5341  ▷  ▷  -with F -do \
5342  ▷  ▷  -finite_field_activity \
5343  ▷  ▷  ▷  -polynomial_division_ranked 24 13 \
5344  ▷  ▷  -end
5345  ▷  pdflatex polynomial_division_24_13.tex
5346  ▷  $(OPEN) polynomial_division_24_13.pdf
5347
5348
5349
5350  mult_polynomials_1024_999_997:
5351  ▷  $(ORBITER) -v 2 \
5352  ▷  ▷  -define F -finite_field -q 2 -end \
5353  ▷  ▷  -with F -do \
```

```
5354  ▷  ▷  -finite_field_activity \
5355  ▷  ▷  ▷  -mult_polynomials 999 997 \
5356  ▷  ▷  -end
5357  ▷  pdflatex polynomial_mult_999_997.tex
5358  ▷  $(OPEN) polynomial_mult_999_997.pdf
5359
5360
5361  polynomial_division_ranked_2_349147_1033:
5362  ▷  $(ORBITER) -v 2 \
5363  ▷  ▷  -define F -finite_field -q 2 -end \
5364  ▷  ▷  -with F -do \
5365  ▷  ▷  -finite_field_activity \
5366  ▷  ▷  -polynomial_division_ranked 349147 1033 \
5367  ▷  ▷  -end
5368  ▷  pdflatex polynomial_division_349147_1033.tex
5369  ▷  $(OPEN) polynomial_division_349147_1033.pdf
5370
5371
5372
5373
5374
5375  mult_polynomials_17_12:
5376  ▷  $(ORBITER) -v 2 \
5377  ▷  ▷  -define F -finite_field -q 2 -end \
5378  ▷  ▷  -with F -do \
5379  ▷  ▷  -finite_field_activity \
5380  ▷  ▷  -mult_polynomials 17 12 -end
5381  ▷  pdflatex polynomial_mult_17_12.tex
5382  ▷  $(OPEN) polynomial_mult_17_12.pdf
5383  ▷
5384  # gives 204
5385
5386  polynomial_division_ranked_2_204_37:
5387  ▷  $(ORBITER) -v 2 \
5388  ▷  ▷  -define F -finite_field -q 2 -end \
5389  ▷  ▷  -with F -do \
5390  ▷  ▷  -finite_field_activity \
5391  ▷  ▷  ▷  -polynomial_division_ranked 204 37 \
5392  ▷  ▷  -end
5393  ▷  pdflatex polynomial_division_204_37.tex
5394  ▷  $(OPEN) polynomial_division_204_37.pdf
5395  ▷
5396  # answer is 18
5397  ▷
5398  ▷
5399
5400
5401
5402
5403  #ToDo:
5404
5405  #test_crc32:
5406  #▷ $(ORBITER) -v 3 \
5407  #▷ ▷  -crc32 "123456789"
5408
5409
5410  Berlekamp_matrix_crc32:
5411  ▷  $(ORBITER) -v 2 \
5412  ▷  ▷  -define F -finite_field -q 2 -end \
```

```
5413  ▷  ▷  -define v -vector -field F -sparse 33 $(CRC32_SPARSE) -end \
5414  ▷  ▷  -with F -do \
5415  ▷  ▷  -finite_field_activity \
5416  ▷  ▷  -Berlekamp_matrix v -end
5417
5418
5419  # N = 2^32-1 = 3 * 5 * 17 * 257 * 65537
5420  # N / 3 = 1431655765
5421  # N / 5 = 858993459
5422  # N / 17 = 252645135
5423  # N / 257 = 16711935
5424  # N / 65537 = 65535
5425
5426
5427
5428  power_mod_inverse:
5429  ▷  $(ORBITER) -v 2 \
5430  ▷  ▷  -define F -finite_field -q 2 -end \
5431  ▷  ▷  -define M -vector -field F -sparse 33 $(CRC32_SPARSE) -end \
5432  ▷  ▷  -define A -vector -field F -sparse 2 "1,1" -end \
5433  ▷  ▷  -with F -do \
5434  ▷  ▷  -finite_field_activity \
5435  ▷  ▷  -polynomial_power_mod A $(TWO_TO_THE_32_MINUS_2) M \
5436  ▷  ▷  -end
5437
5438
5439  #A(X)=X^{31} + X^{25} + X^{22} + X^{21} + X^{15}
5440  #+ X^{11} + X^{10} + X^{9} + X^{7} + X^{6} + X^{4} + X^{3} + X + 1
5441
5442  mult_mod_to_get_one:
5443  ▷  $(ORBITER) -v 2 \
5444  ▷  ▷  -define F -finite_field -q 2 -end \
5445  ▷  ▷  -define M -vector -field F -sparse 33 $(CRC32_SPARSE) -end \
5446  ▷  ▷  -define A -vector -field F -sparse 2 "1,1" -end \
5447  ▷  ▷  -define B -vector -field F -sparse 33 $(INVERSE_SPARSE) -end \
5448  ▷  ▷  -with F -do \
5449  ▷  ▷  -finite_field_activity \
5450  ▷  ▷  ▷  -polynomial_mult_mod A B M \
5451  ▷  ▷  -end
5452
5453  #C(X)=1
5454
5455
5456
5457
5458
5459
5460
5461  Berlekamp_matrix_2_3:
5462  ▷  $(ORBITER) -v 2 \
5463  ▷  ▷  -define F -finite_field -q 2 -end \
5464  ▷  ▷  -define v -vector -field F -dense "1,1,0,1" -end \
5465  ▷  ▷  -with F -do \
5466  ▷  ▷  -finite_field_activity \
5467  ▷  ▷  -Berlekamp_matrix v -end
5468  ▷  pdflatex Berlekamp_matrix_q2_d3.tex
5469  ▷  $(OPEN) Berlekamp_matrix_q2_d3.pdf
5470
5471  # the polynomial X^3+X+1 is irreducible over GF(2) because the rank of the Berlek
```

```
      amp matrix is 2.
5472
5473  Berlekamp_matrix_2_4:
5474  ▷  $(ORBITER) -v 2 \
5475  ▷  ▷  -define F -finite_field -q 2 -end \
5476  ▷  ▷  -define v -vector -field F -dense "1,1,0,0,1" -end \
5477  ▷  ▷  -with F -do \
5478  ▷  ▷  -finite_field_activity \
5479  ▷  ▷  -Berlekamp_matrix v -end
5480
5481  # the polynomial X^4+X+1 is irreducible over GF(2) because the rank of the Berlek
      amp matrix is 3.
5482
5483
5484  Berlekamp_matrix_4_3a:
5485  ▷  $(ORBITER) -v 2 \
5486  ▷  ▷  -define F -finite_field -q 4 -end \
5487  ▷  ▷  -define v -vector -field F -dense "1,3,0,1" -end \
5488  ▷  ▷  -with F -do \
5489  ▷  ▷  -finite_field_activity \
5490  ▷  ▷  -Berlekamp_matrix v -end
5491
5492  Berlekamp_matrix_4_3b:
5493  ▷  $(ORBITER) -v 2 \
5494  ▷  ▷  -define F -finite_field -q 4 -end \
5495  ▷  ▷  -define v -vector -field F -dense "1,3,1,1" -end \
5496  ▷  ▷  -with F -do \
5497  ▷  ▷  -finite_field_activity \
5498  ▷  ▷  -Berlekamp_matrix v -end
5499
5500
5501
5502
5503  find_roots_a:
5504  ▷  $(ORBITER) -v 2 \
5505  ▷  ▷  -define F -finite_field -q 19 -end \
5506  ▷  ▷  -define v -vector -field F -dense "18,1,1" -end \
5507  ▷  ▷  -with F -do \
5508  ▷  ▷  -finite_field_activity \
5509  ▷  ▷  -polynomial_find_roots v -end
5510
5511  find_roots_b:
5512  ▷  $(ORBITER) -v 2 \
5513  ▷  ▷  -define F -finite_field -q 19 -end \
5514  ▷  ▷  -define v -vector -field F -dense "1,3,1" -end \
5515  ▷  ▷  -with F -do \
5516  ▷  ▷  -finite_field_activity \
5517  ▷  ▷  -polynomial_find_roots v -end
5518
5519  find_roots_c:
5520  ▷  $(ORBITER) -v 2 \
5521  ▷  ▷  -define F -finite_field -q 19 -end \
5522  ▷  ▷  -define v -vector -field F -dense "1,16,1" -end \
5523  ▷  ▷  -with F -do \
5524  ▷  ▷  -finite_field_activity \
5525  ▷  ▷  -polynomial_find_roots v -end
5526
5527  find_roots_d:
5528  ▷  $(ORBITER) -v 2 \
```

```
5529  ▷  ▷  -define F -finite_field -q 19 -end \
5530  ▷  ▷  -define v -vector -field F -dense "1,18,1" -end \
5531  ▷  ▷  -with F -do \
5532  ▷  ▷  -finite_field_activity \
5533  ▷  ▷  -polynomial_find_roots v -end
5534
5535  find_roots_e:
5536  ▷  $(ORBITER) -v 2 \
5537  ▷  ▷  -define F -finite_field -q 19 -end \
5538  ▷  ▷  -define v -vector -field F -dense "1,16,3" -end \
5539  ▷  ▷  -with F -do \
5540  ▷  ▷  -finite_field_activity \
5541  ▷  ▷  -polynomial_find_roots v -end
5542
5543
5544  roots_over_F2:
5545  ▷  $(ORBITER) -v 2 \
5546  ▷  ▷  -define F -finite_field -q 2  -end \
5547  ▷  ▷  -define v -vector -field F -dense "0,1,0,1,1,1" -end \
5548  ▷  ▷  -with F -do \
5549  ▷  ▷  -finite_field_activity \
5550  ▷  ▷  -polynomial_find_roots v -end
5551
5552
5553
5554  roots_over_F8:
5555  ▷  $(ORBITER) -v 2 \
5556  ▷  ▷  -define F -finite_field -q 8 -override_polynomial 11 -end \
5557  ▷  ▷  -define v -vector -field F -dense "0,1,0,1,1,1" -end \
5558  ▷  ▷  -with F -do \
5559  ▷  ▷  -finite_field_activity \
5560  ▷  ▷  -polynomial_find_roots v -end
5561
5562
5563
5564
5565  # degree and then order of the field of coefficients:
5566
5567  irred_3_2:
5568  ▷  $(ORBITER) -v 3 \
5569  ▷  ▷  -define F -finite_field -q 2 -end \
5570  ▷  ▷  -with F -do \
5571  ▷  ▷  -finite_field_activity \
5572  ▷  ▷  -make_table_of_irreducible_polynomials 3 -end
5573  ▷  pdflatex Irred_q2_d3.tex
5574  ▷  $(OPEN) Irred_q2_d3.pdf
5575
5576
5577  irred_4_2:
5578  ▷  $(ORBITER) -v 3 \
5579  ▷  ▷  -define F -finite_field -q 2 -end \
5580  ▷  ▷  -with F -do \
5581  ▷  ▷  -finite_field_activity \
5582  ▷  ▷  -make_table_of_irreducible_polynomials 4 -end
5583  ▷  pdflatex Irred_q2_d4.tex
5584  ▷  $(OPEN) Irred_q2_d4.pdf
5585
5586  # 3 polys
5587
```

```
5588  irred_5_2:
5589  ▷  $(ORBITER) -v 3 \
5590  ▷  ▷  -define F -finite_field -q 2 -end \
5591  ▷  ▷  -with F -do \
5592  ▷  ▷  -finite_field_activity \
5593  ▷  ▷  -make_table_of_irreducible_polynomials 5 -end
5594  ▷  pdflatex Irred_q2_d5.tex
5595  ▷  $(OPEN) Irred_q2_d5.pdf
5596
5597  # 6 polys
5598
5599  irred_6_2:
5600  ▷  $(ORBITER) -v 3 \
5601  ▷  ▷  -define F -finite_field -q 2 -end \
5602  ▷  ▷  -with F -do \
5603  ▷  ▷  -finite_field_activity \
5604  ▷  ▷  -make_table_of_irreducible_polynomials 6 -end
5605  ▷  pdflatex Irred_q2_d6.tex
5606  ▷  $(OPEN) Irred_q2_d6.pdf
5607
5608  # 9 polys: 97, 73, 67, 115, 117, 109, 91, 87, 103.
5609
5610  irred_7_2:
5611  ▷  $(ORBITER) -v 3 \
5612  ▷  ▷  -define F -finite_field -q 2 -end \
5613  ▷  ▷  -with F -do \
5614  ▷  ▷  -finite_field_activity \
5615  ▷  ▷  -make_table_of_irreducible_polynomials 7 -end
5616  ▷  pdflatex Irred_q2_d7.tex
5617  ▷  $(OPEN) Irred_q2_d7.pdf
5618
5619  # 18 polys
5620
5621  irred_8_2:
5622  ▷  $(ORBITER) -v 3 \
5623  ▷  ▷  -define F -finite_field -q 2 -end \
5624  ▷  ▷  -with F -do \
5625  ▷  ▷  -finite_field_activity \
5626  ▷  ▷  -make_table_of_irreducible_polynomials 8 -end
5627  ▷  pdflatex Irred_q2_d8.tex
5628  ▷  $(OPEN) Irred_q2_d8.pdf
5629
5630  # 30 polys
5631
5632  irred_9_2:
5633  ▷  $(ORBITER) -v 3 \
5634  ▷  ▷  -define F -finite_field -q 2 -end \
5635  ▷  ▷  -with F -do \
5636  ▷  ▷  -finite_field_activity \
5637  ▷  ▷  -make_table_of_irreducible_polynomials 9 -end
5638  ▷  pdflatex Irred_q2_d9.tex
5639  ▷  $(OPEN) Irred_q2_d9.pdf
5640
5641  # 56 polys
5642
5643  irred_10_2:
5644  ▷  $(ORBITER) -v 3 \
5645  ▷  ▷  -define F -finite_field -q 2 -end \
5646  ▷  ▷  -with F -do \
```

```
5647  ▷  ▷  -finite_field_activity \
5648  ▷  ▷  -make_table_of_irreducible_polynomials 10 -end
5649  ▷  pdflatex Irred_q2_d10.tex
5650  ▷  $(OPEN) Irred_q2_d10.pdf
5651
5652  # 99 polys
5653
5654  irred_2_4:
5655  ▷  $(ORBITER) -v 3 \
5656  ▷  ▷  -define F -finite_field -q 4 -end \
5657  ▷  ▷  -with F -do \
5658  ▷  ▷  -finite_field_activity \
5659  ▷  ▷  -make_table_of_irreducible_polynomials 2 -end
5660  ▷  pdflatex Irred_q4_d2.tex
5661  ▷  $(OPEN) Irred_q4_d2.pdf
5662
5663  # 6 polys
5664
5665
5666  irred_3_4:
5667  ▷  $(ORBITER) -v 6 \
5668  ▷  ▷  -define F -finite_field -q 4 -end \
5669  ▷  ▷  -with F -do \
5670  ▷  ▷  -finite_field_activity \
5671  ▷  ▷  -make_table_of_irreducible_polynomials 3 -end
5672  ▷  pdflatex Irred_q4_d3.tex
5673  ▷  $(OPEN) Irred_q4_d3.pdf
5674
5675  # 20 polys
5676
5677
5678  F2_get_primitive_poly_of_deg_10:
5679  ▷  $(ORBITER) -v 6 \
5680  ▷  ▷  -define F -finite_field -q 2 -end \
5681  ▷  ▷  -with F -do \
5682  ▷  ▷  -finite_field_activity \
5683  ▷  ▷  -get_primitive_polynomial 10 -end
5684
5685
5686  F2_get_primitive_poly_range_2_10:
5687  ▷  $(ORBITER) -v 2 \
5688  ▷  ▷  -define F -finite_field -q 2 -end \
5689  ▷  ▷  -with F -do \
5690  ▷  ▷  -finite_field_activity \
5691  ▷  ▷  -get_primitive_polynomial_in_range 2 10 -end
5692
5693  #| grep //
5694  # for degree 61, it gets stuck in factoring 2^61-1 (which is prime)
5695
5696
5697
5698
5699
5700
5701
5702
5703  ##############################################################################
5704  # Section 5.2: Multivariate Polynomials
5705
```

```
5706
5707
5708  SECTION_MULTIVARIATE_POLYNOMIALS:
5709
5710
5711  test_5_2:
5712  ▷  #make parse_algebraic_formula error message
5713  ▷  make Fermat_cubic_F9
5714  ▷  make Fermat_cubic_F9_c
5715  ▷  make Cremona_map
5716  ▷  make arcs_5_2_q11
5717  ▷  make arcs_5_2_q11_ideal
5718  ▷  make surface_9lines_4E_ideal
5719  ▷  make F_9_q7
5720  ▷  make random_k_subsets_PG_2_11
5721  ▷  #make line_type_in_PG_2_11
5722  ▷  make random_arc_5_2_q11_ideal
5723  ▷  make Endrass_F7.txt
5724  ▷  make octic_prepare
5725  ▷  make affine_map_F5_0
5726  ▷  make affine_map_F5_1
5727  ▷  make affine_map_F5_2
5728  ▷  make affine_map_F5_3
5729  ▷  make affine_map_F5_4
5730  ▷  make permutation_polynomial_F5
5731
5732
5733
5734  parse_algebraic_formula:
5735  ▷  $(ORBITER) -v 6 \
5736  ▷  ▷  -define F -finite_field -q 13 -end \
5737  ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
5738  ▷  ▷  -define R -polynomial_ring \
5739  ▷  ▷  ▷  -field F \
5740  ▷  ▷  ▷  -number_of_variables 3 \
5741  ▷  ▷  ▷  -homogeneous_of_degree 3 \
5742  ▷  ▷  ▷  -monomial_ordering_partition \
5743  ▷  ▷  ▷  -variables "y0,y1,y2" "y_0,y_1,y_2" \
5744  ▷  ▷  -end \
5745  ▷  ▷  -with R -do \
5746  ▷  ▷  ▷  -ring_theoretic_activity \
5747  ▷  ▷  ▷  -parse_equation "eqn" "eqn" "y0^3+y1^3+y2^3" "" "" \
5748  ▷  ▷  -end
5749
5750  Fermat_cubic_F9:
5751  ▷  $(ORBITER) -v 6 \
5752  ▷  ▷  -define F -finite_field -q 9 -end \
5753  ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
5754  ▷  ▷  -define R -polynomial_ring \
5755  ▷  ▷  ▷  -field F \
5756  ▷  ▷  ▷  -number_of_variables 3 \
5757  ▷  ▷  ▷  -homogeneous_of_degree 3 \
5758  ▷  ▷  ▷  -monomial_ordering_partition \
5759  ▷  ▷  ▷  -variables "y0,y1,y2" "y_0,y_1,y_2" \
5760  ▷  ▷  ▷  -end \
5761  ▷  ▷  -define Fermat_cubic -symbolic_object \
5762  ▷  ▷  ▷  -field F \
5763  ▷  ▷  ▷  -text "y0^3+y1^3+y2^3" \
5764  ▷  ▷  ▷  -end \
```

```
5765  ▷  ▷  -with P -do \
5766  ▷  ▷  -projective_space_activity \
5767  ▷  ▷  ▷  -projective_variety R Fermat_cubic "" \
5768  ▷  ▷  -end
5769
5770  Fermat_cubic_F9_c:
5771  ▷  $(ORBITER) -v 6 \
5772  ▷  ▷  -define C -combinatorial_object \
5773  ▷  ▷  ▷  -label Fermat_cubic_F9 Fermat\_cubic\_F9 \
5774  ▷  ▷  ▷  -set_of_points "3, 5, 13, 28, 45, 53, 61, 69, 77, 85" \
5775  ▷  ▷  -end \
5776  ▷  ▷  -define F -finite_field -q 9 -end \
5777  ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
5778  ▷  ▷  -with C -do \
5779  ▷  ▷  -combinatorial_object_activity \
5780  ▷  ▷  ▷  -canonical_form_PG P \
5781  ▷  ▷  ▷  ▷  -save_ago \
5782  ▷  ▷  ▷  ▷  -max_TDO_depth 10 \
5783  ▷  ▷  ▷  -end \
5784  ▷  ▷  -end \
5785  ▷  ▷  -with C -do \
5786  ▷  ▷  -combinatorial_object_activity \
5787  ▷  ▷  ▷  -report \
5788  ▷  ▷  ▷  ▷  -export_flag_orbits \
5789  ▷  ▷  ▷  ▷  -show_TDO \
5790  ▷  ▷  ▷  ▷  -show_TDA \
5791  ▷  ▷  ▷  ▷  -export_labels \
5792  ▷  ▷  ▷  ▷  -dont_show_incidence_matrices \
5793  ▷  ▷  ▷  ▷  -export_group_GAP \
5794  ▷  ▷  ▷  -end \
5795  ▷  ▷  -end
5796  ▷  pdflatex Fermat_cubic_F9_classification.tex
5797  ▷  $(OPEN) Fermat_cubic_F9_classification.pdf
5798
5799
5800  Cremona_map:
5801  ▷  $(ORBITER) -v 6 \
5802  ▷  ▷  -define F -finite_field -q 13 -end \
5803  ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
5804  ▷  ▷  -define R -polynomial_ring \
5805  ▷  ▷  ▷  -field F \
5806  ▷  ▷  ▷  -number_of_variables 3 \
5807  ▷  ▷  ▷  -homogeneous_of_degree 6 \
5808  ▷  ▷  ▷  -monomial_ordering_lex \
5809  ▷  ▷  ▷  -variables "y0,y1,y2" "y_0,y_1,y_2" \
5810  ▷  ▷  ▷  -end \
5811  ▷  ▷  -define Y0 -symbolic_object \
5812  ▷  ▷  ▷  -field F \
5813  ▷  ▷  ▷  -text $(CREMONA_MAP_Y0) \
5814  ▷  ▷  ▷  -end \
5815  ▷  ▷  -define Y1 -symbolic_object \
5816  ▷  ▷  ▷  -field F \
5817  ▷  ▷  ▷  -text $(CREMONA_MAP_Y1) \
5818  ▷  ▷  ▷  -end \
5819  ▷  ▷  -define Y2 -symbolic_object \
5820  ▷  ▷  ▷  -field F \
5821  ▷  ▷  ▷  -text $(CREMONA_MAP_Y2) \
5822  ▷  ▷  ▷  -end \
5823  ▷  ▷  -define Cremona -symbolic_object \
```

```
5824  ▷  ▷  ▷  -field F \
5825  ▷  ▷  ▷  -text "Y0,Y1,Y2" \
5826  ▷  ▷  ▷  -end \
5827  ▷  ▷  -with P -do \
5828  ▷  ▷  -projective_space_activity \
5829  ▷  ▷  ▷  -map R Cremona "" \
5830  ▷  ▷  -end
5831
5832
5833
5834
5835  arcs_5_2_q11:
5836  ▷  $(ORBITER) -v 4 \
5837  ▷  ▷  -define F -finite_field -q 11 -end \
5838  ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
5839  ▷  ▷  -define Control -poset_classification_control \
5840  ▷  ▷  ▷  -problem_label arcs_5_2_q11 \
5841  ▷  ▷  ▷  -W -depth 5 \
5842  ▷  ▷  -end \
5843  ▷  ▷  -with P -do \
5844  ▷  ▷  -projective_space_activity \
5845  ▷  ▷  ▷  -classify_arcs \
5846  ▷  ▷  ▷  ▷  -control Control \
5847  ▷  ▷  ▷  ▷  -target_size 5 \
5848  ▷  ▷  ▷  ▷  -d 2 \
5849  ▷  ▷  ▷  -end \
5850  ▷  ▷  -end
5851  ▷  #pdflatex arcs_5_2_q11_poset.tex
5852  ▷  #$(OPEN) arcs_5_2_q11_poset.pdf
5853
5854
5855  # 2 orbits:
5856  # 0 1 2 3 37
5857  # 0 1 2 3 49
5858
5859
5860  arcs_5_2_q11_ideal:
5861  ▷  $(ORBITER) -v 2 \
5862  ▷  ▷  -define F -finite_field -q 11 -end \
5863  ▷  ▷  -define R -polynomial_ring \
5864  ▷  ▷  ▷  -field F \
5865  ▷  ▷  ▷  -number_of_variables 3 \
5866  ▷  ▷  ▷  -homogeneous_of_degree 2 \
5867  ▷  ▷  ▷  -monomial_ordering_lex \
5868  ▷  ▷  ▷  -variables "x0,x1,x2" "x_0,x_1,x_2" \
5869  ▷  ▷  ▷  -end \
5870  ▷  ▷  -define C -combinatorial_object \
5871  ▷  ▷  ▷  -label arcs_5_2_q11_lvl_5 arcs\_5\_2\_q11\_lvl\_5 \
5872  ▷  ▷  ▷  -file_of_points arcs_5_2_q11_lvl_5 \
5873  ▷  ▷  -end \
5874  ▷  ▷  -with C -do \
5875  ▷  ▷  -combinatorial_object_activity \
5876  ▷  ▷  ▷  -ideal R \
5877  ▷  ▷  -end
5878
5879  #( 0, 1, 2, 3, 37 )
5880  #generator 0 / 1 is 7*x0*x1 + 5*x0*x2 + 10*x1*x2
5881  #We found 12 points on the generator of the ideal
5882  #They are : ( 0, 1, 2, 3, 37, 54, 74, 80, 93, 105, 121, 128 )
```

```
5883
5884  #( 0, 1, 2, 3, 49 )
5885  #generator 0 / 1 is 4*x0*x1 + 8*x0*x2 + 10*x1*x2
5886  #looping over all generators of the ideal:
5887  #generator 0 / 1 is ( 0, 4, 8, 0, 10, 0 ) :
5888  #We found 12 points on the generator of the ideal
5889  #They are : ( 0, 1, 2, 3, 41, 49, 58, 77, 83, 95, 109, 130 )
5890
5891
5892
5893  surface_9lines_4E_ideal:
5894  ▷  $(ORBITER) -v 2 \
5895  ▷  ▷  -define Pts -vector -dense \
5896  ▷  ▷  ▷  $(PTS_OF_SURFACE_ORBIT211_Q3_L9_E4) \
5897  ▷  ▷  ▷  -end \
5898  ▷  ▷  -define F -finite_field -q 3 -end \
5899  ▷  ▷  -define R -polynomial_ring \
5900  ▷  ▷  ▷  -field F \
5901  ▷  ▷  ▷  -number_of_variables 4 \
5902  ▷  ▷  ▷  -homogeneous_of_degree 3 \
5903  ▷  ▷  ▷  -monomial_ordering_lex \
5904  ▷  ▷  ▷  -variables "x0,x1,x2,x3" "x_0,x_1,x_2,x_3" \
5905  ▷  ▷  ▷  -end \
5906  ▷  ▷  -with R -do \
5907  ▷  ▷  ▷  -ring_theoretic_activity \
5908  ▷  ▷  ▷  -ideal "surf_eqn" "surf\_eqn" Pts \
5909  ▷  ▷  ▷  -end
5910
5911
5912  #The ideal has dimension 2
5913  #generators for the ideal:
5914  #0 1 0 0 2 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0
5915  #0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 0
5916  #x0*x0*x1 + 2*x0*x1*x1 + 2*x0*x1*x3
5917  #2*x2*x2*x3 + 2*x2*x3*x3
5918
5919
5920
5921  F_9_q7:
5922  ▷  $(ORBITER) -v 10 \
5923  ▷  ▷  -define F -finite_field -q 7 -end \
5924  ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
5925  ▷  ▷  -define R -polynomial_ring \
5926  ▷  ▷  ▷  -field F \
5927  ▷  ▷  ▷  -number_of_variables 4 \
5928  ▷  ▷  ▷  -homogeneous_of_degree 3 \
5929  ▷  ▷  ▷  -monomial_ordering_partition \
5930  ▷  ▷  ▷  -variables "x0,x1,x2,x3" "x_0,x_1,x_2,x_3" \
5931  ▷  ▷  ▷  -end \
5932  ▷  ▷  -define F_9 -cubic_surface -space P \
5933  ▷  ▷  ▷  -by_equation R "F_9" \
5934  ▷  ▷  ▷  "\DF_9\D" "x0,x1,x2,x3" \
5935  ▷  ▷  ▷  $(SURFACE_F_9) \
5936  ▷  ▷  ▷  "" \
5937  ▷  ▷  ▷  "\Dno parameters\D" "" \
5938  ▷  ▷  -end \
5939  ▷  ▷  -with F_9 -do \
5940  ▷  ▷  -cubic_surface_activity \
5941  ▷  ▷  ▷  -report \
```

```
5942  ▷  ▷  -end
5943  ▷  pdflatex surface_equation_F_9_q7_report.tex
5944  ▷  $(OPEN) surface_equation_F_9_q7_report.pdf
5945
5946
5947
5948
5949  # we create 20 5-subsets of PG(2,11) at random. Note that PG(2,11) has 133 points
      .
5950
5951  random_k_subsets_PG_2_11:
5952  ▷  $(ORBITER) -v 4 \
5953  ▷  ▷  -create_random_k_subsets 133 5 20
5954
5955
5956  #random_k_subsets_n133_k5_nb20.csv
5957
5958
5959  # toDo
5960
5961  line_type_in_PG_2_11:
5962  ▷  $(ORBITER) -v 3 \
5963  ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
5964  ▷  ▷  -define F -finite_field -q 11 -end \
5965  ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
5966  ▷  ▷  -define C -combinatorial_object \
5967  ▷  ▷  ▷  -label random_sets random_sets \
5968  ▷  ▷  ▷  -file_of_points random_k_subsets_n133_k5_nb20.csv \
5969  ▷  ▷  -end \
5970  ▷  ▷  -with C -do \
5971  ▷  ▷  -combinatorial_object_activity \
5972  ▷  ▷  ▷  line_type_old \
5973  ▷  ▷  -end
5974
5975
5976  # the second one is an arc: 3,33,40,83,102
5977
5978  # we compute the ideal:
5979
5980
5981  random_arc_5_2_q11_ideal:
5982  ▷  $(ORBITER) -v 2 \
5983  ▷  ▷  -define F -finite_field -q 11 -end \
5984  ▷  ▷  -define R -polynomial_ring \
5985  ▷  ▷  ▷  -field F \
5986  ▷  ▷  ▷  -number_of_variables 3 \
5987  ▷  ▷  ▷  -homogeneous_of_degree 2 \
5988  ▷  ▷  ▷  -monomial_ordering_lex \
5989  ▷  ▷  ▷  -variables "x0,x1,x2" "x_0,x_1,x_2" \
5990  ▷  ▷  ▷  -end \
5991  ▷  ▷  -define C -combinatorial_object \
5992  ▷  ▷  ▷  -label random_arc random\_arc \
5993  ▷  ▷  ▷  -set_of_points "3,33,40,83,102" \
5994  ▷  ▷  -end \
5995  ▷  ▷  -with C -do \
5996  ▷  ▷  -combinatorial_object_activity \
5997  ▷  ▷  ▷  -ideal R \
5998  ▷  ▷  -end
5999
```

```
6000  #generator 0 / 1 is 10*x0*x0 + 3*x0*x1 + 8*x0*x2 + 2*x1*x1 + 10*x2*x2
6001  #We found 12 points on the generator of the ideal
6002  #They are : ( 3, 15, 19, 33, 40, 42, 46, 50, 83, 88, 102, 108 )
6003
6004
6005
6006
6007  Endrass_F7.txt:
6008  ▷  $(ORBITER) -v 2 \
6009  ▷  ▷  -define F -finite_field -q 7 -end \
6010  ▷  ▷  -define R -polynomial_ring -field F \
6011  ▷  ▷  ▷  -number_of_variables 4 \
6012  ▷  ▷  ▷  -homogeneous_of_degree 8 \
6013  ▷  ▷  ▷  -end \
6014  ▷  ▷  -define eqn -vector -field F -sparse 165 \
6015  ▷  ▷  ▷  $(ENDRASS_SPARSE) -end \
6016  ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
6017  ▷  ▷  -define Endrass_F7 -geometric_object P \
6018  ▷  ▷  ▷  -projective_variety R \
6019  ▷  ▷  ▷  ▷  "Endrass_F7" \
6020  ▷  ▷  ▷  ▷  "Endrass\_F7" \
6021  ▷  ▷  ▷  ▷  eqn \
6022  ▷  ▷  -end \
6023  ▷  ▷  -with Endrass_F7 -do \
6024  ▷  ▷  -combinatorial_object_activity -save \
6025  ▷  ▷  -end
6026
6027
6028
6029  # we created a set of 33 points, called Endrass_F7.txt
6030
6031
6032
6033  octic_prepare:
6034  ▷  $(ORBITER) -v 4 \
6035  ▷  ▷  -define A -vector -format 1 -dense "1,1,1,1" -end \
6036  ▷  ▷  -define D -diophant \
6037  ▷  ▷  ▷  -label octic_monomials \
6038  ▷  ▷  ▷  -coefficient_matrix A \
6039  ▷  ▷  ▷  -RHS "mult=1,EQ=8" \
6040  ▷  ▷  ▷  -x_min_global 0 -x_max_global 8 \
6041  ▷  ▷  -end \
6042  ▷  ▷  -with D -do \
6043  ▷  ▷  ▷  -diophant_activity -solve_mckay \
6044  ▷  ▷  -end
6045  ▷  sort -r octic_monomials_sol.csv >octic_monomials_sorted.txt
6046
6047  #Found 165 solutions with 210 backtrack steps
6048  # 165=binomial(11,3)
6049
6050
6051
6052  affine_map_F5_0:
6053  ▷  $(ORBITER) -v 8 \
6054  ▷  ▷  -define F -finite_field -q 5 -end \
6055  ▷  ▷  -define P -projective_space -n 1 -field F -v 0 -end \
6056  ▷  ▷  -define R -polynomial_ring \
6057  ▷  ▷  ▷  -field F \
6058  ▷  ▷  ▷  -number_of_variables 2 \
```

```
6059  ▷  ▷  ▷  -homogeneous_of_degree 4 \
6060  ▷  ▷  ▷  -monomial_ordering_lex \
6061  ▷  ▷  ▷  -variables "x,y" "x,y" \
6062  ▷  ▷  ▷  -end \
6063  ▷  ▷  -define Y0 -symbolic_object \
6064  ▷  ▷  ▷  -field F \
6065  ▷  ▷  ▷  -text "1-x^4" \
6066  ▷  ▷  ▷  -end \
6067  ▷  ▷  -define Y -symbolic_object \
6068  ▷  ▷  ▷  -field F \
6069  ▷  ▷  ▷  -text "Y0" \
6070  ▷  ▷  ▷  -end \
6071  ▷  ▷  -with P -do \
6072  ▷  ▷  -projective_space_activity \
6073  ▷  ▷  ▷  -affine_map R Y "" \
6074  ▷  ▷  -end
6075
6076  #( 1, 0, 0, 0, 0 )
6077
6078  affine_map_F5_1:
6079  ▷  $(ORBITER) -v 6 \
6080  ▷  ▷  -define F -finite_field -q 5 -end \
6081  ▷  ▷  -define P -projective_space -n 1 -field F -v 0 -end \
6082  ▷  ▷  -define R -polynomial_ring \
6083  ▷  ▷  ▷  -field F \
6084  ▷  ▷  ▷  -number_of_variables 2 \
6085  ▷  ▷  ▷  -homogeneous_of_degree 4 \
6086  ▷  ▷  ▷  -monomial_ordering_lex \
6087  ▷  ▷  ▷  -variables "x,y" "x,y" \
6088  ▷  ▷  ▷  -end \
6089  ▷  ▷  -define Y0 -symbolic_object \
6090  ▷  ▷  ▷  -field F \
6091  ▷  ▷  ▷  -text "1-(x-1)^4" \
6092  ▷  ▷  ▷  -end \
6093  ▷  ▷  -define Y -symbolic_object \
6094  ▷  ▷  ▷  -field F \
6095  ▷  ▷  ▷  -text "Y0" \
6096  ▷  ▷  ▷  -end \
6097  ▷  ▷  -with P -do \
6098  ▷  ▷  -projective_space_activity \
6099  ▷  ▷  ▷  -affine_map R Y "" \
6100  ▷  ▷  -end
6101
6102  #( 0, 1, 0, 0, 0 )
6103
6104  affine_map_F5_2:
6105  ▷  $(ORBITER) -v 6 \
6106  ▷  ▷  -define F -finite_field -q 5 -end \
6107  ▷  ▷  -define P -projective_space -n 1 -field F -v 0 -end \
6108  ▷  ▷  -define R -polynomial_ring \
6109  ▷  ▷  ▷  -field F \
6110  ▷  ▷  ▷  -number_of_variables 2 \
6111  ▷  ▷  ▷  -homogeneous_of_degree 4 \
6112  ▷  ▷  ▷  -monomial_ordering_lex \
6113  ▷  ▷  ▷  -variables "x,y" "x,y" \
6114  ▷  ▷  ▷  -end \
6115  ▷  ▷  -define Y0 -symbolic_object \
6116  ▷  ▷  ▷  -field F \
6117  ▷  ▷  ▷  -text "1-(x-2)^4" \
```

```
6118  ▷  ▷  ▷  -end \
6119  ▷  ▷  -define Y -symbolic_object \
6120  ▷  ▷  ▷  -field F \
6121  ▷  ▷  ▷  -text "Y0" \
6122  ▷  ▷  ▷  -end \
6123  ▷  ▷  -with P -do \
6124  ▷  ▷  -projective_space_activity \
6125  ▷  ▷  ▷  -affine_map R Y "" \
6126  ▷  ▷  -end
6127
6128  #( 0, 0, 1, 0, 0 )
6129
6130  affine_map_F5_3:
6131  ▷  $(ORBITER) -v 6 \
6132  ▷  ▷  -define F -finite_field -q 5 -end \
6133  ▷  ▷  -define P -projective_space -n 1 -field F -v 0 -end \
6134  ▷  ▷  -define R -polynomial_ring \
6135  ▷  ▷  ▷  -field F \
6136  ▷  ▷  ▷  -number_of_variables 2 \
6137  ▷  ▷  ▷  -homogeneous_of_degree 4 \
6138  ▷  ▷  ▷  -monomial_ordering_lex \
6139  ▷  ▷  ▷  -variables "x,y" "x,y" \
6140  ▷  ▷  ▷  -end \
6141  ▷  ▷  -define Y0 -symbolic_object \
6142  ▷  ▷  ▷  -field F \
6143  ▷  ▷  ▷  -text "1-(x-3)^4" \
6144  ▷  ▷  ▷  -end \
6145  ▷  ▷  -define Y -symbolic_object \
6146  ▷  ▷  ▷  -field F \
6147  ▷  ▷  ▷  -text "Y0" \
6148  ▷  ▷  ▷  -end \
6149  ▷  ▷  -with P -do \
6150  ▷  ▷  -projective_space_activity \
6151  ▷  ▷  ▷  -affine_map R Y "" \
6152  ▷  ▷  -end
6153
6154
6155  #( 0, 0, 0, 1, 0 )
6156
6157
6158
6159  affine_map_F5_4:
6160  ▷  $(ORBITER) -v 6 \
6161  ▷  ▷  -define F -finite_field -q 5 -end \
6162  ▷  ▷  -define P -projective_space -n 1 -field F -v 0 -end \
6163  ▷  ▷  -define R -polynomial_ring \
6164  ▷  ▷  ▷  -field F \
6165  ▷  ▷  ▷  -number_of_variables 2 \
6166  ▷  ▷  ▷  -homogeneous_of_degree 4 \
6167  ▷  ▷  ▷  -monomial_ordering_lex \
6168  ▷  ▷  ▷  -variables "x,y" "x,y" \
6169  ▷  ▷  ▷  -end \
6170  ▷  ▷  -define Y0 -symbolic_object \
6171  ▷  ▷  ▷  -field F \
6172  ▷  ▷  ▷  -text "1-(x-4)^4" \
6173  ▷  ▷  ▷  -end \
6174  ▷  ▷  -define Y -symbolic_object \
6175  ▷  ▷  ▷  -field F \
6176  ▷  ▷  ▷  -text "Y0" \
```

```
6177  ▷  ▷  ▷  -end \
6178  ▷  ▷  -with P -do \
6179  ▷  ▷  -projective_space_activity \
6180  ▷  ▷  ▷  -affine_map R Y "" \
6181  ▷  ▷  -end
6182
6183
6184  #( 0, 0, 0, 0, 1 )
6185
6186
6187
6188  permutation_polynomial_F5:
6189  ▷  $(ORBITER) -v 6 \
6190  ▷  ▷  -define F -finite_field -q 5 -end \
6191  ▷  ▷  -define P -projective_space -n 1 -field F -v 0 -end \
6192  ▷  ▷  -define R -polynomial_ring \
6193  ▷  ▷  ▷  -field F \
6194  ▷  ▷  ▷  -number_of_variables 2 \
6195  ▷  ▷  ▷  -homogeneous_of_degree 4 \
6196  ▷  ▷  ▷  -monomial_ordering_lex \
6197  ▷  ▷  ▷  -variables "x,y" "x,y" \
6198  ▷  ▷  ▷  -end \
6199  ▷  ▷  -define Y0 -symbolic_object \
6200  ▷  ▷  ▷  -field F \
6201  ▷  ▷  ▷  -text "3*(1-x^4)+4*(1-(x-2)^4)+1*(1-(x-3)^4)+2*(1-(x-4)^4)" \
6202  ▷  ▷  ▷  -end \
6203  ▷  ▷  -define Y -symbolic_object \
6204  ▷  ▷  ▷  -field F \
6205  ▷  ▷  ▷  -text "Y0" \
6206  ▷  ▷  ▷  -end \
6207  ▷  ▷  -with P -do \
6208  ▷  ▷  -projective_space_activity \
6209  ▷  ▷  ▷  -affine_map R Y "" \
6210  ▷  ▷  -end \
6211  ▷  ▷  -define Yx -symbolic_object \
6212  ▷  ▷  ▷  -field F \
6213  ▷  ▷  ▷  -expand Y \
6214  ▷  ▷  ▷  -end \
6215
6216
6217
6218
6219
6220
6221  ##############################################################################
6222  # Section 5.3: Algebraic Geometry
6223
6224  test_5_3:
6225  ▷  make Veronese_1_3_q7
6226  ▷  make variety_hermitian_curve
6227  ▷  make variety_hermitian_surface
6228
6229  Veronese_1_3_q7:
6230  ▷  $(ORBITER) -v 5 \
6231  ▷  ▷  -define Fq -finite_field -q 7 -end \
6232  ▷  ▷  -define C -symbolic_object \
6233  ▷  ▷  ▷  -field Fq \
6234  ▷  ▷  ▷  -managed_variables "y0,y1" \
6235  ▷  ▷  ▷  -text "y1^3,y0*y1^2,y0^2*y1,y0^3" \
```

```
6236  ▷  ▷  -end \
6237  ▷  ▷  -define P1 -projective_space -n 1 -field Fq -v 0 -end \
6238  ▷  ▷  -define P3 -projective_space -n 3 -field Fq -v 0 -end \
6239  ▷  ▷  -define R2 -polynomial_ring \
6240  ▷  ▷  ▷  -field Fq \
6241  ▷  ▷  ▷  -number_of_variables 2 \
6242  ▷  ▷  ▷  -homogeneous_of_degree 3 \
6243  ▷  ▷  ▷  -monomial_ordering_partition \
6244  ▷  ▷  ▷  -variables "y0,y1" "y_0,y_1" \
6245  ▷  ▷  ▷  -end \
6246  ▷  ▷  -define map -mapping \
6247  ▷  ▷  ▷  -domain P1 \
6248  ▷  ▷  ▷  -codomain P3 \
6249  ▷  ▷  ▷  -ring R2 \
6250  ▷  ▷  ▷  -formula C \
6251  ▷  ▷  ▷  -substitute "" \
6252  ▷  ▷  -end
6253
6254
6255
6256  variety_hermitian_curve:
6257  ▷  echo $(FILE_HERMITIAN_CURVE) >input.csv
6258  ▷  $(ORBITER) -v 8 \
6259  ▷  ▷  -define F -finite_field -q 9 -end \
6260  ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
6261  ▷  ▷  -define R -polynomial_ring \
6262  ▷  ▷  ▷  -field F \
6263  ▷  ▷  ▷  -number_of_variables 3 \
6264  ▷  ▷  ▷  -homogeneous_of_degree 4 \
6265  ▷  ▷  ▷  -monomial_ordering_partition \
6266  ▷  ▷  ▷  -variables "X,Y,Z" "X,Y,Z" \
6267  ▷  ▷  -end \
6268  ▷  ▷  -define O -orbits -classification_by_canonical_form \
6269  ▷  ▷  ▷  -space P \
6270  ▷  ▷  ▷  -ring R \
6271  ▷  ▷  ▷  -input_fname_mask input.csv \
6272  ▷  ▷  ▷  -nb_files 1 \
6273  ▷  ▷  ▷  -output_fname hermitian_curve_q9 \
6274  ▷  ▷  ▷  -label_equation "equation" \
6275  ▷  ▷  ▷  -algorithm_nauty \
6276  ▷  ▷  ▷  -end \
6277  ▷  ▷  -end \
6278  ▷  -with O -do -orbits_activity \
6279  ▷  ▷  -report \
6280  ▷  ▷  -report_options \
6281  ▷  ▷  ▷  -fname hermitian_curve_q9 \
6282  ▷  ▷  -end \
6283  ▷  -end
6284  ▷  pdflatex hermitian_curve_q9_orbits.tex
6285  ▷  $(OPEN) hermitian_curve_q9_orbits.pdf
6286
6287
6288
6289  variety_hermitian_surface:
6290  ▷  echo $(FILE_HERMITIAN_SURFACE) >input.csv
6291  ▷  $(ORBITER) -v 8 \
6292  ▷  ▷  -define F -finite_field -q 9 -end \
6293  ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
6294  ▷  ▷  -define R -polynomial_ring \
```

```
6295  ▷  ▷  ▷  -field F \
6296  ▷  ▷  ▷  -number_of_variables 4 \
6297  ▷  ▷  ▷  -homogeneous_of_degree 4 \
6298  ▷  ▷  ▷  -monomial_ordering_partition \
6299  ▷  ▷  ▷  -variables "X,Y,Z,W" "X,Y,Z,W" \
6300  ▷  ▷  -end \
6301  ▷  ▷  -define O -orbits -classification_by_canonical_form \
6302  ▷  ▷  ▷  -space P \
6303  ▷  ▷  ▷  -ring R \
6304  ▷  ▷  ▷  -input_fname_mask input.csv \
6305  ▷  ▷  ▷  -nb_files 1 \
6306  ▷  ▷  ▷  -output_fname hermitian_surface_q9 \
6307  ▷  ▷  ▷  -label_equation "equation" \
6308  ▷  ▷  ▷  -algorithm_nauty \
6309  ▷  ▷  ▷  -end \
6310  ▷  ▷  -end \
6311  ▷  -with O -do -orbits_activity \
6312  ▷  ▷  -report \
6313  ▷  ▷  -report_options \
6314  ▷  ▷  ▷  -fname hermitian_surface_q9 \
6315  ▷  ▷  -end \
6316  ▷  -end
6317  ▷  pdflatex hermitian_surface_q9_orbits.tex
6318  ▷  $(OPEN) hermitian_surface_q9_orbits.pdf
6319
6320
6321
6322
6323
6324
6325
6326  ##############################################################################
6327  # Chapter 6 - Group Theory
6328  ##############################################################################
6329
6330
6331  test_6:
6332  ▷  make test_6_1
6333  ▷  make test_6_2
6334  ▷  make test_6_3
6335  ▷  make test_6_4
6336  ▷  make test_6_5
6337  ▷  make test_6_6
6338  ▷  #make test_6_7 # magma stuff
6339  ▷  make test_6_8
6340
6341
6342  ##############################################################################
6343  # Section 6.1: Permutation groups
6344
6345
6346  SECTION_PERMUTATION_GROUPS:
6347
6348  test_6_1:
6349  ▷  make Cyclic_6
6350  ▷  make Cyclic_6_group_table
6351  ▷  make Symmetric_3
6352  ▷  make Symmetric_3_group_table
6353  ▷  make Symmetric_3_elements
```

```
6354  ▷  make Symmetric_3_long
6355  ▷  make Symmetric_4
6356  ▷  make Symmetric_4_group_table
6357  ▷  make Symmetric_4_long
6358  ▷  make Group_8_1
6359  ▷  make Group_8_2
6360  ▷  make Group_8_3
6361  ▷  make Group_8_4
6362  ▷  make Group_8_5
6363  ▷  make Group_M12
6364
6365
6366  Cyclic_6:
6367  ▷  $(ORBITER) -v 3 \
6368  ▷  ▷  -define G -permutation_group -cyclic_group 6 -end \
6369  ▷  ▷  -with G -do \
6370  ▷  ▷  ▷  -group_theoretic_activity \
6371  ▷  ▷  ▷  -report \
6372  ▷  ▷  -end
6373  ▷  pdflatex C_6_report.tex
6374  ▷  $(OPEN) C_6_report.pdf
6375
6376
6377  Cyclic_6_group_table:
6378  ▷  $(ORBITER) -v 3 \
6379  ▷  ▷  -define G -permutation_group -cyclic_group 6 -end \
6380  ▷  ▷  -with G -do \
6381  ▷  ▷  -group_theoretic_activity \
6382  ▷  ▷  ▷  -export_group_table \
6383  ▷  ▷  -end
6384  ▷  $(ORBITER) -v 2 \
6385  ▷  ▷  -define all_one_r -vector -repeat 1 6 -end \
6386  ▷  ▷  -define all_one_c -vector -repeat 1 6 -end \
6387  ▷  ▷  -draw_matrix \
6388  ▷  ▷  ▷  -input_csv_file C_6_group_table.csv \
6389  ▷  ▷  ▷  -box_width 50 -bit_depth 24 \
6390  ▷  ▷  ▷  -partition 3 all_one_r all_one_c \
6391  ▷  ▷  -end
6392  ▷  $(OPEN) C_6_group_table_draw.bmp
6393
6394
6395  Symmetric_3:
6396  ▷  $(ORBITER) -v 3 \
6397  ▷  ▷  -define G -permutation_group -symmetric_group 3 -end \
6398  ▷  ▷  -with G -do \
6399  ▷  ▷  -group_theoretic_activity \
6400  ▷  ▷  ▷  -report \
6401  ▷  ▷  -end
6402  ▷  pdflatex Sym_3_report.tex
6403  ▷  $(OPEN) Sym_3_report.pdf
6404
6405
6406  Symmetric_3_group_table:
6407  ▷  $(ORBITER) -v 3 \
6408  ▷  ▷  -define G -permutation_group -symmetric_group 3 -end \
6409  ▷  ▷  -with G -do \
6410  ▷  ▷  -group_theoretic_activity \
6411  ▷  ▷  ▷  -export_group_table \
6412  ▷  ▷  -end
```

```
6413  ▷  $(ORBITER) -v 2 \
6414  ▷  ▷  -define all_one_r -vector -repeat 1 6 -end \
6415  ▷  ▷  -define all_one_c -vector -repeat 1 6 -end \
6416  ▷  ▷  -draw_matrix \
6417  ▷  ▷  ▷  -input_csv_file Sym_3_group_table.csv \
6418  ▷  ▷  ▷  -box_width 50 -bit_depth 24 \
6419  ▷  ▷  ▷  -partition 3 all_one_r all_one_c \
6420  ▷  ▷  -end
6421  ▷  $(OPEN) Sym_3_group_table_draw.bmp
6422
6423  Symmetric_3_elements:
6424  ▷  $(ORBITER) -v 3 \
6425  ▷  ▷  -define G -permutation_group -symmetric_group 3 -end \
6426  ▷  ▷  -with G -do \
6427  ▷  ▷  -group_theoretic_activity \
6428  ▷  ▷  ▷  -save_elements_csv "Symmetric3_elts.csv" \
6429  ▷  ▷  -end
6430  ▷  $(ORBITER) -v 2 \
6431  ▷  ▷  -define Sym3_elts -vector -load_csv_data_column \
6432  ▷  ▷  ▷  Symmetric3_elts.csv 1 -end \
6433  ▷  ▷  -save_matrix_csv Sym3_elts
6434  ▷  $(ORBITER) -v 2 \
6435  ▷  ▷  -define all_one_r -vector -repeat 1 6 -end \
6436  ▷  ▷  -define all_one_c -vector -repeat 1 3 -end \
6437  ▷  ▷  -draw_matrix \
6438  ▷  ▷  ▷  -input_csv_file Sym3_elts_matrix.csv \
6439  ▷  ▷  ▷  -box_width 50 -bit_depth 8 \
6440  ▷  ▷  ▷  -partition 3 \
6441  ▷  ▷  ▷  ▷  all_one_r all_one_c \
6442  ▷  ▷  -end
6443  ▷  $(OPEN) Sym3_elts_matrix_draw.bmp
6444
6445  Symmetric_3_long:
6446  ▷  $(ORBITER) -v 3 \
6447  ▷  ▷  -define G -permutation_group -symmetric_group 3 -end \
6448  ▷  ▷  -with G -do \
6449  ▷  ▷  -group_theoretic_activity \
6450  ▷  ▷  ▷  -export_orbiter \
6451  ▷  ▷  -end \
6452  ▷  ▷  -with G -do \
6453  ▷  ▷  -group_theoretic_activity \
6454  ▷  ▷  ▷  -print_elements_tex \
6455  ▷  ▷  -end \
6456  ▷  ▷  -with G -do \
6457  ▷  ▷  -group_theoretic_activity \
6458  ▷  ▷  ▷  -report \
6459  ▷  ▷  -end \
6460  ▷  ▷  -with G -do \
6461  ▷  ▷  -group_theoretic_activity \
6462  ▷  ▷  ▷  -save_elements_csv "Symmetric3_elts.csv" \
6463  ▷  ▷  -end
6464  ▷  $(ORBITER) -v 3 \
6465  ▷  ▷  -draw_options \
6466  ▷  ▷  ▷  -nodes \
6467  ▷  ▷  ▷  -embedded -radius 250 \
6468  ▷  ▷  ▷  -xin 10000 -yin 10000 \
6469  ▷  ▷  ▷  -xout 1000000 -yout 600000 \
6470  ▷  ▷  ▷  -scale 0.3 -line_width 1.0 \
6471  ▷  ▷  -end \
```

```
6472  ▷  ▷  -tree_draw -file Sym_3_elements_tree.txt -end
6473  ▷  $(ORBITER) -v 2 \
6474  ▷  ▷  -define M -vector -load_csv_data_column \
6475  ▷  ▷  ▷  Symmetric3_elts.csv 1 -end \
6476  ▷  ▷  -save_matrix_csv M
6477  ▷  $(ORBITER) -v 2 \
6478  ▷  ▷  -define all_one_r -vector -repeat 1 6 -end \
6479  ▷  ▷  -define all_one_c -vector -repeat 1 3 -end \
6480  ▷  ▷  -draw_matrix \
6481  ▷  ▷  ▷  -input_csv_file M_matrix.csv \
6482  ▷  ▷  ▷  -box_width 50 -bit_depth 8 \
6483  ▷  ▷  ▷  -partition 3 \
6484  ▷  ▷  ▷  ▷  all_one_r all_one_c \
6485  ▷  ▷  -end
6486  ▷  pdflatex Sym_3_elements_tree_draw.tex
6487  ▷  $(OPEN) Sym_3_elements_tree_draw.pdf
6488  ▷  #pdflatex Perm3_report.tex
6489  ▷  #$(OPEN) Perm3_report.pdf
6490
6491
6492  Symmetric_4:
6493  ▷  $(ORBITER) -v 3 \
6494  ▷  ▷  -define G -permutation_group -symmetric_group 4 -end \
6495  ▷  ▷  -with G -do \
6496  ▷  ▷  -group_theoretic_activity \
6497  ▷  ▷  ▷  -report \
6498  ▷  ▷  -end
6499  ▷  pdflatex Sym_4_report.tex
6500  ▷  $(OPEN) Sym_4_report.pdf
6501
6502
6503  Symmetric_4_group_table:
6504  ▷  $(ORBITER) -v 3 \
6505  ▷  ▷  -define G -permutation_group -symmetric_group 4 -end \
6506  ▷  ▷  -with G -do \
6507  ▷  ▷  -group_theoretic_activity \
6508  ▷  ▷  ▷  -export_group_table \
6509  ▷  ▷  -end
6510  ▷  $(ORBITER) -v 2 \
6511  ▷  ▷  -define all_one_r -vector -repeat 1 24 -end \
6512  ▷  ▷  -define all_one_c -vector -repeat 1 24 -end \
6513  ▷  ▷  -draw_matrix \
6514  ▷  ▷  ▷  -input_csv_file Sym_4_group_table.csv \
6515  ▷  ▷  ▷  -box_width 50 -bit_depth 24 \
6516  ▷  ▷  ▷  -partition 3 all_one_r all_one_c \
6517  ▷  ▷  -end
6518  ▷  $(OPEN) Sym_4_group_table_draw.bmp
6519
6520
6521
6522  Symmetric_4_long:
6523  ▷  $(ORBITER) -v 3 \
6524  ▷  ▷  -define G -permutation_group -symmetric_group 4 -end \
6525  ▷  ▷  -with G -do \
6526  ▷  ▷  -group_theoretic_activity \
6527  ▷  ▷  ▷  -export_orbiter \
6528  ▷  ▷  -end \
6529  ▷  ▷  -with G -do \
6530  ▷  ▷  -group_theoretic_activity \
```

```
6531  ▷  ▷  ▷  -export_group_table \
6532  ▷  ▷  -end \
6533  ▷  ▷  -with G -do \
6534  ▷  ▷  -group_theoretic_activity \
6535  ▷  ▷  ▷  -print_elements_tex \
6536  ▷  ▷  -end \
6537  ▷  ▷  -with G -do \
6538  ▷  ▷  -group_theoretic_activity \
6539  ▷  ▷  ▷  -report \
6540  ▷  ▷  -end \
6541  ▷  ▷  -with G -do \
6542  ▷  ▷  -group_theoretic_activity \
6543  ▷  ▷  ▷  -save_elements_csv "Sym_4_elts.csv" \
6544  ▷  ▷  -end \
6545  ▷  ▷  -with G -do \
6546  ▷  ▷  -group_theoretic_activity \
6547  ▷  ▷  ▷  -export_inversion_graphs "Sym_4_inversion_graphs.csv" \
6548  ▷  ▷  -end
6549  ▷  $(ORBITER) -v 2 \
6550  ▷  ▷  -draw_options \
6551  ▷  ▷  ▷  -nodes \
6552  ▷  ▷  ▷  -embedded -radius 175 \
6553  ▷  ▷  ▷  -xin 10000 -yin 10000 \
6554  ▷  ▷  ▷  -xout 1500000 -yout 600000 \
6555  ▷  ▷  ▷  -scale 0.3 -line_width 1.0 \
6556  ▷  ▷  -end \
6557  ▷  ▷  -tree_draw -file Sym_4_elements_tree.txt -end
6558  ▷  $(ORBITER) -v 2 -draw_matrix \
6559  ▷  ▷  -input_csv_file Sym_4_group_table.csv \
6560  ▷  ▷  -box_width 50 -bit_depth 24 -end
6561  ▷  $(ORBITER) -v 2 \
6562  ▷  ▷  -define M -vector -load_csv_data_column \
6563  ▷  ▷  ▷  Sym_4_elts.csv 1 -end \
6564  ▷  ▷  -save_matrix_csv M
6565  ▷  $(ORBITER) -v 2 \
6566  ▷  ▷  -define all_one_r -vector -repeat 1 24 -end \
6567  ▷  ▷  -define all_one_c -vector -repeat 1 4 -end \
6568  ▷  ▷  -draw_matrix \
6569  ▷  ▷  ▷  -input_csv_file M_matrix.csv \
6570  ▷  ▷  ▷  -box_width 50 -bit_depth 8 \
6571  ▷  ▷  ▷  -partition 3 \
6572  ▷  ▷  ▷  ▷  all_one_r all_one_c \
6573  ▷  ▷  -end
6574  ▷  pdflatex Sym_4_elements_tree_draw.tex
6575  ▷  #$(OPEN) Sym_4_elements_tree_draw.pdf
6576
6577
6578
6579  # C8:
6580
6581  Group_8_1:
6582  ▷  $(ORBITER) -v 2 \
6583  ▷  ▷  -define gens -vector -dense \
6584  ▷  ▷  ▷  "1,2,3,4,5,6,7,0" \
6585  ▷  ▷  -end \
6586  ▷  ▷  -define G -permutation_group -symmetric_group 8 \
6587  ▷  ▷  ▷  -subgroup_by_generators G1 8 1 gens -end \
6588  ▷  ▷  -with G -do \
6589  ▷  ▷  -group_theoretic_activity \
```

```
6590  ▷  ▷  ▷  -export_group_table \
6591  ▷  ▷  -end
6592  ▷  $(ORBITER) -v 2 \
6593  ▷  ▷  -define all_one_r -vector -repeat 1 8 -end \
6594  ▷  ▷  -define all_one_c -vector -repeat 1 8 -end \
6595  ▷  ▷  -draw_matrix \
6596  ▷  ▷  ▷  -input_csv_file \
6597  ▷  ▷  ▷  ▷  Sym_8_Subgroup_G1_8_group_table.csv \
6598  ▷  ▷  ▷  -box_width 50 -bit_depth 24 \
6599  ▷  ▷  ▷  -partition 3 all_one_r all_one_c \
6600  ▷  ▷  -end
6601  ▷  convert Sym_8_Subgroup_G1_8_group_table_draw.bmp \
6602  ▷  ▷  Sym_8_Subgroup_G1_8_group_table_draw.png
6603  ▷  $(OPEN) Sym_8_Subgroup_G1_8_group_table_draw.png
6604
6605
6606
6607  # C4C2:
6608
6609  Group_8_2:
6610  ▷  $(ORBITER) -v 2 \
6611  ▷  ▷  -define gens -vector -dense \
6612  ▷  ▷  ▷  "1,2,3,0,4,5, 0,1,2,3,5,4" \
6613  ▷  ▷  -end \
6614  ▷  ▷  -define G -permutation_group -symmetric_group 6 \
6615  ▷  ▷  ▷  -subgroup_by_generators G2 8 2 gens -end \
6616  ▷  ▷  -with G -do \
6617  ▷  ▷  -group_theoretic_activity \
6618  ▷  ▷  ▷  -export_group_table \
6619  ▷  ▷  -end
6620  ▷  $(ORBITER) -v 2 \
6621  ▷  ▷  -define all_one_r -vector -repeat 1 8 -end \
6622  ▷  ▷  -define all_one_c -vector -repeat 1 8 -end \
6623  ▷  ▷  -draw_matrix \
6624  ▷  ▷  ▷  -input_csv_file \
6625  ▷  ▷  ▷  ▷  Sym_6_Subgroup_G2_8_group_table.csv \
6626  ▷  ▷  ▷  -box_width 50 -bit_depth 24 \
6627  ▷  ▷  ▷  -partition 3 all_one_r all_one_c \
6628  ▷  ▷  -end
6629  ▷  convert Sym_6_Subgroup_G2_8_group_table_draw.bmp \
6630  ▷  ▷  Sym_6_Subgroup_G2_8_group_table_draw.png
6631  ▷  $(OPEN) Sym_6_Subgroup_G2_8_group_table_draw.png
6632
6633
6634  # Group C2C2C2:
6635
6636  Group_8_3:
6637  ▷  $(ORBITER) -v 2 \
6638  ▷  ▷  -define gens -vector -dense \
6639  ▷  ▷  ▷  "1,0,2,3,4,5, 0,1,3,2,4,5, 0,1,2,3,5,4" \
6640  ▷  ▷  -end \
6641  ▷  ▷  -define G -permutation_group -symmetric_group 6 \
6642  ▷  ▷  ▷  -subgroup_by_generators G3 8 3 gens -end \
6643  ▷  ▷  -with G -do \
6644  ▷  ▷  -group_theoretic_activity \
6645  ▷  ▷  ▷  -export_group_table \
6646  ▷  ▷  -end
6647  ▷  $(ORBITER) -v 2 \
6648  ▷  ▷  -define all_one_r -vector -repeat 1 8 -end \
```

```
6649  ▷  ▷  -define all_one_c -vector -repeat 1 8 -end \
6650  ▷  ▷  -draw_matrix \
6651  ▷  ▷  ▷  -input_csv_file \
6652  ▷  ▷  ▷  ▷  Sym_6_Subgroup_G3_8_group_table.csv \
6653  ▷  ▷  ▷  -box_width 50 -bit_depth 24 \
6654  ▷  ▷  ▷  -partition 3 all_one_r all_one_c \
6655  ▷  ▷  -end
6656  ▷  convert Sym_6_Subgroup_G3_8_group_table_draw.bmp \
6657  ▷  ▷  Sym_6_Subgroup_G3_8_group_table_draw.png
6658  ▷  $(OPEN) Sym_6_Subgroup_G3_8_group_table_draw.png
6659
6660
6661  # Diehedral group D4 of order 8:
6662
6663  Group_8_4:
6664  ▷  $(ORBITER) -v 2 \
6665  ▷  ▷  -define gens -vector -dense \
6666  ▷  ▷  ▷  "1,2,3,0, 0,3,2,1" \
6667  ▷  ▷  -end \
6668  ▷  ▷  -define G -permutation_group -symmetric_group 4 \
6669  ▷  ▷  ▷  -subgroup_by_generators G4 8 2 gens -end \
6670  ▷  ▷  -with G -do \
6671  ▷  ▷  -group_theoretic_activity \
6672  ▷  ▷  ▷  -export_group_table \
6673  ▷  ▷  -end
6674  ▷  $(ORBITER) -v 2 \
6675  ▷  ▷  -define all_one_r -vector -repeat 1 8 -end \
6676  ▷  ▷  -define all_one_c -vector -repeat 1 8 -end \
6677  ▷  ▷  -draw_matrix \
6678  ▷  ▷  ▷  -input_csv_file \
6679  ▷  ▷  ▷  ▷  Sym_4_Subgroup_G4_8_group_table.csv \
6680  ▷  ▷  ▷  -box_width 50 -bit_depth 24 \
6681  ▷  ▷  ▷  -partition 3 all_one_r all_one_c \
6682  ▷  ▷  -end
6683  ▷  convert Sym_4_Subgroup_G4_8_group_table_draw.bmp \
6684  ▷  ▷  Sym_4_Subgroup_G4_8_group_table_draw.png
6685  ▷  $(OPEN) Sym_4_Subgroup_G4_8_group_table_draw.png
6686
6687  # Quaternion group:
6688
6689  Group_8_5:
6690  ▷  $(ORBITER) -v 2 \
6691  ▷  ▷  -define gens -vector -dense \
6692  ▷  ▷  ▷  "2,3,1,0,6,7,5,4, 4,5,7,6,1,0,2,3" \
6693  ▷  ▷  -end \
6694  ▷  ▷  -define G -permutation_group -symmetric_group 8 \
6695  ▷  ▷  ▷  -subgroup_by_generators G5 8 2 gens -end \
6696  ▷  ▷  -with G -do \
6697  ▷  ▷  -group_theoretic_activity \
6698  ▷  ▷  ▷  -export_group_table \
6699  ▷  ▷  -end
6700  ▷  $(ORBITER) -v 2 \
6701  ▷  ▷  -define all_one_r -vector -repeat 1 8 -end \
6702  ▷  ▷  -define all_one_c -vector -repeat 1 8 -end \
6703  ▷  ▷  -draw_matrix \
6704  ▷  ▷  ▷  -input_csv_file \
6705  ▷  ▷  ▷  ▷  Sym_8_Subgroup_G5_8_group_table.csv \
6706  ▷  ▷  ▷  -box_width 50 -bit_depth 24 \
6707  ▷  ▷  ▷  -partition 3 all_one_r all_one_c \
```

```
6708  ▷  ▷  -end
6709  ▷  convert Sym_8_Subgroup_G5_8_group_table_draw.bmp \
6710  ▷  ▷  Sym_8_Subgroup_G5_8_group_table_draw.png
6711  ▷  $(OPEN) Sym_8_Subgroup_G5_8_group_table_draw.png
6712
6713
6714
6715  # M12 of order 95040:
6716  #
6717  #(1,4)(3,10)(5,11)(6,12)
6718  #(1,8,9)(2,3,4)(5,12,11)(6,10,7)
6719  #[3,1,9,0,10,11,6,7,8,2,4,5]
6720  #[7,2,3,1,11,9,5,8,0,6,4,10]
6721
6722
6723
6724
6725  Group_M12:
6726  ▷  $(ORBITER) -v 2 \
6727  ▷  ▷  -define gens -vector -dense $(GENERATORS_M12) -end \
6728  ▷  ▷  -define G -permutation_group -symmetric_group 12 \
6729  ▷  ▷  ▷  -subgroup_by_generators M12 95040 2 gens -end \
6730  ▷  ▷  -with G -do \
6731  ▷  ▷  -group_theoretic_activity \
6732  ▷  ▷  ▷  -report \
6733  ▷  ▷  -end
6734  ▷  pdflatex Sym_12_Subgroup_M12_95040_report.tex
6735  ▷  $(OPEN) Sym_12_Subgroup_M12_95040_report.pdf
6736
6737
6738
6739  ##############################################################################
6740  # Section 6.2: Linear Groups
6741
6742
6743  SECTION_LINEAR_GROUPS:
6744
6745  test_6_2:
6746  ▷  make PGL_3_2
6747  ▷  make PGL_4_2
6748  ▷  make PGL_4_2_multiply_all_elements
6749  ▷  make PGL_4_2_export
6750  ▷  make PGL_4_2_export_GAP
6751  ▷  make PGL_4_2_generated
6752  ▷  make PGL_5_2
6753  ▷  #make PGL_5_2_multiply_all_elements too slow
6754  ▷  make L_5_3
6755  ▷  make L_4_5
6756  ▷  make PGL_4_5
6757  ▷  make PGGL_3_4
6758  ▷  make PGGL_3_8
6759  ▷  make AGL_1_27
6760  ▷  make AGGL_2_27
6761  ▷  make SP_4_2
6762  ▷  make PSP_4_4
6763  ▷  make PGOp_4_2
6764  ▷  make PGO_5_2
6765  ▷  make PGGO_5_4
6766  ▷  make PGOp_6_2
```

```
6767  ▷  make PGOm_6_2
6768  ▷  make PSP_6_2
6769  ▷  make PGO_7_2
6770  ▷  make NullPolarity_6_2
6771
6772
6773
6774  PGL_3_2:
6775  ▷  $(ORBITER) -v 2 \
6776  ▷  ▷  -define F -finite_field -q 2 -end \
6777  ▷  ▷  -define G -linear_group -PGL 3 F -end \
6778  ▷  ▷  -with G -do \
6779  ▷  ▷  -group_theoretic_activity \
6780  ▷  ▷  ▷  -report \
6781  ▷  ▷  -end
6782  ▷  pdflatex PGL_3_2_report.tex
6783  ▷  $(OPEN) PGL_3_2_report.pdf
6784
6785
6786  PGL_4_2:
6787  ▷  $(ORBITER) -v 2 \
6788  ▷  ▷  -define F -finite_field -q 2 -end \
6789  ▷  ▷  -define G -linear_group -PGL 4 F -lex_least_base -end \
6790  ▷  ▷  -with G -do \
6791  ▷  ▷  -group_theoretic_activity \
6792  ▷  ▷  ▷  -report \
6793  ▷  ▷  -end
6794  ▷  pdflatex PGL_4_2_report.tex
6795  ▷  $(OPEN) PGL_4_2_report.pdf
6796
6797
6798  PGL_4_2_multiply_all_elements:
6799  ▷  $(ORBITER) -v 5 \
6800  ▷  ▷  -define F -finite_field -q 2 -end \
6801  ▷  ▷  -define G -linear_group -PGL 4 F -lex_least_base -end \
6802  ▷  ▷  -with G -do \
6803  ▷  ▷  -group_theoretic_activity \
6804  ▷  ▷  ▷  -multiply_all_elements_in_lex_order \
6805  ▷  ▷  -end
6806
6807  # 0.37 of a second, dt=37 tps = 100
6808
6809
6810  PGL_4_2_export:
6811  ▷  $(ORBITER) -v 2 \
6812  ▷  ▷  -define F -finite_field -q 2 -end \
6813  ▷  ▷  -define G -linear_group -PGL 4 F -end \
6814  ▷  ▷  -with G -do \
6815  ▷  ▷  -group_theoretic_activity \
6816  ▷  ▷  ▷  -report \
6817  ▷  ▷  -end \
6818  ▷  ▷  -with G -do \
6819  ▷  ▷  -group_theoretic_activity \
6820  ▷  ▷  ▷  -export_orbiter \
6821  ▷  ▷  -end
6822  ▷  pdflatex PGL_4_2_report.tex
6823  ▷  $(OPEN) PGL_4_2_report.pdf
6824
6825  # created by PGL_4_2_export
```

```
6826
6827
6828  PGL_4_2_export_GAP:
6829  ▷  $(ORBITER) -v 2 \
6830  ▷  ▷  -define F -finite_field -q 2 -end \
6831  ▷  ▷  -define G -linear_group -PGL 4 F -end \
6832  ▷  ▷  -with G -do \
6833  ▷  ▷  -group_theoretic_activity \
6834  ▷  ▷  ▷  -report \
6835  ▷  ▷  -end \
6836  ▷  ▷  -with G -do \
6837  ▷  ▷  -group_theoretic_activity \
6838  ▷  ▷  ▷  -export_gap \
6839  ▷  ▷  -end
6840  ▷  #pdflatex PGL_4_2_report.tex
6841  ▷  #$(OPEN) PGL_4_2_report.pdf
6842
6843
6844  PGL_4_2_generated:
6845  ▷  $(ORBITER) -v 2 \
6846  ▷  ▷  -define gens -vector -file PGL_4_2_gens.csv -end \
6847  ▷  ▷  -define G -permutation_group \
6848  ▷  ▷  -bsgs PGL_4_2 "{\rm PGL}(4,2)" 15 20160 "0,1,2,3" 6 gens -end \
6849
6850  PGL_5_2:
6851  ▷  $(ORBITER) -v 2 \
6852  ▷  ▷  -define F -finite_field -q 2 -end \
6853  ▷  ▷  -define G -linear_group -PGL 5 F -lex_least_base -end \
6854  ▷  ▷  -with G -do \
6855  ▷  ▷  -group_theoretic_activity \
6856  ▷  ▷  ▷  -report \
6857  ▷  ▷  -end
6858  ▷  pdflatex PGL_5_2_report.tex
6859  ▷  $(OPEN) PGL_5_2_report.pdf
6860
6861  #The order of the group PGGL(5, 2) is 9999360
6862
6863  PGL_5_2_multiply_all_elements:
6864  ▷  $(ORBITER) -v 5 \
6865  ▷  ▷  -define F -finite_field -q 2 -end \
6866  ▷  ▷  -define G -linear_group -PGL 5 F -lex_least_base -end \
6867  ▷  ▷  -with G -do \
6868  ▷  ▷  -group_theoretic_activity \
6869  ▷  ▷  ▷  -multiply_all_elements_in_lex_order \
6870  ▷  ▷  -end
6871
6872  # time: 9 min :28 sec on Macbook
6873
6874
6875  L_5_3:
6876  ▷  $(ORBITER) -v 2 \
6877  ▷  ▷  -define F -finite_field -q 3 -end \
6878  ▷  ▷  -define G -linear_group -PSL 5 F -end \
6879  ▷  ▷  -with G -do \
6880  ▷  ▷  -group_theoretic_activity \
6881  ▷  ▷  ▷  -report \
6882  ▷  ▷  -end
6883  ▷  pdflatex PSL_5_3_report.tex
6884  ▷  $(OPEN) PSL_5_3_report.pdf
```

```
6885
6886  #PSL(5,3): Order 237783237120 = 121 * 120 * 117 * 108 * 81 * 16
6887
6888
6889
6890  L_4_5:
6891  ▷  $(ORBITER) -v 2 \
6892  ▷  ▷  -define F -finite_field -q 5 -end \
6893  ▷  ▷  -define G -linear_group -PSL 4 F -end \
6894  ▷  ▷  -with G -do \
6895  ▷  ▷  -group_theoretic_activity \
6896  ▷  ▷  ▷  -report \
6897  ▷  ▷  -end
6898  ▷  pdflatex PSL_4_5_report.tex
6899  ▷  $(OPEN) PSL_4_5_report.pdf
6900
6901  #PSL(4,5): Order 7254000000
6902
6903
6904
6905
6906
6907
6908
6909
6910
6911  PGL_4_5:
6912  ▷  $(ORBITER) -v 2 \
6913  ▷  ▷  -define F -finite_field -q 5 -end \
6914  ▷  ▷  -define G -linear_group -PGL 4 F -end \
6915  ▷  ▷  -with G -do \
6916  ▷  ▷  -group_theoretic_activity \
6917  ▷  ▷  ▷  -report \
6918  ▷  ▷  -end
6919  ▷  pdflatex PGL_4_5_report.tex
6920  ▷  $(OPEN) PGL_4_5_report.pdf
6921
6922
6923
6924  PGGL_3_4:
6925  ▷  $(ORBITER) -v 2 \
6926  ▷  ▷  -define G -linear_group -PGGL 3 4 -end \
6927  ▷  ▷  -with G -do \
6928  ▷  ▷  -group_theoretic_activity \
6929  ▷  ▷  ▷  -report \
6930  ▷  ▷  ▷  -report_sylow \
6931  ▷  ▷  ▷  -report_classes \
6932  ▷  ▷  -end
6933  ▷  pdflatex PGGL_3_4_report.tex
6934  ▷  $(OPEN) PGGL_3_4_report.pdf
6935
6936
6937
6938  PGGL_3_8:
6939  ▷  $(ORBITER) -v 2 \
6940  ▷  ▷  -define G -linear_group -PGGL 3 8 -end \
6941  ▷  ▷  -with G -do \
6942  ▷  ▷  -group_theoretic_activity \
6943  ▷  ▷  ▷  -report \
```

```
6944  ▷  ▷  ▷  -report_sylow \
6945  ▷  ▷  ▷  -report_classes \
6946  ▷  ▷  -end
6947  ▷  pdflatex PGGL_3_8_report.tex
6948  ▷  $(OPEN) PGGL_3_8_report.pdf
6949
6950
6951
6952
6953  AGL_1_27:
6954  ▷  $(ORBITER) -v 2 \
6955  ▷  ▷  -define F -finite_field -q 27 -end \
6956  ▷  ▷  -define G -linear_group -AGL 1 F -end \
6957  ▷  ▷  -with G -do \
6958  ▷  ▷  -group_theoretic_activity \
6959  ▷  ▷  ▷  -report \
6960  ▷  ▷  -end
6961  ▷  pdflatex AGL_1_27_report.tex
6962  ▷  $(OPEN) AGL_1_27_report.pdf
6963
6964
6965  AGGL_2_27:
6966  ▷  $(ORBITER) -v 2 \
6967  ▷  ▷  -define F -finite_field -q 27 -end \
6968  ▷  ▷  -define G -linear_group -AGGL 2 F -end \
6969  ▷  ▷  -with G -do \
6970  ▷  ▷  -group_theoretic_activity \
6971  ▷  ▷  ▷  -report \
6972  ▷  ▷  -end
6973  ▷  pdflatex AGGL_2_27_report.tex
6974  ▷  $(OPEN) AGGL_2_27_report.pdf
6975
6976  #The order of the group AGGL(2, 27) is 1117679472
6977
6978
6979  SP_4_2:
6980  ▷  $(ORBITER) -v 2 \
6981  ▷  ▷  -define F -finite_field -q 2 -end \
6982  ▷  ▷  -define G -linear_group -GL 4 F \
6983  ▷  ▷  ▷  -symplectic_group \
6984  ▷  ▷  -end \
6985  ▷  ▷  -with G -do \
6986  ▷  ▷  -group_theoretic_activity \
6987  ▷  ▷  ▷  -report \
6988  ▷  ▷  -end
6989  ▷  pdflatex GL_4_2_Sp_4_2_report.tex
6990  ▷  $(OPEN) GL_4_2_Sp_4_2_report.pdf
6991
6992  # order 720
6993
6994
6995  PSP_4_4:
6996  ▷  $(ORBITER) -v 5 \
6997  ▷  ▷  -define F -finite_field -q 4 -end \
6998  ▷  ▷  -define G -linear_group -PGL 4 F \
6999  ▷  ▷  ▷  -symplectic_group \
7000  ▷  ▷  -end \
7001  ▷  ▷  -with G -do \
7002  ▷  ▷  -group_theoretic_activity \
```

```
7003  ▷  ▷  ▷  -report \
7004  ▷  ▷  -end
7005  ▷  pdflatex PGL_4_4_Sp_4_4_report.tex
7006  ▷  $(OPEN) PGL_4_4_Sp_4_4_report.pdf
7007
7008  #order 979200
7009
7010
7011  PGOp_4_2:
7012  ▷  $(ORBITER) -v 2 \
7013  ▷  ▷  -define F -finite_field -q 2 -end \
7014  ▷  ▷  -define G -linear_group -PGOp 4 F -end \
7015  ▷  ▷  -with G -do \
7016  ▷  ▷  -group_theoretic_activity \
7017  ▷  ▷  ▷  -report \
7018  ▷  ▷  -end
7019  ▷  pdflatex PGOp_4_2_report.tex
7020  ▷  $(OPEN) PGOp_4_2_report.pdf
7021
7022  # order 72
7023
7024
7025  PGO_5_2:
7026  ▷  $(ORBITER) -v 2 \
7027  ▷  ▷  -define F -finite_field -q 2 -end \
7028  ▷  ▷  -define G -linear_group -PGO 5 F -end \
7029  ▷  ▷  -with G -do \
7030  ▷  ▷  -group_theoretic_activity \
7031  ▷  ▷  ▷  -report \
7032  ▷  ▷  -end
7033  ▷  pdflatex PGO_5_2_report.tex
7034  ▷  $(OPEN) PGO_5_2_report.pdf
7035
7036  # order 720
7037
7038
7039  PGGO_5_4:
7040  ▷  $(ORBITER) -v 2 \
7041  ▷  ▷  -define F4 -finite_field -q 4 -end \
7042  ▷  ▷  -define G -linear_group -PGGO 5 F4 -end \
7043  ▷  ▷  -with G -do \
7044  ▷  ▷  -group_theoretic_activity \
7045  ▷  ▷  ▷  -report \
7046  ▷  ▷  -end
7047  ▷  pdflatex PGGO_5_4_report.tex
7048  ▷  $(OPEN) PGGO_5_4_report.pdf
7049
7050  # order 1958400
7051
7052  PGGO_5_4_export_gap:
7053  ▷  $(ORBITER) -v 2 \
7054  ▷  ▷  -define F4 -finite_field -q 4 -end \
7055  ▷  ▷  -define G -linear_group -PGGO 5 F4 -end \
7056  ▷  ▷  -with G -do \
7057  ▷  ▷  -group_theoretic_activity \
7058  ▷  ▷  ▷  -export_gap \
7059  ▷  ▷  -end
7060
7061
```

```
7062  PGGO_5_4_multiply_all_elements:
7063  ▷  $(ORBITER) -v 5 \
7064  ▷  ▷  -define F4 -finite_field -q 4 -end \
7065  ▷  ▷  -define G -linear_group -PGGO 5 F4 -end \
7066  ▷  ▷  -with G -do \
7067  ▷  ▷  -group_theoretic_activity \
7068  ▷  ▷  ▷  -multiply_all_elements_in_lex_order \
7069  ▷  ▷  -end
7070
7071
7072  PGOp_6_2:
7073  ▷  $(ORBITER) -v 2 \
7074  ▷  ▷  -define F -finite_field -q 2 -end \
7075  ▷  ▷  -define G -linear_group -PGOp 6 F -end \
7076  ▷  ▷  -with G -do \
7077  ▷  ▷  -group_theoretic_activity \
7078  ▷  ▷  ▷  -report \
7079  ▷  ▷  -end
7080  ▷  pdflatex PGOp_6_2_report.tex
7081  ▷  $(OPEN) PGOp_6_2_report.pdf
7082
7083  PGOm_6_2:
7084  ▷  $(ORBITER) -v 2 \
7085  ▷  ▷  -define F -finite_field -q 2 -end \
7086  ▷  ▷  -define G -linear_group -PGOm 6 F -end \
7087  ▷  ▷  -with G -do \
7088  ▷  ▷  -group_theoretic_activity \
7089  ▷  ▷  ▷  -report \
7090  ▷  ▷  -end
7091  ▷  pdflatex PGOm_6_2_report.tex
7092  ▷  $(OPEN) PGOm_6_2_report.pdf
7093
7094
7095
7096
7097
7098  # the following two groups are isomorphic:
7099
7100  PSP_6_2:
7101  ▷  $(ORBITER) -v 2 \
7102  ▷  ▷  -define F -finite_field -q 2 -end \
7103  ▷  ▷  -define G -linear_group -PGL 6 F \
7104  ▷  ▷  ▷  -symplectic_group \
7105  ▷  ▷  -end \
7106  ▷  ▷  -with G -do \
7107  ▷  ▷  -group_theoretic_activity \
7108  ▷  ▷  ▷  -report \
7109  ▷  ▷  -end
7110  ▷  pdflatex PGL_6_2_Sp_6_2_report.tex
7111  ▷  $(OPEN) PGL_6_2_Sp_6_2_report.pdf
7112
7113  # group order 1451520, acting on 63 points
7114
7115
7116  PGO_7_2:
7117  ▷  $(ORBITER) -v 2 \
7118  ▷  ▷  -define F -finite_field -q 2 -end \
7119  ▷  ▷  -define G -linear_group -PGO 7 F -end \
7120  ▷  ▷  -with G -do \
```

```
7121  ▷  ▷  -group_theoretic_activity \
7122  ▷  ▷  ▷  -report \
7123  ▷  ▷  -end
7124  ▷  pdflatex PGO_7_2_report.tex
7125  ▷  $(OPEN) PGO_7_2_report.pdf
7126
7127
7128  # group order 1451520, acting on 63 points
7129
7130
7131
7132
7133
7134  NullPolarity_6_2:
7135  ▷  $(ORBITER) -v 2 \
7136  ▷  ▷  -define F -finite_field -q 2 -end \
7137  ▷  ▷  -define G -linear_group -PGL 6 F \
7138  ▷  ▷  ▷  -null_polarity_group \
7139  ▷  ▷  -end \
7140  ▷  ▷  -with G -do \
7141  ▷  ▷  -group_theoretic_activity \
7142  ▷  ▷  ▷  -report \
7143  ▷  ▷  -end
7144  ▷  pdflatex PGL_6_2_NullPolarity_6_2_report.tex
7145  ▷  $(OPEN) PGL_6_2_NullPolarity_6_2_report.pdf
7146
7147  # group order 23040, acting on 63 points
7148
7149
7150
7151  ################################################################################
7152  # Section 6.3: Subgroups
7153
7154  SECTION_SUBGROUPS:
7155
7156
7157  test_6_3:
7158  ▷  make C13
7159  ▷  make C13_generated
7160  ▷  make C13_as_subgroup
7161  ▷  make J1
7162  ▷  make PGL_3_11_singer
7163  ▷  make PGL_3_11_singer_and_frobenius
7164  ▷  make PG_2_4_order_21
7165  ▷  make quaternion
7166  ▷  make cube_group
7167  ▷  make tetra_group
7168  ▷  make Hesse_group
7169  ▷  make Weyl_E8
7170  ▷  make test_subgroup
7171  ▷  make coset_reps
7172  ▷  make coset_reps_read
7173  ▷  make SP_6_2_point_stab_subgroup
7174  ▷  make PGOp_6_2_report
7175  ▷  make PGOm_6_2_report
7176  ▷  make PGOm_6_2_export_magma
7177  ▷  make PGOp_6_2_point_stab_subgroup
7178  ▷  make PGOp_6_2_linear
7179  ▷  make PGOp_6_2_linear_stab_6
```

```
7180
7181
7182
7183
7184  C13:
7185  ▷  $(ORBITER) -v 2 \
7186  ▷  ▷  -define gens -vector -dense $(GEN_C13) -end \
7187  ▷  ▷  -define G -permutation_group \
7188  ▷  ▷  ▷  -bsgs C13 C_{13} 13 13 0 1 \
7189  ▷  ▷  ▷  ▷  gens \
7190  ▷  ▷  ▷  -end \
7191  ▷  ▷  -with G -do \
7192  ▷  ▷  -group_theoretic_activity \
7193  ▷  ▷  ▷  -export_orbiter \
7194  ▷  ▷  -end \
7195  ▷  ▷  -with G -do \
7196  ▷  ▷  -group_theoretic_activity \
7197  ▷  ▷  ▷  -export_group_table \
7198  ▷  ▷  -end \
7199  ▷  ▷  -with G -do \
7200  ▷  ▷  -group_theoretic_activity \
7201  ▷  ▷  ▷  -report \
7202  ▷  ▷  -end \
7203  ▷  ▷  -with G -do \
7204  ▷  ▷  -group_theoretic_activity \
7205  ▷  ▷  ▷  -save_elements_csv "C13_elts.csv" \
7206  ▷  ▷  -end
7207  ▷  pdflatex C13_report.tex
7208  ▷  $(OPEN) C13_report.pdf
7209
7210
7211  C13_generated:
7212  ▷  $(ORBITER) -v 2 \
7213  ▷  ▷  -define gens -vector -file C13_gens.csv -end \
7214  ▷  ▷  -define G -permutation_group \
7215  ▷  ▷  -bsgs C13 "C_{13}" 13 13 "0" 1 gens -end \
7216
7217
7218  C13_as_subgroup:
7219  ▷  $(ORBITER) -v 2 \
7220  ▷  ▷  -define G -permutation_group -symmetric_group 13 \
7221  ▷  ▷  ▷  -subgroup_by_generators C13 13 1 $(GEN_C13) -end \
7222  ▷  ▷  -with G -do \
7223  ▷  ▷  -group_theoretic_activity \
7224  ▷  ▷  ▷  -export_orbiter \
7225  ▷  ▷  -end \
7226  ▷  ▷  -with G -do \
7227  ▷  ▷  -group_theoretic_activity \
7228  ▷  ▷  ▷  -report \
7229  ▷  ▷  -end \
7230  ▷  ▷  -with G -do \
7231  ▷  ▷  -group_theoretic_activity \
7232  ▷  ▷  ▷  -save_elements_csv "C13_elts.csv" \
7233  ▷  ▷  -end
7234  ▷  #pdflatex Perm13_Subgroup_C13_13_report.tex
7235  ▷  #$(OPEN) Perm13_Subgroup_C13_13_report.pdf
7236
7237
7238
```

```
7239
7240  J1:
7241  ▷  $(ORBITER) -v 2 \
7242  ▷  ▷  -define G -linear_group -PGL 7 11 -Janko1 -end \
7243  ▷  ▷  -with G -do \
7244  ▷  ▷  -group_theoretic_activity \
7245  ▷  ▷  ▷  -report \
7246  ▷  ▷  -end
7247  ▷  pdflatex Janko1_report.tex
7248  ▷  $(OPEN) Janko1_report.pdf
7249
7250
7251  PGL_3_11_singer:
7252  ▷  $(ORBITER) -v 2 \
7253  ▷  ▷  -define G -linear_group -PGL 3 11 -singer 19 -end \
7254  ▷  ▷  -with G -do \
7255  ▷  ▷  -group_theoretic_activity \
7256  ▷  ▷  ▷  -report \
7257  ▷  ▷  -end
7258  ▷  pdflatex PGL_3_11_Singer_3_11_19_report.tex
7259  ▷  $(OPEN) PGL_3_11_Singer_3_11_19_report.pdf
7260
7261
7262  PGL_3_11_singer_and_frobenius:
7263  ▷  $(ORBITER) -v 2 \
7264  ▷  ▷  -define G -linear_group -PGL 3 11 -singer_and_frobenius 19 -end \
7265  ▷  ▷  -with G -do \
7266  ▷  ▷  -group_theoretic_activity \
7267  ▷  ▷  ▷  -report \
7268  ▷  ▷  -end
7269  ▷  ▷  pdflatex PGL_3_11_Singer_and_Frob3_11_19_report.tex
7270  ▷  ▷  $(OPEN) PGL_3_11_Singer_and_Frob3_11_19_report.pdf
7271
7272  PG_2_4_order_21:
7273  ▷  $(ORBITER) -v 2 \
7274  ▷  ▷  -define G -linear_group -PGL 3 4 -end \
7275  ▷  ▷  -with G -do \
7276  ▷  ▷  -group_theoretic_activity \
7277  ▷  ▷  ▷  -search_element_of_order 21 \
7278  ▷  ▷  -end
7279
7280
7281
7282
7283  quaternion:
7284  ▷  $(ORBITER) -v 2 \
7285  ▷  ▷  -define G -linear_group -SL 2 3 \
7286  ▷  ▷  -subgroup_by_generators "quaternion" "8" 3 \
7287  ▷  ▷  ▷  "1,1,1,2, 2,1,1,1, 0,2,1,0" \
7288  ▷  ▷  -end \
7289  ▷  ▷  -with G -do \
7290  ▷  ▷  -group_theoretic_activity \
7291  ▷  ▷  ▷  -print_elements_tex \
7292  ▷  ▷  ▷  -report_group_table \
7293  ▷  ▷  ▷  -report \
7294  ▷  ▷  -end
7295  ▷  pdflatex SL_2_3_Subgroup_quaternion_8_report.tex
7296  ▷  $(OPEN) SL_2_3_Subgroup_quaternion_8_report.pdf
7297
```

```
7298
7299  cube_group:
7300  ▷  $(ORBITER) -v 2 \
7301  ▷  ▷  -define gens -vector -dense \
7302  ▷  ▷  ▷  "0,1,0,2,0,0,0,0,1, \
7303  ▷  ▷  ▷  0,0,1,0,1,0,2,0,0, \
7304  ▷  ▷  ▷  2,0,0,0,1,0,0,0,1" \
7305  ▷  ▷  -end \
7306  ▷  ▷  -define G -linear_group -GL 3 3 \
7307  ▷  ▷  -subgroup_by_generators "cube" "48" 3 \
7308  ▷  ▷  ▷  gens \
7309  ▷  ▷  -end \
7310  ▷  ▷  -with G -do \
7311  ▷  ▷  -group_theoretic_activity \
7312  ▷  ▷  ▷  -print_elements_tex \
7313  ▷  ▷  ▷  -report \
7314  ▷  ▷  -end
7315  ▷  pdflatex GL_3_3_Subgroup_cube_48_report.tex
7316  ▷  $(OPEN) GL_3_3_Subgroup_cube_48_report.pdf
7317
7318
7319  tetra_group:
7320  ▷  $(ORBITER) -v 3 \
7321  ▷  ▷  -define G -linear_group -GL 3 3 \
7322  ▷  ▷  -subgroup_by_generators "tetra" "12" 2 \
7323  ▷  ▷  ▷  "0,1,0,0,0,1,1,0,0, 0,0,1,2,0,0,0,2,0" \
7324  ▷  ▷  -end \
7325  ▷  ▷  -with G -do \
7326  ▷  ▷  -group_theoretic_activity \
7327  ▷  ▷  ▷  -print_elements_tex \
7328  ▷  ▷  ▷  -report \
7329  ▷  ▷  -end
7330  ▷  pdflatex GL_3_3_Subgroup_tetra_12_report.tex
7331  ▷  $(OPEN) GL_3_3_Subgroup_tetra_12_report.pdf
7332
7333
7334
7335  Hesse_group:
7336  ▷  $(ORBITER) -v 3 \
7337  ▷  ▷  -define gens -vector -compact \
7338  ▷  ▷  ▷  $(GENERATORS_HESSE_GROUP) \
7339  ▷  ▷  -end \
7340  ▷  ▷  -define G -linear_group -PGGL 3 4 \
7341  ▷  ▷  -subgroup_by_generators "Hesse" "432" 7 gens \
7342  ▷  ▷  -end \
7343  ▷  ▷  -with G -do \
7344  ▷  ▷  -group_theoretic_activity \
7345  ▷  ▷  ▷  -print_elements_tex \
7346  ▷  ▷  ▷  -report \
7347  ▷  ▷  -end
7348  ▷  pdflatex PGGL_3_4_Subgroup_Hesse_432_report.tex
7349  ▷  $(OPEN) PGGL_3_4_Subgroup_Hesse_432_report.pdf
7350
7351
7352
7353  Weyl_E8:
7354  ▷  $(ORBITER) -v 3 \
7355  ▷  ▷  -define gens -vector -dense \
7356  ▷  ▷  ▷  $(GENERATORS_WEYL_GROUP_E8) \
```

```
7357  ▷  ▷  -end \
7358  ▷  ▷  -define G -linear_group -GL 8 3 \
7359  ▷  ▷  -subgroup_by_generators \
7360  ▷  ▷  ▷  "Weyl_E8" "696729600" 2 gens  \
7361  ▷  ▷  -end \
7362  ▷  ▷  -with G -do \
7363  ▷  ▷  -group_theoretic_activity \
7364  ▷  ▷  ▷  -report \
7365  ▷  ▷  -end
7366  ▷  pdflatex GL_8_3_Subgroup_Weyl_E8_696729600_report.tex
7367  ▷  $(OPEN) GL_8_3_Subgroup_Weyl_E8_696729600_report.pdf
7368
7369  # group generators from http://www.math.rwth-aachen.de/~Gabriele.Nebe/LATTICES/E8
      .b.html
7370
7371
7372
7373  test_subgroup:
7374  ▷  $(ORBITER) -v 2 \
7375  ▷  ▷  -define F -finite_field -q 2 -end \
7376  ▷  ▷  -define G1 -linear_group -PGOp 6 F -end \
7377  ▷  ▷  -define G2 -linear_group -PGL 6 F \
7378  ▷  ▷  ▷  -symplectic_group \
7379  ▷  ▷  -end \
7380  ▷  ▷  -with G1 -and G2 -do \
7381  ▷  ▷  -group_theoretic_activity \
7382  ▷  ▷  ▷  -is_subgroup_of \
7383  ▷  ▷  -end
7384
7385
7386  coset_reps:
7387  ▷  $(ORBITER) -v 2 \
7388  ▷  ▷  -define F -finite_field -q 2 -end \
7389  ▷  ▷  -define G1 -linear_group -PGOp 6 F -end \
7390  ▷  ▷  -define G2 -linear_group -PGL 6 F \
7391  ▷  ▷  ▷  -symplectic_group \
7392  ▷  ▷  -end \
7393  ▷  ▷  -with G1 -and G2 -do \
7394  ▷  ▷  -group_theoretic_activity \
7395  ▷  ▷  ▷  -coset_reps \
7396  ▷  ▷  -end
7397  ▷  pdflatex PGOp_6_2_coset_reps.tex
7398  ▷  $(OPEN) PGOp_6_2_coset_reps.pdf
7399
7400  coset_reps_read:
7401  ▷  $(ORBITER) -v 2 \
7402  ▷  ▷  -define F -finite_field -q 2 -end \
7403  ▷  ▷  -define G1 -linear_group -PGOp 6 F -end \
7404  ▷  ▷  -define G2 -linear_group -PGL 6 F \
7405  ▷  ▷  ▷  -symplectic_group \
7406  ▷  ▷  -end \
7407  ▷  ▷  -define CR -vector_ge -action G2 \
7408  ▷  ▷  ▷  -read_csv \
7409  ▷  ▷  ▷  PGOp_6_2_coset_reps.csv Element \
7410  ▷  ▷  -end
7411
7412  SP_6_2_point_stab_subgroup:
7413  ▷  $(ORBITER) -v 2 \
7414  ▷  ▷  -define F -finite_field -q 2 -end \
```

```
7415  ▷  ▷  -define G -linear_group -PGL 6 F \
7416  ▷  ▷  ▷  -symplectic_group \
7417  ▷  ▷  -end \
7418  ▷  ▷  -define G0 -modified_group -from G \
7419  ▷  ▷  ▷  -point_stabilizer 0 \
7420  ▷  ▷  -end \
7421  ▷  ▷  -with G0 -do \
7422  ▷  ▷  -group_theoretic_activity \
7423  ▷  ▷  ▷  -report \
7424  ▷  ▷  -end
7425  ▷  pdflatex PGL_6_2_Sp_6_2_Stab0_report.tex
7426  ▷  $(OPEN) PGL_6_2_Sp_6_2_Stab0_report.pdf
7427
7428
7429  # group of order 23040
7430
7431
7432  PGOp_6_2_report:
7433  ▷  $(ORBITER) -v 2 \
7434  ▷  ▷  -define F -finite_field -q 2 -end \
7435  ▷  ▷  -define G -linear_group -PGOp 6 F -end \
7436  ▷  ▷  -with G -do \
7437  ▷  ▷  -group_theoretic_activity \
7438  ▷  ▷  ▷  -report \
7439  ▷  ▷  -end
7440  ▷  pdflatex PGOp_6_2_report.tex
7441  ▷  $(OPEN) PGOp_6_2_report.pdf
7442
7443  # group order 40320
7444
7445
7446  PGOm_6_2_report:
7447  ▷  $(ORBITER) -v 2 \
7448  ▷  ▷  -define F -finite_field -q 2 -end \
7449  ▷  ▷  -define G -linear_group -PGOm 6 F -end \
7450  ▷  ▷  -with G -do \
7451  ▷  ▷  -group_theoretic_activity \
7452  ▷  ▷  ▷  -report \
7453  ▷  ▷  -end
7454  ▷  pdflatex PGOm_6_2_report.tex
7455  ▷  $(OPEN) PGOm_6_2_report.pdf
7456
7457
7458
7459  PGOm_6_2_export_magma:
7460  ▷  $(ORBITER) -v 2 \
7461  ▷  ▷  -define F -finite_field -q 2 -end \
7462  ▷  ▷  -define G -linear_group -PGOm 6 F -end \
7463  ▷  ▷  -with G -do \
7464  ▷  ▷  -group_theoretic_activity \
7465  ▷  ▷  ▷  -export_magma \
7466  ▷  ▷  -end
7467
7468
7469
7470  PGOp_6_2_point_stab_subgroup:
7471  ▷  $(ORBITER) -v 2 \
7472  ▷  ▷  -define F -finite_field -q 2 -end \
7473  ▷  ▷  -define G -linear_group -PGOp 6 F -end \
```

```
7474  ▷  ▷  -define G0 -modified_group -from G \
7475  ▷  ▷  ▷  -point_stabilizer 0 \
7476  ▷  ▷  -end \
7477  ▷  ▷  -with G0 -do \
7478  ▷  ▷  -group_theoretic_activity \
7479  ▷  ▷  ▷  -report \
7480  ▷  ▷  -end
7481  ▷  pdflatex PGOp_6_2_report.tex
7482  ▷  $(OPEN) PGOp_6_2_report.pdf
7483
7484
7485  # group of order 1152
7486
7487
7488  PGOp_6_2_linear:
7489  ▷  $(ORBITER) -v 2 \
7490  ▷  ▷  -define F -finite_field -q 2 -end \
7491  ▷  ▷  -define G -linear_group -PGL 6 F \
7492  ▷  ▷  ▷  -subgroup_by_generators PGOp_6_2 \
7493  ▷  ▷  ▷  ▷  40320 10 $(PGOp_6_2_GENS) \
7494  ▷  ▷  -end \
7495  ▷  ▷  -with G -do \
7496  ▷  ▷  -group_theoretic_activity \
7497  ▷  ▷  ▷  -report \
7498  ▷  ▷  -end
7499  ▷  pdflatex PGL_6_2_Subgroup_PGOp_6_2_40320_report.tex
7500  ▷  $(OPEN) PGL_6_2_Subgroup_PGOp_6_2_40320_report.pdf
7501
7502
7503
7504  PGOp_6_2_linear_stab_6:
7505  ▷  $(ORBITER) -v 2 \
7506  ▷  ▷  -define F -finite_field -q 2 -end \
7507  ▷  ▷  -define G -linear_group -PGL 6 F \
7508  ▷  ▷  ▷  -subgroup_by_generators PGOp_6_2 \
7509  ▷  ▷  ▷  ▷  40320 10 $(PGOp_6_2_GENS) \
7510  ▷  ▷  -end \
7511  ▷  ▷  -define G6 -modified_group -from G \
7512  ▷  ▷  ▷  -point_stabilizer 6 \
7513  ▷  ▷  -end \
7514  ▷  ▷  -with G6 -do \
7515  ▷  ▷  -group_theoretic_activity \
7516  ▷  ▷  ▷  -report \
7517  ▷  ▷  -end
7518  ▷  pdflatex PGL_6_2_Subgroup_PGOp_6_2_40320_Stab6_report.tex
7519  ▷  $(OPEN) PGL_6_2_Subgroup_PGOp_6_2_40320_Stab6_report.pdf
7520
7521
7522  # group of order 1440, of index 28 in PGOp(6,2)
7523
7524
7525
7526
7527  ##############################################################################
7528  # Section 6.4: New Actions from Old
7529
7530
7531  SECTION_NEW_ACTIONS_FROM_OLD:
7532
```

```
7533
7534  test_6_4:
7535  ▷  make Symmetric_4_on_pairs
7536  ▷  make direct_product_AGL
7537  ▷  make PGGLstar_3_4
7538  ▷  make Group_M11
7539  ▷  make surface_q13_Eckardt
7540  ▷  make surface_q13_Eckardt_on_tritangent_planes
7541  ▷  make Hirschfeld_stab_projectivity_group
7542  ▷  make on_planes
7543  ▷  make on_planes_restricted
7544  ▷  make T3_on_tensors
7545  ▷  make T3r1
7546  ▷  make T4_on_tensors
7547  ▷  make T4r1
7548  ▷  #make PGGL_2_8_on_conic
7549  ▷  make PGL_4_2_wedge
7550  ▷  make PGL_5_3_wedge
7551  ▷  make PGL_4_2_wd
7552  ▷  make PGL_5_3_wd
7553  ▷  make PGL_4_2_wd_reverse
7554
7555
7556
7557
7558  Symmetric_4_on_pairs:
7559  ▷  $(ORBITER) -v 3 \
7560  ▷  ▷  -define G -permutation_group -symmetric_group 4 -end \
7561  ▷  ▷  -define G_on_2subsets -modified_group -from G \
7562  ▷  ▷  ▷  -on_k_subsets 2 \
7563  ▷  ▷  -end \
7564  ▷  ▷  -with G_on_2subsets -do \
7565  ▷  ▷  -group_theoretic_activity \
7566  ▷  ▷  ▷  -report \
7567  ▷  ▷  -end
7568  ▷  pdflatex Sym_4_OnSubsets_2_report.tex
7569  ▷  $(OPEN) Sym_4_OnSubsets_2_report.pdf
7570
7571
7572
7573  direct_product_AGL:
7574  ▷  $(ORBITER) -v 4 \
7575  ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
7576  ▷  ▷  -define G1 -linear_group -AGL 1 3 \
7577  ▷  ▷  ▷  -end \
7578  ▷  ▷  -define G2 -linear_group -AGL 1 7 \
7579  ▷  ▷  ▷  -end \
7580  ▷  ▷  -define G -modified_group -direct_product "G1,G2" \
7581  ▷  ▷  ▷  "21" "1,1,1,1" \
7582  ▷  ▷  -end \
7583  ▷  ▷  -with G -do \
7584  ▷  ▷  -group_theoretic_activity \
7585  ▷  ▷  ▷  -report \
7586  ▷  ▷  -end
7587  ▷  pdflatex product_AGL_1_3_AGL_1_7_report.tex
7588  ▷  $(OPEN) product_AGL_1_3_AGL_1_7_report.pdf
7589
7590  PGGLstar_3_4:
7591  ▷  $(ORBITER) -v 6 \
```

```
7592  ▷  ▷  -define F -finite_field -q 4 -end \
7593  ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
7594  ▷  ▷  -define G0 -linear_group -PGGL 3 F -end \
7595  ▷  ▷  -define G -modified_group -polarity_extension \
7596  ▷  ▷  ▷  G0 P \
7597  ▷  ▷  -end \
7598  ▷  ▷  -with G -do \
7599  ▷  ▷  -group_theoretic_activity \
7600  ▷  ▷  ▷  -report \
7601  ▷  ▷  -end
7602  ▷  pdflatex PGGL_3_4_polarity_ext_report.tex
7603  ▷  $(OPEN) PGGL_3_4_polarity_ext_report.pdf
7604
7605
7606
7607  Group_M11:
7608  ▷  $(ORBITER) -v 2 \
7609  ▷  ▷  -define gens -vector -dense $(GENERATORS_M12) -end \
7610  ▷  ▷  -define G -permutation_group -symmetric_group 12 \
7611  ▷  ▷  ▷  -subgroup_by_generators M12 95040 2 gens -end \
7612  ▷  ▷  -define G0 -modified_group -from G \
7613  ▷  ▷  ▷  -point_stabilizer 0 \
7614  ▷  ▷  -end \
7615  ▷  ▷  -with G0 -do \
7616  ▷  ▷  -group_theoretic_activity \
7617  ▷  ▷  ▷  -report \
7618  ▷  ▷  -end
7619  ▷  pdflatex Sym_12_Subgroup_M12_95040_Stab0_report.tex
7620  ▷  $(OPEN) Sym_12_Subgroup_M12_95040_Stab0_report.pdf
7621
7622
7623
7624  surface_q13_Eckardt:
7625  ▷  $(ORBITER) -v 3 \
7626  ▷  ▷  -define F -finite_field -q 13 -end \
7627  ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
7628  ▷  ▷  -define S -cubic_surface -space P -arc_lifting "0,1,2,3,43,113" -end \
7629  ▷  ▷  -with S -do \
7630  ▷  ▷  -cubic_surface_activity \
7631  ▷  ▷  ▷  -report \
7632  ▷  ▷  -end \
7633  ▷  ▷  -with S -do \
7634  ▷  ▷  -cubic_surface_activity \
7635  ▷  ▷  ▷  -export_something tritangent_planes \
7636  ▷  ▷  -end
7637  ▷  pdflatex surface_arc_lifting_trihedral_q13_arc0_1_2_3_43_113_report.tex
7638  ▷  $(OPEN) surface_arc_lifting_trihedral_q13_arc0_1_2_3_43_113_report.pdf
7639
7640  #surface_equation_F_Hirschfeld_q4_tritangent_planes.csv
7641
7642
7643  surface_q13_Eckardt_on_tritangent_planes:
7644  ▷  $(ORBITER) -v 2 \
7645  ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
7646  ▷  ▷  -draw_options -embedded -end \
7647  ▷  ▷  -define F -finite_field -q 13 -end \
7648  ▷  ▷  -define gens -vector -field F \
7649  ▷  ▷  ▷  -dense $(SURFACE_Q13_STAB_GENS) \
7650  ▷  ▷  -end \
```

```
7651  ▷  ▷  -define TriP -set -file \
7652  ▷  ▷  ▷  surface_arc_lifting_trihedral_q13_arc_0_1_2_3_43_113_tritangent_planes.csv
      \
7653  ▷  ▷  -end \
7654  ▷  ▷  -define G -linear_group -PGL 4 F \
7655  ▷  ▷  ▷  -subgroup_by_generators "stab" \
7656  ▷  ▷  ▷  ▷  24 3 gens \
7657  ▷  ▷  ▷  -end \
7658  ▷  ▷  -define G_on_planes -modified_group -from G \
7659  ▷  ▷  ▷  -on_k_subspaces 3 \
7660  ▷  ▷  -end \
7661  ▷  ▷  -define Gr -modified_group -from G_on_planes \
7662  ▷  ▷  ▷  -restricted_action TriP TriP \
7663  ▷  ▷  -end \
7664  ▷  ▷  -with Gr -do \
7665  ▷  ▷  -group_theoretic_activity \
7666  ▷  ▷  ▷  -report \
7667  ▷  ▷  -end \
7668  ▷  ▷  -define Orb -orbits -group Gr \
7669  ▷  ▷  ▷  -on_points \
7670  ▷  ▷  -end \
7671  ▷  ▷  -with Orb -do -orbits_activity \
7672  ▷  ▷  ▷  -report \
7673  ▷  ▷  -end \
7674  ▷  ▷  -with Orb -do -orbits_activity \
7675  ▷  ▷  ▷  -draw_tree 0 \
7676  ▷  ▷  -end \
7677  ▷  ▷  -with Orb -do -orbits_activity \
7678  ▷  ▷  ▷  -draw_tree 1 \
7679  ▷  ▷  -end \
7680  ▷  ▷  -with Orb -do -orbits_activity \
7681  ▷  ▷  ▷  -stabilizer 36 \
7682  ▷  ▷  -end
7683  ▷  pdflatex PGL_4_13_Gr_4_3_res_TriP_orbits_report.tex
7684  ▷  $(OPEN) PGL_4_13_Gr_4_3_res_TriP_orbits_report.pdf
7685
7686
7687  Hirschfeld_stab_projectivity_group:
7688  ▷  $(ORBITER) -v 4 \
7689  ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
7690  ▷  ▷  -define G -linear_group -PGGL 4 4 \
7691  ▷  ▷  ▷  -subgroup_by_generators "Hirschfeld_Stab" \
7692  ▷  ▷  ▷  51840 6 $(HIRSCHFELD_STAB_GENERATORS) \
7693  ▷  ▷  ▷  -end \
7694  ▷  ▷  -define G0 -modified_group -from G \
7695  ▷  ▷  ▷  -projectivity_subgroup \
7696  ▷  ▷  -end \
7697  ▷  ▷  -with G0 -do \
7698  ▷  ▷  -group_theoretic_activity \
7699  ▷  ▷  ▷  -report \
7700  ▷  ▷  -end
7701  ▷  pdflatex PGGL_4_4_Subgroup_Hirschfeld_Stab_51840_ProjectivitySubgroup_report.te
      x
7702  ▷  $(OPEN) PGGL_4_4_Subgroup_Hirschfeld_Stab_51840_ProjectivitySubgroup_report.pdf

7703
7704
7705
7706
```

```
7707
7708
7709  # related to planes_in_pencil:
7710  # we are computing the action on the planes through the line 0.
7711
7712
7713
7714  on_planes:
7715  ▷  $(ORBITER) -v 2 \
7716  ▷  ▷  -define F -finite_field -q 8 -end \
7717  ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
7718  ▷  ▷  -define G -linear_group -PGL 4 F -end \
7719  ▷  ▷  -define G_on_planes -modified_group -from G \
7720  ▷  ▷  ▷  -on_k_subspaces 3 \
7721  ▷  ▷  -end \
7722  ▷  ▷  -with G_on_planes -do \
7723  ▷  ▷  -group_theoretic_activity \
7724  ▷  ▷  ▷  -apply "0,8,1,6,4,3,7,2,5" \
7725  ▷  ▷  ▷  "1,0,0,0, 0,1,0,0, 0,0,0,2, 0,0,1,1" \
7726  ▷  ▷  -end
7727  ▷  pdflatex PGL_4_8_Gr_4_3_apply.tex
7728  ▷  $(OPEN) PGL_4_8_Gr_4_3_apply.pdf
7729
7730
7731  on_planes_restricted:
7732  ▷  $(ORBITER) -v 2 \
7733  ▷  ▷  -define planes -vector -dense "0,8,1,6,4,3,7,2,5" -end \
7734  ▷  ▷  -define F -finite_field -q 8 -end \
7735  ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
7736  ▷  ▷  -define G -linear_group -PGL 4 F \
7737  ▷  ▷  ▷  -subgroup_by_generators "cyclic" \
7738  ▷  ▷  ▷  9 1 "1,0,0,0, 0,1,0,0, 0,0,0,2, 0,0,1,1" \
7739  ▷  ▷  ▷  -end \
7740  ▷  ▷  -define G_on_planes -modified_group -from G \
7741  ▷  ▷  ▷  -on_k_subspaces 3 \
7742  ▷  ▷  -end \
7743  ▷  ▷  -define Gr -modified_group -from G_on_planes \
7744  ▷  ▷  ▷  -restricted_action planes planes \
7745  ▷  ▷  -end \
7746  ▷  ▷  -with Gr -do \
7747  ▷  ▷  -group_theoretic_activity \
7748  ▷  ▷  ▷  -report \
7749  ▷  ▷  -end
7750  ▷  pdflatex PGL_4_8_Gr_4_3_res_planes_report.tex
7751  ▷  $(OPEN) PGL_4_8_Gr_4_3_res_planes_report.pdf
7752
7753
7754  T3_on_tensors:
7755  ▷  $(ORBITER) -v 2 \
7756  ▷  ▷  -define G \
7757  ▷  ▷  -linear_group -GL_d_q_wr_Sym_n 2 2 3 \
7758  ▷  ▷  ▷  -on_tensors -end \
7759  ▷  ▷  -with G -do \
7760  ▷  ▷  -group_theoretic_activity \
7761  ▷  ▷  ▷  -report \
7762  ▷  ▷  -end
7763  ▷  pdflatex GL_2_2_wreath_Sym3_report.tex
7764  ▷  $(OPEN) GL_2_2_wreath_Sym3_report.pdf
7765  ▷
```

```
7766
7767
7768  T3r1:
7769  ▷  $(ORBITER) -v 4 \
7770  ▷  ▷  -define G \
7771  ▷  ▷  -linear_group -GL_d_q_wr_Sym_n 2 2 3 \
7772  ▷  ▷  ▷  -on_rank_one_tensors -end \
7773  ▷  ▷  -with G -do \
7774  ▷  ▷  -group_theoretic_activity \
7775  ▷  ▷  ▷  -report \
7776  ▷  ▷  -end
7777  ▷  pdflatex GL_2_2_wreath_Sym3_report.tex
7778  ▷  $(OPEN) GL_2_2_wreath_Sym3_report.pdf
7779
7780
7781
7782
7783
7784  T4_on_tensors:
7785  ▷  $(ORBITER) -v 4 \
7786  ▷  ▷  -define G \
7787  ▷  ▷  -linear_group -GL_d_q_wr_Sym_n 2 2 4 \
7788  ▷  ▷  ▷  -on_tensors -end \
7789  ▷  ▷  -with G -do \
7790  ▷  ▷  -group_theoretic_activity \
7791  ▷  ▷  ▷  -report \
7792  ▷  ▷  -end
7793  ▷  pdflatex GL_2_2_wreath_Sym4_report.tex
7794  ▷  $(OPEN) GL_2_2_wreath_Sym4_report.pdf
7795  ▷
7796  ▷
7797  T4r1:
7798  ▷  $(ORBITER) -v 4 \
7799  ▷  ▷  -define G \
7800  ▷  ▷  -linear_group -GL_d_q_wr_Sym_n 2 2 4 \
7801  ▷  ▷  ▷  -on_rank_one_tensors -end \
7802  ▷  ▷  -with G -do \
7803  ▷  ▷  -group_theoretic_activity \
7804  ▷  ▷  ▷  -report \
7805  ▷  ▷  -end
7806  ▷  pdflatex GL_2_2_wreath_Sym4_report.tex
7807  ▷  $(OPEN) GL_2_2_wreath_Sym4_report.pdf
7808
7809
7810  # ToDo
7811
7812  PGGL_2_8_on_conic:
7813  ▷  $(ORBITER) -v 4 \
7814  ▷  ▷  -define G -linear_group \
7815  ▷  ▷  ▷  -PGGL 2 8 -PGL2OnConic \
7816  ▷  ▷  -end \
7817  ▷  ▷  -with G -do \
7818  ▷  ▷  -group_theoretic_activity \
7819  ▷  ▷  ▷  -report \
7820  ▷  ▷  -end
7821  ▷  #pdflatex PGGL_2_8_OnConic_2_8_report.tex
7822  ▷  #$(OPEN) PGGL_2_8_OnConic_2_8_report.pdf
7823
7824  PGL_4_2_wedge:
```

```
7825  ▷  $(ORBITER) -v 3 \
7826  ▷  ▷  -define G -linear_group -PGL 4 2 -end \
7827  ▷  ▷  -define G2 -modified_group -from G \
7828  ▷  ▷  ▷  -on_wedge_product \
7829  ▷  ▷  -end \
7830  ▷  ▷  -with G -do \
7831  ▷  ▷  -group_theoretic_activity \
7832  ▷  ▷  ▷  -report \
7833  ▷  ▷  -end
7834  ▷  pdflatex PGL_4_2_report.tex
7835  ▷  $(OPEN) PGL_4_2_report.pdf
7836
7837
7838
7839
7840  PGL_5_3_wedge:
7841  ▷  $(ORBITER) -v 3 \
7842  ▷  ▷  -define G -linear_group -PGL 5 3 -end \
7843  ▷  ▷  -define G2 -modified_group -from G \
7844  ▷  ▷  ▷  -on_wedge_product \
7845  ▷  ▷  -end \
7846  ▷  ▷  -with G -do \
7847  ▷  ▷  -group_theoretic_activity \
7848  ▷  ▷  ▷  -report \
7849  ▷  ▷  -end
7850  ▷  pdflatex PGL_5_3_report.tex
7851  ▷  $(OPEN) PGL_5_3_report.pdf
7852
7853
7854
7855
7856
7857  PGL_4_2_wd:
7858  ▷  $(ORBITER) -v 3 \
7859  ▷  ▷  -define G -linear_group -PGL 4 2 -wedge_detached -end \
7860  ▷  ▷  -with G -do \
7861  ▷  ▷  -group_theoretic_activity \
7862  ▷  ▷  ▷  -report \
7863  ▷  ▷  -end
7864  ▷  pdflatex PGL_6_2_Wedge_4_2_detached_report.tex
7865  ▷  $(OPEN) PGL_6_2_Wedge_4_2_detached_report.pdf
7866
7867
7868
7869  PGL_5_3_wd:
7870  ▷  $(ORBITER) -v 3 \
7871  ▷  ▷  -define G -linear_group -PGL 5 3 -wedge_detached -end \
7872  ▷  ▷  -with G -do \
7873  ▷  ▷  -group_theoretic_activity \
7874  ▷  ▷  ▷  -report \
7875  ▷  ▷  -end
7876  ▷  pdflatex PGL_10_3_Wedge_5_3_detached_report.tex
7877  ▷  $(OPEN) PGL_10_3_Wedge_5_3_detached_report.pdf
7878
7879
7880  PGL_4_2_wd_reverse:
7881  ▷  $(ORBITER) -v 3 \
7882  ▷  ▷  -define G -linear_group -PGL 4 2 -wedge_detached -end \
7883  ▷  ▷  -with G -do \
```

```
7884  ▷  ▷  -group_theoretic_activity \
7885  ▷  ▷  ▷  -reverse_isomorphism_exterior_square \
7886  ▷  ▷  -end
7887
7888
7889
7890
7891
7892
7893  ################################################################################
7894  # Section 6.5: Group Theoretic Activities
7895
7896
7897  SECTION_GROUP_THEORETIC_ACTIVITIES:
7898
7899
7900  test_6_5:
7901  ▷  make PGL_3_2_elements
7902  ▷  make Sym_3_elements
7903  ▷  make Cycle_13_power
7904  ▷  make Cycle_12_power
7905  ▷  make PGL_3_4_singer
7906  ▷  make GL_2_8_multiply
7907  ▷  make GL_2_7_multiply
7908  ▷  make GL_2_7_inv
7909  ▷  make GL_2_7_power
7910  ▷  make PGL_3_2_classes
7911  ▷  make PGL_4_2_classes_based_on_normal_form
7912  ▷  make PGL_10_2_classes_based_on_normal_form
7913  ▷  make normal_forms_PGL_4_4
7914  ▷  make PGL_4_4_2A_rank
7915  ▷  make PGL_4_4_2A_unrank
7916  ▷  make PGL_4_5_3B_rank
7917  ▷  make PGL_4_5_3B_unrank
7918  ▷  make normal_forms_PGL_4_5
7919  ▷  make on_planes
7920  ▷  make PGL_4_2_random_element
7921
7922
7923
7924  PGL_3_2_elements:
7925  ▷  $(ORBITER) -v 5 \
7926  ▷  ▷  -define G -linear_group -PGL 3 2 -end \
7927  ▷  ▷  -with G -do \
7928  ▷  ▷  -group_theoretic_activity \
7929  ▷  ▷  ▷  -save_elements_csv "PGL_3_2_elements.csv" \
7930  ▷  ▷  -end
7931
7932  # creates PGL_3_2_elements.csv
7933
7934
7935  Sym_3_elements:
7936  ▷  $(ORBITER) -v 3 \
7937  ▷  ▷  -define G -permutation_group -symmetric_group 3 -end \
7938  ▷  ▷  -with G -do \
7939  ▷  ▷  -group_theoretic_activity \
7940  ▷  ▷  ▷  -print_elements_tex \
7941  ▷  ▷  -end
7942  ▷  $(ORBITER) -v 2 \
```

```
7943  ▷  ▷  -draw_options \
7944  ▷  ▷  ▷  -nodes \
7945  ▷  ▷  ▷  -embedded -radius 250 \
7946  ▷  ▷  ▷  -xin 10000 -yin 10000 \
7947  ▷  ▷  ▷  -xout 1000000 -yout 600000 \
7948  ▷  ▷  ▷  -scale 0.3 -line_width 1.0 \
7949  ▷  ▷  -end \
7950  ▷  ▷  -tree_draw -file Sym_3_elements_tree.txt -end
7951  ▷  pdflatex Sym_3_elements_tree_draw.tex
7952  ▷  $(OPEN) Sym_3_elements_tree_draw.pdf
7953
7954
7955
7956  Cycle_13_power:
7957  ▷  $(ORBITER) -v 5 \
7958  ▷  ▷  -define G -permutation_group -symmetric_group 13 -end \
7959  ▷  ▷  -with G -do \
7960  ▷  ▷  -group_theoretic_activity \
7961  ▷  ▷  ▷  -consecutive_powers \
7962  ▷  ▷  ▷  "1,2,3,4,5,6,7,8,9,10,11,12,0" 13 \
7963  ▷  ▷  -end
7964  ▷  pdflatex Sym_13_all_powers.tex
7965  ▷  $(OPEN) Sym_13_all_powers.pdf
7966
7967
7968  Cycle_12_power:
7969  ▷  $(ORBITER) -v 5 \
7970  ▷  ▷  -define G -permutation_group -symmetric_group 12 -end \
7971  ▷  ▷  -with G -do \
7972  ▷  ▷  -group_theoretic_activity \
7973  ▷  ▷  ▷  -consecutive_powers \
7974  ▷  ▷  ▷  "1,2,3,4,5,6,7,8,9,10,11,0" 12 \
7975  ▷  ▷  -end
7976  ▷  pdflatex Sym_12_all_powers.tex
7977  ▷  $(OPEN) Sym_12_all_powers.pdf
7978  ▷
7979
7980
7981
7982
7983  PGL_3_4_singer:
7984  ▷  $(ORBITER) -v 5 \
7985  ▷  ▷  -define G -linear_group -PGL 3 4 -end \
7986  ▷  ▷  -with G -do \
7987  ▷  ▷  -group_theoretic_activity \
7988  ▷  ▷  ▷  -find_singer_cycle \
7989  ▷  ▷  -end
7990
7991
7992  GL_2_8_multiply:
7993  ▷  $(ORBITER) -v 5 \
7994  ▷  ▷  -define G -linear_group -GL 2 8 -end \
7995  ▷  ▷  -with G -do \
7996  ▷  ▷  -group_theoretic_activity \
7997  ▷  ▷  ▷  -multiply "0,1,2,3" "4,5,6,7" \
7998  ▷  ▷  -end
7999  ▷  pdflatex GL_2_8_mult.tex
8000  ▷  $(OPEN) GL_2_8_mult.pdf
8001
```

```
8002
8003  GL_2_7_multiply:
8004  ▷  $(ORBITER) -v 5 \
8005  ▷  ▷  -define G -linear_group -GL 2 7 -end \
8006  ▷  ▷  -with G -do \
8007  ▷  ▷  -group_theoretic_activity \
8008  ▷  ▷  ▷  -multiply "0,1,2,3" "4,5,6,0" \
8009  ▷  ▷  -end
8010  ▷  pdflatex GL_2_7_mult.tex
8011  ▷  $(OPEN) GL_2_7_mult.pdf
8012
8013
8014  GL_2_7_inv:
8015  ▷  $(ORBITER) -v 5 \
8016  ▷  ▷  -define G -linear_group -GL 2 7 -end \
8017  ▷  ▷  -with G -do \
8018  ▷  ▷  -group_theoretic_activity \
8019  ▷  ▷  ▷  -inverse "0,1,2,3" \
8020  ▷  ▷  -end
8021  ▷  pdflatex GL_2_7_inv.tex
8022  ▷  $(OPEN) GL_2_7_inv.pdf
8023
8024  GL_2_7_power:
8025  ▷  $(ORBITER) -v 5 \
8026  ▷  ▷  -define G -linear_group -GL 2 7 -end \
8027  ▷  ▷  -with G -do \
8028  ▷  ▷  -group_theoretic_activity \
8029  ▷  ▷  ▷  -raise_to_the_power "0,1,2,3" 2 \
8030  ▷  ▷  -end
8031  ▷  pdflatex GL_2_7_power.tex
8032  ▷  $(OPEN) GL_2_7_power.pdf
8033
8034
8035  PGL_3_2_classes:
8036  ▷  $(ORBITER) -v 3 \
8037  ▷  ▷  -define G -linear_group -PGL 3 2 -end \
8038  ▷  ▷  -with G -do \
8039  ▷  ▷  -group_theoretic_activity \
8040  ▷  ▷  ▷  -classes_based_on_normal_form \
8041  ▷  ▷  -end
8042  ▷  pdflatex PGL_3_2_classes_based_on_normal_forms_3_2.tex
8043  ▷  $(OPEN) PGL_3_2_classes_based_on_normal_forms_3_2.pdf
8044  ▷  #pdflatex PGL_3_2_classes_out.tex
8045  ▷  #$(OPEN) PGL_3_2_classes_out.pdf
8046
8047  #▷ ▷  -classes \
8048
8049
8050  PGL_4_2_classes_based_on_normal_form:
8051  ▷  $(ORBITER) -v 3 \
8052  ▷  ▷  -define G -linear_group -PGL 4 2 -end \
8053  ▷  ▷  -with G -do \
8054  ▷  ▷  -group_theoretic_activity \
8055  ▷  ▷  ▷  -classes_based_on_normal_form \
8056  ▷  ▷  -end
8057  ▷  pdflatex PGL_4_2_classes_based_on_normal_forms_4_2.tex
8058  ▷  $(OPEN) PGL_4_2_classes_based_on_normal_forms_4_2.pdf
8059
8060
```

```
8061  PGL_10_2_classes_based_on_normal_form:
8062  ▷  $(ORBITER) -v 3 \
8063  ▷  ▷  -define G -linear_group -PGL 10 2 -end \
8064  ▷  ▷  -with G -do \
8065  ▷  ▷  -group_theoretic_activity \
8066  ▷  ▷  ▷  -classes_based_on_normal_form \
8067  ▷  ▷  -end
8068  ▷  pdflatex PGL_10_2_classes_based_on_normal_forms_10_2.tex
8069  ▷  $(OPEN) PGL_10_2_classes_based_on_normal_forms_10_2.pdf
8070
8071
8072
8073  normal_forms_PGL_4_4:
8074  ▷  $(ORBITER) -v 7 \
8075  ▷  ▷  -define G -linear_group -PGGL 4 4 -end \
8076  ▷  ▷  -with G -do \
8077  ▷  ▷  -group_theoretic_activity \
8078  ▷  ▷  ▷  -classes_based_on_normal_form \
8079  ▷  ▷  -end
8080  ▷  pdflatex PGGL_4_4_classes_based_on_normal_forms_4_4.tex
8081  ▷  $(OPEN) PGGL_4_4_classes_based_on_normal_forms_4_4.pdf
8082
8083
8084
8085
8086  PGL_4_4_2A_rank:
8087  ▷  $(ORBITER) -v 6 \
8088  ▷  ▷  -define G -linear_group -PGGL 4 4 -end \
8089  ▷  ▷  -with G -do \
8090  ▷  ▷  -group_theoretic_activity \
8091  ▷  ▷  ▷  -element_rank \
8092  ▷  ▷  ▷  "1,0,0,0, 0,1,0,0, 0,0,1,0, 0,0,0,1, 1" \
8093  ▷  ▷  -end
8094
8095
8096  PGL_4_4_2A_unrank:
8097  ▷  $(ORBITER) -v 6 \
8098  ▷  ▷  -define G -linear_group -PGGL 4 4 -end \
8099  ▷  ▷  -with G -do \
8100  ▷  ▷  -group_theoretic_activity \
8101  ▷  ▷  ▷  -element_unrank "1" \
8102  ▷  ▷  -end
8103  ▷  ▷
8104
8105
8106
8107
8108
8109
8110  PGL_4_5_3B_rank:
8111  ▷  $(ORBITER) -v 6 \
8112  ▷  ▷  -define G -linear_group -PGL 4 5 -end \
8113  ▷  ▷  -with G -do \
8114  ▷  ▷  -group_theoretic_activity \
8115  ▷  ▷  ▷  -element_rank "0,0,0,1, 2,3,0,1, 0,3,4,4, 0,1,2,1" \
8116  ▷  ▷  -end
8117
8118
8119  PGL_4_5_3B_unrank:
```

```
	$(ORBITER) -v 6 \
	  -define G -linear_group -PGL 4 5 -end \
	  -with G -do \
	  -group_theoretic_activity \
	    -element_unrank "701459351" \
	  -end
	
	


normal_forms_PGL_4_5:
	$(ORBITER) -v 7 \
	  -define G -linear_group -PGL 4 5 -end \
	  -with G -do \
	  -group_theoretic_activity \
	    -classes_based_on_normal_form \
	  -end
	pdflatex PGL_4_5_classes_based_on_normal_forms_4_5.tex
	$(OPEN) PGL_4_5_classes_based_on_normal_forms_4_5.pdf


PGL_4_2_random_element:
	$(ORBITER) -v 2 \
	  -define F -finite_field -q 2 -end \
	  -define G -linear_group -PGL 4 F -end \
	  -with G -do \
	  -group_theoretic_activity \
	    -random_element r1 \
	  -end \
	  -print_symbols


#group order H = 20160
#Element :
#1 0 1 1
#0 1 1 0
#0 1 0 1
#1 0 1 0


#coded: ( 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0 )
#Element as permutation:
#(0, 13, 1, 7, 12, 4, 2, 10, 5, 11, 14, 9, 3, 6, 8)
#In list notation:
#( 13, 7, 10, 6, 2, 11, 8, 12, 0, 3, 5, 14, 4, 1, 9 )
#The rank of the element is 9883

#\{0,1,2,3,4\} -> \{13,7,10,6,2\} = \{2,6,7,10,13\}


################################################################################
# Section 6.6: Group Theoretic Activities Based on Magma


SECTION_GROUP_THEORETIC_ACTIVITIES_BASED_ON_MAGMA:


test_6_6:
```

```
8179  ▷  make PSP_4_4_export_magma
8180  ▷  make PGGL_2_4_export_magma
8181  ▷  make Hirschfeld_stab_export_magma
8182  ▷  make PGGL_2_4_classes
8183  ▷  make PGL_7_2_classes
8184  ▷  make PGL_8_2_classes
8185  ▷  make Hirschfeld_stab_classes
8186  ▷  make PGGL_2_4_cent_2A
8187  ▷  make Normalizer_of_H5
8188  ▷  make PGGL_3_4_classes
8189  ▷  make classes_PGGL_4_4
8190  ▷  make subgroups_PGL_4_5
8191  ▷  make classes_PGL_4_5
8192  ▷  #make PGL_4_5_3B_class_again
8193  ▷  make search_primitive_poly_q5_deg3
8194  ▷  make GL_3_5_singer_power
8195  ▷  make PGL_4_5_norm_31
8196  ▷  make Normalizer_of_Z22_in_PGL_2_9
8197  ▷  make PGOm_6_2_classes
8198  ▷  make PGOm_6_2_Class_2A
8199  ▷  make PGOm_6_2_Class_2A_print
8200  ▷  make PGOm_6_2_Class_2A_centralizer
8201  ▷  make PGOm_6_2_Class_2A_centralizer_recover
8202
8203
8204
8205
8206  PSP_4_4_export_magma:
8207  ▷  $(ORBITER) -v 5 \
8208  ▷  ▷  -define F -finite_field -q 4 -end \
8209  ▷  ▷  -define G -linear_group -PGL 4 F \
8210  ▷  ▷  ▷  -symplectic_group \
8211  ▷  ▷  -end \
8212  ▷  ▷  -with G -do \
8213  ▷  ▷  -group_theoretic_activity \
8214  ▷  ▷  ▷  -export_magma \
8215  ▷  ▷  -end
8216
8217  # creates PGL_4_4_Sp_4_4_generators.magma
8218
8219
8220  PGGL_2_4_export_magma:
8221  ▷  $(ORBITER) -v 20 \
8222  ▷  ▷  -define F -finite_field -q 4 \
8223  ▷  ▷  ▷  -compute_related_fields \
8224  ▷  ▷  -end \
8225  ▷  ▷  -define G -linear_group -PGGL 2 F \
8226  ▷  ▷  -end \
8227  ▷  ▷  -with G -do \
8228  ▷  ▷  -group_theoretic_activity \
8229  ▷  ▷  ▷  -report \
8230  ▷  ▷  -end \
8231  ▷  ▷  -with G -do \
8232  ▷  ▷  -group_theoretic_activity \
8233  ▷  ▷  ▷  -export_magma \
8234  ▷  ▷  -end
8235  ▷  pdflatex PGGL_2_4_report.tex
8236  ▷  $(OPEN) PGGL_2_4_report.pdf
8237
```

```
8238
8239
8240
8241  Hirschfeld_stab_export_magma:
8242  ▷  $(ORBITER) -v 9 \
8243  ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
8244  ▷  ▷  -define F -finite_field -q 4 \
8245  ▷  ▷  ▷  -compute_related_fields \
8246  ▷  ▷  -end \
8247  ▷  ▷  -define G -linear_group -PGGL 4 F \
8248  ▷  ▷  ▷  -subgroup_by_generators "Hirschfeld_Stab" \
8249  ▷  ▷  ▷  51840 6 $(HIRSCHFELD_STAB_GENERATORS) \
8250  ▷  ▷  ▷  -end \
8251  ▷  ▷  -with G -do \
8252  ▷  ▷  -group_theoretic_activity \
8253  ▷  ▷  ▷  -export_magma \
8254  ▷  ▷  -end
8255
8256
8257  PGGL_2_4_classes:
8258  ▷  $(ORBITER) -v 3 \
8259  ▷  ▷  -define G \
8260  ▷  ▷  -linear_group -PGGL 2 4 \
8261  ▷  ▷  -end \
8262  ▷  ▷  -with G -do \
8263  ▷  ▷  -group_theoretic_activity \
8264  ▷  ▷  ▷  -classes \
8265  ▷  ▷  -end
8266  ▷  $(MAGMA_PATH)magma PGGL_2_4_classes.magma
8267  ▷  $(ORBITER) -v 3 \
8268  ▷  ▷  -define G \
8269  ▷  ▷  -linear_group -PGGL 2 4 \
8270  ▷  ▷  -end \
8271  ▷  ▷  -with G -do \
8272  ▷  ▷  -group_theoretic_activity \
8273  ▷  ▷  ▷  -classes \
8274  ▷  ▷  -end
8275  ▷  pdflatex PGGL_2_4_classes_out.tex
8276  ▷  $(OPEN) PGGL_2_4_classes_out.pdf
8277  ▷  $(OPEN) PGGL_2_4_classes_out.csv
8278
8279
8280
8281  PGL_7_2_classes:
8282  ▷  $(ORBITER) -v 3 \
8283  ▷  ▷  -define G \
8284  ▷  ▷  -linear_group -PGL 7 2 \
8285  ▷  ▷  -end \
8286  ▷  ▷  -with G -do \
8287  ▷  ▷  -group_theoretic_activity \
8288  ▷  ▷  ▷  -classes \
8289  ▷  ▷  -end
8290  ▷  $(MAGMA_PATH)magma PGL_7_2_classes.magma
8291  ▷  $(ORBITER) -v 3 \
8292  ▷  ▷  -define G \
8293  ▷  ▷  -linear_group -PGL 7 2 \
8294  ▷  ▷  -end \
8295  ▷  ▷  -with G -do \
8296  ▷  ▷  -group_theoretic_activity \
```

```
8297  ▷  ▷  ▷  -classes \
8298  ▷  ▷  -end
8299
8300
8301  PGL_8_2_classes:
8302  ▷  $(ORBITER) -v 3 \
8303  ▷  ▷  -define G \
8304  ▷  ▷  -linear_group -PGL 8 2 \
8305  ▷  ▷  -end \
8306  ▷  ▷  -with G -do \
8307  ▷  ▷  -group_theoretic_activity \
8308  ▷  ▷  ▷  -classes \
8309  ▷  ▷  -end
8310  ▷  $(MAGMA_PATH)magma PGL_8_2_classes.magma
8311  ▷  $(ORBITER) -v 3 \
8312  ▷  ▷  -define G \
8313  ▷  ▷  -linear_group -PGL 8 2 \
8314  ▷  ▷  -end \
8315  ▷  ▷  -with G -do \
8316  ▷  ▷  -group_theoretic_activity \
8317  ▷  ▷  ▷  -classes \
8318  ▷  ▷  -end
8319
8320
8321
8322
8323
8324  Hirschfeld_stab_classes:
8325  ▷  $(ORBITER) -v 9 \
8326  ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
8327  ▷  ▷  -define G -linear_group -PGGL 4 4 \
8328  ▷  ▷  ▷  -subgroup_by_generators "Hirschfeld_Stab" \
8329  ▷  ▷  ▷  51840 6 $(HIRSCHFELD_STAB_GENERATORS) \
8330  ▷  ▷  ▷  -end \
8331  ▷  ▷  -with G -do \
8332  ▷  ▷  -group_theoretic_activity \
8333  ▷  ▷  ▷  -classes \
8334  ▷  ▷  -end
8335  ▷  $(ORBITER) -v 9 \
8336  ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
8337  ▷  ▷  -define G -linear_group -PGGL 4 4 \
8338  ▷  ▷  ▷  -subgroup_by_generators "Hirschfeld_Stab" \
8339  ▷  ▷  ▷  51840 6 $(HIRSCHFELD_STAB_GENERATORS) \
8340  ▷  ▷  ▷  -end \
8341  ▷  ▷  -with G -do \
8342  ▷  ▷  -group_theoretic_activity \
8343  ▷  ▷  ▷  -classes \
8344  ▷  ▷  -end
8345  ▷  pdflatex PGGL_4_4_Subgroup_Hirschfeld_Stab_51840_classes_out.tex
8346  ▷  $(OPEN) PGGL_4_4_Subgroup_Hirschfeld_Stab_51840_classes_out.pdf
8347
8348
8349
8350  PGGL_2_4_cent_2A:
8351  ▷  $(ORBITER) -v 3 \
8352  ▷  ▷  -define G \
8353  ▷  ▷  -linear_group -PGGL 2 4 -end \
8354  ▷  ▷  -with G -do \
8355  ▷  ▷  -group_theoretic_activity \
```

```
8356  ▷  ▷  ▷  -centralizer_of_element "2A" "1,0, 0,1, 1" \
8357  ▷  ▷  ▷  -report \
8358  ▷  ▷  -end
8359  ▷  $(MAGMA_PATH)magma element_2A_centralizer.magma
8360  ▷  $(ORBITER) -v 6 \
8361  ▷  ▷  -define G \
8362  ▷  ▷  -linear_group -PGGL 2 4 -end \
8363  ▷  ▷  -with G -do \
8364  ▷  ▷  -group_theoretic_activity \
8365  ▷  ▷  ▷  -centralizer_of_element "2A" "1,0, 0,1, 1" \
8366  ▷  ▷  ▷  -report \
8367  ▷  ▷  -end
8368  ▷  pdflatex PGGL_2_4_elt_2A_centralizer.tex
8369  ▷  $(OPEN) PGGL_2_4_elt_2A_centralizer.pdf
8370  ▷
8371
8372
8373  Normalizer_of_H5:
8374  ▷  $(ORBITER) -v 2 \
8375  ▷  ▷  -define G -permutation_group -symmetric_group 13 \
8376  ▷  ▷  ▷  -subgroup_by_generators H5 5 1 \
8377  ▷  ▷  ▷  ▷  $(GENERATORS_H5) -end \
8378  ▷  ▷  -with G -do \
8379  ▷  ▷  -group_theoretic_activity \
8380  ▷  ▷  ▷  -normalizer \
8381  ▷  ▷  -end
8382  ▷  pdflatex Sym_13_Subgroup_H5_5_normalizer.tex
8383  ▷  $(OPEN) Sym_13_Subgroup_H5_5_normalizer.pdf
8384
8385
8386
8387
8388  PGGL_3_4_classes:
8389  ▷  $(ORBITER) -v 3 \
8390  ▷  ▷  -define G \
8391  ▷  ▷  -linear_group -PGGL 3 4 \
8392  ▷  ▷  -end \
8393  ▷  ▷  -with G -do \
8394  ▷  ▷  -group_theoretic_activity \
8395  ▷  ▷  ▷  -classes \
8396  ▷  ▷  -end
8397  ▷  pdflatex PGGL_3_4_classes_out.tex
8398  ▷  $(OPEN) PGGL_3_4_classes_out.pdf
8399
8400
8401
8402
8403
8404
8405
8406  classes_PGGL_4_4:
8407  ▷  $(ORBITER) -v 3 \
8408  ▷  ▷  -magma_path $(MAGMA_PATH) \
8409  ▷  ▷  -define G -linear_group -PGGL 4 4 -end \
8410  ▷  ▷  -with G -do \
8411  ▷  ▷  -group_theoretic_activity \
8412  ▷  ▷  ▷  -classes \
8413  ▷  ▷  -end
8414
```

```
8415  # group order 1974067200 = 2^13 * 3^4 * 5^2 * 7 * 17
8416
8417
8418
8419
8420
8421  # the -find_subgroup command is too specialized
8422
8423  subgroups_PGL_4_5:
8424  ▷  $(ORBITER) -v 6 \
8425  ▷  ▷  -define G \
8426  ▷  ▷  -linear_group -PGL 4 5 -end \
8427  ▷  ▷  -with G -do \
8428  ▷  ▷  -group_theoretic_activity \
8429  ▷  ▷  ▷  -find_subgroup 3 \
8430  ▷  ▷  -end
8431  ▷  pdflatex PGL_4_5_report.tex
8432  ▷  $(OPEN) PGL_4_5_report.pdf
8433
8434
8435  classes_PGL_4_5:
8436  ▷  $(ORBITER) -v 6 \
8437  ▷  ▷  -define G \
8438  ▷  ▷  -linear_group -PGL 4 5 -end \
8439  ▷  ▷  -with G -do \
8440  ▷  ▷  -group_theoretic_activity \
8441  ▷  ▷  ▷  -classes \
8442  ▷  ▷  -end
8443  ▷  pdflatex PGL_4_5_classes_out.tex
8444  ▷  $(OPEN) PGL_4_5_classes_out.pdf
8445  ▷
8446  # 163 classes
8447
8448
8449  # two classes of elements of order 3
8450  #Order of element = 3 Class size = 310000 Centralizer order = 93600 Normalizer or
      der = 187200
8451  #of order 3 and with 0 fixed points.
8452  #0,1,0,2,0,1,2,1,4,2,3,1,2,0,4,3,
8453
8454  #Class size = 10075000 Centralizer order = 2880 Normalizer order = 5760
8455  #of order 3 and with 6 fixed points.
8456  #0,0,0,1,2,3,0,1,0,3,4,4,0,1,2,1,
8457
8458
8459  PGL_4_5_3B_class_again:
8460  ▷  $(ORBITER) -v 6 \
8461  ▷  ▷  -define G -linear_group -PGL 4 5 -end \
8462  ▷  ▷  -with G -do \
8463  ▷  ▷  -group_theoretic_activity \
8464  ▷  ▷  ▷  -conjugacy_class_of \
8465  ▷  ▷  ▷  "3B" "0,0,0,1, 2,3,0,1, 0,3,4,4, 0,1,2,1" \
8466  ▷  ▷  -end
8467
8468  # very slow
8469
8470
8471  search_primitive_poly_q5_deg3:
8472  ▷  $(ORBITER) -v 6 \
```

```
8473  ▷  ▷  -define F -finite_field -q 5 -end \
8474  ▷  ▷  -with F -do \
8475  ▷  ▷  -finite_field_activity \
8476  ▷  ▷  -get_primitive_polynomial 3 -end
8477
8478
8479  #OK, we found an irreducible and primitive polynomial X^{3} + X^{2} + 2
8480
8481  GL_3_5_singer_power:
8482  ▷  $(ORBITER) -v 6 -define G \
8483  ▷  ▷  -linear_group -GL 3 5 -end \
8484  ▷  ▷  -with G -do \
8485  ▷  ▷  -group_theoretic_activity \
8486  ▷  ▷  ▷  -raise_to_the_power \
8487  ▷  ▷  ▷  "0,1,0, 0,0,1, 3,0,4" 31 \
8488  ▷  ▷  -end
8489  ▷  pdflatex GL_3_5_power.tex
8490  ▷  $(OPEN) GL_3_5_power.pdf
8491
8492
8493
8494  PGL_4_5_norm_31:
8495  ▷  $(ORBITER) -v 6 -define G \
8496  ▷  ▷  -linear_group -PGL 4 5 -end \
8497  ▷  ▷  -with G -do \
8498  ▷  ▷  -group_theoretic_activity \
8499  ▷  ▷  ▷  -normalizer_of_cyclic_subgroup "31" \
8500  ▷  ▷  ▷  "2,0,0,0, 0,0,1,0, 0,0,0,1, 0,3,0,4"
8501  ▷  ▷  -end
8502  ▷  pdflatex normalizer_of_31_in_PGL_4_5.tex
8503  ▷  $(OPEN) normalizer_of_31_in_PGL_4_5.pdf
8504
8505
8506
8507  Normalizer_of_Z22_in_PGL_2_9:
8508  ▷  $(ORBITER) -v 2 \
8509  ▷  ▷  -define G -linear_group -PGL 2 9 \
8510  ▷  ▷  -subgroup_by_generators Z22 4 2 \
8511  ▷  ▷  ▷  "2,0,0,1, 0,1,1,0" -end \
8512  ▷  ▷  -with G -do \
8513  ▷  ▷  -group_theoretic_activity \
8514  ▷  ▷  ▷  -normalizer \
8515  ▷  ▷  -end
8516  ▷  pdflatex PGL_2_9_Subgroup_Z22_4_normalizer.tex
8517  ▷  $(OPEN) PGL_2_9_Subgroup_Z22_4_normalizer.pdf
8518
8519
8520
8521
8522  PGOm_6_2_classes:
8523  ▷  $(ORBITER) -v 2 \
8524  ▷  ▷  -define F -finite_field -q 2 -end \
8525  ▷  ▷  -define G -linear_group -PGOm 6 F -end \
8526  ▷  ▷  -with G -do \
8527  ▷  ▷  -group_theoretic_activity \
8528  ▷  ▷  ▷  -classes \
8529  ▷  ▷  -end
8530  ▷  pdflatex PGOm_6_2_classes_out.tex
8531  ▷  $(OPEN) PGOm_6_2_classes_out.pdf
```

```
8532
8533
8534  PGOm_6_2_Class_2A:
8535  ▷  $(ORBITER) -v 9 \
8536  ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
8537  ▷  ▷  -define F -finite_field -q 2 -end \
8538  ▷  ▷  -define G -linear_group -PGOm 6 F -end \
8539  ▷  ▷  -with G -do \
8540  ▷  ▷  -group_theoretic_activity \
8541  ▷  ▷  ▷  -conjugacy_class_of "2A" $(PGOM_6_2_CLASS_2A_REP) \
8542  ▷  ▷  -end
8543
8544
8545
8546  #PGOm_6_2_class_of_2A.csv
8547
8548
8549  PGOm_6_2_Class_2A_print:
8550  ▷  $(ORBITER) -v 9 \
8551  ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
8552  ▷  ▷  -define PGOm_6_2_Class2A -vector \
8553  ▷  ▷  ▷  -file $(FILE_NAME_PGOM_6_2_CLASS_2A) -end \
8554  ▷  ▷  -define F -finite_field -q 2 -end \
8555  ▷  ▷  -define G -linear_group -PGOm 6 F -end \
8556  ▷  ▷  -with G -do \
8557  ▷  ▷  -group_theoretic_activity \
8558  ▷  ▷  ▷  -element_processing \
8559  ▷  ▷  ▷  ▷  -input PGOm_6_2_Class2A \
8560  ▷  ▷  ▷  ▷  -print \
8561  ▷  ▷  ▷  -end \
8562  ▷  ▷  -end
8563  ▷  pdflatex PGOm_6_2_Class2A_elements.tex
8564  ▷  $(OPEN) PGOm_6_2_Class2A_elements.pdf
8565
8566
8567
8568
8569  PGOm_6_2_Class_2A_centralizer:
8570  ▷  $(ORBITER) -v 9 \
8571  ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
8572  ▷  ▷  -define F -finite_field -q 2 -end \
8573  ▷  ▷  -define G -linear_group -PGOm 6 F -end \
8574  ▷  ▷  -with G -do \
8575  ▷  ▷  -group_theoretic_activity \
8576  ▷  ▷  ▷  -centralizer_of_element \
8577  ▷  ▷  ▷  ▷  "2Arep" $(PGOM_6_2_CLASS_2A_REP_CLEAN) \
8578  ▷  ▷  -end
8579  ▷  pdflatex PGOm_6_2_elt_2Arep_centralizer.tex
8580  ▷  $(OPEN) PGOm_6_2_elt_2Arep_centralizer.pdf
8581
8582
8583
8584  PGOm_6_2_Class_2A_centralizer_recover:
8585  ▷  $(ORBITER) -v 2 \
8586  ▷  ▷  -define F -finite_field -q 2 -end \
8587  ▷  ▷  -define Class2A_cent_gens -vector \
8588  ▷  ▷  ▷  -dense $(PGOM_6_2_CLASS_2A_CENTRALIZER_GENS) -end \
8589  ▷  ▷  -define G -linear_group -PGL 6 2 \
8590  ▷  ▷  ▷  -subgroup_by_generators "2ACent" \
```

```
8591  ▷  ▷  ▷  1440 9 Class2A_cent_gens \
8592  ▷  ▷  ▷  -end \
8593  ▷  ▷  -with G -do \
8594  ▷  ▷  -group_theoretic_activity \
8595  ▷  ▷  ▷  -report \
8596  ▷  ▷  -end
8597  ▷  pdflatex PGL_6_2_Subgroup_2ACent_1440_report.tex
8598  ▷  $(OPEN) PGL_6_2_Subgroup_2ACent_1440_report.pdf
8599
8600
8601  ################################################################################
8602  # Section 6.7: The GAP Interface
8603
8604
8605  SECTION_GAP_INTERFACE:
8606
8607  test_6_7:
8608  ▷  make Hirschfeld_stab_export_gap
8609  ▷  make Cyclic_6_canonical_image
8610  ▷  make Edge_qurve_q17_canonical_image
8611
8612
8613  Hirschfeld_stab_export_gap:
8614  ▷  $(ORBITER) -v 9 \
8615  ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
8616  ▷  ▷  -define G -linear_group -PGGL 4 4 \
8617  ▷  ▷  ▷  -subgroup_by_generators "Hirschfeld_Stab" \
8618  ▷  ▷  ▷  51840 6 $(HIRSCHFELD_STAB_GENERATORS) \
8619  ▷  ▷  ▷  -end \
8620  ▷  ▷  -with G -do \
8621  ▷  ▷  -group_theoretic_activity \
8622  ▷  ▷  ▷  -export_gap \
8623  ▷  ▷  -end
8624
8625
8626
8627  Cyclic_6_canonical_image:
8628  ▷  $(ORBITER) -v 3 \
8629  ▷  ▷  -define G -permutation_group -cyclic_group 6 -end \
8630  ▷  ▷  -with G -do -group_theoretic_activity \
8631  ▷  ▷  ▷  -canonical_image_GAP 3,5 \
8632  ▷  ▷  -end
8633
8634
8635
8636  Edge_qurve_q17_canonical_image:
8637  ▷  $(ORBITER) -v 3 \
8638  ▷  ▷  -define G -linear_group -PGL 3 17 -end \
8639  ▷  ▷  -with G -do -group_theoretic_activity \
8640  ▷  ▷  ▷  -canonical_image_GAP $(EDGE_CURVE_Q17_AS_POINTS) \
8641  ▷  ▷  -end
8642
8643  #PGL_3_17_canonical_image.gap
8644
8645
8646
8647  ################################################################################
8648  # Section 6.8: Linear Groups, Advanced Topics
8649
```

```
8650
8651  SECTION_LINEAR_GROUPS_ADVANCED_TOPICS:
8652
8653  test_6_8:
8654  ▷  make U_3_3
8655  ▷  make PGL_2_3
8656  ▷  make Co3
8657  ▷  #make Ree_27
8658
8659
8660  U_3_3:
8661  ▷  $(ORBITER) -v 3 \
8662  ▷  ▷  -define F -finite_field -q 9 \
8663  ▷  ▷  ▷  -override_polynomial "17" \
8664  ▷  ▷  -end \
8665  ▷  ▷  -define G -linear_group -PGL 3 F \
8666  ▷  ▷  ▷  -subgroup_by_generators \
8667  ▷  ▷  ▷  "U_3_3" "6048" 2 \
8668  ▷  ▷  ▷  "1,6,4, 5,0,6, 8,5,1, \
8669  ▷  ▷  ▷  6,2,1, 7,8,4, 0,6,6" \
8670  ▷  ▷  ▷  -end \
8671  ▷  ▷  -with G -do \
8672  ▷  ▷  -group_theoretic_activity \
8673  ▷  ▷  ▷  -report \
8674  ▷  ▷  -end
8675  ▷  pdflatex PGL_3_9_Subgroup_U_3_3_6048_report.tex
8676  ▷  #$(OPEN) PGL_3_9_Subgroup_U_3_3_6048_report.pdf
8677
8678
8679
8680
8681
8682  PGL_2_3:
8683  ▷  $(ORBITER) -v 3 \
8684  ▷  ▷  -define G -linear_group -PGL 2 3 -end \
8685  ▷  ▷  -with G -do \
8686  ▷  ▷  -group_theoretic_activity \
8687  ▷  ▷  ▷  -report \
8688  ▷  ▷  ▷  -report_group_table \
8689  ▷  ▷  -end
8690  ▷  pdflatex PGL_2_3_report.tex
8691  ▷  $(OPEN) PGL_2_3_report.pdf
8692  ▷  #pdflatex PGL_2_3_group_table_order_24.tex
8693  ▷  #$(OPEN) PGL_2_3_group_table_order_24.pdf
8694  ▷  #$(OPEN) PGL_2_3_report.pdf
8695
8696
8697
8698
8699
8700
8701  #Co3 from Conway et al., 1985 (ATLAS)
8702  #order = 495766656000
8703  #Co3 from the paper by Suleiman and Wilson 1997
8704
8705
8706  Co3:
8707  ▷  $(ORBITER) -v 2 \
8708  ▷  ▷  -define F -finite_field -q 2 -end \
```

```
8709  ▷  ▷  -define g1 -vector -field F -format 22 -compact $(CONWAY_GEN1) -end \
8710  ▷  ▷  -define g2 -vector -field F -format 22 -compact $(CONWAY_GEN2) -end \
8711  ▷  ▷  -define gens -vector -concatenate g1,g2 -end \
8712  ▷  ▷  -define G -linear_group -PGL 22 2 \
8713  ▷  ▷  ▷  -subgroup_by_generators "Co3" "495766656000" 2 gens \
8714  ▷  ▷  ▷  -end \
8715  ▷  ▷  -with G -do \
8716  ▷  ▷  -group_theoretic_activity \
8717  ▷  ▷  ▷  -report \
8718  ▷  ▷  -end
8719  ▷  pdflatex PGL_22_2_Subgroup_Co3_495766656000_report.tex
8720  ▷  #$(OPEN) PGL_22_2_Subgroup_Co3_495766656000_report.pdf
8721
8722  # needs a lot of memory to run!
8723
8724  Ree_27:
8725  ▷  $(ORBITER) -v 2 \
8726  ▷  ▷  -define F -finite_field -q 27 \
8727  ▷  ▷  ▷  -override_polynomial "34" \
8728  ▷  ▷  -end \
8729  ▷  ▷  -define g1 -vector -field F -format 7 -dense $(Ree_gen1) -end \
8730  ▷  ▷  -define g2 -vector -field F -format 7 -dense $(Ree_gen2) -end \
8731  ▷  ▷  -define gens -vector -concatenate g1,g2 -end \
8732  ▷  ▷  -define G -linear_group -PGL 7 F \
8733  ▷  ▷  ▷  -subgroup_by_generators "Ree_27" "10073444472" 2 gens \
8734  ▷  ▷  ▷  -end \
8735  ▷  ▷  -with G -do \
8736  ▷  ▷  -group_theoretic_activity \
8737  ▷  ▷  ▷  -report \
8738  ▷  ▷  -end
8739
8740  # needs a lot of memory to run!
8741
8742
8743
8744
8745
8746  ################################################################################
8747  # Chapter 7 - Orbit Algorithms
8748  ################################################################################
8749
8750
8751  test_7:
8752  ▷  make test_7_1
8753  ▷  make test_7_2
8754  ▷  make test_7_3
8755  ▷  make test_7_4
8756  ▷  make test_7_5
8757  ▷  make test_7_6
8758  ▷  make test_7_7
8759
8760
8761  ################################################################################
8762  # Section 7.1: Orbit Algorithms
8763
8764
8765  SECTION_ORBIT_ALGORITHMS_SCHREIER_TREES:
8766
8767
```

```
8768  test_7_1:
8769  ▷  make orbits_PGL_4_2_on_points_draw_tree
8770  ▷  make orbits_PGL_4_2_on_points_export_trees
8771  ▷  make DD_PP4_orbit
8772  ▷  make T3r1_orbits
8773  ▷  #make T3r1_orbits_draw
8774  ▷  make 2C_orbit_under_PGGL_4_4_elements_coded.csv
8775  ▷  make orbits_on_conics_q13
8776  ▷  make orbits_cubic_curves_q2
8777  ▷  make orbits_cubic_curves_q2_with_draw_tree
8778  ▷  make poly_orbits_d3_n3_q2.csv
8779  ▷  make poly_orbits_d3_n3_q2_get_ranks
8780  ▷  make T4_orbits
8781  ▷  make T4r1_orbits
8782  ▷  make T4r1_orbits_draw
8783  ▷  make T4r1_orbits_4
8784  ▷  make PGGL_2_8_on_conic_orbits
8785  ▷  make PGGL_7_8_orbits
8786
8787
8788
8789  orbits_PGL_4_2_on_points_draw_tree:
8790  ▷  $(ORBITER) -v 4 \
8791  ▷  ▷  -draw_options -embedded -end \
8792  ▷  ▷  -define G -linear_group -PGL 4 2 -end \
8793  ▷  ▷  -define Orb -orbits -group G \
8794  ▷  ▷  ▷  -on_points \
8795  ▷  ▷  -end \
8796  ▷  ▷  -with Orb -do -orbits_activity \
8797  ▷  ▷  ▷  -report \
8798  ▷  ▷  -end \
8799  ▷  ▷  -with Orb -do -orbits_activity \
8800  ▷  ▷  ▷  -export_something "orbit" 0 \
8801  ▷  ▷  -end
8802  ▷  ▷  -with Orb -do -orbits_activity \
8803  ▷  ▷  ▷  -draw_tree 0 \
8804  ▷  ▷  -end
8805  ▷  pdflatex PGL_4_2_orbits_report.tex
8806  ▷  $(OPEN) PGL_4_2_orbits_report.pdf
8807
8808
8809  orbits_PGL_4_2_on_points_export_trees:
8810  ▷  $(ORBITER) -v 4 \
8811  ▷  ▷  -draw_options -embedded -end \
8812  ▷  ▷  -define G -linear_group -PGL 4 2 -end \
8813  ▷  ▷  -define Orb -orbits -group G \
8814  ▷  ▷  ▷  -on_points \
8815  ▷  ▷  -end \
8816  ▷  ▷  -with Orb -do -orbits_activity \
8817  ▷  ▷  ▷  -report \
8818  ▷  ▷  -end \
8819  ▷  ▷  -with Orb -do -orbits_activity \
8820  ▷  ▷  ▷  -export_trees \
8821  ▷  ▷  -end
8822  ▷  $(ORBITER) -v 3 \
8823  ▷  ▷  -draw_layered_graph \
8824  ▷  ▷  ▷  orbit_PGL_4_2_0.layered_graph \
8825  ▷  ▷  -radius 500 -spanning_tree -embedded \
8826  ▷  ▷  ▷  -line_width 1.1 -x_stretch 1.4 -scale 0.25 \
```

```
8827  ▷  ▷  -end
8828  ▷  pdflatex orbit_PGL_4_2_0_draw.tex
8829  ▷  $(OPEN) orbit_PGL_4_2_0_draw.pdf
8830
8831
8832  DD_PP4_orbit:
8833  ▷  $(ORBITER) -v 4 \
8834  ▷  ▷  -define G1 -linear_group -AGL 1 3 \
8835  ▷  ▷  ▷  -end \
8836  ▷  ▷  -define G2 -linear_group -AGL 1 7 \
8837  ▷  ▷  ▷  -end \
8838  ▷  ▷  -define G -modified_group -direct_product "G1,G2" \
8839  ▷  ▷  ▷  "21" "1,1,1,1" \
8840  ▷  ▷  -end \
8841  ▷  ▷  -with G -do \
8842  ▷  ▷  -group_theoretic_activity \
8843  ▷  ▷  ▷  -report \
8844  ▷  ▷  -end \
8845  ▷  ▷  -define S -vector -dense "0,1,3,13,20" -end \
8846  ▷  ▷  -define Orb -orbits -group G \
8847  ▷  ▷  ▷  -of_one_subset S \
8848  ▷  ▷  -end
8849
8850
8851  T3r1_orbits:
8852  ▷  $(ORBITER) -v 4 \
8853  ▷  ▷  -draw_options -embedded -end \
8854  ▷  ▷  -define G \
8855  ▷  ▷  -linear_group -GL_d_q_wr_Sym_n 2 2 3 \
8856  ▷  ▷  ▷  -on_rank_one_tensors -end \
8857  ▷  ▷  -define Orb -orbits -group G \
8858  ▷  ▷  ▷  -on_points \
8859  ▷  ▷  -end \
8860  ▷  ▷  -with Orb -do -orbits_activity \
8861  ▷  ▷  ▷  -report \
8862  ▷  ▷  -end \
8863  ▷  ▷  -with Orb -do -orbits_activity \
8864  ▷  ▷  ▷  -draw_tree 0 \
8865  ▷  ▷  -end
8866  ▷  pdflatex GL_2_2_wreath_Sym3_orbit_0_tree.tex
8867  ▷  $(OPEN) GL_2_2_wreath_Sym3_orbit_0_tree.pdf
8868
8869
8870  # ToDo: layered_graph file is not written
8871
8872  T3r1_orbits_draw:
8873  ▷  $(ORBITER) -v 3 \
8874  ▷  ▷  -draw_layered_graph \
8875  ▷  ▷  ▷  GL_2_2_wreath_Sym3_res27_0.layered_graph \
8876  ▷  ▷  -radius 500 -spanning_tree -embedded \
8877  ▷  ▷  ▷  -line_width 1.1 -x_stretch 1.4 -scale 0.25 \
8878  ▷  ▷  -end
8879  ▷  #pdflatex GL_2_2_wreath_Sym3_report.tex
8880  ▷  #$(OPEN) GL_2_2_wreath_Sym3_report.pdf
8881  ▷  pdflatex GL_2_2_wreath_Sym3_res27_0_draw.tex
8882  ▷  $(OPEN) GL_2_2_wreath_Sym3_res27_0_draw.pdf
8883
8884
8885  # write GL_2_2_wreath_Sym3_res27_0.layered_graph
```

```
8886
8887
8888
8889  2C_orbit_under_PGGL_4_4_elements_coded.csv:
8890  ▷  $(ORBITER) -v 6 \
8891  ▷  ▷  -define G -linear_group -PGGL 4 4 -end \
8892  ▷  ▷  -with G -do \
8893  ▷  ▷  -group_theoretic_activity \
8894  ▷  ▷  ▷  -conjugacy_class_of \
8895  ▷  ▷  ▷  "2C" "1,0,0,0, 1,1,0,0, 0,0,1,0, 0,0,1,1, 0" \
8896  ▷  ▷  -end
8897
8898  # class of size 64260
8899  # creates:
8900  # PGGL_4_4_class_of_2C.csv
8901  # 1:33 on Mac
8902  #User time: 2:59 on Mac
8903  # now 1:02 on new Mac
8904
8905
8906
8907
8908  orbits_on_conics_q13:
8909  ▷  $(ORBITER) -v 4 \
8910  ▷  ▷  -define F -finite_field -q 13 -end \
8911  ▷  ▷  -define G -linear_group -PGL 3 F -end \
8912  ▷  ▷  -define R -polynomial_ring \
8913  ▷  ▷  ▷  -field F \
8914  ▷  ▷  ▷  -number_of_variables 3 \
8915  ▷  ▷  ▷  -homogeneous_of_degree 2 \
8916  ▷  ▷  ▷  -monomial_ordering_partition \
8917  ▷  ▷  ▷  -variables "X,Y,Z" "X,Y,Z" \
8918  ▷  ▷  -end \
8919  ▷  ▷  -define Orb -orbits -group G \
8920  ▷  ▷  ▷  -on_polynomials R \
8921  ▷  ▷  -end
8922  ▷  #pdflatex poly_orbits_d2_n2_q13.tex
8923  ▷  #$(OPEN) poly_orbits_d2_n2_q13.pdf
8924
8925
8926  orbits_cubic_curves_q2:
8927  ▷  $(ORBITER) -v 4 \
8928  ▷  ▷  -define F -finite_field -q 2 -end \
8929  ▷  ▷  -define G -linear_group -PGL 3 F -end \
8930  ▷  ▷  -define R -polynomial_ring \
8931  ▷  ▷  ▷  -field F \
8932  ▷  ▷  ▷  -number_of_variables 3 \
8933  ▷  ▷  ▷  -homogeneous_of_degree 3 \
8934  ▷  ▷  ▷  -monomial_ordering_partition \
8935  ▷  ▷  ▷  -variables "X,Y,Z" "X,Y,Z" \
8936  ▷  ▷  -end \
8937  ▷  ▷  -define Orb -orbits -group G \
8938  ▷  ▷  ▷  -on_polynomials R \
8939  ▷  ▷  -end
8940  ▷  #pdflatex poly_orbits_d3_n3_q2.tex
8941  ▷  #$(OPEN) poly_orbits_d3_n3_q2.pdf
8942
8943
8944  orbits_cubic_curves_q2_with_draw_tree:
```

```
8945 ▷ $(ORBITER) -v 4 \
8946 ▷ ▷ -draw_options -yout 500000 -radius 150 -nodes_empty \
8947 ▷ ▷ ▷ -line_width 0.5 -y_stretch 0.25 -embedded -end \
8948 ▷ ▷ -define F -finite_field -q 2 -end \
8949 ▷ ▷ -define G -linear_group -PGL 3 F -end \
8950 ▷ ▷ -define R -polynomial_ring \
8951 ▷ ▷ ▷ -field F \
8952 ▷ ▷ ▷ -number_of_variables 3 \
8953 ▷ ▷ ▷ -homogeneous_of_degree 3 \
8954 ▷ ▷ ▷ -monomial_ordering_partition \
8955 ▷ ▷ ▷ -variables "X,Y,Z" "X,Y,Z" \
8956 ▷ ▷ -end \
8957 ▷ ▷ -define Orb -orbits -group G \
8958 ▷ ▷ ▷ -on_polynomials R \
8959 ▷ ▷ -end \
8960 ▷ ▷ -with Orb -do -orbits_activity \
8961 ▷ ▷ ▷ -draw_tree 6 \
8962 ▷ ▷ -end
8963 ▷ pdflatex PGL_3_2_orbit_6_tree.tex
8964 ▷ $(OPEN) PGL_3_2_orbit_6_tree.pdf
8965
8966
8967
8968
8969 poly_orbits_d3_n3_q2.csv:
8970 ▷ $(ORBITER) -v 4 \
8971 ▷ ▷ -draw_options -yout 500000 -radius 15 -nodes_empty \
8972 ▷ ▷ ▷ -line_width 0.5 -y_stretch 0.25 -embedded -end \
8973 ▷ ▷ -define F -finite_field -q 2 -end \
8974 ▷ ▷ -define G -linear_group -PGL 4 F -end \
8975 ▷ ▷ -define R -polynomial_ring \
8976 ▷ ▷ ▷ -field F \
8977 ▷ ▷ ▷ -number_of_variables 4 \
8978 ▷ ▷ ▷ -homogeneous_of_degree 3 \
8979 ▷ ▷ ▷ -monomial_ordering_partition \
8980 ▷ ▷ ▷ -variables "X,Y,Z,W" "X,Y,Z,W" \
8981 ▷ ▷ -end \
8982 ▷ ▷ -define Orb -orbits -group G \
8983 ▷ ▷ ▷ -on_polynomials R \
8984 ▷ ▷ -end \
8985 ▷ ▷ -with Orb -do -orbits_activity \
8986 ▷ ▷ ▷ -report \
8987 ▷ ▷ -end \
8988 ▷ ▷ -with Orb -do -orbits_activity \
8989 ▷ ▷ ▷ -draw_tree 6 \
8990 ▷ ▷ -end
8991 ▷ pdflatex PGL_4_2_orbit_6_tree.tex
8992 ▷ $(OPEN) PGL_4_2_orbit_6_tree.pdf
8993
8994 #written file PGL_4_2_orbit_6_tree.tex of size 79891
8995
8996
8997 poly_orbits_d3_n3_q2_get_ranks:
8998 ▷ $(ORBITER) -v 4 \
8999 ▷ ▷ -define cols -vector -dense 0 -end \
9000 ▷ ▷ -csv_file_select_cols poly_orbits_d3_n3_q2.csv "_reps" cols
9001 ▷ #pdflatex poly_orbits_d3_n3_q2.tex
9002 ▷ #$(OPEN) poly_orbits_d3_n3_q2.pdf
9003
```

```
9004
9005
9006  T4_orbits:
9007  ▷  $(ORBITER) -v 4 \
9008  ▷  ▷  -define G \
9009  ▷  ▷  -linear_group -GL_d_q_wr_Sym_n 2 2 4 \
9010  ▷  ▷  ▷  -on_tensors -end \
9011  ▷  ▷  -define Orb -orbits -group G \
9012  ▷  ▷  ▷  -on_points \
9013  ▷  ▷  -end
9014  ▷  #pdflatex GL_2_2_wreath_Sym4_res65535_orbits.tex
9015  ▷  #$(OPEN) GL_2_2_wreath_Sym4_res65535_orbits.pdf
9016  ▷  #pdflatex GL_2_2_wreath_Sym4_report.tex
9017  ▷  #$(OPEN) GL_2_2_wreath_Sym4_report.pdf
9018
9019
9020
9021
9022  T4r1_orbits:
9023  ▷  $(ORBITER) -v 4 \
9024  ▷  ▷  -define G -linear_group -GL_d_q_wr_Sym_n 2 2 4 \
9025  ▷  ▷  ▷  -on_rank_one_tensors -end \
9026  ▷  ▷  -define Orb -orbits -group G \
9027  ▷  ▷  ▷  -on_points \
9028  ▷  ▷  -end \
9029  ▷  ▷  -with Orb -do -orbits_activity \
9030  ▷  ▷  ▷  -export_trees \
9031  ▷  ▷  -end
9032
9033
9034  # problem in export_trees because of f_load_save = true in Algebra.orbits_on_poin
      ts
9035  # solution set f_load_save = false
9036
9037  #orbit_GL_2_2_wreath_Sym4_res_on_rank_one_tensors_0.layered_graph
9038
9039
9040  T4r1_orbits_draw:
9041  ▷  $(ORBITER) -v 3 \
9042  ▷  ▷  -draw_layered_graph \
9043  ▷  ▷  ▷  orbit_GL_2_2_wreath_Sym4_res_on_rank_one_tensors_0.layered_graph \
9044  ▷  ▷  -radius 400 -spanning_tree -embedded \
9045  ▷  ▷  ▷  -line_width 1.1 -x_stretch 2.5 -scale 0.15 \
9046  ▷  ▷  -end
9047  ▷  pdflatex orbit_GL_2_2_wreath_Sym4_res_on_rank_one_tensors_0_draw.tex
9048  ▷  $(OPEN) orbit_GL_2_2_wreath_Sym4_res_on_rank_one_tensors_0_draw.pdf
9049
9050
9051  T4r1_orbits_4:
9052  ▷  $(ORBITER) -v 4 \
9053  ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
9054  ▷  ▷  -define Control -poset_classification_control \
9055  ▷  ▷  ▷  -problem_label T4r1 -W \
9056  ▷  ▷  ▷  -draw_options -end \
9057  ▷  ▷  -end \
9058  ▷  ▷  -define G -linear_group -GL_d_q_wr_Sym_n 2 2 4 \
9059  ▷  ▷  ▷  -on_rank_one_tensors -end \
9060  ▷  ▷  -define Orb -orbits -group G \
9061  ▷  ▷  ▷  -on_subsets 4 Control \
```

```
9062  ▷  ▷  -end \
9063  ▷  ▷  -with Orb -do -orbits_activity \
9064  ▷  ▷  ▷  -report \
9065  ▷  ▷  ▷  -report_options \
9066  ▷  ▷  ▷  ▷  -draw_poset -type_ordinary \
9067  ▷  ▷  ▷  -end \
9068  ▷  ▷  -end
9069  ▷  pdflatex T4r1_poset.tex
9070  ▷  $(OPEN) T4r1_poset.pdf
9071
9072
9073
9074  # ToDo
9075
9076  PGGL_2_8_on_conic_orbits:
9077  ▷  $(ORBITER) -v 4 \
9078  ▷  ▷  -define G \
9079  ▷  ▷  -linear_group -PGGL 2 8 -PGL2OnConic -end \
9080  ▷  ▷  -define Orb -orbits -group G \
9081  ▷  ▷  ▷  -on_points \
9082  ▷  ▷  -end
9083
9084  #any_group::init_linear_group !LG->f_has_strong_generators
9085
9086
9087  # example from the Fining manual, page 107:
9088
9089  PGGL_7_8_orbits:
9090  ▷  $(ORBITER) -v 4 \
9091  ▷  ▷  -define G \
9092  ▷  ▷  -linear_group -PGGL 7 8 -end \
9093  ▷  ▷  -define Orb -orbits -group G \
9094  ▷  ▷  ▷  -on_points \
9095  ▷  ▷  -end
9096
9097  # 0:42
9098
9099
9100
9101
9102
9103
9104  ##############################################################################
9105  # Section 7.2: Poset Classification
9106
9107
9108  SECTION_POSET_CLASSIFICATION:
9109
9110  test_7_2:
9111  ▷  make poset_of_4subsets
9112  ▷  make poset_of_4subsets_draw
9113  ▷  make poset_of_5subsets
9114  ▷  make poset_of_5subsets_draw
9115  ▷  make Symmetric_4_on_pairs_poset
9116  ▷  make V_3_2_trivial
9117  ▷  make V_4_2_trivial
9118
9119
9120  poset_of_4subsets:
```

```
9121  ▷  $(ORBITER) -v 3 \
9122  ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
9123  ▷  ▷  -define Control -poset_classification_control \
9124  ▷  ▷  ▷  -problem_label poset_4 \
9125  ▷  ▷  ▷  -W -depth 4 \
9126  ▷  ▷  -end \
9127  ▷  ▷  -define G -permutation_group -identity_group 4 -end \
9128  ▷  ▷  -define Orb -orbits -group G \
9129  ▷  ▷  ▷  -on_subsets 4 Control \
9130  ▷  ▷  -end \
9131  ▷  ▷  -with Orb -do -orbits_activity \
9132  ▷  ▷  ▷  -report \
9133  ▷  ▷  ▷  -report_options \
9134  ▷  ▷  ▷  ▷  -draw_poset -type_ordinary \
9135  ▷  ▷  ▷  -end \
9136  ▷  ▷  -end
9137  ▷  pdflatex poset_4_poset.tex
9138  ▷  $(OPEN) poset_4_poset.pdf
9139
9140
9141
9142  poset_of_4subsets_draw:
9143  ▷  $(ORBITER) -v 3 \
9144  ▷  ▷  -draw_layered_graph \
9145  ▷  ▷  ▷  poset_4_poset_lvl_4.layered_graph \
9146  ▷  ▷  ▷  -radius 300 -embedded -line_width 1.1 \
9147  ▷  ▷  ▷  -y_stretch 0.9 -scale 0.25 \
9148  ▷  ▷  -end
9149  ▷  pdflatex poset_4_poset_lvl_4_draw.tex
9150  ▷  $(OPEN) poset_4_poset_lvl_4_draw.pdf
9151
9152
9153  poset_of_5subsets:
9154  ▷  $(ORBITER) -v 3 \
9155  ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
9156  ▷  ▷  -define Control -poset_classification_control \
9157  ▷  ▷  ▷  -problem_label poset_5 \
9158  ▷  ▷  ▷  -W -depth 5 \
9159  ▷  ▷  ▷  -draw_options -radius 150 -end \
9160  ▷  ▷  -end \
9161  ▷  ▷  -define G -permutation_group -identity_group 5 -end \
9162  ▷  ▷  -define Orb -orbits -group G \
9163  ▷  ▷  ▷  -on_subsets 5 Control \
9164  ▷  ▷  -end \
9165  ▷  ▷  -with Orb -do -orbits_activity \
9166  ▷  ▷  ▷  -report \
9167  ▷  ▷  ▷  -report_options \
9168  ▷  ▷  ▷  ▷  -draw_poset -type_ordinary \
9169  ▷  ▷  ▷  -end \
9170  ▷  ▷  -end
9171  ▷  pdflatex poset_5_poset.tex
9172  ▷  $(OPEN) poset_5_poset.pdf
9173
9174
9175  poset_of_5subsets_draw:
9176  ▷  $(ORBITER) -v 3 \
9177  ▷  ▷  -draw_layered_graph \
9178  ▷  ▷  ▷  poset_5_poset_lvl_5.layered_graph \
9179  ▷  ▷  ▷  -radius 300 -embedded \
```

```
9180  ▷  ▷  ▷  -line_width 1.1 -y_stretch 0.9 \
9181  ▷  ▷  ▷  -scale 0.25 \
9182  ▷  ▷  -end
9183  ▷  pdflatex poset_5_poset_lvl_5_draw.tex
9184  ▷  $(OPEN) poset_5_poset_lvl_5_draw.pdf
9185
9186
9187
9188  Symmetric_4_on_pairs_poset:
9189  ▷  $(ORBITER) -v 3 \
9190  ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
9191  ▷  ▷  -define Control -poset_classification_control \
9192  ▷  ▷  ▷  -problem_label Sym4_on2 \
9193  ▷  ▷  ▷  -W -depth 6 \
9194  ▷  ▷  ▷  -draw_options -radius 150 -end \
9195  ▷  ▷  -end \
9196  ▷  ▷  -define G -permutation_group -symmetric_group 4 -end \
9197  ▷  ▷  -define G_on_2subsets -modified_group -from G \
9198  ▷  ▷  ▷  -on_k_subsets 2 \
9199  ▷  ▷  -end \
9200  ▷  ▷  -define Orb -orbits -group G_on_2subsets \
9201  ▷  ▷  ▷  -on_subsets 6 Control \
9202  ▷  ▷  -end \
9203  ▷  ▷  -with Orb -do -orbits_activity \
9204  ▷  ▷  ▷  -report \
9205  ▷  ▷  ▷  -report_options \
9206  ▷  ▷  ▷  ▷  -draw_poset -type_ordinary \
9207  ▷  ▷  ▷  -end \
9208  ▷  ▷  -end
9209  ▷  pdflatex Sym4_on2_poset.tex
9210  ▷  $(OPEN) Sym4_on2_poset.pdf
9211
9212
9213  V_3_2_trivial:
9214  ▷  $(ORBITER) -v 5 \
9215  ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
9216  ▷  ▷  -define Control -poset_classification_control \
9217  ▷  ▷  ▷  -problem_label V_3_2_trivial \
9218  ▷  ▷  ▷  -W -depth 3 \
9219  ▷  ▷  ▷  -draw_options \
9220  ▷  ▷  ▷  ▷  -radius 200 -embedded \
9221  ▷  ▷  ▷  -end \
9222  ▷  ▷  -end \
9223  ▷  ▷  -define G -linear_group -PGL 3 2 -identity_group -end \
9224  ▷  ▷  -define Orb -orbits -group G \
9225  ▷  ▷  ▷  -on_subspaces 3 Control \
9226  ▷  ▷  -end \
9227  ▷  ▷  -with Orb -do -orbits_activity \
9228  ▷  ▷  ▷  -report \
9229  ▷  ▷  ▷  -report_options \
9230  ▷  ▷  ▷  ▷  -draw_poset -type_ordinary \
9231  ▷  ▷  ▷  -end \
9232  ▷  ▷  -end
9233  ▷  pdflatex PGL_3_2_Identity_3_2_poset.tex
9234  ▷  $(OPEN) PGL_3_2_Identity_3_2_poset.pdf
9235
9236
9237
9238  V_4_2_trivial:
```

```
9239  ▷  $(ORBITER) -v 5 \
9240  ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
9241  ▷  ▷  -define Control -poset_classification_control \
9242  ▷  ▷  ▷  -problem_label V_4_2_trivial \
9243  ▷  ▷  ▷  -W -depth 3 \
9244  ▷  ▷  ▷  -draw_options \
9245  ▷  ▷  ▷  ▷  -radius 200 -embedded \
9246  ▷  ▷  ▷  -end \
9247  ▷  ▷  -end \
9248  ▷  ▷  -define G -linear_group -PGL 4 2 -identity_group -end \
9249  ▷  ▷  -define Orb -orbits -group G \
9250  ▷  ▷  ▷  -on_subspaces 4 Control \
9251  ▷  ▷  -end \
9252  ▷  ▷  -with Orb -do -orbits_activity \
9253  ▷  ▷  ▷  -report \
9254  ▷  ▷  ▷  -report_options \
9255  ▷  ▷  ▷  ▷  -draw_poset -type_ordinary \
9256  ▷  ▷  ▷  -end \
9257  ▷  ▷  -end
9258
9259
9260
9261  ##############################################################################
9262  # Section 7.3: Orbits on Subsets
9263
9264
9265  SECTION_ORBITS_ON_SUBSETS:
9266
9267
9268  test_7_3:
9269  ▷  make PG_2_2_subsets
9270  ▷  make PG_3_2_subsets
9271  ▷  make PGL_3_2_singer
9272  ▷  make PGL_3_2_on_lines
9273  ▷  make PGL_2_5_on_subsets
9274  ▷  make PGL_2_7_on_subsets
9275  ▷  make PGGL_2_8_on_subsets
9276  ▷  make PGGL_2_9_on_subsets
9277  ▷  make PGL_2_11_on_subsets
9278  ▷  make PGGL_2_16_on_subsets
9279  ▷  make PGGL_2_32_on_subsets
9280  ▷  make PG_3_4_subsets
9281  ▷  make PGGL_2_9_orbits
9282  ▷  make PGO_5_2_on_subsets
9283
9284
9285
9286  PG_2_2_subsets:
9287  ▷  $(ORBITER) -v 3 \
9288  ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
9289  ▷  ▷  -define Control -poset_classification_control \
9290  ▷  ▷  ▷  -problem_label PGL_3_2 \
9291  ▷  ▷  ▷  -depth 7 \
9292  ▷  ▷  ▷  -draw_options \
9293  ▷  ▷  ▷  ▷  -radius 200 -embedded \
9294  ▷  ▷  ▷  -end \
9295  ▷  ▷  -end \
9296  ▷  ▷  -define F -finite_field -q 2 -end \
9297  ▷  ▷  -define G -linear_group -PGL 3 F -end \
```

```
9298  ▷  ▷  -define Orb -orbits -group G \
9299  ▷  ▷  ▷  -on_subsets 7 Control \
9300  ▷  ▷  -end \
9301  ▷  ▷  -with Orb -do -orbits_activity \
9302  ▷  ▷  ▷  -report \
9303  ▷  ▷  ▷  -report_options \
9304  ▷  ▷  ▷  ▷  -draw_poset -type_aux \
9305  ▷  ▷  ▷  -end \
9306  ▷  ▷  -end \
9307  ▷  ▷  -with G -do \
9308  ▷  ▷  -group_theoretic_activity \
9309  ▷  ▷  ▷  -stats "PG_2_2_subsets" \
9310  ▷  ▷  -end
9311  ▷  pdflatex PGL_3_2_poset_lvl_7.tex
9312  ▷  $(OPEN) PGL_3_2_poset_lvl_7.pdf
9313  ▷  pdflatex PGL_3_2_poset.tex
9314  ▷  $(OPEN) PGL_3_2_poset.pdf
9315
9316
9317  # PG(3,2) has 2^3+2^2+2^1+1 = 15 points:
9318  # PG(3,3) has 3^3+3^2+3^1+1 = 27 + 9 + 3 + 1 = 40 points.
9319
9320
9321
9322
9323
9324  PG_3_2_subsets:
9325  ▷  $(ORBITER) -v 3 \
9326  ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
9327  ▷  ▷  -define Control -poset_classification_control \
9328  ▷  ▷  ▷  -problem_label PGL_4_2 \
9329  ▷  ▷  ▷  -depth 15 \
9330  ▷  ▷  ▷  -draw_options \
9331  ▷  ▷  ▷  ▷  -radius 200 -embedded \
9332  ▷  ▷  ▷  -end \
9333  ▷  ▷  -end \
9334  ▷  ▷  -define F -finite_field -q 2 -end \
9335  ▷  ▷  -define G -linear_group -PGL 4 F -end \
9336  ▷  ▷  -define Orb -orbits -group G \
9337  ▷  ▷  ▷  -on_subsets 15 Control \
9338  ▷  ▷  -end \
9339  ▷  ▷  -with Orb -do -orbits_activity \
9340  ▷  ▷  ▷  -report \
9341  ▷  ▷  ▷  -report_options \
9342  ▷  ▷  ▷  ▷  -draw_poset -type_ordinary \
9343  ▷  ▷  ▷  -end \
9344  ▷  ▷  -end
9345  ▷  pdflatex PGL_4_2_poset.tex
9346  ▷  $(OPEN) PGL_4_2_poset.pdf
9347
9348  PGL_3_2_singer:
9349  ▷  $(ORBITER) -v 3 \
9350  ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
9351  ▷  ▷  -define Control -poset_classification_control \
9352  ▷  ▷  ▷  -problem_label PGL_3_2_singer_1 \
9353  ▷  ▷  ▷  -W -depth 7 \
9354  ▷  ▷  ▷  -draw_options \
9355  ▷  ▷  ▷  ▷  -radius 200 -embedded \
9356  ▷  ▷  ▷  -end \
```

```
9357  ▷  ▷  -end \
9358  ▷  ▷  -define G -linear_group -PGL 3 2 -singer 1 -end \
9359  ▷  ▷  -define Orb -orbits -group G \
9360  ▷  ▷  ▷  -on_subsets 7 Control \
9361  ▷  ▷  -end \
9362  ▷  ▷  -with Orb -do -orbits_activity \
9363  ▷  ▷  ▷  -report \
9364  ▷  ▷  ▷  -report_options \
9365  ▷  ▷  ▷  ▷  -draw_poset -type_ordinary \
9366  ▷  ▷  ▷  -end \
9367  ▷  ▷  -end
9368  ▷  pdflatex PGL_3_2_singer_1_poset.tex
9369  ▷  $(OPEN) PGL_3_2_singer_1_poset.pdf
9370
9371
9372
9373  PGL_3_2_on_lines:
9374  ▷  $(ORBITER) -v 3 \
9375  ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
9376  ▷  ▷  -define Control -poset_classification_control \
9377  ▷  ▷  ▷  -problem_label PGL_3_2_lines \
9378  ▷  ▷  ▷  -W -depth 7 \
9379  ▷  ▷  -end \
9380  ▷  ▷  -define G -linear_group -PGL 3 2 -end \
9381  ▷  ▷  -define G_on_lines -modified_group -from G \
9382  ▷  ▷  ▷  -on_k_subspaces 2 \
9383  ▷  ▷  -end \
9384  ▷  ▷  -define Orb -orbits -group G_on_lines \
9385  ▷  ▷  ▷  -on_subsets 7 Control \
9386  ▷  ▷  -end \
9387  ▷  ▷  -with Orb -do -orbits_activity \
9388  ▷  ▷  ▷  -report \
9389  ▷  ▷  ▷  -report_options \
9390  ▷  ▷  ▷  ▷  -draw_poset -type_ordinary \
9391  ▷  ▷  ▷  -end \
9392  ▷  ▷  -end
9393  ▷  pdflatex PGL_3_2_lines_poset.tex
9394  ▷  $(OPEN) PGL_3_2_lines_poset.pdf
9395
9396
9397
9398  PGL_2_5_on_subsets:
9399  ▷  $(ORBITER) -v 5 \
9400  ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
9401  ▷  ▷  -define Control -poset_classification_control \
9402  ▷  ▷  ▷  -problem_label PGL_2_5 \
9403  ▷  ▷  ▷  -W -depth 6 \
9404  ▷  ▷  -end \
9405  ▷  ▷  -define G -linear_group -PGL 2 5 -end \
9406  ▷  ▷  -define Orb -orbits -group G \
9407  ▷  ▷  ▷  -on_subsets 6 Control \
9408  ▷  ▷  -end \
9409  ▷  ▷  -with Orb -do -orbits_activity \
9410  ▷  ▷  ▷  -report \
9411  ▷  ▷  ▷  -report_options \
9412  ▷  ▷  ▷  ▷  -draw_poset -type_ordinary \
9413  ▷  ▷  ▷  -end \
9414  ▷  ▷  -end
9415  ▷  pdflatex PGL_2_5_poset.tex
```

```
9416  ▷  $(OPEN) PGL_2_5_poset.pdf
9417
9418  PGL_2_7_on_subsets:
9419  ▷  $(ORBITER) -v 10 \
9420  ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
9421  ▷  ▷  -define Control -poset_classification_control \
9422  ▷  ▷  ▷  -problem_label PGL_2_7 \
9423  ▷  ▷  ▷  -W -depth 8 \
9424  ▷  ▷  -end \
9425  ▷  ▷  -define G -linear_group -PGL 2 7 -end \
9426  ▷  ▷  -define Orb -orbits -group G \
9427  ▷  ▷  ▷  -on_subsets 8 Control \
9428  ▷  ▷  -end \
9429  ▷  ▷  -with Orb -do -orbits_activity \
9430  ▷  ▷  ▷  -report \
9431  ▷  ▷  ▷  -report_options \
9432  ▷  ▷  ▷  ▷  -draw_poset -type_ordinary \
9433  ▷  ▷  ▷  -end \
9434  ▷  ▷  -end
9435  ▷  pdflatex PGL_2_7_poset.tex
9436  ▷  $(OPEN) PGL_2_7_poset.pdf
9437
9438  PGGL_2_8_on_subsets:
9439  ▷  $(ORBITER) -v 10 \
9440  ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
9441  ▷  ▷  -define Control -poset_classification_control \
9442  ▷  ▷  ▷  -problem_label PGGL_2_8 \
9443  ▷  ▷  ▷  -W -depth 9 \
9444  ▷  ▷  -end \
9445  ▷  ▷  -define G -linear_group -PGGL 2 8 -end \
9446  ▷  ▷  -define Orb -orbits -group G \
9447  ▷  ▷  ▷  -on_subsets 9 Control \
9448  ▷  ▷  -end \
9449  ▷  ▷  -with Orb -do -orbits_activity \
9450  ▷  ▷  ▷  -report \
9451  ▷  ▷  ▷  -report_options \
9452  ▷  ▷  ▷  ▷  -draw_poset -type_ordinary \
9453  ▷  ▷  ▷  -end \
9454  ▷  ▷  -end
9455  ▷  pdflatex PGGL_2_8_poset.tex
9456  ▷  $(OPEN) PGGL_2_8_poset.pdf
9457
9458
9459  PGGL_2_9_on_subsets:
9460  ▷  $(ORBITER) -v 10 \
9461  ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
9462  ▷  ▷  -define Control -poset_classification_control \
9463  ▷  ▷  ▷  -problem_label PGGL_2_9 \
9464  ▷  ▷  ▷  -W -depth 10 \
9465  ▷  ▷  -end \
9466  ▷  ▷  -define G -linear_group -PGGL 2 9 -end \
9467  ▷  ▷  -define Orb -orbits -group G \
9468  ▷  ▷  ▷  -on_subsets 10 Control \
9469  ▷  ▷  -end \
9470  ▷  ▷  -with Orb -do -orbits_activity \
9471  ▷  ▷  ▷  -report \
9472  ▷  ▷  ▷  -report_options \
9473  ▷  ▷  ▷  ▷  -draw_poset -type_ordinary \
9474  ▷  ▷  ▷  -end \
```

```
9475  ▷  ▷  -end
9476  ▷  pdflatex PGGL_2_9_poset.tex
9477  ▷  $(OPEN) PGGL_2_9_poset.pdf
9478
9479
9480  PGL_2_11_on_subsets:
9481  ▷  $(ORBITER) -v 10 \
9482  ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
9483  ▷  ▷  -define Control -poset_classification_control \
9484  ▷  ▷  ▷  -problem_label PGL_2_11 \
9485  ▷  ▷  ▷  -W -depth 12 \
9486  ▷  ▷  -end \
9487  ▷  ▷  -define G -linear_group -PGL 2 11 -end \
9488  ▷  ▷  -define Orb -orbits -group G \
9489  ▷  ▷  ▷  -on_subsets 12 Control \
9490  ▷  ▷  -end \
9491  ▷  ▷  -with Orb -do -orbits_activity \
9492  ▷  ▷  ▷  -report \
9493  ▷  ▷  ▷  -report_options \
9494  ▷  ▷  ▷  ▷  -draw_poset -type_ordinary \
9495  ▷  ▷  ▷  -end \
9496  ▷  ▷  -end
9497  ▷  pdflatex PGL_2_11_poset.tex
9498  ▷  $(OPEN) PGL_2_11_poset.pdf
9499
9500
9501
9502  PGGL_2_16_on_subsets:
9503  ▷  $(ORBITER) -v 3 \
9504  ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
9505  ▷  ▷  -define Control -poset_classification_control \
9506  ▷  ▷  ▷  -problem_label PGGL_2_16 \
9507  ▷  ▷  ▷  -W -depth 10 \
9508  ▷  ▷  -end \
9509  ▷  ▷  -define G -linear_group -PGGL 2 16 -end \
9510  ▷  ▷  -define Orb -orbits -group G \
9511  ▷  ▷  ▷  -on_subsets 10 Control \
9512  ▷  ▷  -end \
9513  ▷  ▷  -with Orb -do -orbits_activity \
9514  ▷  ▷  ▷  -report \
9515  ▷  ▷  ▷  -report_options \
9516  ▷  ▷  ▷  ▷  -draw_poset -type_ordinary \
9517  ▷  ▷  ▷  -end \
9518  ▷  ▷  -end
9519  ▷  pdflatex PGGL_2_16_poset.tex
9520  ▷  $(OPEN) PGGL_2_16_poset.pdf
9521
9522
9523  PGGL_2_32_on_subsets:
9524  ▷  $(ORBITER) -v 3 \
9525  ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
9526  ▷  ▷  -define Control -poset_classification_control \
9527  ▷  ▷  ▷  -problem_label PGGL_2_32 \
9528  ▷  ▷  ▷  -W -depth 8 \
9529  ▷  ▷  -end \
9530  ▷  ▷  -define G -linear_group -PGGL 2 32 -end \
9531  ▷  ▷  -define Orb -orbits -group G \
9532  ▷  ▷  ▷  -on_subsets 8 Control \
9533  ▷  ▷  -end \
```

```
9534  ▷  ▷  -with Orb -do -orbits_activity \
9535  ▷  ▷  ▷  -report \
9536  ▷  ▷  ▷  -report_options \
9537  ▷  ▷  ▷  ▷  -draw_poset -type_ordinary \
9538  ▷  ▷  ▷  -end \
9539  ▷  ▷  -end
9540  ▷  pdflatex PGGL_2_32_poset.tex
9541  ▷  $(OPEN) PGGL_2_32_poset.pdf
9542
9543
9544  PG_3_4_subsets:
9545  ▷  $(ORBITER) -v 3 \
9546  ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
9547  ▷  ▷  -define Control -poset_classification_control \
9548  ▷  ▷  ▷  -problem_label PGGL_4_4 \
9549  ▷  ▷  ▷  -depth 5 \
9550  ▷  ▷  ▷  -draw_options \
9551  ▷  ▷  ▷  ▷  -radius 200 \
9552  ▷  ▷  ▷  -end \
9553  ▷  ▷  -end \
9554  ▷  ▷  -define G -linear_group -PGGL 4 4 -end \
9555  ▷  ▷  -define Orb -orbits -group G \
9556  ▷  ▷  ▷  -on_subsets 5 Control \
9557  ▷  ▷  -end \
9558  ▷  ▷  -with Orb -do -orbits_activity \
9559  ▷  ▷  ▷  -report \
9560  ▷  ▷  ▷  -report_options \
9561  ▷  ▷  ▷  ▷  -draw_poset -type_ordinary \
9562  ▷  ▷  ▷  -end \
9563  ▷  ▷  -end
9564  ▷  pdflatex PGGL_4_4_poset.tex
9565  ▷  $(OPEN) PGGL_4_4_poset.pdf
9566
9567
9568  PGGL_2_9_orbits:
9569  ▷  $(ORBITER) -v 3 \
9570  ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
9571  ▷  ▷  -define Control -poset_classification_control \
9572  ▷  ▷  ▷  -problem_label PGGL_2_9 -W -depth 5 \
9573  ▷  ▷  ▷  -draw_options -radius 200 -end \
9574  ▷  ▷  -end \
9575  ▷  ▷  -define G -linear_group -PGGL 2 9 -end \
9576  ▷  ▷  -define Orb -orbits -group G \
9577  ▷  ▷  ▷  -on_subsets 5 Control \
9578  ▷  ▷  -end \
9579  ▷  ▷  -with Orb -do -orbits_activity \
9580  ▷  ▷  ▷  -report \
9581  ▷  ▷  ▷  -report_options \
9582  ▷  ▷  ▷  ▷  -draw_poset -type_ordinary \
9583  ▷  ▷  ▷  -end \
9584  ▷  ▷  -end
9585  ▷  pdflatex PGGL_2_9_poset.tex
9586  ▷  $(OPEN) PGGL_2_9_poset.pdf
9587
9588
9589
9590
9591  PGO_5_2_on_subsets:
9592  ▷  $(ORBITER) -v 3 \
```

```
9593  ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
9594  ▷  ▷  -define Control -poset_classification_control \
9595  ▷  ▷  ▷  -problem_label PGO_5_2 \
9596  ▷  ▷  ▷  -depth 15 \
9597  ▷  ▷  ▷  -w \
9598  ▷  ▷  -end \
9599  ▷  ▷  -define F -finite_field -q 2 -end \
9600  ▷  ▷  -define G -linear_group -PGO 5 F -end \
9601  ▷  ▷  -define Orb -orbits -group G \
9602  ▷  ▷  ▷  -on_subsets 15 Control \
9603  ▷  ▷  -end \
9604  ▷  ▷  -with Orb -do -orbits_activity \
9605  ▷  ▷  ▷  -report \
9606  ▷  ▷  ▷  -report_options \
9607  ▷  ▷  ▷  ▷  -draw_poset -type_ordinary \
9608  ▷  ▷  ▷  -end \
9609  ▷  ▷  -end
9610  ▷  pdflatex PGO_5_2_poset.tex
9611  ▷  $(OPEN) PGO_5_2_poset.pdf
9612
9613
9614
9615
9616  ##############################################################################
9617  # Section 7.4: Orbits on Subspaces
9618
9619
9620  SECTION_ORBITS_ON_SUBSPACES:
9621
9622
9623  test_7_4:
9624  ▷  make PGL_5_3_wedge_subspace_orbits
9625  ▷  make subspaces_Op_4_2
9626  ▷  make PGL_4_2_on_subspaces
9627  ▷  make PGL_4_2_singer_on_subspaces
9628  ▷  make PGL_8_2_singer_on_subspaces
9629  ▷  make Op_6_2_orbits_on_subspaces
9630  ▷  make Op_6_3_orbits_on_subspaces
9631  ▷  make Op_6_11_orbits_on_subspaces
9632  ▷  make Op_8_2_orbits_on_subspaces
9633  ▷  make PGO_7_2_on_subspaces
9634
9635
9636
9637  PGL_5_3_wedge_subspace_orbits:
9638  ▷  $(ORBITER) -v 3 \
9639  ▷  ▷  -draw_options -radius 75 -embedded -nodes_empty -end \
9640  ▷  ▷  -define G -linear_group -PGL 5 3 -end \
9641  ▷  ▷  -define G2 -modified_group -from G \
9642  ▷  ▷  ▷  -on_wedge_product \
9643  ▷  ▷  -end \
9644  ▷  ▷  -define Control -poset_classification_control \
9645  ▷  ▷  ▷  -draw_options -radius 200 -embedded -end \
9646  ▷  ▷  ▷  -problem_label PGL_5_3_wedge -W -depth 5 \
9647  ▷  ▷  -end \
9648  ▷  ▷  -define Orb -orbits -group G2 \
9649  ▷  ▷  ▷  -on_subspaces 5 Control \
9650  ▷  ▷  -end \
9651  ▷  ▷  -with Orb -do -orbits_activity \
```

```
9652  ▷  ▷  ▷  -draw_tree 0 \
9653  ▷  ▷  -end
9654  ▷  pdflatex PGL_5_3_OnWedge_tree_lvl_5.tex
9655  ▷  $(OPEN) PGL_5_3_OnWedge_tree_lvl_5.pdf
9656
9657
9658
9659
9660  subspaces_Op_4_2:
9661  ▷  $(ORBITER) -v 5 \
9662  ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
9663  ▷  ▷  -define Control -poset_classification_control \
9664  ▷  ▷  ▷  -draw_options -radius 200 -end \
9665  ▷  ▷  ▷  -problem_label Op_4_2 -W -depth 4 \
9666  ▷  ▷  -end \
9667  ▷  ▷  -define G -linear_group -PGL 4 2 -orthogonal 1 -end \
9668  ▷  ▷  -define Orb -orbits -group G \
9669  ▷  ▷  ▷  -on_subspaces 4 Control \
9670  ▷  ▷  -end \
9671  ▷  ▷  -with Orb -do -orbits_activity \
9672  ▷  ▷  ▷  -report \
9673  ▷  ▷  ▷  -report_options \
9674  ▷  ▷  ▷  ▷  -draw_poset -type_ordinary \
9675  ▷  ▷  ▷  -end \
9676  ▷  ▷  -end
9677  ▷  pdflatex PGL_4_2_Orthogonal_plus_4_2_poset.tex
9678  ▷  $(OPEN) PGL_4_2_Orthogonal_plus_4_2_poset.pdf
9679
9680
9681
9682
9683
9684  PGL_4_2_on_subspaces:
9685  ▷  $(ORBITER) -v 5 \
9686  ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
9687  ▷  ▷  -define Control -poset_classification_control \
9688  ▷  ▷  ▷  -problem_label PGL_4_2 \
9689  ▷  ▷  ▷  -W -depth 4 \
9690  ▷  ▷  -end \
9691  ▷  ▷  -define G -linear_group -PGL 4 2 -end \
9692  ▷  ▷  -define Orb -orbits -group G \
9693  ▷  ▷  ▷  -on_subspaces 4 Control \
9694  ▷  ▷  -end \
9695  ▷  ▷  -with Orb -do -orbits_activity \
9696  ▷  ▷  ▷  -report \
9697  ▷  ▷  ▷  -report_options \
9698  ▷  ▷  ▷  ▷  -draw_poset -type_ordinary \
9699  ▷  ▷  ▷  -end \
9700  ▷  ▷  -end
9701  ▷  pdflatex PGL_4_2_poset.tex
9702  ▷  $(OPEN) PGL_4_2_poset.pdf
9703
9704
9705
9706
9707  PGL_4_2_singer_on_subspaces:
9708  ▷  $(ORBITER) -v 5 \
9709  ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
9710  ▷  ▷  -define Control -poset_classification_control \
```

```
9711  ▷  ▷  ▷  -draw_options -end \
9712  ▷  ▷  ▷  -problem_label PGL_4_2_singer \
9713  ▷  ▷  ▷  -W -depth 4 \
9714  ▷  ▷  -end \
9715  ▷  ▷  -define G -linear_group -PGL 4 2 -singer 1 -end \
9716  ▷  ▷  -define Orb -orbits -group G \
9717  ▷  ▷  ▷  -on_subspaces 4 Control \
9718  ▷  ▷  -end \
9719  ▷  ▷  -with Orb -do -orbits_activity \
9720  ▷  ▷  ▷  -report \
9721  ▷  ▷  ▷  -report_options \
9722  ▷  ▷  ▷  ▷  -draw_poset -type_ordinary \
9723  ▷  ▷  ▷  -end \
9724  ▷  ▷  -end
9725  ▷  pdflatex PGL_4_2_Singer_4_2_1_poset.tex
9726  ▷  $(OPEN) PGL_4_2_Singer_4_2_1_poset.pdf
9727
9728
9729
9730
9731  PGL_8_2_singer_on_subspaces:
9732  ▷  $(ORBITER) -v 5 \
9733  ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
9734  ▷  ▷  -define Control -poset_classification_control \
9735  ▷  ▷  ▷  -draw_options -radius 150 -end \
9736  ▷  ▷  ▷  -problem_label PGL_8_2_singer \
9737  ▷  ▷  ▷  -W -depth 8 \
9738  ▷  ▷  -end \
9739  ▷  ▷  -define G -linear_group -PGL 8 2 -singer 1 -end \
9740  ▷  ▷  -define Orb -orbits -group G \
9741  ▷  ▷  ▷  -on_subspaces 8 Control \
9742  ▷  ▷  -end \
9743  ▷  ▷  -with Orb -do -orbits_activity \
9744  ▷  ▷  ▷  -report \
9745  ▷  ▷  ▷  -report_options \
9746  ▷  ▷  ▷  ▷  -draw_poset -type_ordinary \
9747  ▷  ▷  ▷  -end \
9748  ▷  ▷  -end
9749
9750  # May 7, 2020: 16 sec on Mac
9751  # 1643 orbits in total
9752
9753  Op_6_2_orbits_on_subspaces:
9754  ▷  $(ORBITER) -v 5 \
9755  ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
9756  ▷  ▷  -define Control -poset_classification_control \
9757  ▷  ▷  ▷  -draw_options -radius 200 -end \
9758  ▷  ▷  ▷  -problem_label Op_6_2 -W \
9759  ▷  ▷  ▷  -depth 6 \
9760  ▷  ▷  -end \
9761  ▷  ▷  -define G -linear_group -PGL 6 2 -orthogonal 1 -end \
9762  ▷  ▷  -define Orb -orbits -group G \
9763  ▷  ▷  ▷  -on_subspaces 6 Control \
9764  ▷  ▷  -end
9765
9766
9767
9768  Op_6_3_orbits_on_subspaces:
9769  ▷  $(ORBITER) -v 5 \
```

```
9770  ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
9771  ▷  ▷  -define Control -poset_classification_control \
9772  ▷  ▷  ▷  -draw_options -radius 200 -end \
9773  ▷  ▷  ▷  -problem_label Op_6_3 -W \
9774  ▷  ▷  ▷  -depth 6 \
9775  ▷  ▷  -end \
9776  ▷  ▷  -define G -linear_group -PGL 6 3 -orthogonal 1 -end \
9777  ▷  ▷  -define Orb -orbits -group G \
9778  ▷  ▷  ▷  -on_subspaces 6 Control \
9779  ▷  ▷  -end
9780
9781  # June 3, 2020 on Mac: 0 sec
9782
9783
9784
9785  Op_6_11_orbits_on_subspaces:
9786  ▷  $(ORBITER) -v 5 \
9787  ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
9788  ▷  ▷  -define Control -poset_classification_control \
9789  ▷  ▷  ▷  -draw_options -radius 200 -end \
9790  ▷  ▷  ▷  -problem_label Op_6_11 -W \
9791  ▷  ▷  ▷  -depth 6 \
9792  ▷  ▷  -end \
9793  ▷  ▷  -define G -linear_group -PGL 6 11 -orthogonal 1 -end \
9794  ▷  ▷  -define Orb -orbits -group G \
9795  ▷  ▷  ▷  -on_subspaces 6 Control \
9796  ▷  ▷  -end
9797
9798
9799  # June 3, 2020 on Mac: 12 sec
9800
9801
9802  Op_8_2_orbits_on_subspaces:
9803  ▷  $(ORBITER) -v 5 \
9804  ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
9805  ▷  ▷  -define Control -poset_classification_control \
9806  ▷  ▷  ▷  -draw_options -radius 200 -end \
9807  ▷  ▷  ▷  -problem_label Op_8_2 -W -depth 8 \
9808  ▷  ▷  -end \
9809  ▷  ▷  -define G -linear_group -PGL 8 2 -orthogonal 1 -end \
9810  ▷  ▷  -define Orb -orbits -group G \
9811  ▷  ▷  ▷  -on_subspaces 8 Control \
9812  ▷  ▷  -end \
9813  ▷  ▷  -with Orb -do -orbits_activity \
9814  ▷  ▷  ▷  -report \
9815  ▷  ▷  ▷  -report_options \
9816  ▷  ▷  ▷  ▷  -draw_poset -type_ordinary \
9817  ▷  ▷  ▷  -end \
9818  ▷  ▷  -end
9819  ▷  pdflatex PGL_8_2_Orthogonal_plus_8_2_poset.tex
9820  ▷  $(OPEN) PGL_8_2_Orthogonal_plus_8_2_poset.pdf
9821
9822
9823
9824  PGO_7_2_on_subspaces:
9825  ▷  $(ORBITER) -v 6 \
9826  ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
9827  ▷  ▷  -define Control -poset_classification_control \
9828  ▷  ▷  ▷  -draw_options -radius 200 -end \
```

```
9829  ▷  ▷  ▷  -problem_label O_7_2 \
9830  ▷  ▷  ▷  -W -depth 7 \
9831  ▷  ▷  -end \
9832  ▷  ▷  -define F -finite_field -q 2 -end \
9833  ▷  ▷  -define G -linear_group -PGL 7 F -orthogonal 0 -end \
9834  ▷  ▷  -define Orb -orbits -group G \
9835  ▷  ▷  ▷  -on_subspaces 7 Control \
9836  ▷  ▷  -end
9837
9838
9839
9840  ###############################################################################
9841  # Section 7.5: Orbits on set partitions
9842
9843
9844  SECTION_ORBITS_ON_SET_PARTITIONS:
9845
9846
9847
9848  test_7_5:
9849  ▷  make C6_on_partition
9850  ▷  make PGL_2_17_on_partition
9851
9852
9853  C6_on_partition:
9854  ▷  $(ORBITER) -v 5 \
9855  ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
9856  ▷  ▷  -define Control -poset_classification_control \
9857  ▷  ▷  ▷  -problem_label C6 \
9858  ▷  ▷  ▷  -depth 2 \
9859  ▷  ▷  ▷  -W \
9860  ▷  ▷  ▷  -draw_options \
9861  ▷  ▷  ▷  ▷  -radius 200 -embedded \
9862  ▷  ▷  ▷  -end \
9863  ▷  ▷  -end \
9864  ▷  ▷  -define G -permutation_group -cyclic_group 6 -end \
9865  ▷  ▷  -define Orb -orbits -group G \
9866  ▷  ▷  ▷  -on_partition 2 Control \
9867  ▷  ▷  -end
9868
9869
9870  PGL_2_17_on_partition:
9871  ▷  $(ORBITER) -v 5 \
9872  ▷  ▷  -define Control -poset_classification_control \
9873  ▷  ▷  ▷  -problem_label PGL_2_17 \
9874  ▷  ▷  ▷  -depth 6 \
9875  ▷  ▷  ▷  -W \
9876  ▷  ▷  -end \
9877  ▷  ▷  -define G -linear_group -PGL 2 17 -end \
9878  ▷  ▷  -define Orb -orbits -group G \
9879  ▷  ▷  ▷  -on_partition 6 Control \
9880  ▷  ▷  -end
9881
9882
9883
9884  ###############################################################################
9885  # Section 7.6: Arcs and Caps in Projective Spaces
9886
9887
```

```
9888  SECTION_ARCS_AND_CAPS_IN_PROJECTIVE_SPACES:
9889
9890
9891
9892  test_7_6:
9893  ▷  make PGL_3_27
9894  ▷  make hyperoval_4_classify
9895  ▷  make hyperoval_8_classify
9896  ▷  make hyperoval_16_classify
9897  ▷  make hyperoval_16_1_conic_type
9898  ▷  make hyperoval_16_1_nonconical_type
9899  ▷  make hyperoval_16_2_nonconical_type
9900  ▷  #make hyperoval_16_stab_0_disjoint_sets_graph
9901  ▷  #make nc_arcs_16
9902  ▷  make five_arcs_q13
9903
9904
9905
9906
9907  PGL_3_27:
9908  ▷  $(ORBITER) -v 5 \
9909  ▷  ▷  -define G \
9910  ▷  ▷  -linear_group -PGL 3 27 -end \
9911  ▷  ▷  -with G -do \
9912  ▷  ▷  -group_theoretic_activity \
9913  ▷  ▷  ▷  -report \
9914  ▷  ▷  -end
9915  ▷  pdflatex PGL_3_27_report.tex
9916  ▷  $(OPEN) PGL_3_27_report.pdf
9917  ▷
9918
9919
9920
9921
9922  hyperoval_4_classify:
9923  ▷  $(ORBITER) -v 4 \
9924  ▷  ▷  -define F -finite_field -q 4 -end \
9925  ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
9926  ▷  ▷  -define Control -poset_classification_control \
9927  ▷  ▷  ▷  -problem_label hyperoval_q4 \
9928  ▷  ▷  ▷  -W -depth 6 \
9929  ▷  ▷  -end \
9930  ▷  ▷  -with P -do \
9931  ▷  ▷  -projective_space_activity \
9932  ▷  ▷  ▷  -classify_arcs \
9933  ▷  ▷  ▷  ▷  -control Control \
9934  ▷  ▷  ▷  ▷  -target_size 6 \
9935  ▷  ▷  ▷  ▷  -d 2 \
9936  ▷  ▷  ▷  -end \
9937  ▷  ▷  -end
9938  ▷  #pdflatex hyperoval_q4_poset.tex
9939  ▷  #$(OPEN) hyperoval_q4_poset.pdf
9940
9941
9942
9943
9944  hyperoval_8_classify:
9945  ▷  $(ORBITER) -v 4 \
9946  ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
```

```
9947  ▷  ▷  -define F -finite_field -q 8 -end \
9948  ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
9949  ▷  ▷  -define Control -poset_classification_control \
9950  ▷  ▷  ▷  -problem_label hyperoval_q8 \
9951  ▷  ▷  ▷  -W -depth 10 \
9952  ▷  ▷  ▷  -draw_options \
9953  ▷  ▷  ▷  ▷  -radius 200 \
9954  ▷  ▷  ▷  -end \
9955  ▷  ▷  -end \
9956  ▷  ▷  -with P -do \
9957  ▷  ▷  -projective_space_activity \
9958  ▷  ▷  ▷  -classify_arcs \
9959  ▷  ▷  ▷  ▷  -control Control \
9960  ▷  ▷  ▷  ▷  -target_size 10 \
9961  ▷  ▷  ▷  ▷  -d 2 \
9962  ▷  ▷  ▷  -end \
9963  ▷  ▷  -end
9964  ▷  #pdflatex hyperoval_q8_poset.tex
9965  ▷  #$(OPEN) hyperoval_q8_poset.pdf
9966
9967
9968
9969
9970
9971
9972  hyperoval_16_classify:
9973  ▷  $(ORBITER) -v 4 \
9974  ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
9975  ▷  ▷  -define F -finite_field -q 16 -end \
9976  ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
9977  ▷  ▷  -define Control -poset_classification_control \
9978  ▷  ▷  ▷  -problem_label hyperoval_q16 \
9979  ▷  ▷  ▷  -W -depth 18 \
9980  ▷  ▷  ▷  -draw_options \
9981  ▷  ▷  ▷  ▷  -radius 200 \
9982  ▷  ▷  ▷  -end \
9983  ▷  ▷  -end \
9984  ▷  ▷  -with P -do \
9985  ▷  ▷  -projective_space_activity \
9986  ▷  ▷  ▷  -classify_arcs \
9987  ▷  ▷  ▷  ▷  -control Control \
9988  ▷  ▷  ▷  ▷  -target_size 18 \
9989  ▷  ▷  ▷  ▷  -d 2 \
9990  ▷  ▷  ▷  -end \
9991  ▷  ▷  -end
9992
9993
9994  ▷  #pdflatex hyperoval_q16_poset.tex
9995  ▷  #$(OPEN) hyperoval_q16_poset.pdf
9996
9997
9998
9999
10000
10001 hyperoval_16_1_conic_type:
10002 ▷  $(ORBITER) -v 2 \
10003 ▷  ▷  -define F -finite_field -q 16 -end \
10004 ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
10005 ▷  ▷  -define H_16_1 -geometric_object P \
```

```
10006 ▷  ▷  ▷  -set "hyperoval_16_144" "hyperoval\_16\_144" \
10007 ▷  ▷  ▷  $(HYPEROVAL_16_144) \
10008 ▷  ▷  -end \
10009 ▷  ▷  -with H_16_1 -do \
10010 ▷  ▷  -combinatorial_object_activity \
10011 ▷  ▷  ▷  -save \
10012 ▷  ▷  -end \
10013 ▷  ▷  -with H_16_1 -do \
10014 ▷  ▷  -combinatorial_object_activity \
10015 ▷  ▷  ▷  -conic_type 6 \
10016 ▷  ▷  -end \
10017 ▷  ▷  -print_symbols
10018
10019 hyperoval_16_1_nonconical_type:
10020 ▷  $(ORBITER) -v 2 \
10021 ▷  ▷  -define F -finite_field -q 16 -end \
10022 ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
10023 ▷  ▷  -define H_16_1 -geometric_object P \
10024 ▷  ▷  ▷  -set "hyperoval_16_144" "hyperoval\_16\_144" \
10025 ▷  ▷  ▷  $(HYPEROVAL_16_144) \
10026 ▷  ▷  -end \
10027 ▷  ▷  -with H_16_1 -do \
10028 ▷  ▷  -combinatorial_object_activity \
10029 ▷  ▷  ▷  -save \
10030 ▷  ▷  -end \
10031 ▷  ▷  -with H_16_1 -do \
10032 ▷  ▷  -combinatorial_object_activity \
10033 ▷  ▷  ▷  -non_conical_type \
10034 ▷  ▷  -end \
10035 ▷  ▷  -print_symbols
10036
10037
10038 #We found 17028 non-conical 6 subsets
10039 #Eckardt point number distribution : $13^{252},\,$ $9^{720},\,$ $5^{2304},\,$ $3^
      {13752}$
10040
10041
10042 hyperoval_16_2_nonconical_type:
10043 ▷  $(ORBITER) -v 2 \
10044 ▷  ▷  -define F -finite_field -q 16 -end \
10045 ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
10046 ▷  ▷  -define H_16_2 -geometric_object P \
10047 ▷  ▷  ▷  -set "hyperoval_16_16320" "hyperoval\_16\_16320" \
10048 ▷  ▷  ▷  $(HYPEROVAL_16_16320) \
10049 ▷  ▷  -end \
10050 ▷  ▷  -with H_16_2 -do \
10051 ▷  ▷  -combinatorial_object_activity \
10052 ▷  ▷  ▷  -save \
10053 ▷  ▷  -end \
10054 ▷  ▷  -with H_16_2 -do \
10055 ▷  ▷  -combinatorial_object_activity \
10056 ▷  ▷  ▷  -non_conical_type \
10057 ▷  ▷  -end \
10058 ▷  ▷  -print_symbols
10059
10060 #We found 6188 = {17 choose 5} non-conical 6 subsets
10061 #Eckardt point number distribution : $45^{68},\,$ $13^{2040},\,$ $5^{4080}$
10062
10063 #neighbors_of_0_with_4_removed.csv
```

```
10064 #Row,C0,C1,C2,C3
10065 #0,2,3,9,10
10066 #1,1,3,7,8
10067 #2,10,12,13,15
10068 #3,1,5,10,11
10069 #4,3,5,6,13
10070 #5,8,9,11,12
10071 #6,7,11,13,17
10072 #7,7,10,14,16
10073 #8,1,9,13,16
10074 #9,2,8,13,14
10075 #10,1,2,15,17
10076 #11,6,8,10,17
10077 #12,6,7,9,15
10078 #13,2,6,11,16
10079 #14,5,9,14,17
10080 #15,5,8,15,16
10081 #16,1,6,12,14
10082 #17,2,5,7,12
10083 #18,3,12,16,17
10084 #19,3,11,14,15
10085 #END
10086
10087
10088 # ToDo neighbors_of_0_with_4_removed.csv is missing
10089
10090 hyperoval_16_stab_0_disjoint_sets_graph:
10091 ▷  $(ORBITER) -v 2 \
10092 ▷  ▷  -define G -graph -disjoint_sets_graph \
10093 ▷  ▷  ▷  neighbors_of_0_with_4_removed.csv \
10094 ▷  ▷  -end \
10095 ▷  ▷  -with G -do \
10096 ▷  ▷  ▷  -graph_theoretic_activity \
10097 ▷  ▷  ▷  -find_cliques \
10098 ▷  ▷  ▷  ▷  -target_size 4 \
10099 ▷  ▷  ▷  -end \
10100 ▷  ▷  -end \
10101 ▷  ▷  -print_symbols
10102
10103
10104 # 5 cliques of size 4
10105 #ROW,C0,C1,C2,C3
10106 #0,0,6,15,16
10107 #1,1,2,13,14
10108 #2,3,9,12,18
10109 #3,4,5,7,10
10110 #4,8,11,17,19
10111 #END
10112
10113 #clique 0:
10114 #0,2,3,9,10
10115 #6,7,11,13,17
10116 #15,5,8,15,16
10117 #16,1,6,12,14
10118 # partition: (1,6,12,14|2,3,9,10|5,8,15,16|7,11,13,17)
10119 # 4 is missing, it is the nucleus
10120 # 0 is missing is the chosen point
10121
10122
```

```
10123
10124
10125
10126
10127
10128 # nonconical 6-arcs are used for classifying cubic surfaces:
10129
10130
10131
10132
10133
10134 nc_arcs_16:
10135 ▷  $(ORBITER) -v 4 \
10136 ▷  ▷  -define F -finite_field -q 16 -end \
10137 ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
10138 ▷  ▷  -define Control -poset_classification_control \
10139 ▷  ▷  ▷  -problem_label nc_arcs_q16_d2 \
10140 ▷  ▷  ▷  -W -depth 6 \
10141 ▷  ▷  -end \
10142 ▷  ▷  -with P -do \
10143 ▷  ▷  -projective_space_activity \
10144 ▷  ▷  ▷  -classify_arcs \
10145 ▷  ▷  ▷  ▷  -control Control \
10146 ▷  ▷  ▷  ▷  -target_size 6 \
10147 ▷  ▷  ▷  ▷  -d 2 \
10148 ▷  ▷  ▷  ▷  -conic_test \
10149 ▷  ▷  ▷  -end \
10150 ▷  ▷  -end
10151 ▷  #pdflatex nc_arcs_q16_d2_poset.tex
10152 ▷  #$(OPEN) nc_arcs_q16_d2_poset.pdf
10153
10154
10155 #User time: 0:00
10156
10157
10158
10159 five_arcs_q13:
10160 ▷  $(ORBITER) -v 4 \
10161 ▷  ▷  -define F -finite_field -q 13 -end \
10162 ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
10163 ▷  ▷  -define Control -poset_classification_control \
10164 ▷  ▷  ▷  -problem_label five_arcs_q13 -W -depth 5 \
10165 ▷  ▷  -end \
10166 ▷  ▷  -with P -do \
10167 ▷  ▷  -projective_space_activity \
10168 ▷  ▷  ▷  -classify_arcs \
10169 ▷  ▷  ▷  ▷  -control Control \
10170 ▷  ▷  ▷  ▷  -target_size 5 \
10171 ▷  ▷  ▷  ▷  -d 2 \
10172 ▷  ▷  ▷  -end \
10173 ▷  ▷  -end
10174 ▷  #pdflatex five_arcs_q13_poset.tex
10175 ▷  #$(OPEN) five_arcs_q13_poset.pdf
10176
10177
10178
10179
10180
10181
```

```
10182
10183 ##############################################################################
10184 # Section 7.7: Cubic Curves
10185
10186 SECTION_CUBIC_CURVES:
10187
10188
10189
10190 test_7_7:
10191 ▷  make cubic_curves_PG_2_4
10192 ▷  make cubic_curves_PG_2_4_draw
10193 ▷  make poly_orbits_q4
10194 ▷  make cubic_curves_PG_2_9
10195 ▷  make cubic_curves_PG_2_9_draw
10196 ▷  make curve_orbit_mini
10197 ▷  make curve_orbit
10198
10199
10200
10201 cubic_curves_PG_2_4:
10202 ▷  $(ORBITER) -v 3 \
10203 ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
10204 ▷  ▷  -draw_options -radius 200 -embedded -end \
10205 ▷  ▷  -define F -finite_field -q 4 -end \
10206 ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
10207 ▷  ▷  -define Control -poset_classification_control \
10208 ▷  ▷  ▷  -problem_label cc_4 -W -depth 9 \
10209 ▷  ▷  ▷  -draw_options -radius 200 -embedded -end \
10210 ▷  ▷  -end \
10211 ▷  ▷  -define Arc_control -arc_generator_control \
10212 ▷  ▷  ▷  -projective_space P \
10213 ▷  ▷  ▷  -control Control \
10214 ▷  ▷  ▷  -target_size 9 \
10215 ▷  ▷  ▷  -d 3 \
10216 ▷  ▷  -end \
10217 ▷  ▷  -define Orb -orbits \
10218 ▷  ▷  ▷  -on_cubic_curves Arc_control \
10219 ▷  ▷  -end \
10220 ▷  ▷  -with Orb -do -orbits_activity \
10221 ▷  ▷  ▷  -report \
10222 ▷  ▷  -end
10223 ▷  ▷  -with Orb -do -orbits_activity \
10224 ▷  ▷  ▷  -draw_tree 0 \
10225 ▷  ▷  -end
10226 ▷  pdflatex cc_4_poset_lvl_9.tex
10227 ▷  $(OPEN) cc_4_poset_lvl_9.pdf
10228 ▷  pdflatex Cubic_curves_q4.tex
10229 ▷  $(OPEN) Cubic_curves_q4.pdf
10230
10231
10232 cubic_curves_PG_2_4_draw:
10233 ▷  $(ORBITER) -v 3 \
10234 ▷  ▷  -draw_layered_graph cc_4_poset_lvl_9.layered_graph \
10235 ▷  ▷  ▷  -radius 300 -embedded -line_width 1.1 \
10236 ▷  ▷  ▷  -y_stretch 0.9 -scale 0.25 \
10237 ▷  ▷  -end
10238 ▷  pdflatex cc_4_poset_lvl_9_draw.tex
10239 ▷  $(OPEN) cc_4_poset_lvl_9_draw.pdf
10240
```

```
10241
10242
10243 poly_orbits_q4:
10244 ▷  $(ORBITER) -v 4 \
10245 ▷  ▷  -draw_options -yout 500000 -radius 15 -nodes_empty \
10246 ▷  ▷  ▷  -line_width 0.5 -y_stretch 0.25 -embedded -end \
10247 ▷  ▷  -define F -finite_field -q 4 -print_numerically -end \
10248 ▷  ▷  -define R -polynomial_ring \
10249 ▷  ▷  ▷  -field F \
10250 ▷  ▷  ▷  -number_of_variables 3 \
10251 ▷  ▷  ▷  -homogeneous_of_degree 3 \
10252 ▷  ▷  ▷  -monomial_ordering_partition \
10253 ▷  ▷  ▷  -variables "X0,X1,X2" "X_0,X_1,X_2" \
10254 ▷  ▷  ▷  -end \
10255 ▷  ▷  -define G -linear_group -PGGL 3 F -end \
10256 ▷  ▷  -define Orb -orbits -group G \
10257 ▷  ▷  ▷  -on_polynomials R \
10258 ▷  ▷  -end \
10259 ▷  ▷  -with Orb -do -orbits_activity \
10260 ▷  ▷  ▷  -report \
10261 ▷  ▷  -end \
10262 ▷  ▷  -with Orb -do -orbits_activity \
10263 ▷  ▷  ▷  -export_something "orbit" 0 \
10264 ▷  ▷  -end
10265 ▷  pdflatex poly_orbits_d3_n2_q4.tex
10266 ▷  $(OPEN) poly_orbits_d3_n2_q4.pdf
10267
10268 #orbits_PGGL_3_4_orbit_0.csv
10269
10270
10271 # ToDo: error in cubic_curve::compute_inflexion_points
10272
10273 cubic_curves_PG_2_9:
10274 ▷  $(ORBITER) -v 3 \
10275 ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
10276 ▷  ▷  -draw_options -radius 200 -embedded -end \
10277 ▷  ▷  -define F -finite_field -q 9 -end \
10278 ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
10279 ▷  ▷  -define Control -poset_classification_control \
10280 ▷  ▷  ▷  -problem_label cc_9 -W -depth 9 \
10281 ▷  ▷  ▷  -draw_options -radius 200 -embedded -end \
10282 ▷  ▷  -end \
10283 ▷  ▷  -define Arc_control -arc_generator_control \
10284 ▷  ▷  ▷  -projective_space P \
10285 ▷  ▷  ▷  -control Control \
10286 ▷  ▷  ▷  -target_size 9 \
10287 ▷  ▷  ▷  -d 3 \
10288 ▷  ▷  -end \
10289 ▷  ▷  -define Orb -orbits \
10290 ▷  ▷  ▷  -on_cubic_curves Arc_control \
10291 ▷  ▷  -end \
10292 ▷  ▷  -with Orb -do -orbits_activity \
10293 ▷  ▷  ▷  -report \
10294 ▷  ▷  -end
10295 ▷  ▷  -with Orb -do -orbits_activity \
10296 ▷  ▷  ▷  -draw_tree 0 \
10297 ▷  ▷  -end
10298 ▷  #pdflatex cc_9_poset_lvl_9.tex
10299 ▷  #$(OPEN) cc_9_poset_lvl_9.pdf
```

```
10300
10301
10302 # too big to latex:
10303
10304 cubic_curves_PG_2_9_draw:
10305 ▷  $(ORBITER) -v 3 \
10306 ▷  ▷  -draw_layered_graph cc_9_poset_lvl_9.layered_graph \
10307 ▷  ▷  ▷  -radius 300 -embedded -line_width 1.1 \
10308 ▷  ▷  ▷  -y_stretch 0.9 -scale 0.25 \
10309 ▷  ▷  -end
10310 ▷  #pdflatex cc_9_poset_lvl_9_draw.tex
10311 ▷  #$(OPEN) cc_9_poset_lvl_9_draw.pdf
10312
10313
10314 curve_orbit_mini:
10315 ▷  $(ORBITER) -v 4 \
10316 ▷  ▷  -draw_options -yout 500000 -radius 15 -nodes_empty \
10317 ▷  ▷  ▷  -line_width 0.5 -y_stretch 0.25 -embedded -end \
10318 ▷  ▷  -define F -finite_field -q 3 -print_numerically -end \
10319 ▷  ▷  -define R -polynomial_ring \
10320 ▷  ▷  ▷  -field F \
10321 ▷  ▷  ▷  -number_of_variables 3 \
10322 ▷  ▷  ▷  -homogeneous_of_degree 3 \
10323 ▷  ▷  ▷  -monomial_ordering_partition \
10324 ▷  ▷  ▷  -variables "X0,X1,X2" "X_0,X_1,X_2" \
10325 ▷  ▷  ▷  -end \
10326 ▷  ▷  -define cubic -symbolic_object \
10327 ▷  ▷  ▷  -field F \
10328 ▷  ▷  ▷  -text "X0^3+X1^3+X2^3" \
10329 ▷  ▷  ▷  -end \
10330 ▷  ▷  -define G -linear_group -PGL 3 F -end \
10331 ▷  ▷  -define Orb -orbits -group G \
10332 ▷  ▷  ▷  -of_one_polynomial R cubic \
10333 ▷  ▷  -end \
10334 ▷  ▷  -with Orb -do -orbits_activity \
10335 ▷  ▷  ▷  -export_something "orbit" 0 \
10336 ▷  ▷  -end
10337
10338
10339 curve_orbit:
10340 ▷  $(ORBITER) -v 4 \
10341 ▷  ▷  -draw_options -yout 500000 -radius 15 -nodes_empty \
10342 ▷  ▷  ▷  -line_width 0.5 -y_stretch 0.25 -embedded -end \
10343 ▷  ▷  -define F -finite_field -q 9 -print_numerically -end \
10344 ▷  ▷  -define R -polynomial_ring \
10345 ▷  ▷  ▷  -field F \
10346 ▷  ▷  ▷  -number_of_variables 3 \
10347 ▷  ▷  ▷  -homogeneous_of_degree 3 \
10348 ▷  ▷  ▷  -monomial_ordering_partition \
10349 ▷  ▷  ▷  -variables "X0,X1,X2" "X_0,X_1,X_2" \
10350 ▷  ▷  ▷  -end \
10351 ▷  ▷  -define cubic -symbolic_object \
10352 ▷  ▷  ▷  -field F \
10353 ▷  ▷  ▷  -text $(SURFACE_F9_1_EQN_RESTRICTED) \
10354 ▷  ▷  ▷  -end \
10355 ▷  ▷  -define G -linear_group -PGL 3 F -end \
10356 ▷  ▷  -define Orb -orbits -group G \
10357 ▷  ▷  ▷  -of_one_polynomial R cubic \
10358 ▷  ▷  -end \
```

```
10359 ▷  ▷  -with Orb -do -orbits_activity \
10360 ▷  ▷  ▷  -export_something "orbit" 0 \
10361 ▷  ▷  -end
10362
10363
10364 #orbits_on_polynomials::orbit_of_one_polynomial found an orbit of length 589680
10365 #7:15
10366
10367
10368
10369
10370 ################################################################################
10371 # Chapter 8 - Cubic Surfaces
10372 ################################################################################
10373
10374
10375 test_8:
10376 ▷  make test_8_1
10377 ▷  make test_8_2
10378 ▷  make test_8_3
10379 ▷  make test_8_4
10380 ▷  make test_8_5
10381 ▷  make test_8_6
10382 ▷  make test_8_7
10383
10384
10385
10386
10387
10388 ################################################################################
10389 # Section 8.1: Global Commands for Cubic Surfaces
10390
10391
10392 SECTION_CUBIC_SURFACES_GLOBAL:
10393
10394
10395
10396 test_8_1:
10397 ▷  make surface_4_0_lines_vs_lines
10398 ▷  make surface_4_0_lines_vs_tritangent_planes
10399 ▷  make surface_4_0_dot_products_columns
10400 ▷  make surface_4_0_export_double_sixes
10401 ▷  make surface_4_0_multiply_half_double_six_vs_lines
10402 ▷  make surface_4_0_multiply_half_double_six_vs_incma
10403 ▷  make surface_4_0_dot_products_rows
10404 ▷  make surface_4_0_dot_products_rowspan
10405 ▷  make surface_4_0_single_six_times_lines_vs_lines_rowspan
10406
10407
10408 surface_4_0_lines_vs_lines:
10409 ▷  $(ORBITER) -v 3 \
10410 ▷  ▷  -define F -finite_field -q 4 -end \
10411 ▷  ▷  -define P -projective_space -n 3 -field F -end \
10412 ▷  ▷  -with P -do \
10413 ▷  ▷  -projective_space_activity \
10414 ▷  ▷  ▷  -export_cubic_surface_line_vs_line_incidence_matrix \
10415 ▷  ▷  -end
10416 ▷  $(ORBITER) -v 2 -draw_matrix \
10417 ▷  ▷  -input_csv_file PG_3_4_lines_vs_lines_incma.csv \
```

```
10418 ▷  ▷  -box_width 32 -bit_depth 24 \
10419 ▷  ▷  -partition 3 27 27 \
10420 ▷  ▷  -end
10421 ▷  convert PG_3_4_lines_vs_lines_incma_draw.bmp PG_3_4_lines_vs_lines_incma_draw.p
      ng
10422
10423
10424 # PG_3_4_lines_vs_lines_incma.csv
10425
10426
10427 surface_4_0_lines_vs_tritangent_planes:
10428 ▷  $(ORBITER) -v 3 \
10429 ▷  ▷  -define F -finite_field -q 4 -end \
10430 ▷  ▷  -define P -projective_space -n 3 -field F -end \
10431 ▷  ▷  -with P -do \
10432 ▷  ▷  -projective_space_activity \
10433 ▷  ▷  ▷  -export_cubic_surface_line_tritangent_plane_incidence_matrix \
10434 ▷  ▷  -end
10435 ▷  $(ORBITER) -v 2 -draw_matrix \
10436 ▷  ▷  -input_csv_file PG_3_4_lines_tritplanes_incma.csv \
10437 ▷  ▷  -box_width 32 -bit_depth 24 \
10438 ▷  ▷  -partition 3 27 45 \
10439 ▷  ▷  -end
10440 ▷  convert PG_3_4_lines_tritplanes_incma_draw.bmp PG_3_4_lines_tritplanes_incma_dr
      aw.png
10441
10442
10443
10444
10445 surface_4_0_dot_products_columns:
10446 ▷  $(ORBITER) -v 3 \
10447 ▷  ▷  -define line_vs_trit -vector \
10448 ▷  ▷  ▷  -file PG_3_4_lines_tritplanes_incma.csv \
10449 ▷  ▷  -end \
10450 ▷  ▷  -dot_product_of_columns line_vs_trit
10451 ▷  $(ORBITER) -v 2 -draw_matrix \
10452 ▷  ▷  -input_csv_file line_vs_trit_dot_products_columns.csv \
10453 ▷  ▷  -box_width 32 -bit_depth 24 \
10454 ▷  ▷  -partition 3 45 45 \
10455 ▷  ▷  -end
10456 ▷  convert line_vs_trit_dot_products_columns_draw.bmp line_vs_trit_dot_products_co
      lumns_draw.png
10457
10458
10459
10460 surface_4_0_export_double_sixes:
10461 ▷  $(ORBITER) -v 3 \
10462 ▷  ▷  -define F -finite_field -q 4 -end \
10463 ▷  ▷  -define P -projective_space -n 3 -field F -end \
10464 ▷  ▷  -with P -do \
10465 ▷  ▷  -projective_space_activity \
10466 ▷  ▷  ▷  -export_double_sixes \
10467 ▷  ▷  -end \
10468
10469
10470 surface_4_0_multiply_half_double_six_vs_lines:
10471 ▷  $(ORBITER) -v 3 \
10472 ▷  ▷  -define lines_vs_lines -vector \
10473 ▷  ▷  ▷  -file PG_3_4_lines_vs_lines_incma.csv \
```

```
10474 ▷  ▷  -end \
10475 ▷  ▷  -define single_six -vector \
10476 ▷  ▷  ▷  -file PG_3_4_single_sixes_char_vec.csv \
10477 ▷  ▷  -end \
10478 ▷  ▷  -matrix_multiply_over_Z single_six lines_vs_lines
10479
10480 # single_six_times_lines_vs_lines.csv
10481
10482 surface_4_0_multiply_half_double_six_vs_incma:
10483 ▷  $(ORBITER) -v 3 \
10484 ▷  ▷  -define line_vs_trit -vector \
10485 ▷  ▷  ▷  -file PG_3_4_lines_tritplanes_incma.csv \
10486 ▷  ▷  -end \
10487 ▷  ▷  -define single_six -vector \
10488 ▷  ▷  ▷  -file PG_3_4_single_sixes_char_vec.csv \
10489 ▷  ▷  -end \
10490 ▷  ▷  -matrix_multiply_over_Z single_six line_vs_trit
10491
10492 # 72 x 45:
10493 #single_six_times_line_vs_trit.csv
10494
10495 surface_4_0_dot_products_rows:
10496 ▷  $(ORBITER) -v 3 \
10497 ▷  ▷  -define single_six_vs_trit_planes -vector \
10498 ▷  ▷  ▷  -file single_six_times_line_vs_trit.csv \
10499 ▷  ▷  -end \
10500 ▷  ▷  -dot_product_of_rows single_six_vs_trit_planes
10501
10502 # 72 x 72:
10503 #single_six_vs_trit_planes_dot_products_rows.csv
10504
10505 surface_4_0_dot_products_rowspan:
10506 ▷  $(ORBITER) -v 3 \
10507 ▷  ▷  -define single_six_vs_trit_planes -vector \
10508 ▷  ▷  ▷  -file single_six_times_line_vs_trit.csv \
10509 ▷  ▷  -end \
10510 ▷  ▷  -rowspan_over_R single_six_vs_trit_planes
10511
10512 surface_4_0_single_six_times_lines_vs_lines_rowspan:
10513 ▷  $(ORBITER) -v 3 \
10514 ▷  ▷  -define M -vector \
10515 ▷  ▷  ▷  -file single_six_times_lines_vs_lines.csv \
10516 ▷  ▷  -end \
10517 ▷  ▷  -rowspan_over_R M
10518
10519
10520
10521
10522 ##############################################################################
10523 # Section 8.2: Cubic Surfaces Creation
10524
10525
10526 SECTION_CUBIC_SURFACES_CREATION:
10527
10528
10529
10530 test_8_2:
10531 ▷  make surface_4_0
10532 ▷  make Family_general_F7
```

```
10533 ▷  make surface_eckardt_13_4_12
10534 ▷  make Eckardt_13
10535 ▷  make surface_11_random
10536 ▷  make surface_F_abcd_Eckardt_q31_by_equation
10537 ▷  make surface_q13_Eckardt_by_arc_lifting
10538
10539
10540
10541 surface_4_0:
10542 ▷  $(ORBITER) -v 3 \
10543 ▷  ▷  -define F -finite_field -q 4 -end \
10544 ▷  ▷  -define P -projective_space -n 3 -field F -end \
10545 ▷  ▷  -define S -cubic_surface -space P -catalogue 0 -end \
10546
10547
10548 Family_general_F7:
10549 ▷  $(ORBITER) -v 3 \
10550 ▷  ▷  -define F -finite_field -q 7 -end \
10551 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
10552 ▷  ▷  -define S7_abcd_2_3_3_4 -cubic_surface \
10553 ▷  ▷  ▷  -space P -family_general_abcd 2 3 3 4 \
10554 ▷  ▷  -end
10555
10556
10557 surface_eckardt_13_4_12:
10558 ▷  $(ORBITER) -v 6 \
10559 ▷  ▷  -define F -finite_field -q 13 -end \
10560 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
10561 ▷  ▷  -define Eckardt_4_12 -cubic_surface \
10562 ▷  ▷  ▷  -space P -family_Eckardt 4 12 \
10563 ▷  ▷  -end
10564
10565
10566 Eckardt_13:
10567 ▷  $(ORBITER) -v 3 \
10568 ▷  ▷  -define F -finite_field -q 13 -end \
10569 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
10570 ▷  ▷  -define Eckardt_3_1 -cubic_surface \
10571 ▷  ▷  ▷  -space P -family_Eckardt 3 1 \
10572 ▷  ▷  -end
10573
10574
10575 surface_11_random:
10576 ▷  $(ORBITER) -v 10 \
10577 ▷  ▷  -define F -finite_field -q 11 -end \
10578 ▷  ▷  -define P -projective_space -n 3 -field F -v 6 -end \
10579 ▷  ▷  -define S -cubic_surface -space P -random -end
10580
10581
10582
10583 surface_F_abcd_Eckardt_q31_by_equation:
10584 ▷  $(ORBITER) -v 10 \
10585 ▷  ▷  -define F -finite_field -q 31 -end \
10586 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
10587 ▷  ▷  -define R -polynomial_ring \
10588 ▷  ▷  ▷  -field F \
10589 ▷  ▷  ▷  -number_of_variables 4 \
10590 ▷  ▷  ▷  -homogeneous_of_degree 3 \
10591 ▷  ▷  ▷  -monomial_ordering_partition \
```

```
10592 ▷  ▷  ▷  -variables "X0,X1,X2,X3" "X_0,X_1,X_2,X_3" \
10593 ▷  ▷  ▷  -end \
10594 ▷  ▷  -define F_abcd -cubic_surface -space P \
10595 ▷  ▷  ▷  -by_equation \
10596 ▷  ▷  ▷  R \
10597 ▷  ▷  ▷  "F_abcd" \
10598 ▷  ▷  ▷  "\DF_{a,b,c,d}\D" "X0,X1,X2,X3" \
10599 ▷  ▷  ▷  $(F_abcd_eqn) \
10600 ▷  ▷  ▷  "a,b,c,d" \
10601 ▷  ▷  ▷  "\D(a,b,c,d)=(2,30,30,2)\D" \
10602 ▷  ▷  ▷  "2,30,30,2" \
10603 ▷  ▷  -end
10604
10605
10606 surface_q13_Eckardt_by_arc_lifting:
10607 ▷  $(ORBITER) -v 3 \
10608 ▷  ▷  -define F -finite_field -q 13 -end \
10609 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
10610 ▷  ▷  -define S -cubic_surface -space P -arc_lifting "0,1,2,3,43,113" -end \
10611 ▷  ▷  -with S -do \
10612 ▷  ▷  -cubic_surface_activity \
10613 ▷  ▷  ▷  -report \
10614 ▷  ▷  -end
10615
10616
10617 ################################################################################
10618 # Section 8.3: Cubic Surface Activities
10619
10620
10621 SECTION_CUBIC_SURFACE_ACTIVITIES:
10622
10623
10624
10625 test_8_3:
10626 ▷  make surface_4_0_report
10627 ▷  make surface_4_0_export
10628 ▷  make Hirschfeld_surface_get_incidence_matrix_40_40
10629 ▷  make Hirschfeld_surface_incma_40_40_c
10630 ▷  make surface_4_0_plane_type_of_lines_on_Klein_quadric
10631 ▷  make surface_7_0
10632 ▷  #make surface_7_0_plane_type_of_lines_on_Klein_quadric # slow
10633 ▷  make Family_general_F7_report
10634 ▷  make surface_8_0
10635 ▷  #make surface_8_0_plane_type_of_lines_on_Klein_quadric # slow
10636 ▷  make surface_8_0_catalogue
10637 ▷  make surface_8_0_clean
10638 ▷  make surface_8_0b
10639 ▷  make surface_9_0
10640 ▷  #make surface_9_0_plane_type_of_lines_on_Klein_quadric # slow
10641 ▷  make surface_9_1
10642 ▷  #make surface_9_1_plane_type_of_lines_on_Klein_quadric # slow
10643 ▷  make surface_eckardt_13_4_12_report
10644 ▷  make Eckardt_13_report
10645 ▷  make surface_13_0
10646 ▷  make surface_16_0
10647 ▷  make G13_8
10648 ▷  make F13_8
10649 ▷  make F13_16
10650 ▷  make F13_32
```

```
10651 ▷  #make F13_64a
10652 ▷  #make F13_64b
10653 ▷  #make Colorado1
10654 ▷  #make Colorado2
10655 ▷  #make Colorado3
10656 ▷  #make F13_128a
10657 ▷  #make F13_128b
10658 ▷  #make F13_128c
10659 ▷  make move_two_lines
10660 ▷  make surface_F_abcd_Eckardt_q31
10661 ▷  make surface_F_abcd_Eckardt_q31_by_equation_report
10662 ▷  #make surface_F_abcd_sweep_4_27_q7
10663 ▷  make surface_11_random_report
10664 ▷  make surface_11_1
10665
10666
10667
10668
10669
10670 surface_4_0_report:
10671 ▷  $(ORBITER) -v 3 \
10672 ▷  ▷  -define F -finite_field -q 4 -end \
10673 ▷  ▷  -define P -projective_space -n 3 -field F -end \
10674 ▷  ▷  -define S -cubic_surface -space P -catalogue 0 -end \
10675 ▷  ▷  -with S -do \
10676 ▷  ▷  -cubic_surface_activity \
10677 ▷  ▷  ▷  -report \
10678 ▷  ▷  -end
10679 ▷  pdflatex surface_catalogue_q4_iso0_report.tex
10680 ▷  $(OPEN) surface_catalogue_q4_iso0_report.pdf
10681
10682
10683 surface_4_0_export:
10684 ▷  $(ORBITER) -v 3 \
10685 ▷  ▷  -define F -finite_field -q 4 -end \
10686 ▷  ▷  -define P -projective_space -n 3 -field F -end \
10687 ▷  ▷  -define S -cubic_surface -space P -catalogue 0 -end \
10688 ▷  ▷  -with S -do \
10689 ▷  ▷  -cubic_surface_activity \
10690 ▷  ▷  ▷  -export_something "points" \
10691 ▷  ▷  -end \
10692 ▷  ▷  -with S -do \
10693 ▷  ▷  -cubic_surface_activity \
10694 ▷  ▷  ▷  -export_something "points_off" \
10695 ▷  ▷  -end \
10696 ▷  ▷  -with S -do \
10697 ▷  ▷  -cubic_surface_activity \
10698 ▷  ▷  ▷  -export_something "lines" \
10699 ▷  ▷  -end \
10700 ▷  ▷  -with S -do \
10701 ▷  ▷  -cubic_surface_activity \
10702 ▷  ▷  ▷  -export_something "Hesse_planes" \
10703 ▷  ▷  -end
10704
10705
10706 Hirschfeld_surface_get_incidence_matrix_40_40:
10707 ▷  $(ORBITER) -v 3 \
10708 ▷  ▷  -define points -vector -dense $(HIRSCHFELD_SURFACE_POINTS_OFF) -end \
10709 ▷  ▷  -define planes -vector -dense $(HIRSCHFELD_SURFACE_HESSE_PLANES) -end \
```

```
10710 ▷  ▷  -define F -finite_field -q 4 -end \
10711 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
10712 ▷  ▷  -with P -do \
10713 ▷  ▷  -projective_space_activity \
10714 ▷  ▷  ▷  -restricted_incidence_matrix 1 3 points planes "H_incma_40_40" \
10715 ▷  ▷  -end
10716
10717
10718
10719 Hirschfeld_surface_incma_40_40_c:
10720 ▷  $(ORBITER) -v 5 \
10721 ▷  ▷  -draw_incidence_structure_description \
10722 ▷  ▷  ▷  -width 60 -width_10 6 -end \
10723 ▷  ▷  -define C -combinatorial_object \
10724 ▷  ▷  ▷  -label H_incma_40_40 H\_incma\_40\_40 \
10725 ▷  ▷  ▷  -file_of_incidence_geometries H_incma_40_40.inc 40 40 480 \
10726 ▷  ▷  -end \
10727 ▷  ▷  -with C -do \
10728 ▷  ▷  -combinatorial_object_activity \
10729 ▷  ▷  ▷  -canonical_form \
10730 ▷  ▷  ▷  ▷  -save_ago \
10731 ▷  ▷  ▷  ▷  -save_transversal \
10732 ▷  ▷  ▷  -end \
10733 ▷  ▷  -end \
10734 ▷  ▷  -with C -do \
10735 ▷  ▷  -combinatorial_object_activity \
10736 ▷  ▷  ▷  -report \
10737 ▷  ▷  ▷  ▷  -export_flag_orbits \
10738 ▷  ▷  ▷  ▷  -show_incidence_matrices \
10739 ▷  ▷  ▷  ▷  -export_group_GAP \
10740 ▷  ▷  ▷  -end \
10741 ▷  ▷  -end
10742 ▷  $(ORBITER) -v 2 -draw_matrix \
10743 ▷  ▷  -input_csv_file H_incma_40_40_object0_TDA_flag_orbits.csv \
10744 ▷  ▷  -secondary_input_csv_file H_incma_40_40_object0_TDA.csv \
10745 ▷  ▷  -box_width 32 -bit_depth 24 \
10746 ▷  ▷  -end
10747 ▷  pdflatex H_incma_40_40_classification.tex
10748 ▷  $(OPEN) H_incma_40_40_classification.pdf
10749
10750 surface_4_0_plane_type_of_lines_on_Klein_quadric:
10751 ▷  $(ORBITER) -v 6 \
10752 ▷  ▷  -define L -vector -file surface_catalogue_q4_iso0_lines.csv -end \
10753 ▷  ▷  -define F -finite_field -q 4 -end \
10754 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
10755 ▷  ▷  -with P -do \
10756 ▷  ▷  -projective_space_activity \
10757 ▷  ▷  ▷  -plane_intersection_type_of_klein_image 4 L \
10758 ▷  ▷  -end
10759
10760 #The plane intersection type is (0^244496, 1^112914, 2^15120, 3^4005, 5^270)
10761 #The plane intersection type is (5^270)
10762 #L_highest_weight_objects.csv
10763
10764
10765
10766
10767 surface_7_0:
10768 ▷  $(ORBITER) -v 3 \
```

```
10769 ▷  ▷  -define F -finite_field -q 7 -end \
10770 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
10771 ▷  ▷  -define S -cubic_surface -space P -catalogue 0 -end \
10772 ▷  ▷  -with S -do \
10773 ▷  ▷  -cubic_surface_activity \
10774 ▷  ▷  ▷  -report \
10775 ▷  ▷  -end \
10776 ▷  ▷  -with S -do \
10777 ▷  ▷  -cubic_surface_activity \
10778 ▷  ▷  ▷  -export_something "lines" \
10779 ▷  ▷  -end
10780 ▷  pdflatex surface_catalogue_q7_iso0_report.tex
10781 ▷  $(OPEN) surface_catalogue_q7_iso0_report.pdf
10782
10783
10784
10785 surface_7_0_plane_type_of_lines_on_Klein_quadric:
10786 ▷  $(ORBITER) -v 6 \
10787 ▷  ▷  -define L_7_0 -vector \
10788 ▷  ▷  ▷  -file surface_catalogue_q7_iso0_lines.csv -end \
10789 ▷  ▷  -define F -finite_field -q 7 -end \
10790 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
10791 ▷  ▷  -with P -do \
10792 ▷  ▷  -projective_space_activity \
10793 ▷  ▷  ▷  -plane_intersection_type_of_klein_image 4 L_7_0 \
10794 ▷  ▷  -end
10795
10796 # Time: 4:50
10797 #The plane intersection type is (0^44526575, 1^3529170, 2^112563, 3^8541, 4^324,
      5^27)
10798 #The plane intersection type is (4^324, 5^27)
10799 #L_7_0_highest_weight_objects.csv
10800
10801
10802 Family_general_F7_report:
10803 ▷  $(ORBITER) -v 3 \
10804 ▷  ▷  -define F -finite_field -q 7 -end \
10805 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
10806 ▷  ▷  -define S7_abcd_2_3_3_4 -cubic_surface \
10807 ▷  ▷  ▷  -space P -family_general_abcd 2 3 3 4 \
10808 ▷  ▷  -end \
10809 ▷  ▷  -with S7_abcd_2_3_3_4 -do \
10810 ▷  ▷  -cubic_surface_activity \
10811 ▷  ▷  ▷  -report \
10812 ▷  ▷  -end
10813 ▷  pdflatex surface_family_general_abcd_q7_a2_b3_c3_d4_report.tex
10814 ▷  $(OPEN) surface_family_general_abcd_q7_a2_b3_c3_d4_report.pdf
10815
10816
10817 # Fermat with 18 Eckardt points
10818 # no automorphism group
10819
10820
10821 surface_8_0:
10822 ▷  $(ORBITER) -v 3 \
10823 ▷  ▷  -define F -finite_field -q 8 -end \
10824 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
10825 ▷  ▷  -define S -cubic_surface -space P -catalogue 0 -end \
10826 ▷  ▷  -with S -do \
```

```
10827 ▷  ▷  -cubic_surface_activity \
10828 ▷  ▷  ▷  -report \
10829 ▷  ▷  ▷  -all_quartic_curves \
10830 ▷  ▷  -end \
10831 ▷  ▷  -with S -do \
10832 ▷  ▷  -cubic_surface_activity \
10833 ▷  ▷  ▷  -export_something "lines" \
10834 ▷  ▷  -end \
10835
10836 surface_8_0_plane_type_of_lines_on_Klein_quadric:
10837 ▷  $(ORBITER) -v 6 \
10838 ▷  ▷  -define L_8_0 -vector -file surface_catalogue_q8_iso0_lines.csv -end \
10839 ▷  ▷  -define F -finite_field -q 8 -end \
10840 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
10841 ▷  ▷  -with P -do \
10842 ▷  ▷  -projective_space_activity \
10843 ▷  ▷  ▷  -plane_intersection_type_of_klein_image 4 L_8_0 \
10844 ▷  ▷  -end
10845
10846 #L_8_0_highest_weight_objects.csv
10847 #intersection numbers: ( 0, 0, 0, 0, 192, 30 )
10848 # time: 15:48
10849
10850
10851
10852
10853
10854
10855 surface_8_0_catalogue:
10856 ▷  $(ORBITER) -v 3 \
10857 ▷  ▷  -define F -finite_field -q 8 -end \
10858 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
10859 ▷  ▷  -define S8_0 -cubic_surface -space P -catalogue 0 -end \
10860 ▷  ▷  -with S8_0 -do \
10861 ▷  ▷  -cubic_surface_activity \
10862 ▷  ▷  ▷  -report \
10863 ▷  ▷  -end
10864 ▷  pdflatex surface_catalogue_q8_iso0_report.tex
10865 ▷  $(OPEN) surface_catalogue_q8_iso0_report.pdf
10866
10867
10868
10869 surface_8_0_clean:
10870 ▷  $(ORBITER) -v 3 \
10871 ▷  ▷  -define F -finite_field -q 8 -end \
10872 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
10873 ▷  ▷  -define S8_0 -cubic_surface -space P -catalogue 0 \
10874 ▷  ▷  ▷  -select_double_six "15,11,22,19,24,5,16,10,23,20,25,4" \
10875 ▷  ▷  ▷  -select_double_six "3,2,1,0,5,4,9,8,7,6,11,10" \
10876 ▷  ▷  ▷  -transform_inverse "1,4,4,0,6,0,0,0,6,2,0,1,7,0,4,0,0" \
10877 ▷  ▷  ▷  -transform "4,4,0,0, 0,0,1,0, 1,0,0,0, 0,0,0,1, 0" \
10878 ▷  ▷  ▷  -transform_inverse "2,0,0,0,0,2,0,0,0,0,2,0,1,1,2,3,0" \
10879 ▷  ▷  ▷  -end \
10880 ▷  ▷  -with S8_0 -do \
10881 ▷  ▷  -cubic_surface_activity \
10882 ▷  ▷  ▷  -report \
10883 ▷  ▷  -end
10884 ▷  pdflatex surface_catalogue_q8_iso0_report.tex
10885 ▷  $(OPEN) surface_catalogue_q8_iso0_report.pdf
```

```
10886
10887
10888
10889 # clean equation for Tekirdag-1:
10890
10891 surface_8_0b:
10892 ▷  $(ORBITER) -v 3 \
10893 ▷  ▷  -define F -finite_field -q 8 -end \
10894 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
10895 ▷  ▷  -define S8_0 -cubic_surface -space P -catalogue 0 \
10896 ▷  ▷  ▷  -select_double_six "15,11,22,19,24,5,16,10,23,20,25,4" \
10897 ▷  ▷  ▷  -select_double_six "3,2,1,0,5,4,9,8,7,6,11,10" \
10898 ▷  ▷  ▷  -transform "1,0,0,0,0,1,0,6,0,0,1,6,0,0,0,1,0" \
10899 ▷  ▷  ▷  -transform_inverse "3,1,1,0,0,1,0,0,0,0,1,0,0,0,0,1,0" \
10900 ▷  ▷  ▷  -transform_inverse "2,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0" \
10901 ▷  ▷  -end \
10902 ▷  ▷  -with S8_0 -do \
10903 ▷  ▷  -cubic_surface_activity \
10904 ▷  ▷  ▷  -report \
10905 ▷  ▷  -end
10906
10907
10908
10909
10910
10911
10912 surface_9_0:
10913 ▷  $(ORBITER) -v 3 \
10914 ▷  ▷  -define F -finite_field -q 9 -end \
10915 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
10916 ▷  ▷  -define S -cubic_surface -space P -catalogue 0 -end \
10917 ▷  ▷  -with S -do \
10918 ▷  ▷  -cubic_surface_activity \
10919 ▷  ▷  ▷  -report \
10920 ▷  ▷  ▷  -all_quartic_curves \
10921 ▷  ▷  -end \
10922 ▷  ▷  -with S -do \
10923 ▷  ▷  -cubic_surface_activity \
10924 ▷  ▷  ▷  -export_something "lines" \
10925 ▷  ▷  -end
10926
10927 surface_9_0_plane_type_of_lines_on_Klein_quadric:
10928 ▷  $(ORBITER) -v 6 \
10929 ▷  ▷  -define L_9_0 -vector -file surface_catalogue_q9_iso0_lines.csv -end \
10930 ▷  ▷  -define F -finite_field -q 9 -end \
10931 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
10932 ▷  ▷  -with P -do \
10933 ▷  ▷  -projective_space_activity \
10934 ▷  ▷  ▷  -plane_intersection_type_of_klein_image 4 L_9_0 \
10935 ▷  ▷  -end
10936
10937
10938 #intersection numbers: ( 0, 0, 0, 0, 180, 15 )
10939 #time: 44:42
10940
10941
10942 surface_9_1:
10943 ▷  $(ORBITER) -v 3 \
10944 ▷  ▷  -define F -finite_field -q 9 -end \
```

```
10945 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
10946 ▷  ▷  -define S -cubic_surface -space P -catalogue 1 -end \
10947 ▷  ▷  -with S -do \
10948 ▷  ▷  -cubic_surface_activity \
10949 ▷  ▷  ▷  -report \
10950 ▷  ▷  ▷  -all_quartic_curves \
10951 ▷  ▷  -end \
10952 ▷  ▷  -with S -do \
10953 ▷  ▷  -cubic_surface_activity \
10954 ▷  ▷  ▷  -export_something "lines" \
10955 ▷  ▷  -end
10956 ▷  pdflatex surface_catalogue_q9_iso1_report.tex
10957 ▷  $(OPEN) surface_catalogue_q9_iso1_report.pdf
10958
10959
10960 surface_9_1_plane_type_of_lines_on_Klein_quadric:
10961 ▷  $(ORBITER) -v 6 \
10962 ▷  ▷  -define L_9_1 -vector -file surface_catalogue_q9_iso1_lines.csv -end \
10963 ▷  ▷  -define F -finite_field -q 9 -end \
10964 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
10965 ▷  ▷  -with P -do \
10966 ▷  ▷  -projective_space_activity \
10967 ▷  ▷  ▷  -plane_intersection_type_of_klein_image 4 L_9_1 \
10968 ▷  ▷  -end
10969
10970 #intersection numbers: ( 0, 0, 0, 0, 216 )
10971
10972 #44:36
10973
10974
10975
10976 surface_eckardt_13_4_12_report:
10977 ▷  $(ORBITER) -v 6 \
10978 ▷  ▷  -define F -finite_field -q 13 -end \
10979 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
10980 ▷  ▷  -define Eckardt_4_12 -cubic_surface \
10981 ▷  ▷  ▷  -space P -family_Eckardt 4 12 \
10982 ▷  ▷  -end \
10983 ▷  ▷  -with Eckardt_4_12 -do \
10984 ▷  ▷  -cubic_surface_activity \
10985 ▷  ▷  ▷  -report \
10986 ▷  ▷  -end
10987 ▷  pdflatex surface_family_Eckardt_q13_a4_b12_report.tex
10988 ▷  $(OPEN) surface_family_Eckardt_q13_a4_b12_report.pdf
10989
10990
10991
10992
10993
10994
10995 # 13_0 has 4 Eckardt points
10996 # 13_1 has 6 Eckardt points
10997 # 13_2 has 9 Eckardt points
10998 # 13_3 has 18 Eckardt points
10999 #
11000
11001
11002
11003
```

```
11004 Eckardt_13_report:
11005 ▷  $(ORBITER) -v 3 \
11006 ▷  ▷  -define F -finite_field -q 13 -end \
11007 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
11008 ▷  ▷  -define Eckardt_3_1 -cubic_surface \
11009 ▷  ▷  ▷  -space P -family_Eckardt 3 1 \
11010 ▷  ▷  -end \
11011 ▷  ▷  -with Eckardt_3_1 -do \
11012 ▷  ▷  -cubic_surface_activity \
11013 ▷  ▷  ▷  -report \
11014 ▷  ▷  -end
11015
11016
11017
11018 surface_13_0:
11019 ▷  $(ORBITER) -v 3 \
11020 ▷  ▷  -define F -finite_field -q 13 -end \
11021 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
11022 ▷  ▷  -define S13_0 -cubic_surface -space P -catalogue 0 -end \
11023 ▷  ▷  -with S13_0 -do \
11024 ▷  ▷  -cubic_surface_activity \
11025 ▷  ▷  ▷  -report \
11026 ▷  ▷  -end
11027
11028
11029 # clean equation for Tekirdag-2:
11030
11031
11032 surface_16_0:
11033 ▷  $(ORBITER) -v 3 \
11034 ▷  ▷  -define F -finite_field -q 16 -end \
11035 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
11036 ▷  ▷  -define S16_0 -cubic_surface -space P -catalogue 0 \
11037 ▷  ▷  ▷  -transform "1,0,0,0,0,1,0,12,0,0,1,12,0,0,0,1,0" \
11038 ▷  ▷  ▷  -transform "15,11,4,0,0,0,12,0,0,12,0,0,0,0,0,1,3" \
11039 ▷  ▷  -end \
11040 ▷  ▷  -with S16_0 -do \
11041 ▷  ▷  -cubic_surface_activity \
11042 ▷  ▷  ▷  -report \
11043 ▷  ▷  -end
11044
11045
11046 #▷ ▷  -transform_inverse "3,0,0,0,0,1,1,0,0,0,1,0,0,0,0,1,0" \
11047 #▷ ▷  -transform_inverse "13,12,1,0,12,13,1,0,0,0,1,0,0,0,0,1,0" \
11048 #▷ ▷  -transform_inverse "1,0,0,0,0,1,0,0,12,12,1,0,0,0,0,1,0" \
11049 #▷ ▷  -transform_inverse "12,0,0,0,0,12,0,0,0,0,1,0,0,0,0,1,0"
11050
11051 # rank of lines ( 66591, 26737, 4093, 69904, 28376, 26470, 70160, 69855, 26208, 5
      847, 369, 32230, 529, 30293, 70068, 2178, 261, 28666, 8575, 105, 31694, 0, 51784,
       25209, 22193, 49862, 274 )
11052 # Rank of points on Klein quadric: ( 29181, 4677, 29950, 33, 62496, 429, 1, 9205,
       37, 29964, 29364, 21501, 4656, 54735, 5425, 30105, 754, 6680, 13354, 758, 30106,
       0, 29209, 48736, 25595, 33780, 4657 )
11053
11054
11055 # ai: 29181, 4677, 29950, 33, 62496, 429
11056 # bi: 1, 9205, 37, 29964, 29364, 21501
11057
11058 # Tekirdag-1:
```

```
11059
11060 G13_8:
11061 ▷  $(ORBITER) -v 3 \
11062 ▷  ▷  -define F -finite_field -q 8 -end \
11063 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
11064 ▷  ▷  -define T1 -cubic_surface -space P -family_G13 2 -end \
11065 ▷  ▷  -with T1 -do \
11066 ▷  ▷  -cubic_surface_activity \
11067 ▷  ▷  ▷  -report \
11068 ▷  ▷  -end
11069
11070
11071 F13_8:
11072 ▷  $(ORBITER) -v 3 \
11073 ▷  ▷  -define F -finite_field -q 8 -end \
11074 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
11075 ▷  ▷  -define T1 -cubic_surface -space P -family_F13 2 -end \
11076 ▷  ▷  -with T1 -do \
11077 ▷  ▷  -cubic_surface_activity \
11078 ▷  ▷  ▷  -report \
11079 ▷  ▷  -end
11080
11081
11082
11083
11084 # Tekirdag-2:
11085
11086 F13_16:
11087 ▷  $(ORBITER) -v 3 \
11088 ▷  ▷  -define F -finite_field -q 16 -end \
11089 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
11090 ▷  ▷  -define T2 -cubic_surface -space P -family_F13 2 -end \
11091 ▷  ▷  -with T2 -do \
11092 ▷  ▷  -cubic_surface_activity \
11093 ▷  ▷  ▷  -report \
11094 ▷  ▷  -end
11095
11096
11097 # Tekirdag-3:
11098
11099 F13_32:
11100 ▷  $(ORBITER) -v 3 \
11101 ▷  ▷  -define F -finite_field -q 32 -end \
11102 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
11103 ▷  ▷  -define T3 -cubic_surface -space P -family_F13 2 -end \
11104 ▷  ▷  -with T3 -do \
11105 ▷  ▷  -cubic_surface_activity \
11106 ▷  ▷  ▷  -report \
11107 ▷  ▷  -end
11108
11109
11110 # Kapadokya-1:
11111
11112 F13_64a:
11113 ▷  $(ORBITER) -v 3 \
11114 ▷  ▷  -define F -finite_field -q 64 -end \
11115 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
11116 ▷  ▷  -define K1 -cubic_surface -space P -family_F13 2 -end \
11117 ▷  ▷  -with K1 -do \
```

```
11118 ▷  ▷  -cubic_surface_activity \
11119 ▷  ▷  ▷  -report \
11120 ▷  ▷  -end
11121
11122
11123 # Kapadokya-2:
11124
11125 F13_64b:
11126 ▷  $(ORBITER) -v 3 \
11127 ▷  ▷  -define F -finite_field -q 64 -end \
11128 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
11129 ▷  ▷  -define K2 -cubic_surface -space P -family_F13 18 -end \
11130 ▷  ▷  -with K2 -do \
11131 ▷  ▷  -cubic_surface_activity \
11132 ▷  ▷  ▷  -report \
11133 ▷  ▷  -end
11134
11135
11136 Colorado1:
11137 ▷  $(ORBITER) -v 3 \
11138 ▷  ▷  -define F -finite_field -q 128 -end \
11139 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
11140 ▷  ▷  -define CO-1 -cubic_surface -space P -catalogue 0 \
11141 ▷  ▷  ▷  -transform_inverse "1,0,0,0,0,1,0,96,0,0,1,96,0,0,0,1,0" \
11142 ▷  ▷  -end \
11143 ▷  ▷  -with CO-1 -do \
11144 ▷  ▷  -cubic_surface_activity \
11145 ▷  ▷  ▷  -report \
11146 ▷  ▷  -end
11147
11148
11149 # recognize the arcs from Colorado-1,2,3:
11150
11151
11152
11153 Colorado2:
11154 ▷  $(ORBITER) -v 3 \
11155 ▷  ▷  -define F -finite_field -q 128 -end \
11156 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
11157 ▷  ▷  -define CO-2 -cubic_surface -space P -catalogue 926 \
11158 ▷  ▷  ▷  -transform_inverse "1,0,0,0,0,1,0,32,0,0,1,32,0,0,0,1,0" \
11159 ▷  ▷  -end \
11160 ▷  ▷  -with CO-2 -do \
11161 ▷  ▷  -cubic_surface_activity \
11162 ▷  ▷  ▷  -report \
11163 ▷  ▷  -end
11164
11165 Colorado3:
11166 ▷  $(ORBITER) -v 3 \
11167 ▷  ▷  -define F -finite_field -q 128 -end \
11168 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
11169 ▷  ▷  -define CO-3 -cubic_surface -space P -catalogue 928 \
11170 ▷  ▷  ▷  -transform_inverse "1,0,0,0,0,1,0,59,0,0,1,59,0,0,0,1,0" \
11171 ▷  ▷  -end \
11172 ▷  ▷  -with CO-3 -do \
11173 ▷  ▷  -cubic_surface_activity \
11174 ▷  ▷  ▷  -report \
11175 ▷  ▷  -end
11176
```

```
11177
11178 # Colorado-1:
11179
11180 F13_128a:
11181 ▷  $(ORBITER) -v 3 \
11182 ▷  ▷  -define F -finite_field -q 128 -end \
11183 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
11184 ▷  ▷  -define CO-1 -cubic_surface -space P -family_F13 2 -end \
11185 ▷  ▷  -with CO-1 -do \
11186 ▷  ▷  -cubic_surface_activity \
11187 ▷  ▷  ▷  -report \
11188 ▷  ▷  -end
11189
11190 # Colorado-2:
11191
11192 F13_128b:
11193 ▷  $(ORBITER) -v 3 \
11194 ▷  ▷  -define F -finite_field -q 128 -end \
11195 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
11196 ▷  ▷  -define CO-2 -cubic_surface -space P -family_F13 6 -end \
11197 ▷  ▷  -with CO-2 -do \
11198 ▷  ▷  -cubic_surface_activity \
11199 ▷  ▷  ▷  -report \
11200 ▷  ▷  -end
11201
11202 # Colorado-3:
11203
11204 F13_128c:
11205 ▷  $(ORBITER) -v 3 \
11206 ▷  ▷  -define F -finite_field -q 128 -end \
11207 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
11208 ▷  ▷  -define CO-3 -cubic_surface -space P -family_F13 14 -end \
11209 ▷  ▷  -with CO-3 -do \
11210 ▷  ▷  -cubic_surface_activity \
11211 ▷  ▷  ▷  -report \
11212 ▷  ▷  -end
11213
11214
11215 move_two_lines:
11216 ▷  $(ORBITER) -v 5 \
11217 ▷  ▷  -define F -finite_field -q 8 -end \
11218 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
11219 ▷  ▷  -with P -do -projective_space_activity \
11220 ▷  ▷  ▷  -move_two_lines_in_hyperplane_stabilizer \
11221 ▷  ▷  ▷  65 4680 72 657 \
11222 ▷  ▷  -end
11223
11224
11225
11226
11227
11228 surface_F_abcd_Eckardt_q31:
11229 ▷  $(ORBITER) -v 10 \
11230 ▷  ▷  -define F -finite_field -q 31 -end \
11231 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
11232 ▷  ▷  -define R -polynomial_ring \
11233 ▷  ▷  ▷  -field F \
11234 ▷  ▷  ▷  -number_of_variables 4 \
11235 ▷  ▷  ▷  -homogeneous_of_degree 3 \
```

```
11236 ▷  ▷  ▷  -monomial_ordering_partition \
11237 ▷  ▷  ▷  -variables "X0,X1,X2,X3" "X_0,X_1,X_2,X_3" \
11238 ▷  ▷  ▷  -end \
11239 ▷  ▷  -define F_abcd -symbolic_object \
11240 ▷  ▷  ▷  -field F \
11241 ▷  ▷  ▷  -text $(F_abcd_eqn) \
11242 ▷  ▷  ▷  -end \
11243 ▷  ▷  -define abcd_values -symbolic_object \
11244 ▷  ▷  ▷  -field F \
11245 ▷  ▷  ▷  -text "2,30,30,2" \
11246 ▷  ▷  ▷  -end \
11247 ▷  ▷  -define F_abcd_sub -symbolic_object \
11248 ▷  ▷  ▷  -field F \
11249 ▷  ▷  ▷  -managed_variables "X0,X1,X2,X3" \
11250 ▷  ▷  ▷  -substitute "a,b,c,d" F_abcd abcd_values \
11251 ▷  ▷  -end \
11252 ▷  ▷  -define S -cubic_surface -space P \
11253 ▷  ▷  ▷  -by_symbolic_object \
11254 ▷  ▷  ▷  R \
11255 ▷  ▷  ▷  F_abcd_sub \
11256 ▷  ▷  -end \
11257 ▷  ▷  -with S -do \
11258 ▷  ▷  -cubic_surface_activity \
11259 ▷  ▷  ▷  -report \
11260 ▷  ▷  -end
11261 ▷  pdflatex surface_equation_F_abcd_sub_q31_report.tex
11262 ▷  $(OPEN) surface_equation_F_abcd_sub_q31_report.pdf
11263
11264
11265
11266 surface_F_abcd_Eckardt_q31_by_equation_report:
11267 ▷  $(ORBITER) -v 10 \
11268 ▷  ▷  -define F -finite_field -q 31 -end \
11269 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
11270 ▷  ▷  -define R -polynomial_ring \
11271 ▷  ▷  ▷  -field F \
11272 ▷  ▷  ▷  -number_of_variables 4 \
11273 ▷  ▷  ▷  -homogeneous_of_degree 3 \
11274 ▷  ▷  ▷  -monomial_ordering_partition \
11275 ▷  ▷  ▷  -variables "X0,X1,X2,X3" "X_0,X_1,X_2,X_3" \
11276 ▷  ▷  ▷  -end \
11277 ▷  ▷  -define F_abcd -cubic_surface -space P \
11278 ▷  ▷  ▷  -by_equation \
11279 ▷  ▷  ▷  R \
11280 ▷  ▷  ▷  "F_abcd" \
11281 ▷  ▷  ▷  "\DF_{a,b,c,d}\D" "X0,X1,X2,X3" \
11282 ▷  ▷  ▷  $(F_abcd_eqn) \
11283 ▷  ▷  ▷  "a,b,c,d" \
11284 ▷  ▷  ▷  "\D(a,b,c,d)=(2,30,30,2)\D" \
11285 ▷  ▷  ▷  "2,30,30,2" \
11286 ▷  ▷  -end \
11287 ▷  ▷  -with F_abcd -do \
11288 ▷  ▷  -cubic_surface_activity \
11289 ▷  ▷  ▷  -report \
11290 ▷  ▷  -end
11291 ▷  pdflatex surface_equation_F_abcd_q31_report.tex
11292 ▷  $(OPEN) surface_equation_F_abcd_q31_report.pdf
11293
11294
```

```
11295 # E = 6
11296
11297
11298
11299
11300
11301 surface_F_abcd_sweep_4_27_q7:
11302 ▷  $(ORBITER) -v 3 \
11303 ▷  ▷  -define F -finite_field -q 7 -end \
11304 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
11305 ▷  ▷  -define R -polynomial_ring \
11306 ▷  ▷  ▷  -field F \
11307 ▷  ▷  ▷  -number_of_variables 4 \
11308 ▷  ▷  ▷  -homogeneous_of_degree 3 \
11309 ▷  ▷  ▷  -monomial_ordering_partition \
11310 ▷  ▷  ▷  -variables "X0,X1,X2,X3" "X_0,X_1,X_2,X_3" \
11311 ▷  ▷  ▷  -end \
11312 ▷  ▷  -with P -do \
11313 ▷  ▷  -projective_space_activity \
11314 ▷  ▷  -sweep_4_27 sweep_4_27_q7 \
11315 ▷  ▷  ▷  -space P \
11316 ▷  ▷  ▷  -by_equation R "F_abcd" \
11317 ▷  ▷  ▷  "\DF_{a,b,c,d}\D" "X0,X1,X2,X3" \
11318 ▷  ▷  ▷  $(F_abcd_eqn_no_exponents) \
11319 ▷  ▷  ▷  "a=2,b=3,c=4,d=5" \
11320 ▷  ▷  ▷  "\Da=2,b=3,c=4,d=5\D" \
11321 ▷  ▷  -end \
11322 ▷  ▷  -end \
11323
11324
11325
11326 surface_11_random_report:
11327 ▷  $(ORBITER) -v 10 \
11328 ▷  ▷  -define F -finite_field -q 11 -end \
11329 ▷  ▷  -define P -projective_space -n 3 -field F -v 6 -end \
11330 ▷  ▷  -define S -cubic_surface -space P -random -end \
11331 ▷  ▷  -with S -do \
11332 ▷  ▷  -cubic_surface_activity \
11333 ▷  ▷  ▷  -report \
11334 ▷  ▷  -end
11335 ▷  pdflatex surface_random_q11_report.tex
11336 ▷  $(OPEN) surface_random_q11_report.pdf
11337
11338 surface_11_1:
11339 ▷  $(ORBITER) -v 3 \
11340 ▷  ▷  -define F -finite_field -q 11 -end \
11341 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
11342 ▷  ▷  -define S -cubic_surface -space P -catalogue 1 -end \
11343 ▷  ▷  -with S -do \
11344 ▷  ▷  -cubic_surface_activity \
11345 ▷  ▷  ▷  -report \
11346 ▷  ▷  -end
11347 ▷  pdflatex surface_catalogue_q11_iso1_report.tex
11348 ▷  $(OPEN) surface_catalogue_q11_iso1_report.pdf
11349
11350 ##############################################################################
11351 # Section 8.4: Classification of Cubic Surfaces with 27 lines
11352
11353
```

```
11354 SECTION_CLASSIFICATION_OF_CUBIC_SURFACES_WITH_27_LINES:
11355
11356
11357 test_8_4:
11358 ▷  make surface_classify_q4
11359 ▷  make surface_classify_q4_arc_lifting_two_lines
11360 ▷  make surface_classify_q7
11361 ▷  make surface_classify_q9
11362 ▷  make surface_classify_q11
11363 ▷  make surface_classify_q13
11364 ▷  make Family_general_F31_25_5_5_25_recognize
11365 ▷  make surface_F_abcd_identify_q11
11366 ▷  make Family_general_F11_iso_test
11367
11368
11369
11370 surface_classify_q4:
11371 ▷  $(ORBITER) -v 5 \
11372 ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
11373 ▷  ▷  -define F -finite_field -q 4 -end \
11374 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
11375 ▷  ▷  -define Control -poset_classification_control -W \
11376 ▷  ▷  ▷  -problem_label "surf_q4" \
11377 ▷  ▷  -end \
11378 ▷  ▷  -define report_options -poset_classification_report_options \
11379 ▷  ▷  -end \
11380 ▷  ▷  -define Orb -orbits \
11381 ▷  ▷  ▷  -on_cubic_surfaces P Control \
11382 ▷  ▷  -end \
11383 ▷  ▷  -with Orb -do -orbits_activity \
11384 ▷  ▷  ▷  -report \
11385 ▷  ▷  -end
11386 ▷  pdflatex Surfaces_q4.tex
11387 ▷  $(OPEN) Surfaces_q4.pdf
11388
11389 #▷ ▷  ▷  -draw_poset \
11390
11391
11392 surface_classify_q4_old:
11393 ▷  $(ORBITER) -v 5 \
11394 ▷  ▷  -define F -finite_field -q 4 -end \
11395 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
11396 ▷  ▷  -define Control -poset_classification_control -W \
11397 ▷  ▷  ▷  -problem_label "surf_q4" \
11398 ▷  ▷  -end \
11399 ▷  ▷  -with P -do \
11400 ▷  ▷  -projective_space_activity \
11401 ▷  ▷  ▷  -classify_surfaces_with_double_sixes Surf27 Control \
11402 ▷  ▷  -end \
11403 ▷  ▷  -with Surf27 -do \
11404 ▷  ▷  -classification_of_cubic_surfaces_with_double_sixes_activity \
11405 ▷  ▷  ▷  -report -end \
11406 ▷  ▷  -end \
11407 ▷  ▷  -with Surf27 -do \
11408 ▷  ▷  -classification_of_cubic_surfaces_with_double_sixes_activity \
11409 ▷  ▷  ▷  -create_source_code \
11410 ▷  ▷  -end \
11411 ▷  ▷  -with Surf27 -do \
11412 ▷  ▷  -classification_of_cubic_surfaces_with_double_sixes_activity \
```

```
11413 ▷  ▷  ▷  -stats surf_q4 \
11414 ▷  ▷  -end \
11415 ▷  ▷  -print_symbols
11416 ▷  pdflatex Surfaces_q4.tex
11417 ▷  $(OPEN) Surfaces_q4.pdf
11418
11419 # time: 0:00
11420
11421
11422
11423
11424 surface_classify_q4_arc_lifting_two_lines:
11425 ▷  $(ORBITER) -v 10 \
11426 ▷  ▷  -define F -finite_field -q 4 -end \
11427 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
11428 ▷  ▷  -define Control -poset_classification_control \
11429 ▷  ▷  ▷  -problem_label sixarcs_q4 -W \
11430 ▷  ▷  -end \
11431 ▷  ▷  -with P -do \
11432 ▷  ▷  -projective_space_activity \
11433 ▷  ▷  ▷  -control_six_arcs Control \
11434 ▷  ▷  ▷  -classify_surfaces_through_arcs_and_two_lines \
11435 ▷  ▷  -end
11436 ▷  pdflatex surfaces_arc_lifting_4.tex
11437 ▷  $(OPEN) surfaces_arc_lifting_4.pdf
11438
11439
11440
11441 surface_classify_q7:
11442 ▷  $(ORBITER) -v 5 \
11443 ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
11444 ▷  ▷  -define F -finite_field -q 7 -end \
11445 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
11446 ▷  ▷  -define Control -poset_classification_control -W \
11447 ▷  ▷  ▷  -problem_label "surf_q7" \
11448 ▷  ▷  ▷  -draw_options \
11449 ▷  ▷  ▷  ▷  -xin 400000 \
11450 ▷  ▷  ▷  ▷  -xout 3000000 \
11451 ▷  ▷  ▷  ▷  -yout 3000000 \
11452 ▷  ▷  ▷  ▷  -y_stretch 0.5 \
11453 ▷  ▷  ▷  -end \
11454 ▷  ▷  -end \
11455 ▷  ▷  -define report_options -poset_classification_report_options \
11456 ▷  ▷  ▷  -draw_poset -type_aux \
11457 ▷  ▷  -end \
11458 ▷  ▷  -define Orb -orbits \
11459 ▷  ▷  ▷  -on_cubic_surfaces P Control \
11460 ▷  ▷  -end \
11461 ▷  ▷  -with Orb -do -orbits_activity \
11462 ▷  ▷  ▷  -report \
11463 ▷  ▷  -end
11464 ▷  pdflatex Surfaces_q7.tex
11465 ▷  $(OPEN) Surfaces_q7.pdf
11466
11467
11468
11469 surface_classify_q7_old:
11470 ▷  $(ORBITER) -v 5 \
11471 ▷  ▷  -define F -finite_field -q 7 -end \
```

```
11472 ▷  ▷  -define P -projective_space -n 3 -field F -end \
11473 ▷  ▷  -define Control -poset_classification_control -W -end \
11474 ▷  ▷  -with P -do \
11475 ▷  ▷  -projective_space_activity \
11476 ▷  ▷  ▷  -classify_surfaces_with_double_sixes Surf27 Control \
11477 ▷  ▷  -end \
11478 ▷  ▷  -with Surf27 -do \
11479 ▷  ▷  -classification_of_cubic_surfaces_with_double_sixes_activity \
11480 ▷  ▷  ▷  -report -end \
11481 ▷  ▷  -end \
11482 ▷  ▷  -print_symbols
11483 ▷  pdflatex Surfaces_q7.tex
11484 ▷  $(OPEN) Surfaces_q7.pdf
11485
11486 #neighbors_7.csv
11487
11488
11489 surface_classify_q9:
11490 ▷  $(ORBITER) -v 5 \
11491 ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
11492 ▷  ▷  -define F -finite_field -q 9 -end \
11493 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
11494 ▷  ▷  -define Control -poset_classification_control -W \
11495 ▷  ▷  ▷  -problem_label "surf_q9" \
11496 ▷  ▷  -end \
11497 ▷  ▷  -define report_options -poset_classification_report_options \
11498 ▷  ▷  -end \
11499 ▷  ▷  -define Orb -orbits \
11500 ▷  ▷  ▷  -on_cubic_surfaces P Control \
11501 ▷  ▷  -end \
11502 ▷  ▷  -with Orb -do -orbits_activity \
11503 ▷  ▷  ▷  -report \
11504 ▷  ▷  -end
11505 ▷  pdflatex Surfaces_q9.tex
11506 ▷  $(OPEN) Surfaces_q9.pdf
11507
11508
11509 surface_classify_q9_old:
11510 ▷  $(ORBITER) -v 5 \
11511 ▷  ▷  -define F -finite_field -q 9 -end \
11512 ▷  ▷  -define P -projective_space -n 3 -field F -end \
11513 ▷  ▷  -define Control -poset_classification_control -W -end \
11514 ▷  ▷  -with P -do \
11515 ▷  ▷  -projective_space_activity \
11516 ▷  ▷  ▷  -classify_surfaces_with_double_sixes Surf27 Control \
11517 ▷  ▷  -end \
11518 ▷  ▷  -with Surf27 -do \
11519 ▷  ▷  -classification_of_cubic_surfaces_with_double_sixes_activity \
11520 ▷  ▷  ▷  -report -end \
11521 ▷  ▷  -end \
11522 ▷  ▷  -print_symbols
11523 ▷  pdflatex Surfaces_q9.tex
11524 ▷  $(OPEN) Surfaces_q9.pdf
11525
11526
11527
11528 surface_classify_q11:
11529 ▷  $(ORBITER) -v 5 \
11530 ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
```

```
11531 ▷  ▷  -define F -finite_field -q 11 -end \
11532 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
11533 ▷  ▷  -define Control -poset_classification_control -W \
11534 ▷  ▷  ▷  -problem_label "surf_q11" \
11535 ▷  ▷  -end \
11536 ▷  ▷  -define report_options -poset_classification_report_options \
11537 ▷  ▷  -end \
11538 ▷  ▷  -define Orb -orbits \
11539 ▷  ▷  ▷  -on_cubic_surfaces P Control \
11540 ▷  ▷  -end \
11541 ▷  ▷  -with Orb -do -orbits_activity \
11542 ▷  ▷  ▷  -report \
11543 ▷  ▷  -end
11544 ▷  pdflatex Surfaces_q11.tex
11545 ▷  $(OPEN) Surfaces_q11.pdf
11546
11547
11548 surface_classify_q13:
11549 ▷  $(ORBITER) -v 5 \
11550 ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
11551 ▷  ▷  -define F -finite_field -q 13 -end \
11552 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
11553 ▷  ▷  -define Control -poset_classification_control -W \
11554 ▷  ▷  ▷  -problem_label "surf_q13" \
11555 ▷  ▷  -end \
11556 ▷  ▷  -define report_options -poset_classification_report_options \
11557 ▷  ▷  -end \
11558 ▷  ▷  -define Orb -orbits \
11559 ▷  ▷  ▷  -on_cubic_surfaces P Control \
11560 ▷  ▷  -end \
11561 ▷  ▷  -with Orb -do -orbits_activity \
11562 ▷  ▷  ▷  -report \
11563 ▷  ▷  -end
11564 ▷  pdflatex Surfaces_q13.tex
11565 ▷  $(OPEN) Surfaces_q13.pdf
11566
11567
11568
11569
11570 Family_general_F31_25_5_5_25_recognize:
11571 ▷  $(ORBITER) -v 3 \
11572 ▷  ▷  -define F -finite_field -q 31 -end \
11573 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
11574 ▷  ▷  -define Surf -cubic_surface \
11575 ▷  ▷  ▷  -space P -family_general_abcd 25 5 5 25 \
11576 ▷  ▷  -end \
11577 ▷  ▷  -with Surf -do \
11578 ▷  ▷  -cubic_surface_activity \
11579 ▷  ▷  ▷  -report \
11580 ▷  ▷  -end \
11581 ▷  ▷  -define Control -poset_classification_control -W \
11582 ▷  ▷  ▷  -problem_label "surf_q31" \
11583 ▷  ▷  -end \
11584 ▷  ▷  -with P -do \
11585 ▷  ▷  -projective_space_activity \
11586 ▷  ▷  ▷  -classify_surfaces_with_double_sixes AllSurf Control \
11587 ▷  ▷  -end \
11588 ▷  ▷  -with AllSurf -do \
11589 ▷  ▷  -classification_of_cubic_surfaces_with_double_sixes_activity \
```

```
11590 ▷  ▷  ▷  -recognize Surf \
11591 ▷  ▷  -end
11592
11593 surface_F_abcd_identify_q11:
11594 ▷  $(ORBITER) -v 3 \
11595 ▷  ▷  -define F -finite_field -q 11 -end \
11596 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
11597 ▷  ▷  -define Control -poset_classification_control -W \
11598 ▷  ▷  ▷  -problem_label "surf_q11" \
11599 ▷  ▷  -end \
11600 ▷  ▷  -with P -do \
11601 ▷  ▷  -projective_space_activity \
11602 ▷  ▷  ▷  -classify_surfaces_with_double_sixes AllSurf Control \
11603 ▷  ▷  -end \
11604 ▷  ▷  -with AllSurf -do \
11605 ▷  ▷  -classification_of_cubic_surfaces_with_double_sixes_activity \
11606 ▷  ▷  ▷  -identify_general_abcd \
11607 ▷  ▷  -end
11608
11609 Family_general_F11_iso_test:
11610 ▷  $(ORBITER) -v 3 \
11611 ▷  ▷  -define F -finite_field -q 11 -end \
11612 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
11613 ▷  ▷  -define Surf1 -cubic_surface \
11614 ▷  ▷  ▷  -space P -family_general_abcd 2 4 4 2 \
11615 ▷  ▷  -end \
11616 ▷  ▷  -define Surf2 -cubic_surface \
11617 ▷  ▷  ▷  -space P -family_general_abcd 5 10 7 5 \
11618 ▷  ▷  -end \
11619 ▷  ▷  -define Control -poset_classification_control -W \
11620 ▷  ▷  ▷  -problem_label "surf_q11" \
11621 ▷  ▷  -end \
11622 ▷  ▷  -with P -do \
11623 ▷  ▷  -projective_space_activity \
11624 ▷  ▷  ▷  -classify_surfaces_with_double_sixes AllSurf Control \
11625 ▷  ▷  -end \
11626 ▷  ▷  -with AllSurf -do \
11627 ▷  ▷  -classification_of_cubic_surfaces_with_double_sixes_activity \
11628 ▷  ▷  ▷  -isomorphism_testing Surf1 Surf2 \
11629 ▷  ▷  -end
11630
11631
11632
11633
11634 ##############################################################################
11635 # Section 8.5: Quartic Curves Creation
11636
11637
11638 SECTION_QUARTIC_CURVES_CREATION:
11639
11640
11641 test_8_5:
11642 ▷  make quartic_curve_9_0
11643 ▷  make quartic_curve_edge_q9
11644 ▷  make surface_7_0_make_quartic_curves
11645 ▷  make surface_9_1_make_quartic_curves
11646 ▷  make quartic_curve_9_0_report
11647 ▷  make quartic_curve_13_0_report
11648 ▷  make PG_2_13_rank_lines
```

```
11649 ▷  make PG_2_13_orbits_on_lines
11650 ▷  make quartic_curve_13_1_report
11651 ▷  make quartic_curves_19_report
11652 ▷  make quartic_curve_q9_1
11653
11654
11655 quartic_curve_9_0:
11656 ▷  $(ORBITER) -v 3 \
11657 ▷  ▷  -define F -finite_field -q 9 -end \
11658 ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
11659 ▷  ▷  -define C -quartic_curve -space P -catalogue 0 -end
11660
11661
11662
11663 # the coefficient 1 is actually 4 mod 3:
11664
11665
11666 quartic_curve_edge_q9:
11667 ▷  $(ORBITER) -v 3 \
11668 ▷  ▷  -define F -finite_field -q 9 -end \
11669 ▷  ▷  -define P2 -projective_space -n 2 -field F -end \
11670 ▷  ▷  -define R -polynomial_ring \
11671 ▷  ▷  ▷  -field F \
11672 ▷  ▷  ▷  -number_of_variables 3 \
11673 ▷  ▷  ▷  -homogeneous_of_degree 4 \
11674 ▷  ▷  ▷  -monomial_ordering_partition \
11675 ▷  ▷  ▷  -variables "X,Y,Z" "X,Y,Z" \
11676 ▷  ▷  -end \
11677 ▷  ▷  -define Edge -symbolic_object \
11678 ▷  ▷  ▷  -field F \
11679 ▷  ▷  ▷  -managed_variables "X,Y,Z" \
11680 ▷  ▷  ▷  -text "X^4-Y^4-Z^4+2*f^2*Y^2*Z^2+1*f*X^2*Y*Z" \
11681 ▷  ▷  -end \
11682 ▷  ▷  -define f -symbolic_object \
11683 ▷  ▷  ▷  -field F \
11684 ▷  ▷  ▷  -text "3" \
11685 ▷  ▷  -end \
11686 ▷  ▷  -define Edge_f3 -symbolic_object \
11687 ▷  ▷  ▷  -field F \
11688 ▷  ▷  ▷  -managed_variables "X,Y,Z" \
11689 ▷  ▷  ▷  -substitute "f" Edge f \
11690 ▷  ▷  -end \
11691 ▷  ▷  -define C -quartic_curve \
11692 ▷  ▷  ▷  -space P2 \
11693 ▷  ▷  ▷  -by_symbolic_object R Edge_f3 -end \
11694 ▷  ▷  -with C -do \
11695 ▷  ▷  ▷  -quartic_curve_activity \
11696 ▷  ▷  ▷  ▷  -report \
11697 ▷  ▷  ▷  -end
11698 ▷  pdflatex quartic_curve_equation_Edge_f3_q9_report.tex
11699 ▷  $(OPEN) quartic_curve_equation_Edge_f3_q9_report.pdf
11700
11701
11702
11703 surface_7_0_make_quartic_curves:
11704 ▷  $(ORBITER) -v 3 \
11705 ▷  ▷  -define F -finite_field -q 7 -end \
11706 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
11707 ▷  ▷  -define S -cubic_surface -space P -catalogue 0 -end \
```

```
11708 ▷  ▷  -with S -do \
11709 ▷  ▷  -cubic_surface_activity \
11710 ▷  ▷  ▷  -report \
11711 ▷  ▷  ▷  -all_quartic_curves \
11712 ▷  ▷  -end \
11713 ▷  #pdflatex surface_catalogue_q7_iso0_report.tex
11714 ▷  #$(OPEN) surface_catalogue_q7_iso0_report.pdf
11715
11716
11717
11718
11719 surface_9_1_make_quartic_curves:
11720 ▷  $(ORBITER) -v 3 \
11721 ▷  ▷  -define F -finite_field -q 9 -end \
11722 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
11723 ▷  ▷  -define S -cubic_surface -space P -catalogue 1 -end \
11724 ▷  ▷  -with S -do \
11725 ▷  ▷  -cubic_surface_activity \
11726 ▷  ▷  ▷  -report \
11727 ▷  ▷  ▷  -all_quartic_curves \
11728 ▷  ▷  -end \
11729 ▷  #pdflatex surface_catalogue_q9_iso1_report.tex
11730 ▷  #$(OPEN) surface_catalogue_q9_iso1_report.pdf
11731
11732
11733 quartic_curve_9_0_report:
11734 ▷  $(ORBITER) -v 3 \
11735 ▷  ▷  -define F -finite_field -q 9 -end \
11736 ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
11737 ▷  ▷  -define C -quartic_curve -space P -catalogue 0 -end \
11738 ▷  ▷  -with C -do \
11739 ▷  ▷  ▷  -quartic_curve_activity \
11740 ▷  ▷  ▷  ▷  -report \
11741 ▷  ▷  ▷  -end
11742 ▷  pdflatex quartic_curve_catalogue_q9_iso0_report.tex
11743 ▷  $(OPEN) quartic_curve_catalogue_q9_iso0_report.pdf
11744
11745
11746
11747
11748 quartic_curve_13_0_report:
11749 ▷  $(ORBITER) -v 3 \
11750 ▷  ▷  -define F -finite_field -q 13 -end \
11751 ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
11752 ▷  ▷  -define C -quartic_curve -space P -catalogue 0 \
11753 ▷  ▷  ▷  -transform "10,4,1,11,5,11,4,1,1" \
11754 ▷  ▷  ▷  -transform_inverse "9,1,0,12,9,0,2,10,11" \
11755 ▷  ▷  -end \
11756 ▷  ▷  -with C -do \
11757 ▷  ▷  ▷  -quartic_curve_activity \
11758 ▷  ▷  ▷  ▷  -report \
11759 ▷  ▷  ▷  -end \
11760 ▷  ▷  -with C -do \
11761 ▷  ▷  ▷  -quartic_curve_activity \
11762 ▷  ▷  ▷  ▷  -extract_orbit_on_bitangents_by_length 4 \
11763 ▷  ▷  ▷  -end
11764 ▷  #pdflatex quartic_curve_catalogue_q13_iso0_report.tex
11765 ▷  #$(OPEN) quartic_curve_catalogue_q13_iso0_report.pdf
11766
```

```
11767 # 170, 111, 140, 2
11768
11769
11770 PG_2_13_rank_lines:
11771 ▷  $(ORBITER) -v 2 \
11772 ▷  ▷  -define v -vector -format 4 \
11773 ▷  ▷  -dense "1,0,0, 0,1,0, 1,0,0, 0,0,1, 1,1,1, 0,1,0, 1,1,1, 0,0,1" \
11774 ▷  ▷  -end \
11775 ▷  ▷  -define F -finite_field -q 23 -end \
11776 ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
11777 ▷  ▷  -with P -do \
11778 ▷  ▷  -projective_space_activity \
11779 ▷  ▷  ▷  -rank_lines_in_PG v \
11780 ▷  ▷  -end \
11781
11782
11783 PG_2_13_orbits_on_lines:
11784 ▷  $(ORBITER) -v 5 \
11785 ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
11786 ▷  ▷  -define Control -poset_classification_control \
11787 ▷  ▷  ▷  -problem_label PGL_3_13 \
11788 ▷  ▷  ▷  -depth 4 \
11789 ▷  ▷  ▷  -draw_options -radius 200 -end  \
11790 ▷  ▷  -end \
11791 ▷  ▷  -define G -linear_group -PGL 3 13 -end \
11792 ▷  ▷  -define G_on_lines -modified_group -from G \
11793 ▷  ▷  ▷  -on_k_subspaces 2 \
11794 ▷  ▷  -end \
11795 ▷  ▷  -define Orb -orbits -group G_on_lines \
11796 ▷  ▷  ▷  -on_subsets 4 Control \
11797 ▷  ▷  -end \
11798 ▷  ▷  -with Orb -do -orbits_activity \
11799 ▷  ▷  ▷  -recognize "170, 111, 140, 2" \
11800 ▷  ▷  ▷  -recognize "0,23,24,47" \
11801 ▷  ▷  -end
11802 ▷  #pdflatex PGL_3_13_poset.tex
11803 ▷  #$(OPEN) PGL_3_13_poset.pdf
11804
11805 # stabilizer of {0,23,24,47}
11806 #1,0,0,7,9,0,9,5,3,
11807 #1,3,0,1,12,0,10,9,2,
11808 #1,1,11,7,9,6,10,5,2,
11809 #1,4,11,12,1,8,10,4,2,
11810
11811 quartic_curve_13_1_report:
11812 ▷  $(ORBITER) -v 3 \
11813 ▷  ▷  -define F -finite_field -q 13 -end \
11814 ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
11815 ▷  ▷  -define C -quartic_curve -space P -catalogue 1 -end \
11816 ▷  ▷  -with C -do \
11817 ▷  ▷  ▷  -quartic_curve_activity \
11818 ▷  ▷  ▷  ▷  -report \
11819 ▷  ▷  ▷  -end
11820 ▷  pdflatex quartic_curve_catalogue_q13_iso1_report.tex
11821 ▷  $(OPEN) quartic_curve_catalogue_q13_iso1_report.pdf
11822
11823
11824
11825 quartic_curves_19_report:
```

```
11826 ▷  $(ORBITER) -v 3 \
11827 ▷  ▷  -define F -finite_field -q 19 -end \
11828 ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
11829 ▷  ▷  -loop L 0 $(NB_QUARTIC_CURVES_Q19) 1 \
11830 ▷  ▷  -define C -quartic_curve -space P -catalogue %L -end \
11831 ▷  ▷  -with C -do \
11832 ▷  ▷  ▷  -quartic_curve_activity \
11833 ▷  ▷  ▷  ▷  -report \
11834 ▷  ▷  ▷  -end \
11835 ▷  ▷  -end_loop L
11836 ▷  ▷  pdflatex quartic_curve_catalogue_q19_iso0_report.tex
11837 ▷  ▷  pdflatex quartic_curve_catalogue_q19_iso1_report.tex
11838 ▷  ▷  pdflatex quartic_curve_catalogue_q19_iso2_report.tex
11839 ▷  ▷  pdflatex quartic_curve_catalogue_q19_iso3_report.tex
11840 ▷  ▷  pdflatex quartic_curve_catalogue_q19_iso4_report.tex
11841 ▷  ▷  pdflatex quartic_curve_catalogue_q19_iso5_report.tex
11842 ▷  ▷  pdflatex quartic_curve_catalogue_q19_iso6_report.tex
11843 ▷  ▷  pdflatex quartic_curve_catalogue_q19_iso7_report.tex
11844 ▷  ▷  pdflatex quartic_curve_catalogue_q19_iso8_report.tex
11845 ▷  ▷  pdflatex quartic_curve_catalogue_q19_iso9_report.tex
11846 ▷  ▷  pdflatex quartic_curve_catalogue_q19_iso10_report.tex
11847 ▷  ▷  pdflatex quartic_curve_catalogue_q19_iso11_report.tex
11848 ▷  ▷  pdflatex quartic_curve_catalogue_q19_iso12_report.tex
11849 ▷  ▷  pdflatex quartic_curve_catalogue_q19_iso13_report.tex
11850
11851
11852
11853 quartic_curve_q9_1:
11854 ▷  $(ORBITER) -v 3 \
11855 ▷  ▷  -define F -finite_field -q 9 -end \
11856 ▷  ▷  -define P2 -projective_space -n 2 -field F -end \
11857 ▷  ▷  -define P3 -projective_space -n 3 -field F -end \
11858 ▷  ▷  -define S9_1 -cubic_surface -space P3 -catalogue 1 -end \
11859 ▷  ▷  -define C -quartic_curve -space P2 -from_cubic_surface S9_1 0 -end \
11860 ▷  ▷  -with C -do \
11861 ▷  ▷  ▷  -quartic_curve_activity \
11862 ▷  ▷  ▷  ▷  -report \
11863 ▷  ▷  ▷  -end
11864 ▷  pdflatex quartic_curve_surface_surface_pt_orb_0_report.tex
11865 ▷  $(OPEN) quartic_curve_surface_surface_pt_orb_0_report.pdf
11866
11867
11868 #The points by rank are: ( 5, 18, 25, 36, 39, 40, 42, 43, 47, 50, 51, 54, 55, 58,
       59, 62, 68, 69, 70, 71, 76, 77, 79, 81, 85, 87, 89, 90 )
11869
11870 #eqn15:
11871 #(1,1,1,1,5,1,7,7,5,0,0,0,0,0,0)
11872
11873
11874 # the 27 single points (of the surface) are:
11875 #6, 17, 81, 118, 146, 192, 229, 265, 332, 351, 362, 386, 412, 432, 488, 505, 531,
       544, 576, 666, 715, 717, 723, 739, 760, 762, 805
11876
11877
11878
11879
11880
11881 ##############################################################################
11882 # Section 8.6: Quartic Curve Activities
```

```
11883
11884
11885 SECTION_QUARTIC_CURVE_ACTIVITIES:
11886
11887
11888 test_8_6:
11889 ▷  make quartic_curves_q7
11890 ▷  make quartic_curves_q13
11891 ▷  make quartic_curves_q13_combine
11892 ▷  make quartic_curves_q13_classify
11893 ▷  make quartic_curves_q19
11894 ▷  make quartic_curves_q19_combine
11895 ▷  make quartic_curves_q19_classify
11896
11897
11898
11899
11900
11901 quartic_curves_q7:
11902 ▷  $(ORBITER) -v 3 \
11903 ▷  ▷  -define F -finite_field -q 7 -end \
11904 ▷  ▷  -define P -projective_space -n 3 -field F -end \
11905 ▷  ▷  -loop L 0 $(NB_CUBIC_SURFACES_Q7) 1 \
11906 ▷  ▷  ▷  -define S_%L -cubic_surface -space P -catalogue %L -end \
11907 ▷  ▷  -end_loop L \
11908 ▷  ▷  -print_symbols \
11909 ▷  ▷  -loop L 0 $(NB_CUBIC_SURFACES_Q7) 1 \
11910 ▷  ▷  ▷  -with S_%L -do \
11911 ▷  ▷  ▷  -cubic_surface_activity \
11912 ▷  ▷  ▷  ▷  -export_all_quartic_curves \
11913 ▷  ▷  ▷  -end \
11914 ▷  ▷  -end_loop L \
11915 ▷  ▷  -print_symbols
11916
11917 quartic_curves_q13:
11918 ▷  $(ORBITER) -v 3 \
11919 ▷  ▷  -define F -finite_field -q 13 -end \
11920 ▷  ▷  -define P -projective_space -n 3 -field F -end \
11921 ▷  ▷  -loop L 0 $(NB_CUBIC_SURFACES_Q13) 1 \
11922 ▷  ▷  ▷  -define S_%L -cubic_surface -space P -catalogue %L -end \
11923 ▷  ▷  -end_loop L \
11924 ▷  ▷  -print_symbols \
11925 ▷  ▷  -loop L 0 $(NB_CUBIC_SURFACES_Q13) 1 \
11926 ▷  ▷  ▷  -with S_%L -do \
11927 ▷  ▷  ▷  -cubic_surface_activity \
11928 ▷  ▷  ▷  ▷  -export_all_quartic_curves \
11929 ▷  ▷  ▷  -end \
11930 ▷  ▷  -end_loop L \
11931 ▷  ▷  -print_symbols
11932
11933 quartic_curves_q13_combine:
11934 ▷  $(ORBITER) -v 3 \
11935 ▷  ▷  -csv_file_concatenate_from_mask $(NB_CUBIC_SURFACES_Q13) \
11936 ▷  ▷  ▷  surface_catalogue_q13_iso%ld_quartics.csv \
11937 ▷  ▷  ▷  quartics_q13.csv
11938
11939
11940 quartic_curves_q13_classify:
11941 ▷  $(ORBITER) -v 30 \
```

```
11942 ▷  ▷  -define F -finite_field -q 13 -end \
11943 ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
11944 ▷  ▷  -define R -polynomial_ring \
11945 ▷  ▷  ▷  -field F \
11946 ▷  ▷  ▷  -number_of_variables 3 \
11947 ▷  ▷  ▷  -homogeneous_of_degree 4 \
11948 ▷  ▷  ▷  -monomial_ordering_partition \
11949 ▷  ▷  ▷  -variables "y0,y1,y2" "y_0,y_1,y_2" \
11950 ▷  ▷  -end \
11951 ▷  ▷  -define O -orbits -classification_by_canonical_form \
11952 ▷  ▷  ▷  -space P \
11953 ▷  ▷  ▷  -ring R \
11954 ▷  ▷  ▷  -input_fname_mask quartics_q13.csv \
11955 ▷  ▷  ▷  -nb_files 1 \
11956 ▷  ▷  ▷  -output_fname quartic_curves_q13_classified \
11957 ▷  ▷  ▷  -label_po_go "PO_GO" \
11958 ▷  ▷  ▷  -label_po_index "PO_INDEX" \
11959 ▷  ▷  ▷  -label_po "PO" \
11960 ▷  ▷  ▷  -label_so "orbit" \
11961 ▷  ▷  ▷  -label_equation "curve" \
11962 ▷  ▷  ▷  -label_points "pts_on_curve" \
11963 ▷  ▷  ▷  -label_lines "bitangents" \
11964 ▷  ▷  ▷  -carry_through "NB_E" \
11965 ▷  ▷  ▷  -carry_through "NB_DOUBLE" \
11966 ▷  ▷  ▷  -carry_through "NB_SINGLE" \
11967 ▷  ▷  ▷  -carry_through "NB_ZERO" \
11968 ▷  ▷  ▷  -algorithm_substructure \
11969 ▷  ▷  ▷  -substructure_size 4 \
11970 ▷  ▷  ▷  -end \
11971 ▷  ▷  -end \
11972 ▷  -with O -do -orbits_activity \
11973 ▷  ▷  -report \
11974 ▷  ▷  -report_options \
11975 ▷  ▷  ▷  -fname quartics_q13 \
11976 ▷  ▷  -end \
11977 ▷  -end
11978 ▷  pdflatex quartics_q13_orbits.tex
11979 ▷  $(OPEN) quartics_q13_orbits.pdf
11980
11981
11982
11983 quartic_curves_q19:
11984 ▷  $(ORBITER) -v 3 \
11985 ▷  ▷  -define F -finite_field -q 19 -end \
11986 ▷  ▷  -define P -projective_space -n 3 -field F -end \
11987 ▷  ▷  -loop L 0 $(NB_CUBIC_SURFACES_Q19) 1 \
11988 ▷  ▷  ▷  -define S_%L -cubic_surface -space P -catalogue %L -end \
11989 ▷  ▷  -end_loop L \
11990 ▷  ▷  -print_symbols \
11991 ▷  ▷  -loop L 0 $(NB_CUBIC_SURFACES_Q19) 1 \
11992 ▷  ▷  ▷  -with S_%L -do \
11993 ▷  ▷  ▷  -cubic_surface_activity \
11994 ▷  ▷  ▷  ▷  -export_all_quartic_curves \
11995 ▷  ▷  ▷  -end \
11996 ▷  ▷  -end_loop L \
11997 ▷  ▷  -print_symbols
11998
11999
12000 quartic_curves_q19_combine:
```

```
12001 ▷  $(ORBITER) -v 3 \
12002 ▷  ▷  -csv_file_concatenate_from_mask $(NB_CUBIC_SURFACES_Q19) \
12003 ▷  ▷  ▷  surface_catalogue_q19_iso%ld_quartics.csv \
12004 ▷  ▷  ▷  quartics_q19.csv
12005
12006
12007
12008 quartic_curves_q19_classify:
12009 ▷  $(ORBITER) -v 3 \
12010 ▷  ▷  -define F -finite_field -q 19 -end \
12011 ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
12012 ▷  ▷  -define R -polynomial_ring \
12013 ▷  ▷  ▷  -field F \
12014 ▷  ▷  ▷  -number_of_variables 3 \
12015 ▷  ▷  ▷  -homogeneous_of_degree 4 \
12016 ▷  ▷  ▷  -monomial_ordering_partition \
12017 ▷  ▷  ▷  -variables "y0,y1,y2" "y_0,y_1,y_2" \
12018 ▷  ▷  -end \
12019 ▷  ▷  -define O -orbits -classification_by_canonical_form \
12020 ▷  ▷  ▷  -space P \
12021 ▷  ▷  ▷  -ring R \
12022 ▷  ▷  ▷  -input_fname_mask quartics_q19.csv \
12023 ▷  ▷  ▷  -nb_files 1 \
12024 ▷  ▷  ▷  -output_fname quartic_curves_q19_classified \
12025 ▷  ▷  ▷  -label_po_go "PO_GO" \
12026 ▷  ▷  ▷  -label_po_index "PO_INDEX" \
12027 ▷  ▷  ▷  -label_po "PO" \
12028 ▷  ▷  ▷  -label_so "orbit" \
12029 ▷  ▷  ▷  -label_equation "curve" \
12030 ▷  ▷  ▷  -label_points "pts_on_curve" \
12031 ▷  ▷  ▷  -label_lines "bitangents" \
12032 ▷  ▷  ▷  -carry_through "NB_E" \
12033 ▷  ▷  ▷  -carry_through "NB_DOUBLE" \
12034 ▷  ▷  ▷  -carry_through "NB_SINGLE" \
12035 ▷  ▷  ▷  -carry_through "NB_ZERO" \
12036 ▷  ▷  ▷  -algorithm_substructure \
12037 ▷  ▷  ▷  -substructure_size 4 \
12038 ▷  ▷  ▷  -end \
12039 ▷  ▷  -end \
12040 ▷  -with O -do -orbits_activity \
12041 ▷  ▷  -report \
12042 ▷  ▷  -report_options \
12043 ▷  ▷  ▷  -fname quartics_q19 \
12044 ▷  ▷  -end \
12045 ▷  -end
12046 ▷  pdflatex quartics_q19_orbits.tex
12047 ▷  $(OPEN) quartics_q19_orbits.pdf
12048
12049
12050
12051
12052
12053
12054 ################################################################################
12055 # Section 8.7: Interfaces
12056
12057 SECTION_CUBIC_SURFACES_INTERFACES:
12058
12059
```

```
12060 test_8_7:
12061 ▷  make surface_4_0_export_gap
12062
12063 surface_4_0_export_gap:
12064 ▷  $(ORBITER) -v 3 \
12065 ▷  ▷  -define F -finite_field -q 4 -end \
12066 ▷  ▷  -define P -projective_space -n 3 -field F -end \
12067 ▷  ▷  -define S -cubic_surface -space P -catalogue 0 -end \
12068 ▷  ▷  -with S -do \
12069 ▷  ▷  -cubic_surface_activity \
12070 ▷  ▷  ▷  -export_gap \
12071 ▷  ▷  -end
12072
12073
12074
12075
12076 ##############################################################################
12077 # Chapter 9 - Applications
12078 ##############################################################################
12079
12080
12081 test_9:
12082 ▷  make test_9_1
12083 ▷  make test_9_2
12084 ▷  make test_9_3
12085
12086
12087
12088 ##############################################################################
12089 # Section 9.1: Number Theory
12090
12091
12092 SECTION_NUMBER_THEORY:
12093
12094
12095 test_9_1:
12096
12097
12098
12099
12100
12101
12102
12103 ##############################################################################
12104 # Section 9.2: Representation Theory
12105
12106 SECTION_REPRESENTATION_THEORY:
12107
12108
12109 test_9_2:
12110 ▷  make representation_on_polynomials_of_degree_3
12111 ▷  make representation_tetrahedral_group_on_polynomials_of_degree_3
12112
12113
12114 representation_on_polynomials_of_degree_3:
12115 ▷  $(ORBITER) -v 4 \
12116 ▷  ▷  -define F -finite_field -q 3 -end \
12117 ▷  ▷  -define G -linear_group -PGL 4 F -end \
12118 ▷  ▷  -define R -polynomial_ring \
```

```
12119 ▷  ▷  ▷  -field F \
12120 ▷  ▷  ▷  -number_of_variables 4 \
12121 ▷  ▷  ▷  -homogeneous_of_degree 3 \
12122 ▷  ▷  ▷  -monomial_ordering_partition \
12123 ▷  ▷  ▷  -variables "X,Y,Z,W" "X,Y,Z,W" \
12124 ▷  ▷  -end \
12125 ▷  ▷  -with G -do \
12126 ▷  ▷  -group_theoretic_activity \
12127 ▷  ▷  ▷  -representation_on_polynomials R \
12128 ▷  ▷  -end
12129 ▷  $(ORBITER) -v 2 \
12130 ▷  ▷  -loop L 0 9 1 -draw_matrix \
12131 ▷  ▷  ▷  -input_csv_file PGL_4_3_rep_3_%L.csv \
12132 ▷  ▷  ▷  -box_width 40 -bit_depth 24 -partition 3 20 20 -end \
12133 ▷  ▷  -end_loop L
12134
12135
12136
12137
12138 representation_tetrahedral_group_on_polynomials_of_degree_3:
12139 ▷  $(ORBITER) -v 4 \
12140 ▷  ▷  -define F -finite_field -q 3 -end \
12141 ▷  ▷  -define G -linear_group -GL 3 F \
12142 ▷  ▷  -subgroup_by_generators "tetra" "12" 2 \
12143 ▷  ▷  "0,1,0,0,0,1,1,0,0, 0,0,1,2,0,0,0,2,0" \
12144 ▷  ▷  -end \
12145 ▷  ▷  -define R -polynomial_ring \
12146 ▷  ▷  ▷  -field F \
12147 ▷  ▷  ▷  -number_of_variables 3 \
12148 ▷  ▷  ▷  -homogeneous_of_degree 3 \
12149 ▷  ▷  ▷  -monomial_ordering_partition \
12150 ▷  ▷  ▷  -variables "X,Y,Z" "X,Y,Z" \
12151 ▷  ▷  -end \
12152 ▷  ▷  -with G -do \
12153 ▷  ▷  -group_theoretic_activity \
12154 ▷  ▷  ▷  -representation_on_polynomials R \
12155 ▷  ▷  -end
12156 ▷  $(ORBITER) -v 2 -loop L 0 2 1 -draw_matrix \
12157 ▷  ▷  -input_csv_file GL_3_3_Subgroup_tetra_12_rep_3_%L.csv \
12158 ▷  ▷  -box_width 40 -bit_depth 24 -partition 3 10 10 -end -end_loop L
12159 ▷  $(OPEN) GL_3_3_Subgroup_tetra_12_rep_3_0_draw.bmp
12160 ▷  $(OPEN) GL_3_3_Subgroup_tetra_12_rep_3_1_draw.bmp
12161
12162 # write GL_3_3_Subgroup_tetra_12_rep_3_0.csv
12163
12164
12165 ##############################################################################
12166 # Section 9.3: Cryptography
12167
12168
12169 SECTION_CRYPTOGRAPHY:
12170
12171
12172
12173 test_9_3:
12174 ▷  make EC_add
12175 ▷  make EC_cyclic_subgroup
12176 ▷  make EC_points_13
12177 ▷  make EC_points_199
```

```
12178 ▷  make EC_Koblitz_encoding
12179 ▷  make EC_bsgs
12180 ▷  make EC_bsgs_decode
12181 ▷  make NTRU_Alice1
12182 ▷  make NTRU_Alice2
12183 ▷  make NTRU_Alice_public_key
12184 ▷  make NTRU_encrypt
12185 ▷  make NTRU_decrypt1
12186 ▷  make NTRU_decrypt2
12187 ▷  make NTRU_decrypt3
12188 ▷  make NTRU_decrypt4
12189 ▷  make NTRU_decrypt5
12190 ▷  make inv_59_mod
12191 ▷  make RSA_e
12192 ▷  make RSA_d
12193 ▷  make im1
12194 ▷  make RSA_e1
12195 ▷  make RSA_d1
12196 ▷  make im1061
12197 ▷  make RSA_e2
12198 ▷  make RSA_d2
12199 ▷  make im3
12200 ▷  make RSA_e3
12201 ▷  make RSA_d3
12202 ▷  make im4
12203 ▷  make RSA_e4
12204 ▷  make RSA_d4
12205 ▷  make im5
12206 ▷  make RSA_e5
12207 ▷  make RSA_d5
12208 ▷  make RSA_d6
12209 ▷  make smooth
12210 ▷  make im7
12211 ▷  make RSA_e7
12212 ▷  make im8
12213 ▷  make RSA_e8
12214 ▷  make sqrt_big
12215 ▷  make sqrt_mod_33_41
12216 ▷  #make quadratic_sieve
12217 ▷  make pseudoprime3
12218 ▷  make pseudoprime10
12219 ▷  make PR10_test1
12220 ▷  make pseudoprime11
12221 ▷  make pseudoprime20
12222 ▷  make PR10
12223 ▷  make pseudoprime50
12224 ▷  make pseudoprime51
12225 ▷  make pseudoprime30
12226 ▷  make pseudoprime31
12227 ▷  make pseudoprime33
12228 ▷  make pseudoprime34
12229 ▷  make pseudoprime35
12230 ▷  make pseudoprime36
12231 ▷  #make MATH360_hw2
12232 ▷  make F_256_Rijndahl
12233 ▷  make all_square_roots_mod_n_1549411
12234 ▷  make power_mod_211
12235
12236
```

```
12237
12238 EC_add:
12239 ▷  $(ORBITER) -v 2 \
12240 ▷  ▷  -define F -finite_field -q 11 -end \
12241 ▷  ▷  -with F -do \
12242 ▷  ▷  -finite_field_activity \
12243 ▷  ▷  -EC_add 1 3 "1,4" "1,4" -end
12244
12245 EC_cyclic_subgroup:
12246 ▷  $(ORBITER) -v 2 \
12247 ▷  ▷  -define F -finite_field -q 11 -end \
12248 ▷  ▷  -with F -do \
12249 ▷  ▷  -finite_field_activity \
12250 ▷  ▷  -EC_cyclic_subgroup 1 3 "1,4" -end
12251
12252
12253 EC_points_13:
12254 ▷  $(ORBITER) -v 2 \
12255 ▷  ▷  -define F -finite_field -q 13 -end \
12256 ▷  ▷  -with F -do \
12257 ▷  ▷  -finite_field_activity \
12258 ▷  ▷  -EC_points "EC_2_5_q13" 2 5 -end
12259 ▷  $(ORBITER) -v 2 -draw_matrix \
12260 ▷  ▷  -input_csv_file EC_2_5_q13_points_xy.csv \
12261 ▷  ▷  -box_width 20 -bit_depth 24 \
12262 ▷  ▷  -partition 2 "1,1,1,1,1,1,1,1,1,1,1,1,1" "1,1,1,1,1,1,1,1,1,1,1,1,1" -end
12263
12264
12265
12266
12267 EC_points_199:
12268 ▷  $(ORBITER) -v 2 \
12269 ▷  ▷  -define F -finite_field -q 199 -end \
12270 ▷  ▷  -with F -do \
12271 ▷  ▷  -finite_field_activity \
12272 ▷  ▷  -EC_points "EC_5_7_q199" 5 7 -end
12273 ▷  $(ORBITER) -v 2 \
12274 ▷  ▷  -draw_matrix -input_csv_file EC_5_7_q199_points_xy.csv \
12275 ▷  ▷  -box_width 10 -bit_depth 24 \
12276 ▷  ▷  -partition 2 199 199 -end
12277
12278 EC_Koblitz_encoding:
12279 ▷  $(ORBITER) -v 6 -seed 17 \
12280 ▷  ▷  -define F -finite_field -q 199 -end \
12281 ▷  ▷  -with F -do \
12282 ▷  ▷  -finite_field_activity \
12283 ▷  ▷  -EC_Koblitz_encoding 5 7 67 "147,164" "DEADBEEF" \
12284 ▷  ▷  -end
12285
12286 EC_bsgs:
12287 ▷  $(ORBITER) -v 2 \
12288 ▷  ▷  -define F -finite_field -q 199 -end \
12289 ▷  ▷  -with F -do \
12290 ▷  ▷  -finite_field_activity \
12291 ▷  ▷  -EC_bsgs 5 7 "147,164" 212 \
12292 ▷  ▷  "172,158,45,195,50,22,10,103,55,33,50,22,\
12293 ▷  ▷  145,105,31,74,73,155,67,60,25,6" \
12294 ▷  ▷  -end
12295
```

```
12296 EC_bsgs_decode:
12297 ▷  $(ORBITER) -v 2 \
12298 ▷  ▷  -define F -finite_field -q 199 -end \
12299 ▷  ▷  -with F -do \
12300 ▷  ▷  -finite_field_activity \
12301 ▷  ▷  -EC_bsgs_decode 5 7 "129,176" 212 \
12302 ▷  ▷  "127,188,51,141,85,29,106,90,41,105,179,71,\
12303 ▷  ▷  171,2,16,197,183,72,27,129,37,10" \
12304 ▷  ▷  "50,179,169,13,153,169,115,116,188,110,176" \
12305 ▷  ▷  -end
12306
12307
12308
12309 NTRU_Alice1:
12310 ▷  $(ORBITER) -v 2 \
12311 ▷  ▷  -define F -finite_field -q $(NTRU_Q) -end \
12312 ▷  ▷  -with F -do \
12313 ▷  ▷  -finite_field_activity \
12314 ▷  ▷  -extended_gcd_for_polynomials \
12315 ▷  ▷  ▷  $(NTRU_XN1) $(ALICE_PRIVATE_F) \
12316 ▷  ▷  -end
12317
12318
12319
12320
12321 NTRU_Alice2:
12322 ▷  $(ORBITER) -v 2 \
12323 ▷  ▷  -define F -finite_field -q $(NTRU_P) -end \
12324 ▷  ▷  -with F -do \
12325 ▷  ▷  -finite_field_activity \
12326 ▷  ▷  -extended_gcd_for_polynomials \
12327 ▷  ▷  ▷  $(NTRU_XN1) $(ALICE_PRIVATE_F) \
12328 ▷  ▷  -end
12329
12330
12331 NTRU_Alice_public_key:
12332 ▷  $(ORBITER) -v 2 \
12333 ▷  ▷  -define F -finite_field -q $(NTRU_Q) -end \
12334 ▷  ▷  -with F -do \
12335 ▷  ▷  -finite_field_activity \
12336 ▷  ▷  -polynomial_mult_mod $(ALICE_PRIVATE_F) \
12337 ▷  ▷  ▷  $(ALICE_PRIVATE_G) $(NTRU_XN1) \
12338 ▷  ▷  -end
12339
12340
12341 NTRU_encrypt:
12342 ▷  $(ORBITER) -v 2 \
12343 ▷  ▷  -define F -finite_field -q $(NTRU_Q) -end \
12344 ▷  ▷  -with F -do \
12345 ▷  ▷  -finite_field_activity \
12346 ▷  ▷  -NTRU_encrypt \
12347 ▷  ▷  $(NTRU_N) $(NTRU_P) $(ALICE_PUBLIC_KEY) \
12348 ▷  ▷  $(BOB_ONE_TIME_KEY) $(BOB_MESSAGE) \
12349 ▷  ▷  -end
12350
12351
12352
12353 NTRU_decrypt1:
12354 ▷  $(ORBITER) -v 2 \
```

```
12355 ▷  ▷  -define F -finite_field -q $(NTRU_Q) -end \
12356 ▷  ▷  -with F -do \
12357 ▷  ▷  -finite_field_activity \
12358 ▷  ▷  -polynomial_mult_mod $(ALICE_PRIVATE_F) \
12359 ▷  ▷  ▷  $(BOB_ENCRYPT) $(NTRU_XN1) \
12360 ▷  ▷  -end
12361
12362
12363
12364 NTRU_decrypt2:
12365 ▷  $(ORBITER) -v 2 \
12366 ▷  ▷  -define F -finite_field -q $(NTRU_Q) -end \
12367 ▷  ▷  -with F -do \
12368 ▷  ▷  -finite_field_activity \
12369 ▷  ▷  -polynomial_center_lift $(ALICE_C1) -end
12370
12371 NTRU_decrypt3:
12372 ▷  $(ORBITER) -v 2 \
12373 ▷  ▷  -define F -finite_field -q $(NTRU_P) -end \
12374 ▷  ▷  -with F -do \
12375 ▷  ▷  -finite_field_activity \
12376 ▷  ▷  -polynomial_reduce_mod_p $(ALICE_C2) -end
12377
12378 NTRU_decrypt4:
12379 ▷  $(ORBITER) -v 2 \
12380 ▷  ▷  -define F -finite_field -q $(NTRU_Q) -end \
12381 ▷  ▷  -with F -do \
12382 ▷  ▷  -finite_field_activity \
12383 ▷  ▷  -polynomial_mult_mod $(ALICE_PRIVATE_FP) \
12384 ▷  ▷  ▷  $(ALICE_C3) $(NTRU_XN1) \
12385 ▷  ▷  -end
12386
12387
12388
12389 NTRU_decrypt5:
12390 ▷  $(ORBITER) -v 2 \
12391 ▷  ▷  -define F -finite_field -q $(NTRU_P) -end \
12392 ▷  ▷  -with F -do \
12393 ▷  ▷  -finite_field_activity \
12394 ▷  ▷  -polynomial_center_lift $(ALICE_C4) -end
12395
12396 #A(X)= - X^{5} + X^{3} + X^{2} - X + 1
12397 #plaintext BOB_MESSAGE
12398
12399
12400
12401 ######
12402
12403
12404 inv_59_mod:
12405 ▷  $(ORBITER) -v 2 -inverse_mod 59 10200
12406
12407 # the inverse of 59 mod 10200 is 2939
12408
12409
12410
12411 RSA_e:
12412 ▷  $(ORBITER) -v 2 \
12413 ▷  ▷  -RSA 59 10403 2 "1921,1605,1804,2116,0518"
```

```
12414
12415
12416 RSA_d:
12417 ▷  $(ORBITER) -v 2 \
12418 ▷  ▷  -RSA 2939 10403 2 "902,3509,9833,3548,5181"
12419
12420
12421 im1:
12422 ▷  $(ORBITER) -v 2 -inverse_mod 869 1843488
12423
12424 #the inverse of 869 mod 1843488 is 386093
12425
12426
12427 # FUNFACTOR:
12428
12429 RSA_e1:
12430 ▷  $(ORBITER) -v 2 \
12431 ▷  ▷  -RSA 386093 1846303 3 "62114,60103,201518"
12432
12433
12434 RSA_d1:
12435 ▷  $(ORBITER) -v 2 \
12436 ▷  ▷  -RSA 869 1846303 3 "1248407,345776,317846"
12437
12438
12439
12440
12441
12442 ##############################
12443 # 5503*4603 = 25330309
12444 # 5502*4602 = 25320204
12445
12446
12447 im1061:
12448 ▷  $(ORBITER) -v 2 \
12449 ▷  ▷  -inverse_mod 1061 25320204
12450 ▷
12451 # the inverse of 1061 mod 25320204 is 2076209
12452
12453
12454
12455 RSA_e2:
12456 ▷  $(ORBITER) -v 2 \
12457 ▷  ▷  -RSA_encrypt_text 2076209 25330309 3 creamcheese
12458
12459 #-RSA_encrypt_text 386093 1846303 creamcheese
12460 #408918,1735142,239809,654636
12461
12462
12463 RSA_d2:
12464 ▷  $(ORBITER) -v 2 \
12465 ▷  ▷  -RSA 1061 25330309 3 "19019931,1619805,740498,2671344"
12466
12467
12468 ##############################
12469 # 7253*8171 = 59264263
12470 # 7252*8170 = 59248840
12471
12472
```

```
12473 im3:
12474 ▷  $(ORBITER) -v 2 \
12475 ▷  ▷  -inverse_mod 2909 59248840
12476 ▷
12477 #the inverse of 2909 mod 59248840 is 4358629
12478
12479 RSA_e3:
12480 ▷  $(ORBITER) -v 2 \
12481 ▷  ▷  -RSA_encrypt_text 2909 59264263 3 encrypted
12482
12483 RSA_d3:
12484 ▷  $(ORBITER) -v 2 \
12485 ▷  ▷  -RSA 4358629 59264263 3 "35270141,9642524,49091707"
12486
12487 #51403,182516,200504 = encrypted
12488
12489
12490 ####
12491 # 7879 * 7901 = 62251979
12492 # 7878 * 7900 = 62236200
12493
12494 # e =
12495
12496 im4:
12497 ▷  $(ORBITER) -v 2 -inverse_mod 583 62236200
12498
12499 # the inverse of 583 mod 62236200 is 32559247
12500
12501 RSA_e4:
12502 ▷  $(ORBITER) -v 2 \
12503 ▷  ▷  -RSA_encrypt_text 583 62251979 3 venividivici
12504
12505 #-RSA_encrypt_text 583 62251979 venividivici
12506 #40513610,53979973,56449676,35068535
12507
12508 RSA_d4:
12509 ▷  $(ORBITER) -v 2 \
12510 ▷  ▷  -RSA 32559247 62251979 "40513610,53979973,56449676,35068535"
12511
12512
12513
12514 ######
12515 # 7369 * 7127 = 52518863
12516 # 7368 * 7126 = 52504368
12517
12518
12519 im5:
12520 ▷  $(ORBITER) -v 2 -inverse_mod 173 52504368
12521
12522 #the inverse of 173 mod 52504368 is 38543669
12523 ▷
12524 RSA_e5:
12525 ▷  $(ORBITER) -v 2 \
12526 ▷  ▷  -RSA_encrypt_text 38543669 52518863 3 fascinating
12527
12528 #-RSA_encrypt_text 38543669 52518863 fascinating
12529 #31526751,8962078,51045732,51894467
12530 ▷  ▷
12531
```

```
12532 RSA_d5:
12533 ▷  $(ORBITER) -v 2 \
12534 ▷  ▷  -RSA 173 52518863 "31526751,8962078,51045732,51894467"
12535
12536
12537 RSA_d6:
12538 ▷  $(ORBITER) -v 2 \
12539 ▷  ▷  -RSA 47177497 55040413 "28702119,48926559"
12540
12541
12542 smooth:
12543 ▷  $(ORBITER) -v 2 \
12544 ▷  ▷  -sift_smooth 100000 100 "2,3,5,7,11,13,17,19"
12545
12546
12547 ▷  ▷
12548 ######
12549 # 1999 * 7907 = 15806093
12550 # 1998 * 7906 = 15796188
12551
12552 im7:
12553 ▷  $(ORBITER) -v 2 -inverse_mod 3221 15796188
12554
12555 #the inverse of 3221 mod 15796188 is 10048553
12556
12557
12558 RSA_e7:
12559 ▷  $(ORBITER) -v 2 \
12560 ▷  ▷  -RSA_encrypt_text 10048553 15806093 3 beachandfun
12561 ▷  ▷
12562 #
12563 # 7853 * 7673 = 60256069
12564 # 7852 * 7672 = 60240544
12565
12566 im8:
12567 ▷  $(ORBITER) -v 2 -inverse_mod 9017 60240544
12568
12569
12570 #the inverse of 9017 mod 60240544 is 14430473
12571
12572 RSA_e8:
12573 ▷  $(ORBITER) -v 2 \
12574 ▷  ▷  -RSA_encrypt_text 9017 60256069 3 strawberry
12575
12576
12577 sqrt_big:
12578 ▷  $(ORBITER) -v 2 -square_root 1002001
12579 ▷
12580 sqrt_mod_33_41:
12581 ▷  $(ORBITER) -v 2 -square_root_mod 33 41
12582 ▷
12583 quadratic_sieve:
12584 ▷  $(ORBITER) -v 5 -quadratic_sieve 31 500 1
12585
12586
12587
12588 ######
12589
12590 pseudoprime3:
```

```
12591 ▷  $(ORBITER) -v 5 \
12592 ▷  ▷  -seed 2531011 -find_pseudoprime 3 5 0 0
12593 ▷  pdflatex pseudoprime_3.tex
12594 ▷  $(OPEN) pseudoprime_3.pdf
12595
12596 pseudoprime10:
12597 ▷  $(ORBITER) -v 5 \
12598 ▷  ▷  -seed 2531011 -find_pseudoprime 10 5 5 5
12599 ▷  pdflatex pseudoprime_10.tex
12600 ▷  $(OPEN) pseudoprime_10.pdf
12601
12602 # 4460190157
12603
12604
12605 PR10_test1:
12606 ▷  $(ORBITER) -v 5 -power_mod 1293 2230095078 4460190157
12607 ▷  $(ORBITER) -v 5 -power_mod 9865 2230095078 4460190157
12608 ▷  $(ORBITER) -v 5 -power_mod 19645 2230095078 4460190157
12609 ▷  $(ORBITER) -v 5 -power_mod 974586571 2230095078 4460190157
12610 ▷  $(ORBITER) -v 5 -power_mod 974586571 1486730052 4460190157
12611 ▷  $(ORBITER) -v 5 -power_mod 974586571 15222492 4460190157
12612 ▷  $(ORBITER) -v 5 -power_mod 974586571 284796 4460190157
12613
12614
12615
12616 pseudoprime11:
12617 ▷  $(ORBITER) -v 5 \
12618 ▷  ▷  -seed 2531011 -find_pseudoprime 11 5 5 5
12619 ▷  pdflatex pseudoprime_11.tex
12620 ▷  $(OPEN) pseudoprime_11.pdf
12621
12622 # 63814633367
12623
12624 # product is 284625399616057168619
12625
12626
12627 pseudoprime20:
12628 ▷  $(ORBITER) -v 5 \
12629 ▷  ▷  -seed 2531011 -find_pseudoprime 20 5 5 5
12630 ▷  pdflatex pseudoprime_20.tex
12631 ▷  $(OPEN) pseudoprime_20.pdf
12632
12633
12634 PR10:
12635 ▷  $(ORBITER) -v 5 -primitive_root 4460190157
12636
12637
12638 # mistake! long integer overflow
12639 # a primitive root modulo 165222861 is 1293
12640
12641
12642
12643
12644 pseudoprime50:
12645 ▷  $(ORBITER) -v 5 \
12646 ▷  ▷  -seed 2531011 -find_pseudoprime 50 5 0 0
12647 ▷  pdflatex pseudoprime_50.tex
12648 ▷  $(OPEN) pseudoprime_50.pdf
12649
```

```
12650
12651 #913227928785812181814313921709869262623366883544 73
12652
12653 pseudoprime51:
12654 ▷  $(ORBITER) -v 5 \
12655 ▷  ▷  -seed 2531011 -find_pseudoprime 51 5 5 5
12656 ▷  pdflatex pseudoprime_51.tex
12657 ▷  $(OPEN) pseudoprime_51.pdf
12658
12659 #754600727746834470214089702490004944659715367045417
12660
12661 # product 689122459660508196061999944232643157323352953244006584366617444032440495729140943799043266615861002 41
12662
12663 pseudoprime30:
12664 ▷  $(ORBITER) -v 5 \
12665 ▷  ▷  -seed 2531011 -find_pseudoprime 30 5 5 5
12666 ▷  pdflatex pseudoprime_30.tex
12667 ▷  $(OPEN) pseudoprime_30.pdf
12668
12669 # 2865255654745045169145955963 87
12670
12671 pseudoprime31:
12672 ▷  $(ORBITER) -v 5 \
12673 ▷  ▷  -seed 2531011 -find_pseudoprime 31 5 5 5
12674 ▷  pdflatex pseudoprime_31.tex
12675 ▷  $(OPEN) pseudoprime_31.pdf
12676 #877726676542264552372412985333 1
12677
12678 # 2514911323283298698837184692002835573476743643265896783515097
12679
12680 # maybe 2 seconds
12681
12682
12683 pseudoprime33:
12684 ▷  $(ORBITER) -v 5 \
12685 ▷  ▷  -seed 2531011 -find_pseudoprime 33 5 5 5
12686 ▷  pdflatex pseudoprime_33.tex
12687 ▷  $(OPEN) pseudoprime_33.pdf
12688
12689
12690 #3716741994982953455433630044598 91
12691
12692
12693 pseudoprime34:
12694 ▷  $(ORBITER) -v 5 \
12695 ▷  ▷  -seed 2531011 -find_pseudoprime 34 5 5 5
12696 ▷  pdflatex pseudoprime_34.tex
12697 ▷  $(OPEN) pseudoprime_34.pdf
12698
12699 #9309708224110488378214945245346817
12700
12701 # 346017835175896253191287297973187452884914223861967789078606101694 7
12702 # 18 sec
12703
12704
12705
12706 pseudoprime35:
12707 ▷  $(ORBITER) -v 5 \
```

```
12708 ▷  ▷  -seed 2531011 -find_pseudoprime 35 5 5 5
12709 ▷  pdflatex pseudoprime_35.tex
12710 ▷  $(OPEN) pseudoprime_35.pdf
12711
12712 #81329557792505271120435930267680203
12713
12714 pseudoprime36:
12715 ▷  $(ORBITER) -v 5 \
12716 ▷  ▷  -seed 2531011 -find_pseudoprime 36 5 5 5
12717 ▷  pdflatex pseudoprime_36.tex
12718 ▷  $(OPEN) pseudoprime_36.pdf
12719
12720 #162624680891993404333363207561599139
12721
12722 # 1322619338309310524253791935022035413521944132364163666548426253214 5217
12723 # factoring takes 46 seconds
12724
12725
12726 #ToDo:
12727
12728 MATH360_hw2:
12729 ▷  $(ORBITER) -v 3 \
12730 ▷  ▷  -define F -finite_field -q 16 -end \
12731 ▷  ▷  -with F -do -finite_field_activity \
12732 ▷  ▷  -parse_and_evaluate "test" "" "a+b" "a=8,b=14" -end
12733 ▷  $(ORBITER) -v 3 \
12734 ▷  ▷  -define F -finite_field -q 16 -end \
12735 ▷  ▷  -with F -do -finite_field_activity \
12736 ▷  ▷  -parse_and_evaluate "test" "" "a*b" "a=9,b=13" -end
12737 ▷  $(ORBITER) -v 3 \
12738 ▷  ▷  -define F -finite_field -q 16 -end \
12739 ▷  ▷  -with F -do -finite_field_activity \
12740 ▷  ▷  -parse_and_evaluate "test" "" "a*a*a*a*a" "a=9" -end
12741 ▷  $(ORBITER) -v 3 \
12742 ▷  ▷  -define F -finite_field -q 16 -end \
12743 ▷  ▷  -with F -do -finite_field_activity \
12744 ▷  ▷  -parse_and_evaluate "test" "" "(a+b)*(a+b)" "a=5,b=7" -end
12745 ▷  $(ORBITER) -v 3 \
12746 ▷  ▷  -define F -finite_field -q 16 -end \
12747 ▷  ▷  -with F -do -finite_field_activity \
12748 ▷  ▷  -parse_and_evaluate "test" "" "a*a+b*b" "a=5,b=7" -end
12749
12750
12751 F_256_Rijndahl:
12752 ▷  $(ORBITER) -v 3 \
12753 ▷  ▷  -define F -finite_field -q 256 -override_polynomial 283 -end \
12754 ▷  ▷  -with F -do -finite_field_activity -cheat_sheet_GF -end
12755
12756
12757
12758
12759
12760
12761
12762 all_square_roots_mod_n_1549411:
12763 ▷  $(ORBITER) -v 3 -all_square_roots_mod_n 1075922 1549411
12764 ▷
12765
12766
```

```
12767 power_mod_211:
12768 ▷  $(ORBITER) -v 3 -power_function_mod_n 2 211
12769 ▷  $(ORBITER) -v 3 \
12770 ▷  ▷  -plot_function power_function_k2_n211.csv
12771 ▷  $(ORBITER) -v 2 -draw_matrix \
12772 ▷  ▷  -input_csv_file power_function_k2_n211_graph.csv \
12773 ▷  ▷  -box_width 10 -bit_depth 8 \
12774 ▷  ▷  -partition 3 211 211 -end
12775 ▷  $(OPEN) power_function_k2_n211_graph_draw.bmp
12776
12777
12778
12779 ################################################################################
12780 # Chapter 10 - Coding Theory
12781 ################################################################################
12782
12783
12784 test_10:
12785 ▷  make test_10_1
12786 ▷  make test_10_2
12787 ▷  make test_10_3
12788 ▷  make test_10_4
12789 ▷  make test_10_5
12790 ▷  make test_10_6
12791 ▷  make test_10_7
12792 ▷  make test_10_8
12793 ▷  make test_10_9
12794 ▷  make test_10_10
12795
12796
12797
12798
12799 ################################################################################
12800 # Section 10.1: Coding Theory
12801
12802 SECTION_CODING_THEORY_INTRODUCTION:
12803
12804 test_10_1:
12805 ▷  make Allen_Gates_noise_1_percent
12806 ▷  make Hamming_space_4_2_distance_matrix
12807 ▷  make Hamming_space_4_2_distance_matrix_draw
12808 ▷  make Hamming_code_macwilliams
12809 ▷  make code_5_2_3_diagram
12810 ▷  make Hamming_5_2_graph
12811 ▷  make Hamming_5_2_with_5_2_3_code
12812 ▷  make Sylvester_Hadamard_code_8
12813
12814
12815 Allen_Gates_noise_1_percent:
12816 ▷  cp $(MY_PATH)/examples/users_guide/allen_Gates.bmp .
12817 ▷  $(ORBITER) -v 3 \
12818 ▷  ▷  -random_noise_in_bitmap_file \
12819 ▷  ▷  allen_Gates.bmp \
12820 ▷  ▷  allen_Gates_1.bmp \
12821 ▷  ▷  1 100
12822 ▷  $(OPEN) allen_Gates_1.bmp
12823
12824
12825 Hamming_space_4_2_distance_matrix:
```

```
12826 ▷  $(ORBITER) -v 2 \
12827 ▷  ▷  -define F -finite_field -q 2 -end \
12828 ▷  ▷  -with F -do -coding_theoretic_activity \
12829 ▷  ▷  ▷  -Hamming_space_distance_matrix 4 \
12830 ▷  ▷  -end
12831
12832
12833 Hamming_space_4_2_distance_matrix_draw:
12834 ▷  $(ORBITER) -v 2 \
12835 ▷  ▷  -draw_matrix \
12836 ▷  ▷  ▷  -input_csv_file Hamming_n4_q2.csv \
12837 ▷  ▷  ▷  -box_width 20 -bit_depth 24 \
12838 ▷  ▷  ▷  -partition 4 16 16 \
12839 ▷  ▷  -end
12840 ▷  $(OPEN) Hamming_n4_q2_draw.bmp
12841
12842
12843
12844 Hamming_code_macwilliams:
12845 ▷  $(ORBITER) -v 2 \
12846 ▷  ▷  -make_macwilliams_system 7 4 2
12847 ▷  pdflatex MacWilliams_n7_k4_q2.tex
12848 ▷  $(OPEN) MacWilliams_n7_k4_q2.pdf
12849
12850
12851
12852
12853
12854 code_5_2_3_diagram:
12855 ▷  $(ORBITER) -v 2 \
12856 ▷  ▷  -define F -finite_field -q 2 -end \
12857 ▷  ▷  -with F -do -coding_theoretic_activity \
12858 ▷  ▷  ▷  -general_code_binary 5 "code_5_2_3" \
12859 ▷  ▷  ▷  $(CODE_5_2_3_CODEWORDS) \
12860 ▷  ▷  ▷  -metric_balls 1 \
12861 ▷  ▷  -end
12862 ▷  $(ORBITER) -v 2 \
12863 ▷  ▷  -draw_matrix \
12864 ▷  ▷  ▷  -input_csv_file code_5_2_3_char_func_5_4.csv \
12865 ▷  ▷  ▷  -box_width 25 -bit_depth 24 \
12866 ▷  ▷  ▷  -partition 4 8 4 \
12867 ▷  ▷  -end
12868
12869
12870
12871 Hamming_5_2_graph:
12872 ▷  $(ORBITER) -v 2 \
12873 ▷  ▷  -define G -graph -Hamming 5 2 -end \
12874 ▷  ▷  -with G -do \
12875 ▷  ▷  -graph_theoretic_activity -export_csv -end \
12876 ▷  ▷  -with G -do \
12877 ▷  ▷  -graph_theoretic_activity -export_graphviz -end \
12878 ▷  ▷  -with G -do \
12879 ▷  ▷  -graph_theoretic_activity -save -end
12880 ▷  $(ORBITER) -v 2 -draw_matrix \
12881 ▷  ▷  -input_csv_file Hamming_5_2.csv \
12882 ▷  ▷  -box_width 8 -bit_depth 24 -partition 4 32 32 -end
12883 ▷  dot -Tpng Hamming_5_2.gv >Hamming_5_2.png
12884
```

```
12885
12886 Hamming_5_2_with_5_2_3_code:
12887 ▷  $(ORBITER) -v 2 \
12888 ▷  ▷  -define G -graph -Hamming 5 2 \
12889 ▷  ▷  ▷  -subset "_code_5_2_3" "\\_code\\_5\\_2\\_3" \
12890 ▷  ▷  ▷  $(CODE_5_2_3_CODEWORDS) -end \
12891 ▷  ▷  -with G -do \
12892 ▷  ▷  -graph_theoretic_activity -export_csv -end \
12893 ▷  ▷  -with G -do \
12894 ▷  ▷  -graph_theoretic_activity -export_graphviz -end \
12895 ▷  ▷  -with G -do \
12896 ▷  ▷  -graph_theoretic_activity -save -end \
12897 ▷  ▷  -with G -do \
12898 ▷  ▷  -graph_theoretic_activity -automorphism_group -end
12899 ▷  pdflatex Hamming_5_2_code_5_2_3_report.tex
12900 ▷  $(OPEN) Hamming_5_2_code_5_2_3_report.pdf
12901
12902 # group has order 32
12903
12904
12905 # n must be a power of 2, and at least 4:
12906 # minus one is represented as 2.
12907
12908 Sylvester_Hadamard_code_8:
12909 ▷  $(ORBITER) -v 2 \
12910 ▷  ▷  -define F -finite_field -q 3 -end \
12911 ▷  ▷  -with F -do -coding_theoretic_activity \
12912 ▷  ▷  ▷  -Sylvester_Hadamard_code 8 \
12913 ▷  ▷  -end
12914
12915
12916
12917
12918 ################################################################################
12919 # Section 10.2: Linear codes
12920
12921 SECTION_CODING_THEORY_LINEAR_CODES:
12922
12923
12924
12925
12926 test_10_2:
12927 ▷  make RM_3_1
12928 ▷  make code_6
12929 ▷  make Hamming_generator
12930 ▷  make simplex_code
12931 ▷  make Hamming_generator
12932 ▷  make Hamming_code
12933 ▷  make RM_3_1_and_codewords
12934 ▷  make RM_3_1_from_generator_matrix
12935 ▷  make RM_4_1_and_codewords
12936 ▷  make RM_5_1_and_codewords
12937 ▷  make Hamming_code_by_rows
12938 ▷  make Hamming_code_long
12939 ▷  make Hamming_code_diagram
12940 ▷  make Hamming_code_words
12941 ▷  make Hamming_weight_enumerator
12942 ▷  make Hamming_minimum_distance
12943 ▷  make Golay23_minimum_distance
```

```
12944 ▷  make code_Hamming_systematic
12945 ▷  make Hamming_RREF
12946 ▷  make Hamming_nullspace
12947
12948
12949
12950 RM_3_1:
12951 ▷  $(ORBITER) -v 2 \
12952 ▷  ▷  -define F -finite_field -q 2 -end \
12953 ▷  ▷  -define C -code -field F \
12954 ▷  ▷  ▷  -Reed_Muller 3 \
12955 ▷  ▷  -end \
12956 ▷  ▷  -with C -and F -do -coding_theoretic_activity \
12957 ▷  ▷  ▷  -export_magma RM_3_1.magma \
12958 ▷  ▷  -end \
12959 ▷  ▷  -with C -do -coding_theoretic_activity \
12960 ▷  ▷  ▷  -report \
12961 ▷  ▷  -end
12962 ▷  pdflatex RM_3_code_n8_k4_q2.tex
12963 ▷  $(OPEN) RM_3_code_n8_k4_q2.pdf
12964
12965
12966 code_6:
12967 ▷  $(ORBITER) -v 2 \
12968 ▷  ▷  -define F -finite_field -q 2 -end \
12969 ▷  ▷  -define C -code -field F \
12970 ▷  ▷  ▷  -basis 6 "60,50,41" \
12971 ▷  ▷  -end \
12972 ▷  ▷  -with C -and F -do -coding_theoretic_activity \
12973 ▷  ▷  ▷  -make_diagram \
12974 ▷  ▷  -end \
12975 ▷  ▷  -with C -do -coding_theoretic_activity \
12976 ▷  ▷  ▷  -report \
12977 ▷  ▷  -end
12978 ▷  $(ORBITER) -v 2 -draw_matrix \
12979 ▷  ▷  -input_csv_file by_basis_n6_k3_char_func_6_8.csv \
12980 ▷  ▷  -box_width 20 -bit_depth 24 \
12981 ▷  ▷  -partition 2 "1,1,1,1,1,1,1,1" "1,1,1,1,1,1,1,1" \
12982 ▷  ▷  -end
12983 ▷  pdflatex by_basis_n6_k3_code_n6_k3_q2.tex
12984 ▷  $(OPEN) by_basis_n6_k3_code_n6_k3_q2.pdf
12985 ▷  $(OPEN) by_basis_n6_k3_char_func_6_8_draw.bmp
12986
12987 # linear code with generator matrix
12988 # 111100 = 60
12989 # 110010 = 50
12990 # 101001 = 41
12991
12992
12993
12994
12995 simplex_code:
12996 ▷  $(ORBITER) -v 2 \
12997 ▷  ▷  -define F -finite_field -q 2 -end \
12998 ▷  ▷  -define v -vector -field F -format 3 \
12999 ▷  ▷  ▷  -dense $(SIMPLEX_CODE_GENERATOR) \
13000 ▷  ▷  -end \
13001 ▷  ▷  -define C -code -field F \
13002 ▷  ▷  ▷  -generator_matrix v \
```

```
13003 ▷  ▷  -end \
13004 ▷  ▷  -with C -do -coding_theoretic_activity \
13005 ▷  ▷  ▷  -report \
13006 ▷  ▷  -end
13007 ▷  pdflatex by_genma_n7_k3_code_n7_k3_q2.tex
13008 ▷  $(OPEN) by_genma_n7_k3_code_n7_k3_q2.pdf
13009
13010
13011
13012 Hamming_generator:
13013 ▷  $(ORBITER) -v 2 \
13014 ▷  ▷  -define F -finite_field -q 2 -end \
13015 ▷  ▷  -define v -vector -field F -format 3 \
13016 ▷  ▷  ▷  -dense $(SIMPLEX_CODE_GENERATOR) \
13017 ▷  ▷  -end \
13018 ▷  ▷  -with F -do \
13019 ▷  ▷  -finite_field_activity \
13020 ▷  ▷  ▷  -nullspace v \
13021 ▷  ▷  -end
13022 ▷  pdflatex nullspace_3_7.tex
13023 ▷  $(OPEN) nullspace_3_7.pdf
13024
13025 # basis in binary:
13026 # 67,37,22,15
13027
13028
13029
13030
13031 Hamming_code:
13032 ▷  $(ORBITER) -v 2 \
13033 ▷  ▷  -define F -finite_field -q 2 -end \
13034 ▷  ▷  -define v -vector -field F -format 3 \
13035 ▷  ▷  ▷  -dense $(SIMPLEX_CODE_GENERATOR) \
13036 ▷  ▷  -end \
13037 ▷  ▷  -define C -code -field F \
13038 ▷  ▷  ▷  -generator_matrix v \
13039 ▷  ▷  ▷  -dual -end \
13040 ▷  ▷  -end \
13041 ▷  ▷  -with C -do -coding_theoretic_activity \
13042 ▷  ▷  ▷  -export_magma Hamming.magma \
13043 ▷  ▷  -end
13044
13045 # writes Hamming.magma
13046
13047
13048
13049 RM_3_1_and_codewords:
13050 ▷  $(ORBITER) -v 2 \
13051 ▷  ▷  -define F -finite_field -q 2 -end \
13052 ▷  ▷  -define C -code -field F \
13053 ▷  ▷  ▷  -Reed_Muller 3 \
13054 ▷  ▷  -end \
13055 ▷  ▷  -with C -and F -do -coding_theoretic_activity \
13056 ▷  ▷  ▷  -export_magma RM_3_1.magma \
13057 ▷  ▷  -end \
13058 ▷  ▷  -with C -and F -do -coding_theoretic_activity \
13059 ▷  ▷  ▷  -export_codewords RM_3_1_codewords.csv \
13060 ▷  ▷  -end \
13061 ▷  ▷  -with C -and F -do -coding_theoretic_activity \
```

```
13062 ▷  ▷  ▷  -export_genma RM_3_1_genma.csv \
13063 ▷  ▷  -end
13064
13065
13066 RM_3_1_from_generator_matrix:
13067 ▷  $(ORBITER) -v 2 \
13068 ▷  ▷  -define F -finite_field -q 2 -end \
13069 ▷  ▷  -define genma -vector -format 8 -field F \
13070 ▷  ▷  ▷  -compact $(CODE_RM_3_1_GENMA) \
13071 ▷  ▷  -end \
13072 ▷  ▷  -define C -code -field F \
13073 ▷  ▷  ▷  -generator_matrix genma \
13074 ▷  ▷  -end
13075 ▷  #pdflatex code_n8_k4_q2.tex
13076 ▷  #$(OPEN) code_n8_k4_q2.pdf
13077
13078 #Codewords: (0,255,170,85,204,51,102,153,240,15,90,165,60,195,150,105)
13079
13080
13081 RM_4_1_and_codewords:
13082 ▷  $(ORBITER) -v 2 \
13083 ▷  ▷  -define F -finite_field -q 2 -end \
13084 ▷  ▷  -define C -code -field F -Reed_Muller 4 -end \
13085 ▷  ▷  -with C -and F -do -coding_theoretic_activity \
13086 ▷  ▷  ▷  -export_magma RM_4_1.magma \
13087 ▷  ▷  -end \
13088 ▷  ▷  -with C -and F -do -coding_theoretic_activity \
13089 ▷  ▷  ▷  -export_codewords RM_4_1_codewords.csv \
13090 ▷  ▷  -end \
13091 ▷  ▷  -with C -and F -do -coding_theoretic_activity \
13092 ▷  ▷  ▷  -export_genma RM_4_1_genma.csv \
13093 ▷  ▷  -end
13094
13095 RM_5_1_and_codewords:
13096 ▷  $(ORBITER) -v 2 \
13097 ▷  ▷  -define F -finite_field -q 2 -end \
13098 ▷  ▷  -define C -code -field F -Reed_Muller 5 -end \
13099 ▷  ▷  -with C -and F -do -coding_theoretic_activity \
13100 ▷  ▷  ▷  -export_magma RM_5_1.magma \
13101 ▷  ▷  -end \
13102 ▷  ▷  -with C -and F -do -coding_theoretic_activity \
13103 ▷  ▷  ▷  -export_codewords RM_5_1_codewords.csv \
13104 ▷  ▷  -end \
13105 ▷  ▷  -with C -and F -do -coding_theoretic_activity \
13106 ▷  ▷  ▷  -export_genma RM_5_1_genma.csv \
13107 ▷  ▷  -end
13108
13109
13110
13111
13112 Hamming_code_by_rows:
13113 ▷  $(ORBITER) -v 2 \
13114 ▷  ▷  -define F -finite_field -q 2 -end \
13115 ▷  ▷  -define v -vector -dense $(HAMMING_CODE_ROWS_IN_BINARY_RANKS) -end \
13116 ▷  ▷  -define C -code -field F \
13117 ▷  ▷  ▷  -basis 7 v \
13118 ▷  ▷  -end
13119 ▷  #pdflatex code_n7_k4_q2.tex
13120 ▷  #$(OPEN) code_n7_k4_q2.pdf
```

```
13121
13122
13123
13124 Hamming_code_long:
13125 ▷  $(ORBITER) -v 2 \
13126 ▷  ▷  -define F -finite_field -q 2 -end \
13127 ▷  ▷  -define C -code -field F \
13128 ▷  ▷  ▷  -long_code 7 4 \
13129 ▷  ▷  ▷  "0,5,6" \
13130 ▷  ▷  ▷  "1,4,6" \
13131 ▷  ▷  ▷  "2,4,5" \
13132 ▷  ▷  ▷  "3,4,5,6" \
13133 ▷  ▷  -end
13134
13135
13136 Hamming_code_diagram:
13137 ▷  $(ORBITER) -v 2 \
13138 ▷  ▷  -define F -finite_field -q 2 -end \
13139 ▷  ▷  -define v -vector -dense $(HAMMING_CODE_ROWS_IN_BINARY_RANKS) -end \
13140 ▷  ▷  -define C -code -field F \
13141 ▷  ▷  ▷  -basis 7 v \
13142 ▷  ▷  -end \
13143 ▷  ▷  -with C -and F -do -coding_theoretic_activity \
13144 ▷  ▷  ▷  -make_diagram \
13145 ▷  ▷  -end
13146 ▷  $(ORBITER) -v 2 \
13147 ▷  ▷  -draw_matrix \
13148 ▷  ▷  ▷  -input_csv_file by_basis_n7_k4_char_func_7_16.csv \
13149 ▷  ▷  ▷  -box_width 25 -bit_depth 24 \
13150 ▷  ▷  ▷  -partition 4 16 8 \
13151 ▷  ▷  -end
13152 ▷  $(OPEN) by_basis_n7_k4_char_func_7_16_draw.bmp
13153
13154
13155 Hamming_code_words:
13156 ▷  $(ORBITER) -v 2 \
13157 ▷  ▷  -define v -vector -dense $(HAMMING_CODE_CODEWORDS) -end \
13158 ▷  ▷  -define F -finite_field -q 2 -end \
13159 ▷  ▷  -loop_over i v \
13160 ▷  ▷  ▷  -with F -do -coding_theoretic_activity \
13161 ▷  ▷  ▷  -general_code_binary 7 "Hamming_7_4_word_%i" "%i[v]" \
13162 ▷  ▷  ▷  -metric_balls 1 \
13163 ▷  ▷  ▷  -end \
13164 ▷  ▷  -end_loop_over i
13165 ▷  $(ORBITER) -v 2 \
13166 ▷  ▷  -loop i 0 16 1 -draw_matrix \
13167 ▷  ▷  ▷  -input_csv_file Hamming_7_4_word_%i_char_func_7_1.csv \
13168 ▷  ▷  ▷  -box_width 25 -bit_depth 8 \
13169 ▷  ▷  ▷  -partition 4 16 8 -end \
13170 ▷  ▷  ▷  -system "convert Hamming_7_4_word_%i_char_func_7_1_draw.bmp \
13171 ▷  ▷  ▷  -frame 8 Hamming_7_4_word_%i_char_func_7_1_draw.png" \
13172 ▷  ▷  -end_loop i
13173
13174
13175
13176 Hamming_weight_enumerator:
13177 ▷  $(ORBITER) -v 2 \
13178 ▷  ▷  -define F -finite_field -q 2 -end \
13179 ▷  ▷  -define v -vector -field F -format 4 \
```

```
13180 ▷  ▷  ▷  -dense $(HAMMING_CODE_GENERATOR) \
13181 ▷  ▷  -end \
13182 ▷  ▷  -define C -code -field F \
13183 ▷  ▷  ▷  -generator_matrix v \
13184 ▷  ▷  -end \
13185 ▷  ▷  -with C -do \
13186 ▷  ▷  -coding_theoretic_activity \
13187 ▷  ▷  ▷  -weight_enumerator \
13188 ▷  ▷  -end
13189
13190 Hamming_minimum_distance:
13191 ▷  $(ORBITER) -v 2 \
13192 ▷  ▷  -define F -finite_field -q 2 -end \
13193 ▷  ▷  -define v -vector -field F -format 4 \
13194 ▷  ▷  ▷  -dense $(HAMMING_CODE_GENERATOR) \
13195 ▷  ▷  -end \
13196 ▷  ▷  -with F -do \
13197 ▷  ▷  -coding_theoretic_activity \
13198 ▷  ▷  ▷  -minimum_distance v \
13199 ▷  ▷  -end
13200
13201
13202 Golay23_minimum_distance:
13203 ▷  $(ORBITER) -v 2 \
13204 ▷  ▷  -define F -finite_field -q 2 -end \
13205 ▷  ▷  -define v -vector -field F -format 12 \
13206 ▷  ▷  ▷  -dense $(GOLAY23_CODE_GENERATOR) \
13207 ▷  ▷  -end \
13208 ▷  ▷  -with F -do \
13209 ▷  ▷  -coding_theoretic_activity \
13210 ▷  ▷  ▷  -minimum_distance v \
13211 ▷  ▷  -end
13212
13213 #d=7 in 0 sec
13214
13215
13216
13217
13218 code_Hamming_systematic:
13219 ▷  $(ORBITER) -v 2 \
13220 ▷  ▷  -define F -finite_field -q 2 -end \
13221 ▷  ▷  -define v -vector -dense $(HAMMING_CODE_ROWS_IN_BINARY_RANKS) -end \
13222 ▷  ▷  -define C -code -field F \
13223 ▷  ▷  ▷  -basis 7 v \
13224 ▷  ▷  -end
13225 ▷  #$(ORBITER) -v 2 -draw_matrix \
13226 ▷  #▷ -input_csv_file code_matrix_16_8.csv \
13227 ▷  #▷ -box_width 25 -bit_depth 8 -partition 2 16 8 -end
13228 ▷  #$(OPEN) code_matrix_16_8_draw.bmp
13229 ▷  #pdflatex code_7_16.tex
13230 ▷  #$(OPEN) code_7_16.pdf
13231
13232
13233 #▷ $(DRAW_PATH)/draw_matrix.out -read code_matrix_256_256.csv -box_width 10
13234
13235 # 1 0 0 0 0 1 1 = 67
13236 # 0 1 0 0 1 0 1 = 37
13237 # 0 0 1 0 1 1 0 = 22
13238 # 0 0 0 1 1 1 1 = 15
```

```
13239
13240
13241 # ToDo
13242
13243 Hamming_RREF:
13244 ▷  $(ORBITER) -v 2 \
13245 ▷  ▷  -define F -finite_field -q 2 -end \
13246 ▷  ▷  -define v -vector -format 4 -field F \
13247 ▷  ▷  ▷  -dense $(HAMMING_CODE_GENERATOR) \
13248 ▷  ▷  -end \
13249 ▷  ▷  -with F -do \
13250 ▷  ▷  -finite_field_activity \
13251 ▷  ▷  ▷  -RREF v \
13252 ▷  ▷  -end
13253 ▷  pdflatex rref_m4_n7_q2.tex
13254 ▷  $(OPEN) rref_m4_n7_q2.pdf
13255 ▷  #pdflatex RREF_example_q2_4_7.tex
13256 ▷  #gs -sDEVICE=png16 -dFIXEDMEDIA \
13257 ▷  #▷ -dDEVICEWIDTHPOINTS=500 -dDEVICEHEIGHTPOINTS=450 \
13258 ▷  #▷ -r240 -oHamming_dual_page%02d.png \
13259 ▷  #▷ RREF_example_q2_4_7.pdf
13260
13261
13262
13263 Hamming_nullspace:
13264 ▷  $(ORBITER) -v 2 \
13265 ▷  ▷  -define F2 -finite_field -q 2 -end \
13266 ▷  ▷  -define v -vector -format 4 -field F2 \
13267 ▷  ▷  ▷  -dense $(HAMMING_CODE_GENERATOR) \
13268 ▷  ▷  -end \
13269 ▷  ▷  -with F2 -do \
13270 ▷  ▷  -finite_field_activity \
13271 ▷  ▷  -nullspace v \
13272 ▷  ▷  -end
13273 ▷  pdflatex nullspace_4_7.tex
13274 ▷  $(OPEN) nullspace_4_7.pdf
13275
13276
13277 #check equations of the Hamming code:
13278 # a4+a5+a6+a7 =1+0+1+0=0 mod2 OK.
13279 # a2+a3+a6+a7 =0+1+1+0=0 mod2 OK.
13280 # a1+a3+a5+a7 =1+1+0+0=0 mod2 OK.
13281
13282 #1010101
13283 #0110011
13284 #0001111
13285
13286
13287
13288
13289
13290 ##############################################################################
13291 # Section 10.3: Coding Theory - Golay codes
13292
13293 SECTION_CODING_THEORY_GOLAY_CODES:
13294
13295
13296 test_10_3:
13297 ▷  make Golay23_code_words
```

```
13298 ▷  make Golay23_code_diagram
13299 ▷  #make Golay23_code_diagram_draw
13300 ▷  make Golay24_make_code
13301 ▷  make Golay24_make_aut_gens
13302 ▷  make Golay24_make_aut_gens_classes
13303 ▷  make M24_elt_2A
13304 ▷  make M24_elt_2B
13305 ▷  make M24_2A_restricted
13306 ▷  make M24_2B_restricted
13307 ▷  make Golay24_2A_fixed_subcode
13308 ▷  make Golay24_2B_fixed_subcode
13309 ▷  make Golay24_2A_fix_code
13310 ▷  make Golay24_2B_fix_code
13311 ▷  make Golay24_2A_fix_weight12
13312 ▷  make Golay24_2A_fix_weight12_c
13313 ▷  make Golay24_2A_fix_weight12_graph_24
13314
13315
13316
13317
13318 Golay23_code_words:
13319 ▷  $(ORBITER) -v 2 \
13320 ▷  ▷  -define v -vector -dense $(GOLAY_23_COLUMN_RANKS_PROJECTIVELY) -end \
13321 ▷  ▷  -define F -finite_field -q 2 -end \
13322 ▷  ▷  -define C -code -field F \
13323 ▷  ▷  ▷  -projective_set 12 v -end \
13324 ▷  ▷  -with C -and F -do -coding_theoretic_activity \
13325 ▷  ▷  ▷  -export_magma Golay23.magma \
13326 ▷  ▷  -end \
13327 ▷  ▷  -with C -and F -do -coding_theoretic_activity \
13328 ▷  ▷  ▷  -export_codewords Golay23_codewords.csv \
13329 ▷  ▷  -end \
13330 ▷  ▷  -with C -and F -do -coding_theoretic_activity \
13331 ▷  ▷  ▷  -export_genma Golay23_genma.csv \
13332 ▷  ▷  -end
13333 ▷  #pdflatex code_n23_k12_q2.tex
13334 ▷  #$(OPEN) code_n23_k12_q2.pdf
13335
13336
13337 Golay23_code_diagram:
13338 ▷  $(ORBITER) -v 2 \
13339 ▷  ▷  -define F -finite_field -q 2 -end \
13340 ▷  ▷  -define words -vector -file Golay23_codewords.csv -end \
13341 ▷  ▷  -with F -do \
13342 ▷  ▷  -coding_theoretic_activity \
13343 ▷  ▷  ▷  -general_code_binary 23 "Golay_23" \
13344 ▷  ▷  ▷  words \
13345 ▷  ▷  ▷  -embellish 4 \
13346 ▷  ▷  -end
13347
13348
13349 Golay23_code_diagram_draw:
13350 ▷  $(ORBITER) -v 2 \
13351 ▷  ▷  -draw_matrix \
13352 ▷  ▷  ▷  -input_csv_file Golay_23_diagram_01_23_4096.csv \
13353 ▷  ▷  ▷  -box_width 4 -bit_depth 8 \
13354 ▷  ▷  ▷  -partition 20 4096 2048 \
13355 ▷  ▷  -end
13356
```

```
13357
13358 Golay24_make_code:
13359 ▷  $(ORBITER) -v 2 \
13360 ▷  ▷  -define F -finite_field -q 2 -end \
13361 ▷  ▷  -define v -vector \
13362 ▷  ▷  ▷  -dense $(GOLAY_24_CODE_PROJECTIVE) \
13363 ▷  ▷  -end \
13364 ▷  ▷  -define C -code -field F \
13365 ▷  ▷  ▷  -projective_set 12 v \
13366 ▷  ▷  -end
13367
13368
13369 Golay24_make_aut_gens:
13370 ▷  $(ORBITER) -v 2 \
13371 ▷  ▷  -define F -finite_field -q 2 -end \
13372 ▷  ▷  -define G -linear_group -PGL 12 2 \
13373 ▷  ▷  -subgroup_by_generators "M24" "244823040" 12 \
13374 ▷  ▷  ▷  $(GOLAY_24_CODE_AUT_GENS) \
13375 ▷  ▷  -end \
13376 ▷  ▷  -with G -do \
13377 ▷  ▷  -group_theoretic_activity \
13378 ▷  ▷  ▷  -report \
13379 ▷  ▷  -end
13380 ▷  pdflatex PGL_12_2_Subgroup_M24_244823040_report.tex
13381 ▷  $(OPEN) PGL_12_2_Subgroup_M24_244823040_report.pdf
13382
13383
13384 Golay24_make_aut_gens_classes:
13385 ▷  $(ORBITER) -v 3 \
13386 ▷  ▷  -define F -finite_field -q 2 -end \
13387 ▷  ▷  -define G -linear_group -PGL 12 2 \
13388 ▷  ▷  -subgroup_by_generators "M24" "244823040" 12 \
13389 ▷  ▷  ▷  $(GOLAY_24_CODE_AUT_GENS) \
13390 ▷  ▷  -end \
13391 ▷  ▷  -with G -do \
13392 ▷  ▷  -group_theoretic_activity \
13393 ▷  ▷  ▷  -classes \
13394 ▷  ▷  -end
13395 ▷  $(MAGMA_PATH)magma M24_classes.magma
13396 ▷  $(ORBITER) -v 3 \
13397 ▷  ▷  -define F -finite_field -q 2 -end \
13398 ▷  ▷  -define G -linear_group -PGL 12 2 \
13399 ▷  ▷  -subgroup_by_generators "M24" "244823040" 12 \
13400 ▷  ▷  ▷  $(GOLAY_24_CODE_AUT_GENS) \
13401 ▷  ▷  -end \
13402 ▷  ▷  -with G -do \
13403 ▷  ▷  -group_theoretic_activity \
13404 ▷  ▷  ▷  -classes \
13405 ▷  ▷  -end
13406 ▷  pdflatex PGL_12_2_Subgroup_M24_244823040_classes_out.tex
13407 ▷  $(OPEN) PGL_12_2_Subgroup_M24_244823040_classes_out.pdf
13408
13409
13410
13411 M24_elt_2A:
13412 ▷  $(ORBITER) -v 2 \
13413 ▷  ▷  -define F -finite_field -q 2 -end \
13414 ▷  ▷  -define G -linear_group -PGL 12 2 \
13415 ▷  ▷  -subgroup_by_generators "M24_2A" "2" 1 \
```

```
13416 ▷  ▷  ▷  $(M24_ELEMENT_2A) \
13417 ▷  ▷  -end \
13418 ▷  ▷  -with G -do \
13419 ▷  ▷  -group_theoretic_activity \
13420 ▷  ▷  ▷  -report \
13421 ▷  ▷  -end
13422 ▷  pdflatex PGL_12_2_Subgroup_M24_2A_2_report.tex
13423 ▷  $(OPEN) PGL_12_2_Subgroup_M24_2A_2_report.pdf
13424
13425 M24_elt_2B:
13426 ▷  $(ORBITER) -v 2 \
13427 ▷  ▷  -define F -finite_field -q 2 -end \
13428 ▷  ▷  -define G -linear_group -PGL 12 2 \
13429 ▷  ▷  -subgroup_by_generators "M24_2B" "2" 1 \
13430 ▷  ▷  ▷  $(M24_ELEMENT_2B) \
13431 ▷  ▷  -end \
13432 ▷  ▷  -with G -do \
13433 ▷  ▷  -group_theoretic_activity \
13434 ▷  ▷  ▷  -report \
13435 ▷  ▷  -end
13436 ▷  pdflatex PGL_12_2_Subgroup_M24_2B_2_report.tex
13437 ▷  $(OPEN) PGL_12_2_Subgroup_M24_2B_2_report.pdf
13438
13439 M24_2A_restricted:
13440 ▷  $(ORBITER) -v 2 \
13441 ▷  ▷  -define v -vector \
13442 ▷  ▷  ▷  -dense $(GOLAY_24_CODE_PROJECTIVE) \
13443 ▷  ▷  -end \
13444 ▷  ▷  -define F -finite_field -q 2 -end \
13445 ▷  ▷  -define G -linear_group -PGL 12 2 \
13446 ▷  ▷  ▷  -subgroup_by_generators "M24_2A" "2" 1 \
13447 ▷  ▷  ▷  $(M24_ELEMENT_2A) \
13448 ▷  ▷  -end \
13449 ▷  ▷  -define M24_2Ar -modified_group -from G -restricted_action v \\_on\\_code -en
      d \
13450 ▷  ▷  -with M24_2Ar -do \
13451 ▷  ▷  ▷  -group_theoretic_activity \
13452 ▷  ▷  ▷  ▷  -print_elements_tex  \
13453 ▷  ▷  -end
13454 ▷  pdflatex PGL_12_2_res_v_elements.tex
13455 ▷  $(OPEN) PGL_12_2_res_v_elements.pdf
13456
13457
13458 # 8 fixed points, 8 cycles of length 2.
13459
13460
13461
13462
13463
13464 M24_2B_restricted:
13465 ▷  $(ORBITER) -v 2 \
13466 ▷  ▷  -define v -vector \
13467 ▷  ▷  ▷  -dense $(GOLAY_24_CODE_PROJECTIVE) \
13468 ▷  ▷  -end \
13469 ▷  ▷  -define F -finite_field -q 2 -end \
13470 ▷  ▷  -define G -linear_group -PGL 12 2 \
13471 ▷  ▷  ▷  -subgroup_by_generators "M24_2B" "2" 1 \
13472 ▷  ▷  ▷  $(M24_ELEMENT_2B) \
13473 ▷  ▷  -end \
```

```
13474 ▷  ▷  -define M24_2Br -modified_group \
13475 ▷  ▷  ▷  -from G -restricted_action v v \
13476 ▷  ▷  -end \
13477 ▷  ▷  -with M24_2Br -do \
13478 ▷  ▷  ▷  -group_theoretic_activity \
13479 ▷  ▷  ▷  ▷  -print_elements_tex  \
13480 ▷  ▷  -end
13481 ▷  pdflatex PGL_12_2_res24_elements.tex
13482 ▷  $(OPEN) PGL_12_2_res24_elements.pdf
13483
13484 # 12 cycles of length 2
13485
13486
13487
13488
13489 Golay24_2A_fixed_subcode:
13490 ▷  $(ORBITER) -v 2 \
13491 ▷  ▷  -define perm -vector \
13492 ▷  ▷  ▷  -dense $(M24_2A_PERM) \
13493 ▷  ▷  -end \
13494 ▷  ▷  -define F -finite_field -q 2 -end \
13495 ▷  ▷  -define v -vector \
13496 ▷  ▷  ▷  -dense $(GOLAY_24_CODE_PROJECTIVE) \
13497 ▷  ▷  -end \
13498 ▷  ▷  -define C -code -field F \
13499 ▷  ▷  ▷  -projective_set 12 v \
13500 ▷  ▷  -end \
13501 ▷  ▷  -with C -and F -do -coding_theoretic_activity \
13502 ▷  ▷  ▷  -fixed_code perm \
13503 ▷  ▷  -end
13504
13505 # 256 codewords
13506
13507 Golay24_2B_fixed_subcode:
13508 ▷  $(ORBITER) -v 2 \
13509 ▷  ▷  -define perm -vector \
13510 ▷  ▷  ▷  -dense $(M24_2B_PERM) \
13511 ▷  ▷  -end \
13512 ▷  ▷  -define F -finite_field -q 2 -end \
13513 ▷  ▷  -define v -vector \
13514 ▷  ▷  ▷  -dense $(GOLAY_24_CODE_PROJECTIVE) \
13515 ▷  ▷  -end \
13516 ▷  ▷  -define C -code -field F \
13517 ▷  ▷  ▷  -projective_set 12 v \
13518 ▷  ▷  -end \
13519 ▷  ▷  -with C -and F -do -coding_theoretic_activity \
13520 ▷  ▷  ▷  -fixed_code perm \
13521 ▷  ▷  -end
13522
13523 # 64 codewords
13524
13525
13526 Golay24_2A_fix_code:
13527 ▷  $(ORBITER) -v 2 \
13528 ▷  ▷  -define F -finite_field -q 2 -end \
13529 ▷  ▷  -define v -vector \
13530 ▷  ▷  ▷  -format 8 \
13531 ▷  ▷  ▷  -dense $(GOLAY24_2A_FIX_SUBCODE) \
13532 ▷  ▷  -end \
```

```
13533 ▷  ▷  -define C -code -field F \
13534 ▷  ▷  ▷  -generator_matrix v \
13535 ▷  ▷  -end \
13536 ▷  ▷  -with C -do \
13537 ▷  ▷  -coding_theoretic_activity \
13538 ▷  ▷  ▷  -weight_enumerator \
13539 ▷  ▷  -end \
13540 ▷  ▷  -with C -do \
13541 ▷  ▷  -coding_theoretic_activity \
13542 ▷  ▷  ▷  -export_codewords_by_weight Golay24_2A_fix \
13543 ▷  ▷  -end
13544
13545 #1y^{24} + 71x^8y^{16} + 112x^{12}y^{12} + 71x^{16}y^8 + 1x^{24}
13546 #weight enumerator:
13547 #( 1, 0, 0, 0, 0, 0, 0, 0, 71, 0, 0, 0, 112, 0, 0, 0, 71, 0, 0, 0, 0, 0, 0, 0, 1
      )
13548
13549 # 112 codewords of weight 12:
13550 #Golay24_2A_fix_of_weight_12.csv
13551
13552
13553 Golay24_2B_fix_code:
13554 ▷  $(ORBITER) -v 2 \
13555 ▷  ▷  -define F -finite_field -q 2 -end \
13556 ▷  ▷  -define v -vector \
13557 ▷  ▷  ▷  -format 6 \
13558 ▷  ▷  ▷  -dense $(GOLAY24_2B_FIX_SUBCODE) \
13559 ▷  ▷  -end \
13560 ▷  ▷  -define C -code -field F \
13561 ▷  ▷  ▷  -generator_matrix v \
13562 ▷  ▷  -end \
13563 ▷  ▷  -with C -do \
13564 ▷  ▷  -coding_theoretic_activity \
13565 ▷  ▷  ▷  -weight_enumerator \
13566 ▷  ▷  -end \
13567 ▷  ▷  -with C -do \
13568 ▷  ▷  -coding_theoretic_activity \
13569 ▷  ▷  ▷  -export_codewords_by_weight Golay24_2B_fix \
13570 ▷  ▷  -end
13571
13572 #1y^{24} + 15x^8y^{16} + 32x^{12}y^{12} + 15x^{16}y^8 + 1x^{24}
13573 #weight enumerator:
13574 #( 1, 0, 0, 0, 0, 0, 0, 0, 15, 0, 0, 0, 32, 0, 0, 0, 15, 0, 0, 0, 0, 0, 0, 0, 1 )

13575
13576 Golay24_2A_fix_weight12:
13577 ▷  $(ORBITER) -v 2 \
13578 ▷  ▷  -define F -finite_field -q 2 -end \
13579 ▷  ▷  -define v -vector \
13580 ▷  ▷  ▷  -load_csv_data_column Golay24_2A_fix_of_weight_12.csv 1 \
13581 ▷  ▷  -end \
13582 ▷  ▷  -define D -design -list_of_blocks_coded \
13583 ▷  ▷  ▷  24 12 v \
13584 ▷  ▷  -end \
13585 ▷  ▷  -with D -do \
13586 ▷  ▷  ▷  -design_activity \
13587 ▷  ▷  ▷  ▷  -export_inc \
13588 ▷  ▷  -end \
13589 ▷  ▷  -with D -do \
```

```
13590 ▷  ▷  ▷  -design_activity \
13591 ▷  ▷  ▷  ▷  -intersection_matrix \
13592 ▷  ▷  -end
13593
13594 #sets_v24_inc.txt
13595 #sets_v24_AAt.csv
13596
13597
13598 Golay24_2A_fix_weight12_c:
13599 ▷  $(ORBITER) -v 3 \
13600 ▷  ▷  -draw_incidence_structure_description \
13601 ▷  ▷  ▷  -width 60 -width_10 6 -end \
13602 ▷  ▷  -define C -combinatorial_object \
13603 ▷  ▷  ▷  -label Golay24_2A_fix_weight12 Golay24\_2A\_fix\_weight12 \
13604 ▷  ▷  ▷  -file_of_incidence_geometries blocks_v24_k12_inc.txt 24 112 1344 \
13605 ▷  ▷  -end \
13606 ▷  ▷  -with C -do \
13607 ▷  ▷  -combinatorial_object_activity \
13608 ▷  ▷  ▷  -canonical_form \
13609 ▷  ▷  ▷  ▷  -save_ago \
13610 ▷  ▷  ▷  ▷  -save_transversal \
13611 ▷  ▷  ▷  -end \
13612 ▷  ▷  -end \
13613 ▷  ▷  -with C -do \
13614 ▷  ▷  -combinatorial_object_activity \
13615 ▷  ▷  ▷  -report \
13616 ▷  ▷  ▷  ▷  -export_flag_orbits \
13617 ▷  ▷  ▷  ▷  -show_incidence_matrices \
13618 ▷  ▷  ▷  ▷  -export_group_GAP \
13619 ▷  ▷  ▷  -end \
13620 ▷  ▷  -end
13621 ▷  pdflatex Golay24_2A_fix_weight12_classification.tex
13622 ▷  $(OPEN) Golay24_2A_fix_weight12_classification.pdf
13623
13624
13625 # stabilizer of order 2752512
13626
13627 Golay24_2A_fix_weight12_graph_24:
13628 ▷  $(ORBITER) -v 3 \
13629 ▷  ▷  -define Gamma -graph \
13630 ▷  ▷  ▷  -load_adjacency_matrix_from_csv_and_select_value \
13631 ▷  ▷  ▷  blocks_v24_k12_AAt.csv 24 \
13632 ▷  ▷  -end \
13633 ▷  ▷  -with Gamma -do \
13634 ▷  ▷  ▷  -graph_theoretic_activity -automorphism_group \
13635 ▷  ▷  -end \
13636 ▷  ▷  -with Gamma -do \
13637 ▷  ▷  ▷  -graph_theoretic_activity \
13638 ▷  ▷  ▷  -properties \
13639 ▷  ▷  -end
13640 ▷  #pdflatex _report.tex
13641 ▷  #$(OPEN) _report.pdf
13642
13643 #Degree type: (14^{16},\, 7^8 )
13644
13645
13646 ##############################################################################
13647 # Section 10.4: Coding Theory - CRC codes
13648
```

```
13649 SECTION_CODING_THEORY_CRC_CODES:
13650
13651
13652 test_10_4:
13653 ▷  make CRC_3_128_10
13654 ▷  make crc32_test
13655 ▷  make crc32_test_hexdata
13656 ▷  make crc32_Berlekamp_matrix
13657 ▷  make CRC_F256_roots_771
13658 ▷  make CRC_F256_BCH_code_d2
13659 ▷  make CRC_F256_BCH_write_code_for_division_d2
13660 ▷  make CRC_F256_BCH_code_d3
13661 ▷  make CRC_F256_BCH_write_code_for_division_Bravo
13662 ▷  make CRC_F256_BCH_code_d7
13663 ▷  make CRC_F256_BCH_write_code_for_division_Charlie
13664 ▷  make F256_BCH_code_d16
13665 ▷  make F256_BCH_write_code_for_division_d16
13666 ▷  make F256_BCH_code_d16_division
13667 ▷  make F256_BCH_code_d16_error
13668 ▷  make F_256
13669 ▷  make CRC_F_16
13670 ▷  make nth_roots_256
13671 ▷  make CRC_F16_roots_51
13672 ▷  make CRC_F16_BCH_code_d3
13673 ▷  make CRC_F16_BCH_code_d3_mindist_dual
13674 ▷  make CRC_F16_BCH_code_d3_dual_weight_enumerator
13675 ▷  make CRC_F16_BCH_code_d3_projective_set
13676 ▷  make CRC_F16_BCH_code_d3_macwilliams
13677 ▷  make CRC_F16_BCH_write_code_for_division_Delta
13678 ▷  make crc_encode_16
13679 ▷  make crc_encode_32
13680 ▷  make introduce_errors_16_500000
13681 ▷  make introduce_errors_32_100000
13682 ▷  make check_errors_16
13683 ▷  make check_errors_32
13684 ▷  make extract_block
13685 ▷  make crc_test_alfa
13686 ▷  make crc_test_bravo_10M_3
13687 ▷  make error_polynomial_bravo_552
13688 ▷  make crc_test_crc32_10M_3
13689 ▷  make crc_test_bravo_10M_4
13690 ▷  make crc_test_crc32_10M_4
13691 ▷  make crc_test_bravo_10M_5
13692 ▷  make crc_test_crc32_10M_5
13693 ▷  make crc_test_bravo_10M_6
13694 ▷  make crc_test_crc32_10M_6
13695 ▷  make crc_test_bravo_10M_7
13696 ▷  make crc_test_crc32_10M_7
13697 ▷  make crc_test_bravo_10M_8
13698 ▷  make crc_test_crc32_10M_8
13699 ▷  make crc_test_bravo_10_loop
13700 ▷  make crc_test_crc32_10_loop
13701 ▷  make crc_test_crc32_20M_3
13702 ▷  make crc_test_charlie
13703
13704
13705
13706 CRC_3_128_10:
13707 ▷  $(ORBITER) -v 1 \
```

```
13708 ▷  ▷  -define F -finite_field -q 2 -end \
13709 ▷  ▷  -with F -do -coding_theoretic_activity \
13710 ▷  ▷  ▷  -find_CRC_polynomials 3 128 10 \
13711 ▷  ▷  -end
13712
13713
13714
13715
13716 crc32_test:
13717 ▷  $(ORBITER) -v 3 \
13718 ▷  ▷  -define F -finite_field -q 2 -end \
13719 ▷  ▷  -with F -do -coding_theoretic_activity \
13720 ▷  ▷  ▷  -crc32 "123456789" \
13721 ▷  ▷  -end
13722
13723 crc32_test_hexdata:
13724 ▷  $(ORBITER) -v 3 \
13725 ▷  ▷  -define F -finite_field -q 2 -end \
13726 ▷  ▷  -with F -do -coding_theoretic_activity \
13727 ▷  ▷  ▷  -crc32_hexdata "7BD11C4010" \
13728 ▷  ▷  -end
13729
13730 crc32_Berlekamp_matrix:
13731 ▷  $(ORBITER) -v 2 \
13732 ▷  ▷  -define F -finite_field -q 2 -end \
13733 ▷  ▷  -define v -vector -field F -sparse 33 $(CRC32_ETHERNET) -end \
13734 ▷  ▷  -with F -do \
13735 ▷  ▷  -finite_field_activity \
13736 ▷  ▷  ▷  -Berlekamp_matrix v \
13737 ▷  ▷  -end
13738
13739
13740 CRC_F256_roots_771:
13741 ▷  $(ORBITER) -v 3 \
13742 ▷  ▷  -define F -finite_field -q 256 -end \
13743 ▷  ▷  -with F -do -finite_field_activity \
13744 ▷  ▷  ▷  -nth_roots 771 \
13745 ▷  ▷  -end
13746
13747
13748 # alfa:
13749 # 255 = 3 * 5 * 17
13750 # 771 = 3 * 257
13751
13752 CRC_F256_BCH_code_d2:
13753 ▷  $(ORBITER) -v 2 \
13754 ▷  ▷  -define F -finite_field -q 256 -end \
13755 ▷  ▷  -define C -code -field F \
13756 ▷  ▷  ▷  -BCH 771 2 \
13757 ▷  ▷  -end \
13758 ▷  ▷  -with C -and F -do -coding_theoretic_activity \
13759 ▷  ▷  ▷  -export_magma BCH_lq8_n771_d2.magma \
13760 ▷  ▷  -end
13761 ▷  #pdflatex BCH_codes_q256_n771_d2.tex
13762 ▷  #$(OPEN) BCH_codes_q256_n771_d2.pdf
13763
13764 #Q=X^{2} + 167X + 214
13765 # degree of polynomial = 3
13766 #-dense "214,167,1"
```

```
13767 #-sparse "214,0,167,1,1,2"
13768
13769
13770
13771 CRC_F256_BCH_write_code_for_division_d2:
13772 ▷  $(ORBITER) -v 2 \
13773 ▷  ▷  -define F -finite_field -q 256 -end \
13774 ▷  ▷  -define A -vector -field F -sparse 772 "1,771,1,0" -end \
13775 ▷  ▷  -define B -vector -field F -dense $(CRC_POLY_Q256_DEG2_DENSE) -end \
13776 ▷  ▷  -with F -do \
13777 ▷  ▷  -coding_theoretic_activity \
13778 ▷  ▷  ▷  -write_code_for_division \
13779 ▷  ▷  ▷  alfa A B \
13780 ▷  ▷  -end
13781 ▷  g++ crc_alfa.cpp -o crc_alfa.out
13782 ▷  ./crc_alfa.out
13783
13784
13785
13786
13787
13788 # bravo:
13789
13790
13791
13792 # degree of polynomial = 4
13793 #-dense "1,23,27,213,1"
13794 #-sparse "1,0,23,1,27,2,213,3,1,4"
13795
13796 CRC_F256_BCH_code_d3:
13797 ▷  $(ORBITER) -v 2 \
13798 ▷  ▷  -define F -finite_field -q 256 -end \
13799 ▷  ▷  -define C -code -field F \
13800 ▷  ▷  ▷  -BCH 771 3 \
13801 ▷  ▷  -end \
13802 ▷  ▷  -with C -and F -do -coding_theoretic_activity \
13803 ▷  ▷  ▷  -export_magma BCH_lq8_n771_d3.magma \
13804 ▷  ▷  -end
13805 ▷  #pdflatex BCH_codes_q256_n771_d3.tex
13806 ▷  #$(OPEN) BCH_codes_q256_n771_d3.pdf
13807
13808
13809
13810 CRC_F256_BCH_write_code_for_division_Bravo:
13811 ▷  $(ORBITER) -v 2 \
13812 ▷  ▷  -define F -finite_field -q 256 -end \
13813 ▷  ▷  -define A -vector -field F -sparse 772 "1,771,1,0" -end \
13814 ▷  ▷  -define B -vector -field F -dense $(CRC_POLY_BRAVO_DENSE) -end \
13815 ▷  ▷  -with F -do \
13816 ▷  ▷  -coding_theoretic_activity \
13817 ▷  ▷  ▷  -write_code_for_division \
13818 ▷  ▷  ▷  bravo A B \
13819 ▷  ▷  -end
13820 ▷  g++ crc_bravo.cpp -o crc_bravo.out
13821 ▷  ./crc_bravo.out
13822
13823
13824 # Charlie
13825
```

```
13826 CRC_F256_BCH_code_d7:
13827 ▷  $(ORBITER) -v 2 \
13828 ▷  ▷  -define F -finite_field -q 256 -end \
13829 ▷  ▷  -define C -code -field F \
13830 ▷  ▷  ▷  -BCH 771 7 \
13831 ▷  ▷  -end \
13832 ▷  ▷  -with C -and F -do -coding_theoretic_activity \
13833 ▷  ▷  ▷  -export_magma BCH_lq8_n771_d7.magma \
13834 ▷  ▷  -end
13835 ▷  pdflatex BCH_codes_q256_n771_d7.tex
13836 ▷  $(OPEN) BCH_codes_q256_n771_d7.pdf
13837
13838
13839 # polynomial of degree 12:
13840 #-dense "1,126,25,1,196,209,244,3,121,126,35,65,1"
13841 #-sparse "1,0,126,1,25,2,1,3,196,4,209,5,244,6,3,7,121,8,126,9,35,10,65,11,1,12"
13842
13843
13844 CRC_F256_BCH_write_code_for_division_Charlie:
13845 ▷  $(ORBITER) -v 2 \
13846 ▷  ▷  -define F -finite_field -q 256 -end \
13847 ▷  ▷  -define A -vector -field F -sparse 772 "1,771,1,0" -end \
13848 ▷  ▷  -define B -vector -field F -dense $(CRC_POLY_CHARLIE_DENSE) -end \
13849 ▷  ▷  -with F -do \
13850 ▷  ▷  -coding_theoretic_activity \
13851 ▷  ▷  ▷  -write_code_for_division \
13852 ▷  ▷  ▷  charlie A B \
13853 ▷  ▷  -end
13854 ▷  g++ crc_charlie.cpp -o crc_charlie.out
13855 ▷  ./crc_charlie.out
13856
13857
13858
13859
13860
13861
13862 F256_BCH_code_d16:
13863 ▷  $(ORBITER) -v 3 \
13864 ▷  ▷  -define F -finite_field -q 256 -end \
13865 ▷  ▷  -define C -code -field F \
13866 ▷  ▷  ▷  -BCH 771 16 \
13867 ▷  ▷  -end
13868 ▷  pdflatex BCH_codes_q256_n771_d16.tex
13869 ▷  $(OPEN) BCH_codes_q256_n771_d16.pdf
13870
13871 #generator polynomial is X^{30} + 253X^{29} + 174X^{28} + 109X^{27} + 97X^{26} +
      144X^{25} + 112X^{24} + 212X^{23} + 192X^{22} + 169X^{21} + 24X^{20} + 150X^{19}
      + 110X^{18} + 248X^{17} + 3X^{16} + 193X^{15} + 194X^{14} + 205X^{13} + 9X^{12} +
       56X^{11} + 95X^{10} + 199X^{9} + 108X^{8} + 58X^{7} + 160X^{6} + 148X^{5} + 138X
      ^{4} + 24X^{3} + 210X^{2} + 26X + 1
13872
13873
13874
13875
13876
13877 F256_BCH_write_code_for_division_d16:
13878 ▷  $(ORBITER) -v 2 \
13879 ▷  ▷  -define F -finite_field -q 256 -end \
13880 ▷  ▷  -define A -vector -field F -sparse 772 "1,771,1,0" -end \
```

```
13881 ▷  ▷  -define B -vector -field F -dense $(POLY_Q256_DEG30_DENSE) -end \
13882 ▷  ▷  -with F -do \
13883 ▷  ▷  -coding_theoretic_activity \
13884 ▷  ▷  ▷  -write_code_for_division \
13885 ▷  ▷  ▷  check_q256_n771_r30 A B \
13886 ▷  ▷  -end
13887 ▷  g++ crc_check_q256_n771_r30.cpp -o crc_check_q256_n771_r30.out
13888 ▷  ./crc_check_q256_n771_r30.out
13889
13890
13891
13892 F256_BCH_code_d16_division:
13893 ▷  $(ORBITER) -v 2 \
13894 ▷  ▷  -define F -finite_field -q 256 -end \
13895 ▷  ▷  -define A -vector -field F -sparse 772 "1,771,1,0" -end \
13896 ▷  ▷  -define B -vector -field F -dense $(POLY_Q256_DEG30_DENSE) -end \
13897 ▷  ▷  -with F -do \
13898 ▷  ▷  -finite_field_activity \
13899 ▷  ▷  -polynomial_division A B -end
13900
13901
13902
13903 F256_BCH_code_d16_error:
13904 ▷  $(ORBITER) -v 2 \
13905 ▷  ▷  -define F -finite_field -q 256 -end \
13906 ▷  ▷  -define A -vector -field F -sparse 771 "2,30,3,31,55,770" -end \
13907 ▷  ▷  -define B -vector -field F -dense $(POLY_Q256_DEG30_DENSE) -end \
13908 ▷  ▷  -with F -do \
13909 ▷  ▷  -finite_field_activity \
13910 ▷  ▷  -polynomial_division A B -end
13911
13912
13913
13914
13915 # delta:
13916 # 15 = 3 * 5
13917 # 51 = 3 * 17
13918
13919 F_256:
13920 ▷  $(ORBITER) -v 3 \
13921 ▷  ▷  -define F -finite_field -q 256 -end \
13922 ▷  ▷  -with F -do -finite_field_activity -cheat_sheet_GF -end
13923 ▷  pdflatex GF_256.tex
13924 ▷  $(OPEN) GF_256.pdf
13925
13926
13927 # subfield F16: X^4+X+1
13928
13929
13930 # F16: X^4 + X3 + 1 not compatible with F_256 and x mapsto x^17
13931
13932 CRC_F_16:
13933 ▷  $(ORBITER) -v 3 \
13934 ▷  ▷  -define F -finite_field -q 16 -override_polynomial 19 -end \
13935 ▷  ▷  -with F -do -finite_field_activity -cheat_sheet_GF -end
13936 ▷  pdflatex GF_16_poly19.tex
13937 ▷  $(OPEN) GF_16_poly19.pdf
13938
13939 nth_roots_256:
```

```
13940 ▷  $(ORBITER) -v 3 \
13941 ▷  ▷  -define F -finite_field -q 256 -end \
13942 ▷  ▷  -with F -do -finite_field_activity \
13943 ▷  ▷  ▷  -nth_roots 51 \
13944 ▷  ▷  -end
13945 ▷  pdflatex Nth_roots_q2_n51.tex
13946 ▷  $(OPEN) Nth_roots_q2_n51.pdf
13947
13948
13949 CRC_F16_roots_51:
13950 ▷  $(ORBITER) -v 3 \
13951 ▷  ▷  -define F -finite_field -q 16 -override_polynomial 19 -end \
13952 ▷  ▷  -with F -do -finite_field_activity \
13953 ▷  ▷  ▷  -nth_roots 51 \
13954 ▷  ▷  -end
13955 ▷  pdflatex Nth_roots_q16_n51.tex
13956 ▷  $(OPEN) Nth_roots_q16_n51.pdf
13957
13958 CRC_F16_BCH_code_d3:
13959 ▷  $(ORBITER) -v 2 \
13960 ▷  ▷  -define F -finite_field -q 16 -override_polynomial 19 -end \
13961 ▷  ▷  -define C -code -field F \
13962 ▷  ▷  ▷  -BCH 51 3 \
13963 ▷  ▷  -end \
13964 ▷  ▷  -with C -and F -do -coding_theoretic_activity \
13965 ▷  ▷  ▷  -export_magma BCH_q16_n51_d3.magma \
13966 ▷  ▷  -end \
13967 ▷  ▷  -with C -and F -do -coding_theoretic_activity \
13968 ▷  ▷  ▷  -export_checkma BCH_q16_n51_d3_check.csv \
13969 ▷  ▷  -end \
13970 ▷  ▷  -with C -and F -do -coding_theoretic_activity \
13971 ▷  ▷  ▷  -export_checkma_as_projective_set BCH_q16_n51_d3_check_ranks.csv \
13972 ▷  ▷  -end
13973 ▷  #pdflatex BCH_codes_q256_n771_d2.tex
13974 ▷  #$(OPEN) BCH_codes_q256_n771_d2.pdf
13975
13976 #BCH_q16_n51_d3_check.csv
13977 #BCH_q16_n51_d3_check_ranks.csv
13978
13979 CRC_F16_BCH_code_d3_mindist_dual:
13980 ▷  $(ORBITER) -v 2 \
13981 ▷  ▷  -define F -finite_field -q 16 -override_polynomial 19 -end \
13982 ▷  ▷  -define v -vector -file BCH_q16_n51_d3_check.csv \
13983 ▷  ▷  ▷  -field F -format 4 -file BCH_q16_n51_d3_check.csv -end \
13984 ▷  ▷  -with F -do \
13985 ▷  ▷  -coding_theoretic_activity \
13986 ▷  ▷  ▷  -minimum_distance v \
13987 ▷  ▷  -end
13988
13989 CRC_F16_BCH_code_d3_dual_weight_enumerator:
13990 ▷  $(ORBITER) -v 2 \
13991 ▷  ▷  -define F -finite_field -q 16 -override_polynomial 19 -end \
13992 ▷  ▷  -define v -vector -file BCH_q16_n51_d3_check.csv \
13993 ▷  ▷  ▷  -field F -format 4 -file BCH_q16_n51_d3_check.csv -end \
13994 ▷  ▷  -define C -code -field F \
13995 ▷  ▷  ▷  -generator_matrix v \
13996 ▷  ▷  ▷  -end \
13997 ▷  ▷  -with C -do \
13998 ▷  ▷  -coding_theoretic_activity \
```

```
13999 ▷  ▷  ▷  -weight_enumerator \
14000 ▷  ▷  -end
14001
14002 #weight enumerator:
14003 #( 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3060, 0, 18360, 0, 25755, 0
      , 18360, 0 )
14004 #1y^{51} + 3060x^{44}y^7 + 18360x^{46}y^5 + 25755x^{48}y^3 + 18360x^{50}y
14005 # 1 + 3060 + 18360 + 25755 + 18360 = 65536 = 2^16 = 16^4
14006
14007
14008 CRC_F16_BCH_code_d3_projective_set:
14009 ▷  $(ORBITER) -v 2 \
14010 ▷  ▷  -define F -finite_field -q 16 -override_polynomial 19 -end \
14011 ▷  ▷  -define v -vector -file BCH_q16_n51_d3_check_ranks.csv \
14012 ▷  ▷  ▷  -file BCH_q16_n51_d3_check_ranks.csv -end \
14013
14014
14015
14016
14017 CRC_F16_BCH_code_d3_macwilliams:
14018 ▷  $(ORBITER) -v 2 \
14019 ▷  ▷  -make_macwilliams_system 51 4 16
14020
14021
14022
14023 CRC_F16_BCH_write_code_for_division_Delta:
14024 ▷  $(ORBITER) -v 2 \
14025 ▷  ▷  -define F -finite_field -q 16 -override_polynomial 19 -end \
14026 ▷  ▷  -define A -vector -field F -sparse 52 "1,51,1,0" -end \
14027 ▷  ▷  -define B -vector -field F -dense $(CRC_POLY_DELTA_DENSE) -end \
14028 ▷  ▷  -with F -do \
14029 ▷  ▷  -coding_theoretic_activity \
14030 ▷  ▷  ▷  -write_code_for_division \
14031 ▷  ▷  ▷  delta A B \
14032 ▷  ▷  -end
14033 ▷  g++ crc_delta.cpp -o crc_delta.out
14034 ▷  ./crc_delta.out
14035
14036 #g++ crc_delta.cpp -o crc_delta.out
14037 #./crc_delta.out
14038 #0,0,0,0,0,
14039
14040
14041
14042
14043 crc_encode_16:
14044 ▷  $(ORBITER) -v 3 \
14045 ▷  ▷  -define F -finite_field -q 2 -end \
14046 ▷  ▷  -with F -do \
14047 ▷  ▷  ▷  -coding_theoretic_activity \
14048 ▷  ▷  ▷  ▷  -crc_encode_file_based \
14049 ▷  ▷  ▷  ▷  $(CRC_FILE).$(CRC_FILE_EXTENSION) \
14050 ▷  ▷  ▷  ▷  $(CRC_FILE)_crc16.bin crc16 771 \
14051 ▷  ▷  -end
14052
14053 #-rw-r--r--  1 betten  staff    646576 Aug 24 14:35 allen_Gates_crc16.bin
14054 #-rw-r--r--  1 betten  staff  21656232 Aug 24 15:35 javad-allahyari-Fs1E2JXM3Gc-u
      nsplash_crc16.bin
```

```
14055
14056
14057
14058 crc_encode_32:
14059 ▷  $(ORBITER) -v 3 \
14060 ▷  ▷  -define F -finite_field -q 2 -end \
14061 ▷  ▷  -with F -do \
14062 ▷  ▷  ▷  -coding_theoretic_activity \
14063 ▷  ▷  ▷  ▷  -crc_encode_file_based \
14064 ▷  ▷  ▷  ▷  $(CRC_FILE).$(CRC_FILE_EXTENSION) \
14065 ▷  ▷  ▷  ▷  $(CRC_FILE)_crc32.bin crc32 771 \
14066 ▷  ▷  -end
14067
14068
14069 #-rw-r--r--  1 betten  staff    648262 Aug 24 14:34 allen_Gates_crc32.bin
14070
14071
14072
14073 #crc_encode_new:
14074 #▷ $(ORBITER) -v 3 \
14075 #▷ ▷  -define F -finite_field -q 256 -end \
14076 #▷ ▷  -with F -do \
14077 #▷ ▷  ▷  -coding_theoretic_activity \
14078 #▷ ▷  ▷  -crc_new_file_based $(CRC_FILE).$(CRC_FILE_EXTENSION) \
14079 #▷ ▷  -end
14080
14081
14082 introduce_errors_16_500000:
14083 ▷  $(ORBITER) -v 3 \
14084 ▷  ▷  -introduce_errors \
14085 ▷  ▷  ▷  -input $(CRC_FILE)_crc16.bin \
14086 ▷  ▷  ▷  -output $(CRC_FILE)_crc16_e.bin \
14087 ▷  ▷  ▷  -block_based_error_generator \
14088 ▷  ▷  ▷  -block_length 771 \
14089 ▷  ▷  ▷  -threshold 500000 \
14090 ▷  ▷  ▷  -file_based_error_generator 500000 \
14091 ▷  ▷  ▷  -nb_repeats 30 \
14092 ▷  ▷  -end
14093
14094
14095 introduce_errors_32_100000:
14096 ▷  $(ORBITER) -v 3 \
14097 ▷  ▷  -introduce_errors \
14098 ▷  ▷  ▷  -input $(CRC_FILE)_crc32.bin \
14099 ▷  ▷  ▷  -output $(CRC_FILE)_crc32_e.bin \
14100 ▷  ▷  ▷  -block_based_error_generator \
14101 ▷  ▷  ▷  -block_length 771 \
14102 ▷  ▷  ▷  -threshold 100000 \
14103 ▷  ▷  ▷  -file_based_error_generator 100000 \
14104 ▷  ▷  ▷  -nb_repeats 30 \
14105 ▷  ▷  -end
14106
14107
14108 check_errors_16:
14109 ▷  $(ORBITER) -v 3 \
14110 ▷  ▷  -check_errors \
14111 ▷  ▷  ▷  -input $(CRC_FILE)_crc16_e.bin \
14112 ▷  ▷  ▷  -output $(CRC_FILE)_recovered.$(CRC_FILE_EXTENSION) \
14113 ▷  ▷  ▷  -crc_type crc16 \
```

```
14114 ▷  ▷  ▷  -error_log $(CRC_FILE)_crc16_e_pattern.csv \
14115 ▷  ▷  ▷  -block_length 771 \
14116 ▷  ▷  -end
14117
14118 check_errors_32:
14119 ▷  $(ORBITER) -v 3 \
14120 ▷  ▷  -check_errors \
14121 ▷  ▷  ▷  -input $(CRC_FILE)_crc32_e.bin \
14122 ▷  ▷  ▷  -output $(CRC_FILE)_recovered.$(CRC_FILE_EXTENSION) \
14123 ▷  ▷  ▷  -crc_type crc32 \
14124 ▷  ▷  ▷  -error_log $(CRC_FILE)_crc32_e_pattern.csv \
14125 ▷  ▷  ▷  -block_length 771 \
14126 ▷  ▷  -end
14127
14128
14129 extract_block:
14130 ▷  $(ORBITER) -v 3 \
14131 ▷  ▷  -extract_block \
14132 ▷  ▷  ▷  -input $(CRC_FILE)_crc32_e.bin \
14133 ▷  ▷  ▷  -output $(CRC_FILE)_recovered.$(CRC_FILE_EXTENSION) \
14134 ▷  ▷  ▷  -error_log $(CRC_FILE)_crc32_e_pattern.csv \
14135 ▷  ▷  ▷  -block_length 771 \
14136 ▷  ▷  ▷  -selected_block 813731 \
14137 ▷  ▷  -end
14138
14139
14140
14141 # alfa has d = 2, so it can detect any 1 error at the F256 level
14142
14143 crc_test_alfa:
14144 ▷  $(ORBITER) -v 3 \
14145 ▷  ▷  -crc_test "alfa" $(TEN_MILLION) 3
14146
14147
14148 #crc_codes::test_crc_object Nb_test = 10000000 k = 2 nb_undetected=89, time=0:24
14149 #crc_codes::test_crc_object Nb_test = 10000000 k = 3 nb_undetected=153, time=0:26

14150
14151
14152
14153 # bravo has d = 3, so it can detect any 2 errors at the F256 level
14154
14155 crc_test_bravo_10M_3:
14156 ▷  $(ORBITER) -v 3 \
14157 ▷  ▷  -crc_test "bravo" $(TEN_MILLION) 3
14158
14159 #crc_codes::test_crc_object Nb_test = 10000000 k = 3 nb_undetected=0, time=0:27
14160 #23754:crc_codes::test_crc_object Nb_test = 10000000 k = 552 nb_undetected=1, tim
      e=11:54
14161 #12596,0,( 737, 417, 171, 634, 158, 30, 480, 688, 94, 739, 147, 694, 724, 741, 51
      , 411, 753, 622, 592, 667, 25, 709, 55, 450, 255, 744, 28, 718, 82, 740, 372, 144
      , 435, 505, 478, 233, 325, 231, 27, 518, 492, 350, 621, 468, 297, 141, 735, 338,
      227, 434, 54, 180, 390, 245, 301, 617, 423, 408, 521, 42, 697, 719, 668, 618, 107
      , 746, 441, 376, 705, 353, 368, 254, 393, 291, 762, 188, 506, 580, 419, 525, 500,
       652, 433, 130, 52, 212, 511, 191, 182, 100, 676, 706, 248, 268, 365, 626, 303, 5
      , 490, 524, 237, 210, 326, 619, 485, 767, 432, 721, 391, 650, 88, 627, 177, 700,
      354, 682, 38, 360, 70, 636, 609, 766, 203, 7, 369, 584, 475, 84, 87, 251, 680, 47
      9, 256, 121, 473, 604, 614, 265, 585, 40, 458, 551, 163, 661, 358, 610, 349, 748,
       371, 310, 31, 136, 153, 649, 662, 547, 120, 324, 162, 671, 62, 725, 603, 645, 67
```

```
      0, 554, 364, 127, 312, 708, 386, 286, 532, 58, 658, 366, 74, 442, 699, 707, 164,
      102, 593, 657, 293, 638, 370, 716, 454, 126, 204, 232, 637, 409, 491, 666, 402, 2
      09, 639, 560, 578, 467, 24, 629, 93, 453, 352, 425, 406, 37, 684, 189, 648, 186,
      345, 691, 673, 193, 264, 247, 714, 267, 540, 215, 134, 729, 588, 295, 294, 241, 5
      77, 398, 722, 46, 214, 60, 640, 770, 477, 106, 97, 381, 280, 675, 418, 29, 385, 2
      69, 151, 665, 109, 514, 305, 377, 95, 544, 570, 552, 128, 152, 599, 397, 415, 149
      , 451, 738, 743, 302, 140, 273, 103, 284, 396, 726, 327, 221, 288, 200, 344, 285,
       112, 542, 22, 313, 263, 218, 61, 157, 362, 559, 507, 243, 664, 206, 550, 242, 24
      4, 230, 145, 761, 413, 679, 392, 447, 17, 330, 156, 279, 752, 487, 731, 651, 388,
       535, 154, 502, 108, 452, 363, 240, 569, 564, 713, 704, 495, 589, 83, 15, 483, 26
      1, 537, 595, 219, 573, 516, 252, 199, 337, 616, 339, 486, 555, 653, 728, 137, 216
      , 56, 211, 734, 449, 44, 26, 348, 426, 91, 139, 238, 539, 444, 754, 672, 695, 543
      , 659, 601, 654, 742, 300, 69, 115, 756, 122, 582, 374, 236, 656, 533, 755, 512,
      594, 404, 65, 611, 316, 703, 361, 318, 80, 92, 12, 720, 378, 763, 228, 429, 623,
      608, 712, 33, 612, 185, 747, 11, 620, 78, 234, 202, 165, 498, 173, 160, 494, 768,
       75, 306, 482, 641, 53, 222, 379, 460, 258, 567, 50, 531, 548, 207, 424, 320, 383
      , 124, 757, 249, 10, 405, 575, 597, 587, 341, 328, 336, 513, 317, 172, 681, 169,
      187, 430, 351, 523, 81, 553, 389, 496, 689, 461, 607, 359, 436, 105, 229, 8, 101,
       176, 663, 43, 701, 356, 630, 562, 678, 272, 224, 387, 271, 311, 456, 530, 401, 2
      78, 239, 520, 529, 287, 343, 59, 367, 342, 21, 503, 420, 466, 322, 146, 504, 563,
       196, 545, 558, 161, 394, 257, 49, 99, 598, 6, 77, 143, 68, 118, 133, 690, 410, 2
      35, 536, 48, 205, 270, 445, 283, 198, 660, 90, 47, 57, 727, 41, 463, 76, 581, 576
      , 192, 250, 686, 509, 34, 175, 246, 400, 168, 111, 334, 403, 517, 677, 373, 497,
      431, 470, 501, 174, 225 ),( 100, 202, 178, 171, 102, 243, 43, 238, 132, 200, 126,
       107, 61, 252, 150, 5, 18, 51, 161, 248, 230, 166, 207, 48, 138, 132, 221, 248, 2
      28, 42, 126, 117, 253, 85, 133, 165, 56, 125, 224, 92, 223, 129, 108, 179, 197, 2
      24, 163, 241, 196, 207, 53, 73, 183, 98, 172, 7, 65, 90, 35, 15, 59, 219, 95, 41,
       93, 52, 51, 19, 205, 78, 66, 6, 218, 47, 109, 228, 45, 134, 141, 173, 225, 93, 1
      94, 5, 164, 100, 199, 10, 118, 30, 163, 110, 152, 247, 212, 214, 146, 104, 164, 7
      , 117, 87, 129, 159, 235, 129, 233, 224, 72, 250, 167, 128, 208, 220, 150, 195, 4
      0, 168, 73, 21, 98, 185, 128, 27, 30, 74, 251, 134, 186, 40, 109, 138, 181, 202,
      137, 86, 72, 230, 117, 18, 123, 180, 10, 23, 122, 47, 241, 16, 193, 71, 189, 203,
       212, 68, 241, 143, 191, 62, 228, 255, 82, 12, 178, 134, 161, 178, 251, 119, 66,
      209, 33, 32, 17, 124, 47, 96, 191, 20, 10, 4, 82, 36, 65, 36, 164, 113, 230, 30,
      33, 176, 75, 54, 130, 25, 171, 227, 1, 41, 182, 164, 87, 232, 210, 74, 161, 84, 2
      4, 187, 106, 106, 52, 153, 247, 152, 252, 245, 182, 253, 213, 203, 45, 168, 193,
      190, 42, 105, 201, 1, 211, 224, 153, 189, 89, 187, 59, 238, 53, 95, 105, 103, 24,
       2, 62, 250, 57, 70, 68, 110, 45, 45, 226, 254, 194, 103, 186, 140, 209, 1, 37, 8
      7, 15, 200, 26, 22, 216, 251, 175, 137, 33, 228, 132, 152, 86, 179, 106, 68, 69,
      49, 202, 146, 67, 168, 130, 223, 55, 234, 177, 88, 146, 195, 100, 176, 210, 177,
      61, 161, 1, 58, 5, 191, 30, 49, 183, 83, 125, 13, 170, 211, 93, 110, 219, 93, 246
      , 120, 169, 97, 206, 86, 241, 208, 195, 199, 53, 31, 133, 253, 181, 247, 192, 209
      , 19, 159, 175, 39, 247, 154, 12, 225, 43, 231, 198, 83, 125, 33, 175, 79, 242, 2
      24, 150, 251, 213, 125, 155, 230, 61, 109, 32, 239, 227, 40, 8, 133, 186, 118, 23
      2, 157, 218, 41, 24, 168, 32, 205, 172, 216, 36, 94, 179, 37, 212, 13, 35, 171, 1
      6, 171, 39, 69, 241, 165, 62, 91, 136, 4, 227, 168, 53, 6, 226, 251, 58, 251, 38,
       234, 245, 255, 168, 119, 169, 47, 179, 223, 28, 32, 17, 185, 49, 44, 196, 122, 8
      9, 116, 157, 223, 252, 204, 40, 119, 89, 8, 7, 178, 235, 52, 12, 51, 217, 60, 104
      , 100, 59, 111, 14, 244, 49, 223, 225, 245, 187, 61, 242, 174, 29, 145, 172, 177,
       131, 188, 157, 76, 104, 244, 94, 39, 96, 246, 131, 77, 255, 70, 211, 201, 166, 1
      05, 68, 129, 4, 80, 20, 89, 58, 222, 159, 208, 82, 255, 245, 121, 210, 149, 202,
      105, 212, 42, 35, 188, 3, 121, 23, 139, 50, 24, 224, 20, 140, 119, 55, 118, 53, 1
      55, 19, 70, 138, 178, 202, 153, 78, 44, 90, 109, 32, 253, 64, 130, 238, 243, 87,
      12, 204, 63, 116, 163, 82, 25, 14, 31, 197, 193, 203, 235, 79, 86, 118, 118, 88,
      190, 26, 55, 179, 156, 153, 89, 227, 221 )
14162 #crc_codes::test_crc_object Nb_test = 10000000 k = 552 nb_undetected=1, time=11:5
      4
14163 #crc_codes::test_crc_object done
14164 #interface_coding_theory::worker done
```

```
14165
14166
14167 error_polynomial_bravo_552:
14168 ▷  $(ORBITER) -v 3 \
14169 ▷  ▷  -define F -finite_field -q 256 -end \
14170 ▷  ▷  -define Coeff -vector -dense $(ERROR_POLY_BRAVO_COEFF) -end \
14171 ▷  ▷  -define Place -vector -dense $(ERROR_POLY_BRAVO_PLACE) -end \
14172 ▷  ▷  -with F -do -finite_field_activity \
14173 ▷  ▷  ▷  -assemble_monopoly 771 Coeff Place \
14174 ▷  ▷  -end
14175
14176
14177
14178 crc_test_crc32_10M_3:
14179 ▷  $(ORBITER) -v 3 \
14180 ▷  ▷  -crc_test "crc32" $(TEN_MILLION) 3
14181
14182
14183 #crc_codes::test_crc_object Nb_test = 10000000 k = 3 nb_undetected=0, time=3:31
14184
14185
14186
14187 crc_test_bravo_10M_4:
14188 ▷  $(ORBITER) -v 3 \
14189 ▷  ▷  -crc_test "bravo" $(TEN_MILLION) 4
14190
14191 #crc_codes::test_crc_object Nb_test = 10000000 k = 4 nb_undetected=0, time=0:28
14192
14193 crc_test_crc32_10M_4:
14194 ▷  $(ORBITER) -v 3 \
14195 ▷  ▷  -crc_test "crc32" $(TEN_MILLION) 4
14196
14197 #crc_codes::test_crc_object Nb_test = 10000000 k = 4 nb_undetected=0, time=3:36
14198
14199
14200 crc_test_bravo_10M_5:
14201 ▷  $(ORBITER) -v 3 \
14202 ▷  ▷  -crc_test "bravo" $(TEN_MILLION) 5
14203
14204
14205 #crc_codes::test_crc_object Nb_test = 10000000 k = 5 nb_undetected=0, time=0:30
14206
14207
14208 crc_test_crc32_10M_5:
14209 ▷  $(ORBITER) -v 3 \
14210 ▷  ▷  -crc_test "crc32" $(TEN_MILLION) 5
14211
14212 #crc_codes::test_crc_object Nb_test = 10000000 k = 5 nb_undetected=0, time=3:41
14213
14214
14215
14216
14217 crc_test_bravo_10M_6:
14218 ▷  $(ORBITER) -v 3 \
14219 ▷  ▷  -crc_test "bravo" $(TEN_MILLION) 6
14220
14221
14222 #crc_codes::test_crc_object Nb_test = 10000000 k = 6 nb_undetected=0, time=0:30
14223
```

```
14224
14225 crc_test_crc32_10M_6:
14226 ▷  $(ORBITER) -v 3 \
14227 ▷  ▷  -crc_test "crc32" $(TEN_MILLION) 6
14228
14229 #crc_codes::test_crc_object Nb_test = 10000000 k = 6 nb_undetected=0, time=3:45
14230
14231
14232
14233 crc_test_bravo_10M_7:
14234 ▷  $(ORBITER) -v 3 \
14235 ▷  ▷  -crc_test "bravo" $(TEN_MILLION) 7
14236
14237
14238 #crc_codes::test_crc_object Nb_test = 10000000 k = 7 nb_undetected=0, time=0:31
14239
14240
14241 crc_test_crc32_10M_7:
14242 ▷  $(ORBITER) -v 3 \
14243 ▷  ▷  -crc_test "crc32" $(TEN_MILLION) 7
14244
14245 #crc_codes::test_crc_object Nb_test = 10000000 k = 7 nb_undetected=0, time=3:47
14246
14247
14248 crc_test_bravo_10M_8:
14249 ▷  $(ORBITER) -v 3 \
14250 ▷  ▷  -crc_test "bravo" $(TEN_MILLION) 8
14251
14252
14253 #crc_codes::test_crc_object Nb_test = 10000000 k = 8 nb_undetected=0, time=0:32
14254
14255 crc_test_crc32_10M_8:
14256 ▷  $(ORBITER) -v 3 \
14257 ▷  ▷  -crc_test "crc32" $(TEN_MILLION) 8
14258
14259 #crc_codes::test_crc_object Nb_test = 10000000 k = 8 nb_undetected=0, time=3:50
14260
14261
14262 crc_test_bravo_10_loop:
14263 ▷  $(ORBITER) -v 3 \
14264 ▷  ▷  -loop i 2 770 1 \
14265 ▷  ▷  ▷  -crc_test "bravo" $(TEN_MILLION) %i \
14266 ▷  ▷  -end_loop i
14267
14268
14269 crc_test_crc32_10_loop:
14270 ▷  $(ORBITER) -v 3 \
14271 ▷  ▷  -loop i 2 770 1 \
14272 ▷  ▷  ▷  -crc_test "crc32" $(TEN_MILLION) %i \
14273 ▷  ▷  -end_loop i
14274
14275
14276 #### 20 Million:
14277
14278
14279
14280 crc_test_crc32_20M_3:
14281 ▷  $(ORBITER) -v 3 \
14282 ▷  ▷  -crc_test "crc32" 20000000 3
```

```
14283
14284
14285
14286
14287 # charlie has d = 7, so it can detect any 6 errors at the F256 level
14288
14289
14290 crc_test_charlie:
14291 ▷  $(ORBITER) -v 3 \
14292 ▷  ▷  -crc_test "charlie" 10000000 7
14293
14294
14295
14296
14297 ################################################################################
14298 # Section 10.5: Coding Theory - Reed-Muller codes
14299
14300 SECTION_CODING_THEORY_REED_MULLER_CODES:
14301
14302
14303
14304
14305 test_10_5:
14306 ▷  make RM_3_export
14307 ▷  make RM_3_Hamming_space_diagram
14308 ▷  make RM_3_draw
14309 ▷  make RM_3_split
14310 ▷  make RM_3_distance_draw
14311 ▷  make RM_3_algebraic_normal_form
14312 ▷  make RM_6
14313 ▷  make RM_6_genma_draw
14314 ▷  make RM_6_codewords_matrix_draw
14315 ▷  make RM_6_codewords_draw
14316
14317
14318
14319 RM_3_export:
14320 ▷  $(ORBITER) -v 2 \
14321 ▷  ▷  -define F -finite_field -q 2 -end \
14322 ▷  ▷  -define C -code -field F \
14323 ▷  ▷  ▷  -Reed_Muller 3 \
14324 ▷  ▷  -end \
14325 ▷  ▷  -with C -and F -do -coding_theoretic_activity \
14326 ▷  ▷  ▷  -export_magma RM_3.magma \
14327 ▷  ▷  -end \
14328 ▷  ▷  -with C -and F -do -coding_theoretic_activity \
14329 ▷  ▷  ▷  -export_codewords RM_3_codewords.csv \
14330 ▷  ▷  -end
14331
14332
14333
14334
14335 RM_3_Hamming_space_diagram:
14336 ▷  $(ORBITER) -v 2 \
14337 ▷  ▷  -define F -finite_field -q 2 -end \
14338 ▷  ▷  -define C -code -field F \
14339 ▷  ▷  ▷  -Reed_Muller 3 \
14340 ▷  ▷  -end \
14341 ▷  ▷  -with C -do \
```

```
14342 ▷  ▷  -coding_theoretic_activity \
14343 ▷  ▷  ▷  -make_diagram \
14344 ▷  ▷  ▷  -metric_balls 1 \
14345 ▷  ▷  -end
14346
14347 # creates
14348 # RM_3_8_16.tex
14349 # RM_3_diagram_8_16.csv
14350 # RM_3_diagram_01_8_16.csv
14351 # RM_3_distance_8_16.csv
14352
14353
14354 RM_3_draw:
14355 ▷  $(ORBITER) -v 2 \
14356 ▷  ▷  -draw_matrix \
14357 ▷  ▷  ▷  -input_csv_file RM_3_distance_H_8_16.csv \
14358 ▷  ▷  ▷  -box_width 25 -bit_depth 8 \
14359 ▷  ▷  ▷  -partition 4 16 16 \
14360 ▷  ▷  -end
14361 ▷  $(ORBITER) -v 2 \
14362 ▷  ▷  -draw_matrix \
14363 ▷  ▷  ▷  -input_csv_file RM_3_char_func_8_16.csv \
14364 ▷  ▷  ▷  -box_width 25 -bit_depth 8 \
14365 ▷  ▷  ▷  -partition 4 16 16 \
14366 ▷  ▷  -end
14367 ▷  $(ORBITER) -v 2 \
14368 ▷  ▷  -draw_matrix \
14369 ▷  ▷  ▷  -input_csv_file RM_3_idx_8_16.csv \
14370 ▷  ▷  ▷  -box_width 25 -bit_depth 8 \
14371 ▷  ▷  ▷  -partition 4 16 16 \
14372 ▷  ▷  -end
14373 ▷  convert RM_3_distance_H_8_16_draw.bmp RM_3_distance_H_8_16_draw.png
14374 ▷  convert RM_3_char_func_8_16_draw.bmp RM_3_char_func_8_16_draw.png
14375 ▷  convert RM_3_idx_8_16_draw.bmp RM_3_idx_8_16_draw.png
14376 ▷  $(OPEN) RM_3_distance_H_8_16_draw.bmp
14377
14378
14379 RM_3_split:
14380 ▷  #$(ORBITER) -split_by_values RM_3_distance_8_16.csv
14381 ▷  $(ORBITER) -split_by_values RM_3_distance_H_8_16.csv
14382
14383 RM_3_distance_draw:
14384 ▷  $(ORBITER) -v 2 \
14385 ▷  ▷  -loop L 0 3 1 \
14386 ▷  ▷  ▷  -draw_matrix \
14387 ▷  ▷  ▷  ▷  -input_csv_file RM_3_distance_H_8_16_value%L.csv \
14388 ▷  ▷  ▷  ▷  -box_width 25 -bit_depth 8 \
14389 ▷  ▷  ▷  ▷  -partition 5 16 16 \
14390 ▷  ▷  ▷  -end \
14391 ▷  ▷  -end_loop L
14392 ▷  convert RM_3_distance_H_8_16_value0_draw.bmp RM_3_distance_H_8_16_value0_draw.p
      ng
14393 ▷  convert RM_3_distance_H_8_16_value1_draw.bmp RM_3_distance_H_8_16_value1_draw.p
      ng
14394 ▷  convert RM_3_distance_H_8_16_value2_draw.bmp RM_3_distance_H_8_16_value2_draw.p
      ng
14395
14396 RM_3_algebraic_normal_form:
14397 ▷  $(ORBITER) -v 3 \
```

```
14398 ▷  ▷  -define F -finite_field -q 2 -end \
14399 ▷  ▷  -loop L 0 3 1 \
14400 ▷  ▷  ▷  -define v%L -vector -field F \
14401 ▷  ▷  ▷  ▷  -file RM_3_distance_8_16_value%L.csv \
14402 ▷  ▷  ▷  -end \
14403 ▷  ▷  ▷  -with F -do -finite_field_activity \
14404 ▷  ▷  ▷  ▷  -algebraic_normal_form 8 v%L \
14405 ▷  ▷  ▷  -end \
14406 ▷  ▷  -end_loop L
14407
14408
14409 RM_6:
14410 ▷  $(ORBITER) -v 2 \
14411 ▷  ▷  -define F -finite_field -q 2 -end \
14412 ▷  ▷  -define C -code -field F \
14413 ▷  ▷  ▷  -long_code 64 7 \
14414 ▷  ▷  ▷  $(RM_6_GENERATOR_1) \
14415 ▷  ▷  ▷  $(RM_6_GENERATOR_2) \
14416 ▷  ▷  ▷  $(RM_6_GENERATOR_3) \
14417 ▷  ▷  ▷  $(RM_6_GENERATOR_4) \
14418 ▷  ▷  ▷  $(RM_6_GENERATOR_5) \
14419 ▷  ▷  ▷  $(RM_6_GENERATOR_6) \
14420 ▷  ▷  ▷  $(RM_6_GENERATOR_7) \
14421 ▷  ▷  -end \
14422 ▷  ▷  -with C -and F -do -coding_theoretic_activity \
14423 ▷  ▷  ▷  -export_genma RM6_genma.csv \
14424 ▷  ▷  -end \
14425 ▷  ▷  -with C -and F -do -coding_theoretic_activity \
14426 ▷  ▷  ▷  -export_codewords_long RM6_codewords.csv \
14427 ▷  ▷  -end \
14428
14429 RM_6_genma_draw:
14430 ▷  $(ORBITER) -v 2 \
14431 ▷  ▷  -draw_matrix \
14432 ▷  ▷  ▷  -input_csv_file RM6_genma.csv \
14433 ▷  ▷  ▷  -box_width 25 -bit_depth 8 \
14434 ▷  ▷  ▷  -partition 3 7 64 \
14435 ▷  ▷  -end
14436 ▷  convert RM6_genma_draw.bmp RM6_genma_draw.png
14437 ▷  $(OPEN) RM6_genma_draw.png
14438
14439
14440 RM_6_codewords_matrix_draw:
14441 ▷  $(ORBITER) -v 2 \
14442 ▷  ▷  -draw_matrix \
14443 ▷  ▷  ▷  -input_csv_file RM6_codewords.csv \
14444 ▷  ▷  ▷  -box_width 25 -bit_depth 8 \
14445 ▷  ▷  ▷  -partition 3 128 64 \
14446 ▷  ▷  -end
14447 ▷  convert RM6_codewords_draw.bmp RM6_codewords_draw.png
14448 ▷  $(OPEN) RM6_codewords_draw.png
14449
14450
14451 RM_6_codewords_draw:
14452 ▷  $(ORBITER) -v 4 \
14453 ▷  ▷  -loop i 0 128 1 \
14454 ▷  ▷  ▷  -csv_file_select_rows RM6_codewords.csv %i \
14455 ▷  ▷  ▷  -reformat RM6_codewords_select_%i.csv RM6_codewords_ref_%i.csv 8 \
14456 ▷  ▷  ▷  -draw_matrix \
```

```
14457 ▷  ▷  ▷  ▷  -input_csv_file RM6_codewords_ref_%i.csv \
14458 ▷  ▷  ▷  ▷  -box_width 25 -bit_depth 8 \
14459 ▷  ▷  ▷  ▷  -partition 3 8 8 \
14460 ▷  ▷  ▷  -end \
14461 ▷  ▷  ▷  -system "convert RM6_codewords_ref_%i_draw.bmp -frame 8 %i.png" \
14462 ▷  ▷  -end_loop i
14463 ▷  convert 0.png 1.png 2.png 3.png 4.png 5.png \
14464 ▷  ▷  6.png 7.png 8.png 9.png 10.png +append a0
14465 ▷  convert 11.png 12.png 13.png 14.png 15.png \
14466 ▷  ▷  16.png 17.png 18.png 19.png 20.png 21.png +append a1
14467 ▷  convert 22.png 23.png 24.png 25.png 26.png \
14468 ▷  ▷  27.png 28.png 29.png 30.png 31.png 32.png +append a2
14469 ▷  convert 33.png 34.png 35.png 36.png 37.png \
14470 ▷  ▷  38.png 39.png 40.png 41.png 42.png 43.png +append a3
14471 ▷  convert 44.png 45.png 46.png 47.png 48.png \
14472 ▷  ▷  49.png 50.png 51.png 52.png 53.png 54.png +append a4
14473 ▷  convert 55.png 56.png 57.png 58.png 59.png \
14474 ▷  ▷  60.png 61.png 62.png 63.png 64.png 65.png +append a5
14475 ▷  convert 66.png 67.png 68.png 69.png 70.png \
14476 ▷  ▷  71.png 72.png 73.png 74.png 75.png 76.png +append a6
14477 ▷  convert 77.png 78.png 79.png 80.png 81.png \
14478 ▷  ▷  82.png 83.png 84.png 85.png 86.png 87.png +append a7
14479 ▷  convert 88.png 89.png 90.png 91.png 92.png \
14480 ▷  ▷  93.png 94.png 95.png 96.png 97.png 98.png +append a8
14481 ▷  convert 99.png 100.png 101.png 102.png 103.png \
14482 ▷  ▷  104.png 105.png 106.png 107.png 108.png 109.png +append a9
14483 ▷  convert 110.png 111.png 112.png 113.png 114.png \
14484 ▷  ▷  115.png 116.png 117.png 118.png 119.png 120.png +append a10
14485 ▷  convert 121.png 122.png 123.png 124.png 125.png \
14486 ▷  ▷  126.png 127.png 0.png 0.png 0.png 0.png +append a11
14487 ▷  convert a0 a1 a2 a3 a4 a5 a6 a7 a8 a9 a10 \
14488 ▷  ▷  a11 -append poster_RM_1_6.png
14489
14490
14491
14492
14493
14494 ##############################################################################
14495 # Section 10.6: Coding Theory - BCH Codes
14496
14497 SECTION_CODING_THEORY_BCH_CODES:
14498
14499
14500 test_10_6:
14501 ▷  make draw_cyclotomic_mod_51_q16
14502 ▷  make draw_cyclotomic_mod_21_q8
14503 ▷  make F_8_BCH_code_d3
14504 ▷  make F_8_BCH_code_d4
14505 ▷  make F_8_BCH_code_d5
14506 ▷  make F_8_BCH_code_d5_minimum_distance
14507 ▷  make F_8_BCH_code_d7
14508 ▷  make F8_BCH_code_n63_d43
14509 ▷  #make F8_BCH_code_n63_d43_minimum_distance
14510 ▷  make F2_BCH_code_n21
14511 ▷  make F7_RS_code_n6
14512 ▷  make F_64_again
14513 ▷  make BCH_15_5
14514 ▷  make draw_mod_31
14515 ▷  make PR127
```

```
14516 ▷  make draw_mod_127_power
14517 ▷  make draw_mod_251
14518 ▷  make draw_mod_255_cyclotomic_1
14519 ▷  make draw_mod_255_cyclotomic_3
14520 ▷  make draw_mod_255_cyclotomic_1_and_3
14521 ▷  make draw_mod_63_4_cyclotomic_3_6
14522 ▷  make BCH_F_64
14523 ▷  make BCH_21_poly_mult_mod_F4
14524 ▷  make BCH_21_poly_division_a
14525 ▷  make BCH_21_poly_division_b
14526 ▷  make BCH_21_poly_division_ab
14527 ▷  make BCH_21_generator_matrix
14528 ▷  #make BCH_21_15_weight_enumerator
14529 ▷  make BCH_21_15_dual
14530 ▷  make BCH_21_6_weight_enumerator
14531 ▷  make BCH_21_6_4_macwilliams
14532 ▷  make BCH_21_15_4_field_reduction
14533 ▷  make BCH_21_poly_division_c
14534 ▷  make F16_roots_5
14535 ▷  make F64_roots_21
14536 ▷  make BCH_F256_roots_771
14537 ▷  make BCH_F256_BCH_code_d16
14538
14539
14540 draw_cyclotomic_mod_51_q16:
14541 ▷  $(ORBITER) -v 2 \
14542 ▷  ▷  -draw_options \
14543 ▷  ▷  ▷  -radius 100 \
14544 ▷  ▷  ▷  -line_width 1.0 -embedded \
14545 ▷  ▷  -end \
14546 ▷  ▷  -draw_mod_n \
14547 ▷  ▷  ▷  -n 51 \
14548 ▷  ▷  ▷  -cyclotomic_sets 16 "1,2" \
14549 ▷  ▷  ▷  -file cyclotomic_51 \
14550 ▷  ▷  -end
14551 ▷  pdflatex cyclotomic_51_draw.tex
14552 ▷  $(OPEN) cyclotomic_51_draw.pdf
14553
14554 #▷ ▷  ▷  -cyclotomic_sets 8 "1,2,4,5,7,10,13" \
14555
14556
14557 draw_cyclotomic_mod_21_q8:
14558 ▷  $(ORBITER) -v 2 \
14559 ▷  ▷  -draw_options \
14560 ▷  ▷  ▷  -radius 100 \
14561 ▷  ▷  ▷  -line_width 1.0 -embedded \
14562 ▷  ▷  -end \
14563 ▷  ▷  -draw_mod_n \
14564 ▷  ▷  ▷  -n 21 \
14565 ▷  ▷  ▷  -file mod_21_cyclotomic \
14566 ▷  ▷  ▷  -cyclotomic_sets 8 "1,2,4,5,7,10,13" \
14567 ▷  ▷  -end
14568 ▷  pdflatex mod_21_cyclotomic_draw.tex
14569 ▷  $(OPEN) mod_21_cyclotomic_draw.pdf
14570
14571
14572 F_8_BCH_code_d3:
14573 ▷  $(ORBITER) -v 3 \
14574 ▷  ▷  -define F -finite_field -q 8 -override_polynomial 11 -end \
```

```
14575 ▷  ▷  -define C -code -field F \
14576 ▷  ▷  ▷  -BCH 21 3 \
14577 ▷  ▷  -end
14578 ▷  pdflatex BCH_codes_q8_n21_d3.tex
14579 ▷  $(OPEN) BCH_codes_q8_n21_d3.pdf
14580
14581 #generator polynomial is X^{4} + 4X^{3} + 4X^{2} + 3X + 4
14582
14583 F_8_BCH_code_d4:
14584 ▷  $(ORBITER) -v 3 \
14585 ▷  ▷  -define F -finite_field -q 8 -override_polynomial 11 -end \
14586 ▷  ▷  -define C -code -field F \
14587 ▷  ▷  ▷  -BCH 21 4 \
14588 ▷  ▷  -end
14589
14590 #generator polynomial is X^{5} + 6X^{4} + 7X^{3} + 2X + 3
14591
14592
14593 F_8_BCH_code_d5:
14594 ▷  $(ORBITER) -v 3 \
14595 ▷  ▷  -define F -finite_field -q 8 -override_polynomial 11 -end \
14596 ▷  ▷  -define C -code -field F \
14597 ▷  ▷  ▷  -BCH 21 5 \
14598 ▷  ▷  -end
14599 ▷  pdflatex BCH_codes_q8_n21_d5.tex
14600 ▷  $(OPEN) BCH_codes_q8_n21_d5.pdf
14601
14602 #-override_polynomial 11
14603 #generator polynomial is X^{7} + 3X^{6} + 3X^{5} + 2X^{4} + X^{3} + 2X^{2} + X +
      2
14604
14605 F_8_BCH_code_d5_minimum_distance:
14606 ▷  $(ORBITER) -v 2 \
14607 ▷  ▷  -define F -finite_field -q 8 -override_polynomial 11 -end \
14608 ▷  ▷  -define v -vector -format 14 -field F \
14609 ▷  ▷  ▷  -compact $(CODE_BCH_F8_N21_D5_GENMA_OVERRIDE_POLYNOMIAL11) \
14610 ▷  ▷  -end \
14611 ▷  ▷  -with F -do \
14612 ▷  ▷  -coding_theoretic_activity \
14613 ▷  ▷  ▷  -minimum_distance v \
14614 ▷  ▷  -end
14615 # important: use the same polynomial as when creating the code.
14616 #
14617 # d=5
14618
14619
14620
14621 F_8_BCH_code_d7:
14622 ▷  $(ORBITER) -v 3 \
14623 ▷  ▷  -define F -finite_field -q 8 -override_polynomial 11 -end \
14624 ▷  ▷  -define C -code -field F \
14625 ▷  ▷  ▷  -BCH 21 7 \
14626 ▷  ▷  -end
14627
14628
14629
14630
14631 F8_BCH_code_n63_d43:
14632 ▷  $(ORBITER) -v 3 \
```

```
14633 ▷  ▷  -define F -finite_field -q 8 -override_polynomial 11 -end \
14634 ▷  ▷  -define C -code -field F \
14635 ▷  ▷  ▷  -BCH 63 43 \
14636 ▷  ▷  -end
14637 ▷  pdflatex BCH_codes_q8_n63_d43.tex
14638 ▷  $(OPEN) BCH_codes_q8_n63_d43.pdf
14639
14640
14641
14642
14643
14644 F8_BCH_code_n63_d43_minimum_distance:
14645 ▷  $(ORBITER) -v 2 \
14646 ▷  ▷  -define F -finite_field -q 8 -override_polynomial 11 -end \
14647 ▷  ▷  -define v -vector -format 9 -field F \
14648 ▷  ▷  ▷  -compact $(CODE_BCH_F8_N63_K9_D43_GENMA) \
14649 ▷  ▷  -end \
14650 ▷  ▷  -with F -do \
14651 ▷  ▷  -coding_theoretic_activity \
14652 ▷  ▷  ▷  -minimum_distance v \
14653 ▷  ▷  -end
14654
14655
14656 #coding_theory_domain::do_minimum_distance The minimum distance is d = 45, comput
      ed in 0 days, 0 hours, 1 minutes, 32 seconds
14657 #1:32
14658
14659
14660
14661
14662
14663
14664 F2_BCH_code_n21:
14665 ▷  $(ORBITER) -v 3 \
14666 ▷  ▷  -define F -finite_field -q 2 -end \
14667 ▷  ▷  -define C -code -field F \
14668 ▷  ▷  ▷  -BCH 21 3 \
14669 ▷  ▷  -end
14670
14671
14672
14673 F7_RS_code_n6:
14674 ▷  $(ORBITER) -v 3 \
14675 ▷  ▷  -define F -finite_field -q 7 -end \
14676 ▷  ▷  -define C -code -field F \
14677 ▷  ▷  ▷  -Reed_Solomon 6 3 \
14678 ▷  ▷  -end
14679
14680
14681 F_64_again:
14682 ▷  $(ORBITER) -v 3 \
14683 ▷  ▷  -define F -finite_field -q 64 -end \
14684 ▷  ▷  -with F -do \
14685 ▷  ▷  ▷  -finite_field_activity \
14686 ▷  ▷  ▷  -cheat_sheet_GF \
14687 ▷  ▷  -end
14688 ▷  pdflatex GF_64.tex
14689 ▷  $(OPEN) GF_64.pdf
14690
```

```
14691
14692
14693 BCH_15_5:
14694 ▷  $(ORBITER) \
14695 ▷  ▷  -define F -finite_field -q 2 -end \
14696 ▷  ▷  -define C -code -field F \
14697 ▷  ▷  ▷  -BCH 15 5 \
14698 ▷  ▷  -end
14699
14700
14701
14702
14703
14704
14705 draw_mod_31:
14706 ▷  $(ORBITER) -v 2 \
14707 ▷  ▷  -draw_options -embedded -end \
14708 ▷  ▷  -draw_mod_n \
14709 ▷  ▷  ▷  -n 31 \
14710 ▷  ▷  ▷  -file mod_31 \
14711 ▷  ▷  ▷  -draw_mod_n_power_cycle 2 \
14712 ▷  ▷  -end
14713 ▷  pdflatex mod_31_draw.tex
14714 ▷  $(OPEN) mod_31_draw.pdf
14715
14716
14717 PR127:
14718 ▷  $(ORBITER) -v 5 -primitive_root 127
14719
14720
14721 draw_mod_127_power:
14722 ▷  $(ORBITER) -v 2 \
14723 ▷  ▷  -draw_options -scale 0.4 -embedded -end \
14724 ▷  ▷  -draw_mod_n \
14725 ▷  ▷  ▷  -n 127 \
14726 ▷  ▷  ▷  -file mod_127 \
14727 ▷  ▷  ▷  -power_cycle 3 \
14728 ▷  ▷  -end
14729 ▷  pdflatex mod_127_draw.tex
14730 ▷  $(OPEN) mod_127_draw.pdf
14731
14732 draw_mod_251:
14733 ▷  $(ORBITER) -v 2 \
14734 ▷  ▷  -draw_options -nodes_empty -radius 10 -embedded -end \
14735 ▷  ▷  -draw_mod_n \
14736 ▷  ▷  ▷  -n 251 \
14737 ▷  ▷  ▷  -file mod_251 \
14738 ▷  ▷  -end
14739 ▷  pdflatex mod_251_draw.tex
14740 ▷  $(OPEN) mod_251_draw.pdf
14741
14742 #-draw_mod_n_inverse
14743
14744
14745 draw_mod_255_cyclotomic_1:
14746 ▷  $(ORBITER) -v 2 \
14747 ▷  ▷  -draw_options -nodes_empty -radius 10 \
14748 ▷  ▷  ▷  -line_width 0.4 -embedded -end \
14749 ▷  ▷  -draw_mod_n \
```

```
14750 ▷  ▷  ▷  -n 255 \
14751 ▷  ▷  ▷  -file mod_255_cyclotomic_1 \
14752 ▷  ▷  ▷  -cyclotomic_sets 2 "1" \
14753 ▷  ▷  -end
14754 ▷  pdflatex mod_255_cyclotomic_1_draw.tex
14755 ▷  $(OPEN) mod_255_cyclotomic_1_draw.pdf
14756
14757 draw_mod_255_cyclotomic_3:
14758 ▷  $(ORBITER) -v 2 \
14759 ▷  ▷  -draw_options -nodes_empty -radius 10 \
14760 ▷  ▷  ▷  -line_width 0.4 -embedded -end \
14761 ▷  ▷  -draw_mod_n \
14762 ▷  ▷  ▷  -n 255 \
14763 ▷  ▷  ▷  -file mod_255_cyclotomic_3 \
14764 ▷  ▷  ▷  -cyclotomic_sets 2 "3" \
14765 ▷  ▷  -end
14766 ▷  pdflatex mod_255_cyclotomic_3_draw.tex
14767 ▷  $(OPEN) mod_255_cyclotomic_3_draw.pdf
14768
14769 draw_mod_255_cyclotomic_1_and_3:
14770 ▷  $(ORBITER) -v 2 \
14771 ▷  ▷  -draw_options -nodes_empty -radius 10 \
14772 ▷  ▷  ▷  -line_width 0.4 -embedded -end \
14773 ▷  ▷  -draw_mod_n \
14774 ▷  ▷  ▷  -n 255 \
14775 ▷  ▷  ▷  -file mod_255_cyclotomic_1_and_3 \
14776 ▷  ▷  ▷  -cyclotomic_sets 2 "1,3" \
14777 ▷  ▷  -end
14778 ▷  pdflatex mod_255_cyclotomic_1_and_3_draw.tex
14779 ▷  $(OPEN) mod_255_cyclotomic_1_and_3_draw.pdf
14780
14781 draw_mod_63_4_cyclotomic_3_6:
14782 ▷  $(ORBITER) -v 2 \
14783 ▷  ▷  -draw_options -radius 20 \
14784 ▷  ▷  ▷  -line_width 0.1 -embedded -end \
14785 ▷  ▷  -draw_mod_n \
14786 ▷  ▷  ▷  -n 63 \
14787 ▷  ▷  ▷  -file mod_63_4_cyclotomic_3_6 \
14788 ▷  ▷  ▷  -cyclotomic_sets 4 "3,6" \
14789 ▷  ▷  ▷  -cyclotomic_sets_thickness 50 \
14790 ▷  ▷  -end
14791 ▷  pdflatex mod_63_4_cyclotomic_3_6_draw.tex
14792 ▷  $(OPEN) mod_63_4_cyclotomic_3_6_draw.pdf
14793
14794 BCH_F_64:
14795 ▷  $(ORBITER) -v 3 \
14796 ▷  ▷  -define F -finite_field -q 64 -end \
14797 ▷  ▷  -with F -do -finite_field_activity \
14798 ▷  ▷  ▷  -cheat_sheet_GF \
14799 ▷  ▷  -end
14800 ▷  pdflatex GF_64.tex
14801
14802
14803
14804
14805 BCH_21_poly_mult_mod_F4:
14806 ▷  $(ORBITER) -v 2 \
14807 ▷  ▷  -define F -finite_field -q 4 -end \
14808 ▷  ▷  -with F -do \
```

```
14809 ▷  ▷  -finite_field_activity \
14810 ▷  ▷  ▷  -polynomial_mult_mod "1,0,2,1" "1,0,3,1" \
14811 ▷  ▷  ▷  "1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1" \
14812 ▷  ▷  -end
14813
14814 #C(X)=X^{6} + X^{5} + X^{4} + X^{2} + 1
14815
14816 # poly 1,0,1,0,1,1,1
14817
14818
14819 BCH_21_poly_division_a:
14820 ▷  $(ORBITER) -v 2 \
14821 ▷  ▷  -define F -finite_field -q 4 -end \
14822 ▷  ▷  -with F -do \
14823 ▷  ▷  -finite_field_activity \
14824 ▷  ▷  -polynomial_division \
14825 ▷  ▷  ▷  "1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1" \
14826 ▷  ▷  ▷  "1,0,2,1" \
14827 ▷  ▷  -end
14828
14829 BCH_21_poly_division_b:
14830 ▷  $(ORBITER) -v 2 \
14831 ▷  ▷  -define F -finite_field -q 4 -end \
14832 ▷  ▷  -with F -do \
14833 ▷  ▷  -finite_field_activity \
14834 ▷  ▷  -polynomial_division \
14835 ▷  ▷  ▷  "1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1" \
14836 ▷  ▷  ▷  "1,0,3,1" \
14837 ▷  ▷  -end
14838
14839
14840 BCH_21_poly_division_ab:
14841 ▷  $(ORBITER) -v 2 \
14842 ▷  ▷  -define F -finite_field -q 4 -end \
14843 ▷  ▷  -with F -do \
14844 ▷  ▷  -finite_field_activity \
14845 ▷  ▷  -polynomial_division \
14846 ▷  ▷  ▷  "1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1" \
14847 ▷  ▷  ▷  "1,0,1,0,1,1,1" \
14848 ▷  ▷  -end
14849
14850 BCH_21_generator_matrix:
14851 ▷  $(ORBITER) -v 2 \
14852 ▷  ▷  -define F -finite_field -q 4 -end \
14853 ▷  ▷  -with F -do \
14854 ▷  ▷  -coding_theoretic_activity \
14855 ▷  ▷  ▷  -generator_matrix_cyclic_code \
14856 ▷  ▷  ▷  ▷  21 "1,0,1,0,1,1,1" \
14857 ▷  ▷  -end
14858
14859
14860
14861
14862 BCH_21_15_weight_enumerator:
14863 ▷  $(ORBITER) -v 2 \
14864 ▷  ▷  -define F -finite_field -q 4 -end \
14865 ▷  ▷  -define v -vector -field F -format 15 \
14866 ▷  ▷  ▷  -dense $(BCH_21_15_GENERATOR_MATRIX) \
14867 ▷  ▷  -end \
```

```
14868 ▷  ▷  -define C -code -field F \
14869 ▷  ▷  ▷  -generator_matrix v \
14870 ▷  ▷  -end \
14871 ▷  ▷  -with C -do \
14872 ▷  ▷  -coding_theoretic_activity \
14873 ▷  ▷  ▷  -weight_enumerator \
14874 ▷  ▷  -end
14875
14876 # too slow!
14877
14878 BCH_21_15_dual:
14879 ▷  $(ORBITER) -v 2 \
14880 ▷  ▷  -define F -finite_field -q 4 -end \
14881 ▷  ▷  -define v -vector -field F -format 15 \
14882 ▷  ▷  ▷  -dense $(BCH_21_15_GENERATOR_MATRIX) -end \
14883 ▷  ▷  -with F -do -finite_field_activity \
14884 ▷  ▷  ▷  -nullspace v \
14885 ▷  ▷  -end
14886
14887
14888 BCH_21_6_weight_enumerator:
14889 ▷  $(ORBITER) -v 2 \
14890 ▷  ▷  -define F -finite_field -q 4 -end \
14891 ▷  ▷  -define v -vector -format 6 -field F \
14892 ▷  ▷  ▷  -dense $(BCH_21_6_GENERATOR_MATRIX) \
14893 ▷  ▷  -end \
14894 ▷  ▷  -define C -code -field F \
14895 ▷  ▷  ▷  -generator_matrix v \
14896 ▷  ▷  -end \
14897 ▷  ▷  -with C -do \
14898 ▷  ▷  -coding_theoretic_activity \
14899 ▷  ▷  ▷  -weight_enumerator \
14900 ▷  ▷  -end
14901
14902 # 1y^{21} + 63x^8y^{13} + 294x^{12}y^9 + 756x^{14}y^7 + 1890x^{16}y^5 + 1092x^{18
      }y^3
14903
14904 #( 1, 0, 0, 0, 0, 0, 0, 0, 63, 0, 0, 0, 294, 0, 756, 0, 1890, 0, 1092, 0, 0, 0 )
14905
14906
14907
14908 BCH_21_6_4_macwilliams:
14909 ▷  $(ORBITER) -v 2 \
14910 ▷  ▷  -make_macwilliams_system 21 6 4
14911 ▷  pdflatex MacWilliams_n21_k6_q4.tex
14912 ▷  $(OPEN) MacWilliams_n21_k6_q4.pdf
14913
14914
14915
14916 #ww := [1, 0, 0, 84, 252, 1575, 10080, 58032, 319662, 1411116, 5133744, 15282792,
       37951620, 79336530, 135622080, 190615824, 213273081, 188911548, 125744304, 59721
      732, 17767512, 2580255]
14917
14918
14919 BCH_21_15_4_field_reduction:
14920 ▷  $(ORBITER) -v 2 \
14921 ▷  ▷  -define F -finite_field -q 4 -end \
14922 ▷  ▷  -with F -do \
14923 ▷  ▷  -finite_field_activity \
```

```
▷  ▷  ▷  -field_reduction "BCH_21_15_4" 2 15 21 \
▷  ▷  ▷  ▷  $(BCH_21_15_GENERATOR_MATRIX) \
▷  ▷  -end
▷  $(ORBITER) -v 2 \
▷  ▷  -draw_matrix \
▷  ▷  ▷  -input_csv_file BCH_21_15_4.csv \
▷  ▷  ▷  -box_width 20 -bit_depth 24 \
▷  ▷  ▷  -partition 4 "30" "42" \
▷  ▷  -end
▷  pdflatex field_reduction_Q4_q2_15_21.tex
▷  $(OPEN) BCH_21_15_4_draw.bmp
▷  #$(OPEN) field_reduction_Q4_q2_15_21.pdf

# poly of degree 12: 1,0,1,0,1,0,0,0,1,0,0,0,1

BCH_21_poly_division_c:
▷  $(ORBITER) -v 2 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -with F -do \
▷  ▷  -finite_field_activity \
▷  ▷  ▷  -polynomial_division \
▷  ▷  ▷  "1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1" \
▷  ▷  ▷  "1,0,1,0,1,0,0,0,1,0,0,0,1" \
▷  ▷  -end


F16_roots_5:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -with F -do -finite_field_activity \
▷  ▷  ▷  -nth_roots 5 \
▷  ▷  -end
▷  pdflatex Nth_roots_q2_n5.tex
▷  $(OPEN) Nth_roots_q2_n5.pdf



F64_roots_21:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 2 -end \
▷  ▷  -with F -do -finite_field_activity \
▷  ▷  ▷  -nth_roots 21 \
▷  ▷  -end
▷  pdflatex Nth_roots_q2_n21.tex
▷  $(OPEN) Nth_roots_q2_n21.pdf



BCH_F256_roots_771:
▷  $(ORBITER) -v 3 \
▷  ▷  -define F -finite_field -q 256 -end \
▷  ▷  -with F -do -finite_field_activity \
▷  ▷  ▷  -nth_roots 771 \
▷  ▷  -end


BCH_F256_BCH_code_d16:
▷  $(ORBITER) -v 3 \
```

```
14982 ▷  ▷  -define F -finite_field -q 256 -end \
14983 ▷  ▷  -define C -code -field F \
14984 ▷  ▷  ▷  -BCH 771 16 \
14985 ▷  ▷  -end
14986 ▷  pdflatex BCH_codes_q256_n771_d16.tex
14987 ▷  $(OPEN) BCH_codes_q256_n771_d16.pdf
14988
14989
14990 #generator polynomial is X^{30} + 253X^{29} + 174X^{28} + 109X^{27} + 97X^{26} +
      144X^{25} + 112X^{24} + 212X^{23} + 192X^{22} + 169X^{21} + 24X^{20} + 150X^{19}
      + 110X^{18} + 248X^{17} + 3X^{16} + 193X^{15} + 194X^{14} + 205X^{13} + 9X^{12} +
       56X^{11} + 95X^{10} + 199X^{9} + 108X^{8} + 58X^{7} + 160X^{6} + 148X^{5} + 138X
      ^{4} + 24X^{3} + 210X^{2} + 26X + 1
14991
14992
14993
14994
14995
14996
14997
14998
14999 ###############################################################################
15000 # Section 10.7: Coding Theory - Reed Solomon codes
15001
15002 SECTION_CODING_THEORY_REED_SOLOMON_CODES:
15003
15004
15005 test_10_7:
15006 ▷  make RS_6_3
15007 ▷  make RS_6_3_weight_enumerator
15008 ▷  make RREF_RS_6_4_7_weight_enumerator
15009 ▷  make Code_RS_11
15010 ▷  #make Code_RS_11_weight_enumerator
15011 ▷  make RS_7_3
15012 ▷  make RS_7_3_weight_enumerator
15013 ▷  make RS_7_3_field_reduction
15014 ▷  make RS_7_3_reduced_weight_enumerator
15015
15016
15017
15018 RS_6_3:
15019 ▷  $(ORBITER) -v 2 \
15020 ▷  ▷  -define F -finite_field -q 7 -end \
15021 ▷  ▷  -define C -code -field F -Reed_Solomon 6 3 -end \
15022 ▷  ▷  -with C -do -coding_theoretic_activity \
15023 ▷  ▷  ▷  -export_magma RS_6_3.magma \
15024 ▷  ▷  -end
15025 ▷  pdflatex RS_codes_q7_n6_d3.tex
15026 ▷  $(OPEN) RS_codes_q7_n6_d3.pdf
15027
15028
15029 RS_6_3_weight_enumerator:
15030 ▷  $(ORBITER) -v 2 \
15031 ▷  ▷  -define F -finite_field -q 7 -end \
15032 ▷  ▷  -define C -code -field F -Reed_Solomon 6 3 -end \
15033 ▷  ▷  -with C -do -coding_theoretic_activity \
15034 ▷  ▷  ▷  -export_magma RS_6_3.magma \
15035 ▷  ▷  -end \
15036 ▷  ▷  -with C -do \
```

```
15037 ▷  ▷  -coding_theoretic_activity \
15038 ▷  ▷  ▷  -weight_enumerator \
15039 ▷  ▷  -end
15040
15041 #[6,4,3]_7 code.
15042 #( 1, 0, 0, 120, 360, 972, 948 )
15043
15044 #1y^6 + 90x^4y^2 + 108x^5y + 144x^6
15045
15046 RREF_RS_6_4_7_weight_enumerator:
15047 ▷  $(ORBITER) -v 2 \
15048 ▷  ▷  -define F -finite_field -q 7 -end \
15049 ▷  ▷  -define v -vector -format 4 -field F \
15050 ▷  ▷  ▷  -compact $(CODE_RS_6_4_7) \
15051 ▷  ▷  -end \
15052 ▷  ▷  -define C -code -field F \
15053 ▷  ▷  ▷  -generator_matrix v \
15054 ▷  ▷  -end \
15055 ▷  ▷  -with C -do \
15056 ▷  ▷  -coding_theoretic_activity \
15057 ▷  ▷  ▷  -weight_enumerator \
15058 ▷  ▷  -end
15059
15060
15061 #1y^6 + 120x^3y^3 + 360x^4y^2 + 972x^5y + 948x^6
15062 #weight enumerator:
15063 #( 1, 0, 0, 120, 360, 972, 948 )
15064
15065
15066
15067
15068
15069 Code_RS_11:
15070 ▷  $(ORBITER) -v 2 \
15071 ▷  ▷  -define F -finite_field -q 11 -end \
15072 ▷  ▷  -define C -code -field F -Reed_Solomon 10 3 -end \
15073 ▷  ▷  -with C -do -coding_theoretic_activity \
15074 ▷  ▷  ▷  -export_magma RS_11_3.magma \
15075 ▷  ▷  -end
15076
15077
15078 Code_RS_11_weight_enumerator:
15079 ▷  $(ORBITER) -v 2 \
15080 ▷  ▷  -define F -finite_field -q 11 -end \
15081 ▷  ▷  -define C -code -field F -Reed_Solomon 10 3 -end \
15082 ▷  ▷  -with C -do -coding_theoretic_activity \
15083 ▷  ▷  ▷  -export_magma RS_11_3.magma \
15084 ▷  ▷  -end \
15085 ▷  ▷  -with C -do \
15086 ▷  ▷  -coding_theoretic_activity \
15087 ▷  ▷  ▷  -weight_enumerator \
15088 ▷  ▷  -end
15089
15090
15091 #1*y^(10) + 1200*x^3*y^7 + 16800*x^4*y^6 + 209160*x^5*y^5 + 1734600*x^6*y^4 + 991
      8000*x^7*y^3 + 37189800*x^8*y^2 + 82644700*x^9*y + 82644620*x^(10)
15092
15093
15094
```

```
15095 RS_7_3:
15096 ▷  $(ORBITER) -v 2 \
15097 ▷  ▷  -define F -finite_field -q 8 -end \
15098 ▷  ▷  -define C -code -field F -Reed_Solomon 7 3 -end \
15099 ▷  ▷  -with C -do -coding_theoretic_activity \
15100 ▷  ▷  ▷  -export_magma RS_7_3.magma \
15101 ▷  ▷  -end
15102 ▷  pdflatex RS_codes_q8_n7_d3.tex
15103 ▷  $(OPEN) RS_codes_q8_n7_d3.pdf
15104
15105
15106 RS_7_3_weight_enumerator:
15107 ▷  $(ORBITER) -v 2 \
15108 ▷  ▷  -define F -finite_field -q 8 -end \
15109 ▷  ▷  -define C -code -field F -Reed_Solomon 7 3 -end \
15110 ▷  ▷  -with C -do -coding_theoretic_activity \
15111 ▷  ▷  ▷  -export_magma RS_7_3.magma \
15112 ▷  ▷  -end \
15113 ▷  ▷  -with C -do \
15114 ▷  ▷  -coding_theoretic_activity \
15115 ▷  ▷  ▷  -weight_enumerator \
15116 ▷  ▷  -end
15117 ▷  pdflatex RS_codes_q8_n7_d3.tex
15118 ▷  $(OPEN) RS_codes_q8_n7_d3.pdf
15119
15120
15121
15122 RS_7_3_field_reduction:
15123 ▷  $(ORBITER) -v 2 \
15124 ▷  ▷  -define F -finite_field -q 8 -end \
15125 ▷  ▷  -define C -code -field F -Reed_Solomon 7 3 -end \
15126 ▷  ▷  -with F -do \
15127 ▷  ▷  -finite_field_activity \
15128 ▷  ▷  -field_reduction "RS_8_red_2" \
15129 ▷  ▷  ▷  2 5 7 $(CODE_RS_F8_N7_K5_D3_GENMA) \
15130 ▷  ▷  -end
15131 ▷  $(ORBITER) -v 2 \
15132 ▷  ▷  -draw_matrix -input_csv_file RS_8_red_2.csv \
15133 ▷  ▷  -box_width 40 -bit_depth 24 \
15134 ▷  ▷  -partition 4 "3,3,3,3,3" "3,3,3,3,3,3,3" -end
15135 ▷  pdflatex field_reduction_Q8_q2_5_7.tex
15136 ▷  $(OPEN) field_reduction_Q8_q2_5_7.pdf
15137
15138
15139 RS_7_3_reduced_weight_enumerator:
15140 ▷  $(ORBITER) -v 2 \
15141 ▷  ▷  -define F -finite_field -q 2 -end \
15142 ▷  ▷  -define v -vector -format 15 -field F \
15143 ▷  ▷  ▷  -compact $(RS_8_reduced) \
15144 ▷  ▷  -end \
15145 ▷  ▷  -define C -code -field F \
15146 ▷  ▷  ▷  -generator_matrix v \
15147 ▷  ▷  -end \
15148 ▷  ▷  -with C -do \
15149 ▷  ▷  -coding_theoretic_activity \
15150 ▷  ▷  ▷  -weight_enumerator \
15151 ▷  ▷  -end
15152
15153
```

```
15154
15155
15156 ##############################################################################
15157 # Section 10.8: Coding Theory - Twisted tensor product codes
15158
15159 SECTION_CODING_THEORY_TWISTED_TENSOR_PRODUCT_CODES:
15160
15161
15162 test_10_8:
15163 ▷  ▷  make TTP_A_4
15164
15165
15166 TTP_A_4:
15167 ▷  $(ORBITER) -v 2 \
15168 ▷  ▷  -define F4 -finite_field -q 4 -end \
15169 ▷  ▷  -define F16 -finite_field -q 16 -end \
15170 ▷  ▷  -define C -code -field F4 -ttpA F16 -end \
15171 ▷  ▷  -with C -do -coding_theoretic_activity \
15172 ▷  ▷  ▷  -export_magma TTP_A_q4.magma \
15173 ▷  ▷  -end \
15174 ▷  ▷  -with C -do -coding_theoretic_activity \
15175 ▷  ▷  ▷  -report \
15176 ▷  ▷  -end
15177 ▷  pdflatex code_n18_k9_q4.tex
15178 ▷  $(OPEN) code_n18_k9_q4.pdf
15179
15180
15181
15182 ##############################################################################
15183 # Section 10.9: Coding Theory - Bounds
15184
15185 SECTION_CODING_THEORY_BOUNDS:
15186
15187
15188 test_10_9:
15189 ▷  make bounds_for_d_given_n6_k4_q7
15190 ▷  make bounds_for_d_given_n15_k6_q2
15191 ▷  make coding_theory_bounds_q2
15192 ▷  make coding_theory_bounds_q8
15193 ▷  make GV_n15_k6_d5
15194 ▷  make bounds_for_d_given_n12_k4_q13
15195 ▷  make GV_n15_k6_d5_weight_enumerator
15196 ▷  make code_n15_k6_d6_a_we
15197 ▷  make code_n15_k6_d6_RREF
15198 ▷  make code_n15_k6_d6_check_RREF
15199
15200
15201 bounds_for_d_given_n6_k4_q7:
15202 ▷  $(ORBITER) -v 2 \
15203 ▷  ▷  -make_bounds_for_d_given_n_and_k_and_q 6 4 7
15204 ▷
15205 bounds_for_d_given_n15_k6_q2:
15206 ▷  $(ORBITER) -v 2 \
15207 ▷  ▷  -make_bounds_for_d_given_n_and_k_and_q 15 6 2
15208
15209 #n = 15 k=6 q=2
15210 #d_GV = 5
15211 #d_singleton = 10
15212 #d_hamming = 6
```

```
15213 #d_plotkin = 7
15214 #d_griesmer = 6
15215
15216
15217 coding_theory_bounds_q2:
15218 ▷  $(ORBITER) -v 2 -table_of_bounds 12 2
15219 ▷  $(ORBITER) -v 2 \
15220 ▷  ▷  -csv_file_latex 1 table_of_bounds_n12_q2.csv
15221 ▷  pdflatex table_of_bounds_n12_q2.tex
15222 ▷  $(OPEN) table_of_bounds_n12_q2.pdf
15223
15224 # produces table_of_bounds_n20_q2.csv
15225
15226 coding_theory_bounds_q8:
15227 ▷  $(ORBITER) -v 2 -table_of_bounds 20 8
15228
15229
15230 GV_n15_k6_d5:
15231 ▷  $(ORBITER) -v 2 \
15232 ▷  ▷  -define F -finite_field -q 2 -end \
15233 ▷  ▷  -define C -code -field F \
15234 ▷  ▷  ▷  -Gilbert_Varshamov 15 6 5 \
15235 ▷  ▷  -end
15236
15237
15238 # [15,6] code created
15239
15240
15241
15242 bounds_for_d_given_n12_k4_q13:
15243 ▷  $(ORBITER) -v 2 \
15244 ▷  ▷  -make_bounds_for_d_given_n_and_k_and_q 12 4 13
15245
15246
15247
15248
15249 GV_n15_k6_d5_weight_enumerator:
15250 ▷  $(ORBITER) -v 2 \
15251 ▷  ▷  -define F -finite_field -q 2 -end \
15252 ▷  ▷  -define v -vector -format 6 -field F \
15253 ▷  ▷  ▷  -compact $(CODE_GV_N15_K6) \
15254 ▷  ▷  -end \
15255 ▷  ▷  -define C -code -field F \
15256 ▷  ▷  ▷  -generator_matrix v \
15257 ▷  ▷  -end \
15258 ▷  ▷  -with C -do \
15259 ▷  ▷  -coding_theoretic_activity \
15260 ▷  ▷  ▷  -weight_enumerator \
15261 ▷  ▷  -end
15262
15263 #1y^{15} + 27x^6y^9 + 24x^8y^7 + 9x^{10}y^5 + 3x^{12}y^3
15264 # surprise: d = 6
15265
15266
15267 code_n15_k6_d6_a_we:
15268 ▷  $(ORBITER) -v 2 \
15269 ▷  ▷  -define F -finite_field -q 2 -end \
15270 ▷  ▷  -define v -vector -format 6 -field F \
15271 ▷  ▷  ▷  -compact $(CODE_15_6_6_A) \
```

```
15272 ▷  ▷  -end \
15273 ▷  ▷  -define C -code -field F \
15274 ▷  ▷  ▷  -generator_matrix v \
15275 ▷  ▷  -end \
15276 ▷  ▷  -with C -do \
15277 ▷  ▷  -coding_theoretic_activity \
15278 ▷  ▷  ▷  -weight_enumerator \
15279 ▷  ▷  -end
15280
15281 #1y^{15} + 27x^6y^9 + 24x^8y^7 + 9x^{10}y^5 + 3x^{12}y^3
15282
15283
15284
15285 # weight enumerator
15286 #1y^{15} + 28x^6y^9 + 21x^8y^7 + 12x^{10}y^5 + 2x^{12}y^3
15287
15288
15289 code_n15_k6_d6_RREF:
15290 ▷  $(ORBITER) -v 2 \
15291 ▷  ▷  -define F -finite_field -q 2 -end \
15292 ▷  ▷  -define v -vector -format 6 -field F \
15293 ▷  ▷  ▷  -compact $(CODE_GV_N15_K6) \
15294 ▷  ▷  -end \
15295 ▷  ▷  -with F -do -finite_field_activity \
15296 ▷  ▷  ▷  -RREF v \
15297 ▷  ▷  -end
15298 ▷  #pdflatex RREF_example_q2_6_15.tex
15299 ▷  #$(OPEN) RREF_example_q2_6_15.pdf
15300
15301 code_n15_k6_d6_check_RREF:
15302 ▷  $(ORBITER) -v 2 \
15303 ▷  ▷  -define F -finite_field -q 2 -end \
15304 ▷  ▷  -define v -vector -format 9 -field F \
15305 ▷  ▷  ▷  -compact $(CODE_GV_N15_K6_CHECK) \
15306 ▷  ▷  -end \
15307 ▷  ▷  -with F -do -finite_field_activity \
15308 ▷  ▷  -RREF v \
15309 ▷  ▷  -end
15310 ▷  #pdflatex RREF_example_q2_9_15.tex
15311 ▷  #$(OPEN) RREF_example_q2_9_15.pdf
15312
15313
15314
15315
15316
15317 ##############################################################################
15318 # Section 10.10: Coding Theory - Classification
15319
15320 SECTION_CODING_THEORY_CLASSIFICATION:
15321
15322 test_10_10:
15323 ▷  make codes_8_4_4
15324 ▷  #make codes_8_4_4_draw
15325 ▷  #make codes_14_4_9_3
15326 ▷  make codes_15_6_6_2
15327 ▷  make codes_d4
15328 ▷  make codes_24_12_8
15329 ▷  #make codes_24_12_8_draw
15330 ▷  make glynn_arc
```

```
15331 ▷  make five_points_in_general
15332 ▷  make codes_q13_12_4
15333
15334
15335 # code classification:
15336
15337 codes_8_4_4:
15338 ▷  $(ORBITER) -v 6 \
15339 ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
15340 ▷  ▷  -define Control -poset_classification_control \
15341 ▷  ▷  ▷  -problem_label codes_8_4_4 \
15342 ▷  ▷  ▷  -draw_options \
15343 ▷  ▷  ▷  ▷  -embedded -radius 250 \
15344 ▷  ▷  ▷  ▷  -line_width 1.0 -spanning_tree \
15345 ▷  ▷  ▷  -end \
15346 ▷  ▷  -end \
15347 ▷  ▷  -define G \
15348 ▷  ▷  -linear_group -PGL 4 2 -end \
15349 ▷  ▷  -with G -do \
15350 ▷  ▷  -group_theoretic_activity \
15351 ▷  ▷  ▷  -linear_codes Control 3 8 \
15352 ▷  ▷  -end
15353 ▷  #pdflatex codes_8_4_4_poset_lvl_8.tex
15354 ▷  #$(OPEN) codes_8_4_4_poset_lvl_8.pdf
15355 ▷  #pdflatex codes_8_4_4_poset.tex
15356 ▷  #$(OPEN) codes_8_4_4_poset.pdf
15357
15358
15359 # ToDo: the previous command does not produce the graph
15360
15361 codes_8_4_4_draw:
15362 ▷  $(ORBITER) -v 3 \
15363 ▷  ▷  -draw_layered_graph \
15364 ▷  ▷  ▷  codes_8_4_4_poset_lvl_8.layered_graph \
15365 ▷  ▷  ▷  -radius 250 -embedded -line_width 1.0 \
15366 ▷  ▷  ▷  -y_stretch 1.0 -scale 0.5 \
15367 ▷  ▷  -end
15368 ▷  pdflatex codes_8_4_4_poset_lvl_8_draw.tex
15369 ▷  $(OPEN) codes_8_4_4_poset_lvl_8_draw.pdf
15370
15371
15372
15373 codes_14_4_9_3:
15374 ▷  $(ORBITER) -v 6 \
15375 ▷  ▷  -define Control -poset_classification_control \
15376 ▷  ▷  ▷  -problem_label codes_14_4_9_3 \
15377 ▷  ▷  ▷  -draw_options \
15378 ▷  ▷  ▷  ▷  -embedded -radius 250 \
15379 ▷  ▷  ▷  -end \
15380 ▷  ▷  -end \
15381 ▷  ▷  -define G \
15382 ▷  ▷  -linear_group -PGL 10 3 -end \
15383 ▷  ▷  -with G -do \
15384 ▷  ▷  -group_theoretic_activity \
15385 ▷  ▷  ▷  -linear_codes Control 9 14 \
15386 ▷  ▷  -end
15387 ▷  #pdflatex codes_14_4_9_3_poset_lvl_13.tex
15388 ▷  #$(OPEN) codes_14_4_9_3_poset_lvl_13.pdf
15389
```

```
15390 #very slow
15391
15392
15393 # ToDo: report is not generated
15394
15395 codes_15_6_6_2:
15396 ▷  $(ORBITER) -v 6 \
15397 ▷  ▷  -define Control -poset_classification_control \
15398 ▷  ▷  ▷  -problem_label codes_15_6_6_2 \
15399 ▷  ▷  ▷  -draw_options \
15400 ▷  ▷  ▷  ▷  -embedded -radius 250 \
15401 ▷  ▷  ▷  -end \
15402 ▷  ▷  -end \
15403 ▷  ▷  -define G \
15404 ▷  ▷  -linear_group -PGL 9 2 -end \
15405 ▷  ▷  -with G -do \
15406 ▷  ▷  -group_theoretic_activity \
15407 ▷  ▷  ▷  -linear_codes Control 6 15 \
15408 ▷  ▷  -end
15409 ▷  #pdflatex codes_15_6_6_2_poset_lvl_15.tex
15410 ▷  #$(OPEN) codes_15_6_6_2_poset_lvl_15.pdf
15411
15412
15413
15414
15415
15416 codes_d4:
15417 ▷  $(ORBITER) -v 3 \
15418 ▷  ▷  -define Control -poset_classification_control \
15419 ▷  ▷  ▷  -W \
15420 ▷  ▷  ▷  -problem_label codes_r4_d4 \
15421 ▷  ▷  ▷  -draw_options -embedded -end \
15422 ▷  ▷  -end \
15423 ▷  ▷  -define G -linear_group -PGL 4 2 -end \
15424 ▷  ▷  -with G -do \
15425 ▷  ▷  -group_theoretic_activity \
15426 ▷  ▷  ▷  -linear_codes Control 4 100 \
15427 ▷  ▷  -end
15428
15429
15430 codes_24_12_8:
15431 ▷  $(ORBITER) -v 6 \
15432 ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
15433 ▷  ▷  -define Control -poset_classification_control \
15434 ▷  ▷  ▷  -problem_label codes_24_12_8 \
15435 ▷  ▷  ▷  -draw_options -embedded -radius 250 \
15436 ▷  ▷  ▷  -line_width 1.0 -spanning_tree -end \
15437 ▷  ▷  -end \
15438 ▷  ▷  -define G \
15439 ▷  ▷  -linear_group -PGL 12 2 -end \
15440 ▷  ▷  -with G -do \
15441 ▷  ▷  -group_theoretic_activity \
15442 ▷  ▷  ▷  -linear_codes Control 8 24 \
15443 ▷  ▷  -end
15444 ▷  #pdflatex codes_24_12_8_poset.tex
15445 ▷  #$(OPEN) codes_24_12_8_poset.pdf
15446
15447 #codes_24_12_8_poset_lvl_24.layered_graph
15448
```

```
15449 codes_24_12_8_draw:
15450 ▷  $(ORBITER) -v 3 \
15451 ▷  ▷  -draw_layered_graph \
15452 ▷  ▷  ▷  codes_24_12_8_poset_lvl_24.layered_graph \
15453 ▷  ▷  ▷  -radius 100 -spanning_tree -embedded \
15454 ▷  ▷  ▷  -line_width 0.5 -x_stretch 1.4 \
15455 ▷  ▷  ▷  -scale 0.25 -nodes_empty \
15456 ▷  ▷  -end
15457 ▷  pdflatex codes_24_12_8_poset_lvl_24_draw.tex
15458 ▷  $(OPEN) codes_24_12_8_poset_lvl_24_draw.pdf
15459
15460 glynn_arc:
15461 ▷  $(ORBITER) -v 5 \
15462 ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
15463 ▷  ▷  -define Control -poset_classification_control \
15464 ▷  ▷  ▷  -problem_label glynn_arc \
15465 ▷  ▷  ▷  -draw_options -embedded -radius 250 \
15466 ▷  ▷  ▷  -line_width 1.0 -spanning_tree -end \
15467 ▷  ▷  -end \
15468 ▷  ▷  -define G \
15469 ▷  ▷  -linear_group -PGGL 5 9 -end \
15470 ▷  ▷  -with G -do \
15471 ▷  ▷  -group_theoretic_activity \
15472 ▷  ▷  ▷  -linear_codes Control 6 10 \
15473 ▷  ▷  -end
15474 ▷  #pdflatex glynn_arc_poset.tex
15475 ▷  #$(OPEN) glynn_arc_poset.pdf
15476
15477
15478
15479
15480
15481 five_points_in_general:
15482 ▷  $(ORBITER) -v 5 \
15483 ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
15484 ▷  ▷  -define Control -poset_classification_control \
15485 ▷  ▷  ▷  -problem_label five_points_in_general \
15486 ▷  ▷  ▷  -draw_options \
15487 ▷  ▷  ▷  ▷  -embedded -radius 250 \
15488 ▷  ▷  ▷  ▷  -line_width 1.0 -spanning_tree \
15489 ▷  ▷  ▷  -end \
15490 ▷  ▷  -end \
15491 ▷  ▷  -define G \
15492 ▷  ▷  -linear_group -PGL 4 2 -end \
15493 ▷  ▷  -with G -do \
15494 ▷  ▷  -group_theoretic_activity \
15495 ▷  ▷  -linear_codes Control 4 5 \
15496 ▷  ▷  -end
15497 ▷  #pdflatex five_points_in_general_poset.tex
15498 ▷  #$(OPEN) five_points_in_general_poset.pdf
15499
15500
15501
15502
15503 codes_q13_12_4:
15504 ▷  $(ORBITER) -v 6 \
15505 ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
15506 ▷  ▷  -define Control -poset_classification_control \
15507 ▷  ▷  ▷  -problem_label codes_q13 \
```

```
15508 ▷  ▷  -end \
15509 ▷  ▷  -define G \
15510 ▷  ▷  -linear_group -PGL 4 13 -end \
15511 ▷  ▷  -with G -do \
15512 ▷  ▷  -group_theoretic_activity \
15513 ▷  ▷  -linear_codes Control 6 12 \
15514 ▷  ▷  -end
15515 ▷  #pdflatex codes_q13_poset.tex
15516 ▷  #$(OPEN) codes_q13_poset.pdf
15517
15518
15519
15520
15521 ###############################################################################
15522 # Chapter 11 - Combinatorics
15523 ###############################################################################
15524
15525
15526
15527 test_11:
15528 ▷  make test_11_1
15529 ▷  make test_11_2
15530 ▷  make test_11_3
15531 ▷  make test_11_4 # slow
15532 ▷  make test_11_5 # slow
15533 ▷  make test_11_6
15534 ▷  make test_11_7
15535 ▷  make test_11_8
15536 ▷  make test_11_9
15537 ▷  make test_11_10
15538
15539
15540
15541 ###############################################################################
15542 # Section 11.1: Combinatorics
15543
15544
15545 SECTION_COMBINATORICS:
15546 ▷
15547
15548
15549 test_11_1:
15550 ▷  make Sym_4_conj_classes
15551 ▷  make Sym_10_conj_classes
15552 ▷  make Sym_15_conj_classes
15553 ▷  make Char_Sym_4
15554 ▷  make Char_Sym_5
15555 ▷  #make Char_Sym_6
15556 ▷  make all_subsets_6_3
15557 ▷  make random_k_subsets
15558 ▷  make rank_k_subsets_test
15559 ▷  make Walsh_matrix_4
15560 ▷  make Dedekind_10_10
15561 ▷  make Dedekind_30_2
15562 ▷  make Dedekind_100_2
15563 ▷  make elementary_symmetric_functions_4
15564 ▷  make elementary_symmetric_functions_8
15565 ▷  make domino_portrait
15566 ▷  make domino_portrait_input
```

```
15567
15568
15569
15570 Sym_4_conj_classes:
15571 ▷  $(ORBITER) -v 2 -conjugacy_classes_Sym_n 4
15572
15573 Sym_10_conj_classes:
15574 ▷  $(ORBITER) -v 2 -conjugacy_classes_Sym_n 10
15575 ▷  $(OPEN) classes_Sym_10.csv
15576
15577 Sym_15_conj_classes:
15578 ▷  $(ORBITER) -v 2 -conjugacy_classes_Sym_n 15
15579
15580 Char_Sym_4:
15581 ▷  $(ORBITER) -v 2 -character_table_symmetric_group 4
15582
15583 Char_Sym_5:
15584 ▷  $(ORBITER) -v 2 -character_table_symmetric_group 5
15585
15586
15587 # too slow:
15588
15589 Char_Sym_6:
15590 ▷  $(ORBITER) -v 2 -character_table_symmetric_group 6
15591
15592 all_subsets_6_3:
15593 ▷  $(ORBITER) -v 2 -tree_of_all_k_subsets 6 3
15594 ▷  $(ORBITER) -v 20 \
15595 ▷  ▷  -draw_options -embedded \
15596 ▷  ▷  ▷  -nodes -radius 80 \
15597 ▷  ▷  ▷  -xin 5000 -yin 5000 \
15598 ▷  ▷  ▷  -xout 1000000 -yout 500000 \
15599 ▷  ▷  ▷  -scale 0.5 -line_width 1.0 \
15600 ▷  ▷  -end \
15601 ▷  ▷  -tree_draw -file all_k_subsets_n6_k3.tree -end
15602 ▷  pdflatex all_k_subsets_n6_k3_draw.tex
15603 ▷  $(OPEN) all_k_subsets_n6_k3_draw.pdf
15604
15605 random_k_subsets:
15606 ▷  $(ORBITER) -v 4 \
15607 ▷  ▷  -create_random_k_subsets 10 5 20
15608
15609
15610 rank_k_subsets_test:
15611 ▷  $(ORBITER) -v 2 \
15612 ▷  ▷  -define Blocks -vector -format 7 \
15613 ▷  ▷  ▷  -dense "0,1,2,0,3,4,1,3,5,2,4,5,3,6,7,1,6,8,0,6,9" \
15614 ▷  ▷  ▷  -end \
15615 ▷  ▷  -rank_k_subset 10 3 Blocks
15616
15617
15618 Walsh_matrix_4:
15619 ▷  $(ORBITER) -v 3 \
15620 ▷  ▷  -define F -finite_field -q 2 -end \
15621 ▷  ▷  -with F -do -finite_field_activity \
15622 ▷  ▷  ▷  -Walsh_matrix 4 -end
15623 ▷  $(ORBITER) -v 2 -draw_matrix \
15624 ▷  ▷  -input_csv_file Walsh_01_4.csv \
15625 ▷  ▷  -box_width 10 -bit_depth 24 -partition 3 16 16 -end
```

```
15626 ▷  #pdflatex GF_2.tex
15627 ▷  #$(OPEN) GF_2.pdf
15628
15629
15630 Dedekind_10_10:
15631 ▷  $(ORBITER) -v 3 -Dedekind_numbers 2 10 2 10
15632 ▷  $(ORBITER) -v 3 -csv_file_latex 1 Dedekind_2_10_2_10.csv
15633 ▷  pdflatex Dedekind_2_10_2_10.tex
15634 ▷  $(OPEN) Dedekind_2_10_2_10.pdf
15635
15636
15637 Dedekind_30_2:
15638 ▷  $(ORBITER) -v 3 -Dedekind_numbers 2 30 2 2
15639
15640
15641 Dedekind_100_2:
15642 ▷  $(ORBITER) -v 3 -Dedekind_numbers 2 100 2 2
15643
15644
15645
15646 elementary_symmetric_functions_4:
15647 ▷  $(ORBITER) -make_elementary_symmetric_functions 4 4
15648
15649 elementary_symmetric_functions_8:
15650 ▷  $(ORBITER) -make_elementary_symmetric_functions 8 8
15651
15652
15653
15654 domino_portrait:
15655 ▷  cp $(MY_PATH)/examples/users_guide/anton_28x32.? .
15656 ▷  $(ORBITER) -v 3 -domino_portrait 7 4 anton_28x32 -end
15657
15658 domino_portrait_input:
15659 ▷  cp $(MY_PATH)/examples/users_guide/anton_28x32.? .
15660 ▷  $(ORBITER) -v 2 \
15661 ▷  ▷  -define all_one_r -vector -repeat 1 28 -end \
15662 ▷  ▷  -define all_one_c -vector -repeat 1 32 -end \
15663 ▷  ▷  -draw_matrix \
15664 ▷  ▷  ▷  -grayscale \
15665 ▷  ▷  ▷  -invert_colors \
15666 ▷  ▷  ▷  -input_csv_file anton_28x32_m.csv \
15667 ▷  ▷  ▷  -box_width 20 -bit_depth 8 \
15668 ▷  ▷  ▷  -partition 3 \
15669 ▷  ▷  ▷  ▷  all_one_c all_one_r \
15670 ▷  ▷  -end
15671 ▷  $(OPEN) anton_28x32_m_draw.bmp
15672
15673
15674
15675 ##############################################################################
15676 # Section 11.2: Combinatorial Objects
15677
15678 SECTION_COMBINATORIAL_OBJECTS:
15679
15680
15681 test_11_2:
15682 ▷  make Hirschfeld_q4_from_set
15683 ▷  make hyperoval_16_create
15684 ▷  make EC_read
```

```
15685 ▷  #make PG_3_5_assume_31_read
15686 ▷  #make LS_AG_2_3_read
15687 ▷  make geo_7_3_read
15688 ▷  #make Desargues_path_lex_least_read
15689 ▷  make geo_pasch_read
15690 ▷  make geo_pasch_given
15691
15692
15693
15694 Hirschfeld_q4_from_set:
15695 ▷  $(ORBITER) -v 4 \
15696 ▷  ▷  -define H -set -here \
15697 ▷  ▷  ▷  $(HIRSCHFELD_SURFACE_Q4_SET_OF_POINTS) \
15698 ▷  ▷  -end \
15699 ▷  ▷  -define C -combinatorial_object \
15700 ▷  ▷  ▷  -label Hirschfeld_surface Hirschfeld\_surface \
15701 ▷  ▷  ▷  -set_of_points H \
15702 ▷  ▷  -end
15703
15704
15705
15706 hyperoval_16_create:
15707 ▷  $(ORBITER) -v 2 \
15708 ▷  ▷  -define C -combinatorial_object \
15709 ▷  ▷  ▷  -label hyperoval_q16 hyperoval\_q16 \
15710 ▷  ▷  ▷  -set_of_points $(HYPEROVAL_16_16320) \
15711 ▷  ▷  ▷  -set_of_points $(HYPEROVAL_16_144) \
15712 ▷  ▷  -end \
15713
15714
15715 EC_read: elliptic_curve_b1_c3_q11.txt
15716 ▷  $(ORBITER) -v 4 \
15717 ▷  ▷  -define C -combinatorial_object \
15718 ▷  ▷  ▷  -label EC EC \
15719 ▷  ▷  ▷  -file_of_points elliptic_curve_b1_c3_q11.txt \
15720 ▷  ▷  -end
15721
15722
15723
15724 PG_3_5_assume_31_read:
15725 ▷  $(ORBITER) -v 2 \
15726 ▷  ▷  -define C -combinatorial_object \
15727 ▷  ▷  ▷  -label packings_25 packings_25 \
15728 ▷  ▷  ▷  -file_of_packings_through_spread_table \
15729 ▷  ▷  ▷  ▷  H31_packings.csv \
15730 ▷  ▷  ▷  ▷  SPREAD_TABLES_5_REG/spread_25_spreads.csv \
15731 ▷  ▷  ▷  ▷  5 \
15732 ▷  ▷  -end
15733
15734
15735
15736 LS_AG_2_3_read:
15737 ▷  $(ORBITER) -v 2 \
15738 ▷  ▷  -define C -combinatorial_object \
15739 ▷  ▷  ▷  -label designs designs \
15740 ▷  ▷  ▷  -file_of_designs \
15741 ▷  ▷  ▷  solutions.csv 9 84 3 12 \
15742 ▷  ▷  -end
15743
```

```
15744
15745
15746 geo_7_3_read:
15747 ▷  $(ORBITER) -v 10 \
15748 ▷  ▷  -draw_incidence_structure_description \
15749 ▷  ▷  ▷  -width 60 -width_10 6 -end \
15750 ▷  ▷  -define C -combinatorial_object \
15751 ▷  ▷  ▷  -label geo_7_3 geo\_7\_3 \
15752 ▷  ▷  ▷  -file_of_incidence_geometries \
15753 ▷  ▷  ▷  ▷  7_3.inc 7 7 21 \
15754 ▷  ▷  -end
15755
15756
15757
15758 Desargues_path_lex_least_read:
15759 ▷  echo $(DESARGUES_PATH_LEX_LEAST) >Desargues_path_lex_least.inc
15760 ▷  $(ORBITER) -v 10 \
15761 ▷  ▷  -draw_incidence_structure_description \
15762 ▷  ▷  ▷  -width 60 -width_10 6 -end \
15763 ▷  ▷  -define C -combinatorial_object \
15764 ▷  ▷  ▷  -label Desargues_path_lex_least Desargues\_path\_lex\_least \
15765 ▷  ▷  ▷  -file_of_incidence_geometries_by_row_ranks \
15766 ▷  ▷  ▷  ▷  Desargues_path_lex_least.inc 10 10 3 \
15767 ▷  ▷  -end
15768
15769
15770
15771
15772 geo_pasch_read:
15773 ▷  $(ORBITER) -v 10 \
15774 ▷  ▷  -define C -combinatorial_object \
15775 ▷  ▷  ▷  -label Pasch Pasch \
15776 ▷  ▷  ▷  -file_of_incidence_geometries \
15777 ▷  ▷  ▷  ▷  pasch.inc 6 4 12 \
15778 ▷  ▷  -end
15779
15780 geo_pasch_given:
15781 ▷  $(ORBITER) -v 10 \
15782 ▷  ▷  -define C -combinatorial_object \
15783 ▷  ▷  ▷  -label Pasch Pasch \
15784 ▷  ▷  ▷  -incidence_geometry \
15785 ▷  ▷  ▷  ▷  "0,1,4,6,8,11,13,14,17,19,22,23" \
15786 ▷  ▷  ▷  ▷  6 4 12 \
15787 ▷  ▷  -end
15788
15789
15790
15791
15792 ##############################################################################
15793 # Section 11.3: Combinatorial Linear Spaces
15794
15795 SECTION_COMBINATORIAL_LINEAR_SPACES:
15796
15797
15798 test_11_3:
15799 ▷  make linsp6
15800 ▷  make linsp7
15801 ▷  make linsp30_pt_types
15802 ▷  make linsp30_pt_distribution
```

```
15803
15804
15805 linsp6:
15806 ▷  $(ORBITER) -v 4 \
15807 ▷  ▷  -define A -vector -format 1 -dense "15,10,6,3,1" -end \
15808 ▷  ▷  -define D -diophant -label linsp6 \
15809 ▷  ▷  -coefficient_matrix A \
15810 ▷  ▷  -RHS "mult=1,EQ=15" \
15811 ▷  ▷  -x_min_global 0 \
15812 ▷  ▷  -x_max_global 15 \
15813 ▷  ▷  -end \
15814 ▷  ▷  -with D -do \
15815 ▷  ▷  ▷  -diophant_activity -solve_mckay \
15816 ▷  ▷  -end
15817 ▷
15818 # Found 15 solutions with 22 backtrack steps
15819
15820
15821
15822
15823 linsp7:
15824 ▷  $(ORBITER) -v 4 \
15825 ▷  ▷  -define A -vector -format 1 -dense "21,15,10,6,3,1" -end \
15826 ▷  ▷  -define D -diophant -label linsp7 \
15827 ▷  ▷  -coefficient_matrix A \
15828 ▷  ▷  -RHS "mult=1,EQ=21" \
15829 ▷  ▷  -x_min_global 0 \
15830 ▷  ▷  -x_max_global 21 \
15831 ▷  ▷  -end \
15832 ▷  ▷  -with D -do \
15833 ▷  ▷  ▷  -diophant_activity -solve_mckay \
15834 ▷  ▷  -end
15835 ▷
15836
15837 # 32 solutions in 45 backtrack steps
15838
15839
15840
15841
15842
15843
15844
15845 linsp30_pt_types:
15846 ▷  $(ORBITER) -v 4 \
15847 ▷  ▷  -define A -vector -format 1 -dense "6,4,3" -end \
15848 ▷  ▷  -define D -diophant \
15849 ▷  ▷  ▷  -label linsp30_pt_types \
15850 ▷  ▷  ▷  -coefficient_matrix A \
15851 ▷  ▷  ▷  -RHS "mult=1,EQ=29" -x_bounds "0,1,0,27,0,24" \
15852 ▷  ▷  -end \
15853 ▷  ▷  -with D -do \
15854 ▷  ▷  ▷  -diophant_activity -solve_mckay \
15855 ▷  ▷  -end
15856
15857 linsp30_pt_distribution:
15858 ▷  $(ORBITER) -v 4 \
15859 ▷  ▷  -define A -vector -format 6 -dense \
15860 ▷  ▷  ▷  "1,1,1,1,1,1,0,0,5,2,5,2,1,5,3,7,10,1,10,1,0,10,3,21" \
15861 ▷  ▷  -end \
```

```
15862 ▷  ▷  -define D -diophant \
15863 ▷  ▷  ▷  -label linsp30_pt_distribution \
15864 ▷  ▷  ▷  -coefficient_matrix A \
15865 ▷  ▷  ▷  -RHS "mult=1,EQ=30" \
15866 ▷  ▷  ▷  -RHS "mult=1,EQ=7" \
15867 ▷  ▷  ▷  -RHS "mult=1,EQ=135" \
15868 ▷  ▷  ▷  -RHS "mult=1,EQ=96" \
15869 ▷  ▷  ▷  -RHS "mult=1,LE=351" \
15870 ▷  ▷  ▷  -RHS "mult=1,LE=276" \
15871 ▷  ▷  ▷  -x_min_global 0 -x_max_global 30 \
15872 ▷  ▷  -end \
15873 ▷  ▷  -with D -do \
15874 ▷  ▷  ▷  -diophant_activity -solve_mckay \
15875 ▷  ▷  -end \
15876 ▷  ▷  -with D -do \
15877 ▷  ▷  ▷  -diophant_activity -draw_as_bitmap 20 8 \
15878 ▷  ▷  -end
15879
15880
15881
15882 ################################################################################
15883 # Section 11.4: Combinatorial Linear Spaces
15884
15885 test_11_4:
15886 ▷  make geo_pasch
15887 ▷  make geo_petersen
15888 ▷  make geo_7_3
15889 ▷  make geo_7_3_no_square_test
15890 ▷  make geo_7_3_no_square_test_draw
15891 ▷  make geo_7_3_orderly
15892 ▷  make geo_7_3_orderly_draw
15893 ▷  make geo_7_3_orderly_mem_debug
15894 ▷  make geo_8_3
15895 ▷  make geo_9_3
15896 ▷  make geo_10_3
15897 ▷  make geo_10_3_inc_draw
15898 ▷  make geo_10_3_orderly
15899 ▷  make geo_10_3_orderly_mem_debug
15900 ▷  make geo_10_3_tree
15901 ▷  make geo_10_3_tree_path
15902 ▷  #make Desargues_path_lex_least_draw
15903 ▷  #make Desargues_path_can_anc_draw
15904 ▷  make geo_11_3
15905 ▷  make geo_12_3
15906 ▷  make geo_12_3_orderly
15907 ▷  make geo_13_3
15908 ▷  make geo_13_3_orderly
15909 ▷  make geo_14_3
15910 ▷  make geo_14_3_orderly
15911 ▷  #make 15_3.inc
15912 ▷  make geo_15_3_g4
15913 ▷  make geo_17_3_g4_orderly
15914 ▷  make geo_18_3_g4
15915 ▷  make geo_19_3_g4
15916 ▷  make geo_20_3_g4
15917 ▷  make geo_21_3_g4
15918 ▷  make geo_15_4
15919 ▷  make geo_16_4_g4
15920 ▷  make geo_16
```

```
15921 ▷  make geo_40_4_g4
15922 ▷  make geo_63_3_g6
15923 ▷  make geo_LSQ6.inc
15924
15925
15926 geo_pasch:
15927 ▷  $(ORBITER) -v 8 \
15928 ▷  ▷  -define Test_lines -set -loop 1 7 1 -end \
15929 ▷  ▷  -define Geo -geometry_builder \
15930 ▷  ▷  ▷  -V 6 -B 4 -TDO 2 -fuse 1 \
15931 ▷  ▷  ▷  -fname_GEO pasch \
15932 ▷  ▷  ▷  -output_to_inc_file \
15933 ▷  ▷  ▷  -test Test_lines \
15934 ▷  ▷  -end
15935
15936
15937 geo_petersen:
15938 ▷  $(ORBITER) -v 8 \
15939 ▷  ▷  -define Test_lines -set -loop 3 11 1 -end \
15940 ▷  ▷  -define Geo -geometry_builder \
15941 ▷  ▷  ▷  -V 10 -B 15 -TDO 3 -fuse 1 \
15942 ▷  ▷  ▷  -fname_GEO petersen -girth 5 \
15943 ▷  ▷  ▷  -output_to_inc_file \
15944 ▷  ▷  ▷  -search_tree \
15945 ▷  ▷  ▷  -test Test_lines \
15946 ▷  ▷  -end
15947
15948 geo_7_3:
15949 ▷  $(ORBITER) -v 2 \
15950 ▷  ▷  -define Test_lines -set -loop 3 8 1 -end \
15951 ▷  ▷  -define Geo -geometry_builder \
15952 ▷  ▷  ▷  -V 7 -B 7 -TDO 3 \
15953 ▷  ▷  ▷  -fuse 1 \
15954 ▷  ▷  ▷  -fname_GEO 7_3 \
15955 ▷  ▷  ▷  -output_to_inc_file \
15956 ▷  ▷  ▷  -test Test_lines \
15957 ▷  ▷  -end
15958
15959 geo_7_3_no_square_test:
15960 ▷  $(ORBITER) -v 2 \
15961 ▷  ▷  -define Test_lines -set -loop 3 8 1 -end \
15962 ▷  ▷  -define Geo -geometry_builder \
15963 ▷  ▷  ▷  -V 7 -B 7 -TDO 3 \
15964 ▷  ▷  ▷  -fuse 1 \
15965 ▷  ▷  ▷  -fname_GEO 7_3_nst \
15966 ▷  ▷  ▷  -output_to_inc_file \
15967 ▷  ▷  ▷  -test Test_lines \
15968 ▷  ▷  ▷  -no_square_test \
15969 ▷  ▷  -end
15970
15971 geo_7_3_no_square_test_draw:
15972 ▷  $(ORBITER) -v 10 \
15973 ▷  ▷  -draw_incidence_structure_description \
15974 ▷  ▷  ▷  -width 60 -width_10 6 -end \
15975 ▷  ▷  -define C -combinatorial_object \
15976 ▷  ▷  ▷  -label geo_7_3_nst geo_7_3_nst \
15977 ▷  ▷  ▷  -file_of_incidence_geometries 7_3_nst.inc 7 7 21 \
15978 ▷  ▷  -end \
15979 ▷  ▷  -with C -do \
```

```
15980 ▷  ▷  -combinatorial_object_activity \
15981 ▷  ▷  ▷  -draw_incidence_matrices \
15982 ▷  ▷  ▷  7_3_nst \
15983 ▷  ▷  -end
15984 ▷  pdflatex 7_3_nst_incma.tex
15985 ▷  $(OPEN) 7_3_nst_incma.pdf
15986
15987
15988
15989 geo_7_3_orderly:
15990 ▷  $(ORBITER) -v 200 \
15991 ▷  ▷  -define Test_lines -set -loop 3 8 1 -end \
15992 ▷  ▷  -define Geo -geometry_builder \
15993 ▷  ▷  ▷  -V 7 -B 7 -TDO 3 \
15994 ▷  ▷  ▷  -fuse 1 -fname_GEO 7_3 \
15995 ▷  ▷  ▷  -output_to_inc_file \
15996 ▷  ▷  ▷  -test Test_lines \
15997 ▷  ▷  ▷  -search_tree \
15998 ▷  ▷  ▷  -orderly \
15999 ▷  ▷  -end
16000
16001 geo_7_3_orderly_draw:
16002 ▷  $(ORBITER) -v 20 \
16003 ▷  ▷  -draw_options -embedded -radius 50 \
16004 ▷  ▷  ▷  -xin 10000 -yin 10000 \
16005 ▷  ▷  ▷  -xout 1000000 -yout 1000000 \
16006 ▷  ▷  ▷  -nodes_empty \
16007 ▷  ▷  ▷  -scale 0.5 -line_width 0.3 \
16008 ▷  ▷  -end \
16009 ▷  ▷  -tree_draw -file 7_3_tree.txt -end
16010 ▷  pdflatex 7_3_tree_draw.tex
16011 ▷  $(OPEN) 7_3_tree_draw.pdf
16012
16013 geo_7_3_orderly_mem_debug:
16014 ▷  $(ORBITER) -v 20 \
16015 ▷  ▷  -memory_debug 2 \
16016 ▷  ▷  -define Test_lines -set -loop 3 8 1 -end \
16017 ▷  ▷  -define Geo -geometry_builder \
16018 ▷  ▷  ▷  -V 7 -B 7 -TDO 3 \
16019 ▷  ▷  ▷  -fuse 1 -fname_GEO 7_3 \
16020 ▷  ▷  ▷  -output_to_inc_file \
16021 ▷  ▷  ▷  -test Test_lines \
16022 ▷  ▷  ▷  -search_tree \
16023 ▷  ▷  ▷  -orderly \
16024 ▷  ▷  -end
16025
16026 geo_8_3:
16027 ▷  $(ORBITER) -v 2 \
16028 ▷  ▷  -define Test_lines -set -loop 3 9 1 -end \
16029 ▷  ▷  -define Geo -geometry_builder \
16030 ▷  ▷  ▷  -V 8 -B 8 -TDO 3 \
16031 ▷  ▷  ▷  -fuse 1 -fname_GEO 8_3 \
16032 ▷  ▷  ▷  -output_to_inc_file \
16033 ▷  ▷  ▷  -test Test_lines \
16034 ▷  ▷  -end
16035
16036 #-print_at_line 4
16037 # 1 geo: 0 11 18 29 30 38 44 54
16038 # ago=48
```

```
16039
16040
16041
16042
16043
16044 geo_9_3:
16045 ▷  $(ORBITER) -v 2 \
16046 ▷  ▷  -define Test_lines -set -loop 3 10 1 -end \
16047 ▷  ▷  -define Geo -geometry_builder \
16048 ▷  ▷  ▷  -V 9 -B 9 -TDO 3 \
16049 ▷  ▷  ▷  -fuse 1 -fname_GEO 9_3 \
16050 ▷  ▷  ▷  -output_to_inc_file \
16051 ▷  ▷  ▷  -test Test_lines \
16052 ▷  ▷  -end
16053
16054
16055 geo_10_3:
16056 ▷  $(ORBITER) -v 2 \
16057 ▷  ▷  -define Test_lines -set -loop 4 11 1 -end \
16058 ▷  ▷  -define Geo -geometry_builder \
16059 ▷  ▷  ▷  -V 10 -B 10 -TDO 3 -fuse 1 \
16060 ▷  ▷  ▷  -fname_GEO 10_3 \
16061 ▷  ▷  ▷  -output_to_inc_file \
16062 ▷  ▷  ▷  -output_to_sage_file \
16063 ▷  ▷  ▷  -output_to_blocks_file \
16064 ▷  ▷  ▷  -test Test_lines \
16065 ▷  ▷  -end
16066
16067
16068
16069 # 10 geos
16070 # 8/26/2021: 0 sec on Mac
16071
16072
16073 geo_10_3_inc_draw:
16074 ▷  $(ORBITER) -v 10 \
16075 ▷  ▷  -draw_incidence_structure_description \
16076 ▷  ▷  ▷  -width 60 -width_10 6 -end \
16077 ▷  ▷  -define C -combinatorial_object \
16078 ▷  ▷  ▷  -label geo_10_3 geo\_10\_3 \
16079 ▷  ▷  ▷  -file_of_incidence_geometries \
16080 ▷  ▷  ▷  10_3.inc 10 10 30 \
16081 ▷  ▷  -end \
16082 ▷  ▷  -with C -do \
16083 ▷  ▷  -combinatorial_object_activity \
16084 ▷  ▷  ▷  -draw_incidence_matrices \
16085 ▷  ▷  ▷  10_3_inc \
16086 ▷  ▷  -end
16087 ▷  pdflatex 10_3_inc_incma.tex
16088 ▷  $(OPEN) 10_3_inc_incma.pdf
16089
16090
16091 geo_10_3_orderly:
16092 ▷  $(ORBITER) -v 20 \
16093 ▷  ▷  -define Test_lines -set -loop 4 11 1 -end \
16094 ▷  ▷  -define Geo -geometry_builder \
16095 ▷  ▷  ▷  -V 10 -B 10 -TDO 3 -fuse 1 \
16096 ▷  ▷  ▷  -fname_GEO 10_3 \
16097 ▷  ▷  ▷  -output_to_inc_file \
```

```
16098 ▷  ▷  ▷  -test Test_lines \
16099 ▷  ▷  ▷  -orderly \
16100 ▷  ▷  -end
16101
16102 geo_10_3_orderly_mem_debug:
16103 ▷  $(ORBITER) -v 2 \
16104 ▷  ▷  -memory_debug 2 \
16105 ▷  ▷  -define Test_lines -set -loop 4 11 1 -end \
16106 ▷  ▷  -define Geo -geometry_builder \
16107 ▷  ▷  ▷  -V 10 -B 10 -TDO 3 -fuse 1 \
16108 ▷  ▷  ▷  -fname_GEO 10_3 \
16109 ▷  ▷  ▷  -output_to_inc_file \
16110 ▷  ▷  ▷  -test Test_lines \
16111 ▷  ▷  ▷  -orderly \
16112 ▷  ▷  -end
16113
16114
16115 geo_10_3_tree:
16116 ▷  $(ORBITER) -v 20 \
16117 ▷  ▷  -define Test_lines -set -loop 0 11 1 -end \
16118 ▷  ▷  -define GEO -geometry_builder \
16119 ▷  ▷  ▷  -V 10 -B 10 -TDO 3 -fuse 1 \
16120 ▷  ▷  ▷  -fname_GEO 10_3 \
16121 ▷  ▷  ▷  -output_to_inc_file \
16122 ▷  ▷  ▷  -search_tree \
16123 ▷  ▷  ▷  -test Test_lines \
16124 ▷  ▷  -end
16125 ▷  $(ORBITER) -v 20 \
16126 ▷  ▷  -draw_options -embedded -radius 40 \
16127 ▷  ▷  ▷  -paperheight 220 \
16128 ▷  ▷  ▷  -paperwidth 330 \
16129 ▷  ▷  ▷  -xin 10000 -yin 10000 \
16130 ▷  ▷  ▷  -xout 1000000 -yout 500000 \
16131 ▷  ▷  ▷  -scale 2 -line_width 0.3 \
16132 ▷  ▷  ▷  -nodes_empty \
16133 ▷  ▷  -end \
16134 ▷  ▷  -tree_draw \
16135 ▷  ▷  ▷  -file 10_3_tree.txt \
16136 ▷  ▷  -end
16137 ▷  pdflatex 10_3_tree_draw.tex
16138 ▷  $(OPEN) 10_3_tree_draw.pdf
16139
16140
16141
16142
16143 geo_10_3_tree_path:
16144 ▷  $(ORBITER) -v 20 \
16145 ▷  ▷  -define Test_lines -set -loop 0 11 1 -end \
16146 ▷  ▷  -define GEO -geometry_builder \
16147 ▷  ▷  ▷  -V 10 -B 10 -TDO 3 -fuse 1 \
16148 ▷  ▷  ▷  -fname_GEO 10_3 \
16149 ▷  ▷  ▷  -output_to_inc_file \
16150 ▷  ▷  ▷  -search_tree \
16151 ▷  ▷  ▷  -test Test_lines \
16152 ▷  ▷  -end
16153 ▷  $(ORBITER) -v 20 \
16154 ▷  ▷  -draw_options -embedded -radius 20 \
16155 ▷  ▷  ▷  -paperheight 220 \
16156 ▷  ▷  ▷  -paperwidth 330 \
```

```
16157 ▷  ▷  ▷  -xin 10000 -yin 10000 \
16158 ▷  ▷  ▷  -xout 1000000 -yout 500000 \
16159 ▷  ▷  ▷  -scale 2 -line_width 0.3 \
16160 ▷  ▷  -end \
16161 ▷  ▷  -tree_draw \
16162 ▷  ▷  ▷  -restrict 2 \
16163 ▷  ▷  ▷  -file 10_3_tree.txt \
16164 ▷  ▷  ▷  -select_path "0,0,15,26,46,56,72,80,93,106,119" \
16165 ▷  ▷  -end
16166 ▷  pdflatex 10_3_tree_draw.tex
16167 ▷  $(OPEN) 10_3_tree_draw.pdf
16168
16169 #▷ ▷  ▷  -nodes_empty \
16170 #-sideways
16171
16172
16173 # ToDo:
16174 #Desargues_path_lex_least.inc is missing
16175
16176 Desargues_path_lex_least_draw:
16177 ▷  echo $(DESARGUES_PATH_LEX_LEAST) >Desargues_path_lex_least.inc
16178 ▷  $(ORBITER) -v 10 \
16179 ▷  ▷  -draw_incidence_structure_description \
16180 ▷  ▷  ▷  -width 60 -width_10 6 -end \
16181 ▷  ▷  -define C -combinatorial_object \
16182 ▷  ▷  ▷  -label geo_10_3 geo\_10\_3 \
16183 ▷  ▷  ▷  -file_of_incidence_geometries_by_row_ranks \
16184 ▷  ▷  ▷  Desargues_path_lex_least.inc 10 10 3 \
16185 ▷  ▷  -end \
16186 ▷  ▷  -with C -do \
16187 ▷  ▷  -combinatorial_object_activity \
16188 ▷  ▷  ▷  -draw_incidence_matrices \
16189 ▷  ▷  ▷  Desargues_path_lex_least \
16190 ▷  ▷  -end
16191 ▷  pdflatex Desargues_path_lex_least_incma.tex
16192 ▷  $(OPEN) Desargues_path_lex_least_incma.pdf
16193
16194
16195 Desargues_path_can_anc_draw:
16196 ▷  echo $(DESARGUES_PATH_CANONICAL_ANCESTOR) >Desargues_path_can_anc.inc
16197 ▷  $(ORBITER) -v 10 \
16198 ▷  ▷  -draw_incidence_structure_description \
16199 ▷  ▷  ▷  -width 60 -width_10 6 -end \
16200 ▷  ▷  -define C -combinatorial_object \
16201 ▷  ▷  ▷  -label geo_10_3 geo\_10\_3 \
16202 ▷  ▷  ▷  -file_of_incidence_geometries_by_row_ranks Desargues_path_can_anc.inc 10 10
       3 \
16203 ▷  ▷  -end \
16204 ▷  ▷  -with C -do \
16205 ▷  ▷  -combinatorial_object_activity \
16206 ▷  ▷  ▷  -draw_incidence_matrices \
16207 ▷  ▷  ▷  Desargues_path_can_anc \
16208 ▷  ▷  -end
16209 ▷  pdflatex Desargues_path_can_anc_incma.tex
16210 ▷  $(OPEN) Desargues_path_can_anc_incma.pdf
16211
16212
16213
16214 geo_11_3:
```

```
16215 ▷  $(ORBITER) -v 2 \
16216 ▷  ▷  -define Test_lines -set -loop 4 12 1 -end \
16217 ▷  ▷  -define Geo -geometry_builder \
16218 ▷  ▷  ▷  -V 11 -B 11 -TDO 3 \
16219 ▷  ▷  ▷  -fuse 1 -fname_GEO 11_3 \
16220 ▷  ▷  ▷  -output_to_inc_file \
16221 ▷  ▷  ▷  -test Test_lines \
16222 ▷  ▷  -end
16223
16224 # 31 geos
16225 # 8/26/2021: 0 sec on Mac
16226
16227 geo_12_3:
16228 ▷  $(ORBITER) -v 2 \
16229 ▷  ▷  -define Test_lines -set -loop 4 13 1 -end \
16230 ▷  ▷  -define Geo -geometry_builder \
16231 ▷  ▷  ▷  -V 12 -B 12 -TDO 3 \
16232 ▷  ▷  ▷  -fuse 1 -fname_GEO 12_3 \
16233 ▷  ▷  ▷  -output_to_inc_file \
16234 ▷  ▷  ▷  -test Test_lines \
16235 ▷  ▷  ▷  -output_to_sage_file \
16236 ▷  ▷  -end
16237
16238 # 229 geos
16239 #User time: 0.45 of a second, dt=45 tps = 100
16240 #nb_calls_to_densenauty=24586
16241
16242
16243 geo_12_3_orderly:
16244 ▷  $(ORBITER) -v 2 \
16245 ▷  ▷  -define Test_lines -set -loop 4 13 1 -end \
16246 ▷  ▷  -define Geo -geometry_builder \
16247 ▷  ▷  ▷  -V 12 -B 12 -TDO 3 \
16248 ▷  ▷  ▷  -fuse 1 -fname_GEO 12_3 \
16249 ▷  ▷  ▷  -output_to_inc_file \
16250 ▷  ▷  ▷  -test Test_lines \
16251 ▷  ▷  ▷  -orderly \
16252 ▷  ▷  -end
16253
16254
16255
16256 geo_13_3:
16257 ▷  $(ORBITER) -v 2 \
16258 ▷  ▷  -define Test_lines -set -loop 4 14 1 -end \
16259 ▷  ▷  -define Geo -geometry_builder \
16260 ▷  ▷  ▷  -V 13 -B 13 -TDO 3 \
16261 ▷  ▷  ▷  -fuse 1 -fname_GEO 13_3 \
16262 ▷  ▷  ▷  -output_to_inc_file \
16263 ▷  ▷  ▷  -test Test_lines \
16264 ▷  ▷  -end
16265
16266 # 2036 geos, 96, 39, 13, 12^4, 8^3, 6^16, 4^30, 3^20, 2^190, 1^1770
16267 #User time: 0:4
16268 #nb_calls_to_densenauty=216777
16269
16270 geo_13_3_orderly:
16271 ▷  $(ORBITER) -v 2 \
16272 ▷  ▷  -define Test_lines -set -loop 4 14 1 -end \
16273 ▷  ▷  -define Geo -geometry_builder \
```

```
16274 ▷  ▷  ▷  -V 13 -B 13 -TDO 3 \
16275 ▷  ▷  ▷  -fuse 1 -fname_GEO 13_3 \
16276 ▷  ▷  ▷  -output_to_inc_file \
16277 ▷  ▷  ▷  -test Test_lines \
16278 ▷  ▷  ▷  -orderly \
16279 ▷  ▷  -end
16280
16281
16282
16283 geo_14_3:
16284 ▷  $(ORBITER) -v 2 \
16285 ▷  ▷  -define Test_lines -set -loop 4 15 1 -end \
16286 ▷  ▷  -define Geo -geometry_builder \
16287 ▷  ▷  ▷  -V 14 -B 14 -TDO 3 \
16288 ▷  ▷  ▷  -fuse 1 -fname_GEO 14_3 \
16289 ▷  ▷  ▷  -output_to_inc_file \
16290 ▷  ▷  ▷  -test Test_lines \
16291 ▷  ▷  -end
16292
16293 # 21399 geos, 56448, 128, 24^2, 16^3, 14^3, 12^7, 8^15, 7, 6^12, 4^91, 3^19, 2^91
      6, 1^20328
16294 #User time: 0:55
16295 #nb_calls_to_densenauty=2089344
16296
16297
16298 geo_14_3_orderly:
16299 ▷  $(ORBITER) -v 2 \
16300 ▷  ▷  -define Test_lines -set -loop 4 15 1 -end \
16301 ▷  ▷  -define Geo -geometry_builder \
16302 ▷  ▷  ▷  -V 14 -B 14 -TDO 3 \
16303 ▷  ▷  ▷  -fuse 1 -fname_GEO 14_3 \
16304 ▷  ▷  ▷  -output_to_inc_file \
16305 ▷  ▷  ▷  -test Test_lines \
16306 ▷  ▷  ▷  -orderly \
16307 ▷  ▷  -end
16308
16309 #User time: 0:50
16310
16311
16312 15_3.inc:
16313 ▷  $(ORBITER) -v 2 \
16314 ▷  ▷  -define Test_lines -set -loop 4 16 1 -end \
16315 ▷  ▷  -define Geo -geometry_builder \
16316 ▷  ▷  ▷  -V 15 -B 15 -TDO 3 \
16317 ▷  ▷  ▷  -fuse 1 -fname_GEO 15_3 \
16318 ▷  ▷  ▷  -output_to_inc_file \
16319 ▷  ▷  ▷  -test Test_lines \
16320 ▷  ▷  ▷  -special_test_not_orderly \
16321 ▷  ▷  -end
16322
16323 # 245342 geos, 8064, 720, 192^2, 128, 72, 48^6, 32, 30^2, 24^2, 20^2, 18, 16^10,
      15^2, 12^11, 10^3, 8^34, 6^59, 5^5, 4^180, 3^69, 2^3709, 1^241240
16324 ## 8 min 11 sec on Mac
16325 # running out of memory
16326
16327
16328 geo_15_3_g4:
16329 ▷  $(ORBITER) -v 2 \
16330 ▷  ▷  -define Test_lines -set -loop 4 16 1 -end \
```

```
16331 ▷  ▷  -define Geo -geometry_builder \
16332 ▷  ▷  ▷  -V 15 -B 15 -TDO 3 \
16333 ▷  ▷  ▷  -fuse 1 -fname_GEO 15_3_g4 \
16334 ▷  ▷  ▷  -output_to_inc_file \
16335 ▷  ▷  ▷  -girth 4 \
16336 ▷  ▷  ▷  -search_tree \
16337 ▷  ▷  ▷  -special_test_not_orderly \
16338 ▷  ▷  ▷  -test Test_lines \
16339 ▷  ▷  -end
16340 ▷  $(ORBITER) -v 2 \
16341 ▷  ▷  -draw_options -embedded -radius 50 \
16342 ▷  ▷  ▷  -nodes_empty \
16343 ▷  ▷  ▷  -scale 0.5 -line_width 0.3 -end \
16344 ▷  ▷  -tree_draw -file 15_3_g4_tree.txt -end
16345 ▷  pdflatex 15_3_g4_tree_draw.tex
16346 ▷  $(OPEN) 15_3_g4_tree_draw.pdf
16347
16348 # The unique Cremona Richmond configuration with group of order 720
16349 # cubic surface with 15 real lines and 15 real tritangent planes.
16350 #User time: 0 of a second, dt=0 tps = 100
16351 #nb_calls_to_densenauty=23
16352
16353 #-sideways
16354
16355
16356
16357
16358 geo_17_3_g4_orderly:
16359 ▷  $(ORBITER) -v 2 \
16360 ▷  ▷  -memory_debug 2 \
16361 ▷  ▷  -define Test_lines -set -loop 4 18 1 -end \
16362 ▷  ▷  -define Geo -geometry_builder \
16363 ▷  ▷  ▷  -V 17 -B 17 -TDO 3 \
16364 ▷  ▷  ▷  -fuse 1 -fname_GEO 17_3_g4 \
16365 ▷  ▷  ▷  -output_to_inc_file \
16366 ▷  ▷  ▷  -girth 4 \
16367 ▷  ▷  ▷  -test Test_lines \
16368 ▷  ▷  ▷  -orderly \
16369 ▷  ▷  -end
16370
16371 # 1 sol
16372
16373 geo_18_3_g4:
16374 ▷  $(ORBITER) -v 2 \
16375 ▷  ▷  -define Test_lines -set -loop 4 19 1 -end \
16376 ▷  ▷  -define Geo -geometry_builder \
16377 ▷  ▷  ▷  -V 18 -B 18 -TDO 3 \
16378 ▷  ▷  ▷  -fuse 1 -fname_GEO 18_3_g4 \
16379 ▷  ▷  ▷  -output_to_inc_file \
16380 ▷  ▷  ▷  -girth 4 \
16381 ▷  ▷  ▷  -search_tree \
16382 ▷  ▷  ▷  -special_test_not_orderly \
16383 ▷  ▷  ▷  -test Test_lines \
16384 ▷  ▷  -end
16385
16386 # 4 sol, 1 sec
16387
16388
16389 geo_19_3_g4:
```

```
16390 ▷  $(ORBITER) -v 2 \
16391 ▷  ▷  -define Test_lines -set -loop 4 20 1 -end \
16392 ▷  ▷  -define Geo -geometry_builder \
16393 ▷  ▷  ▷  -V 19 -B 19 -TDO 3 \
16394 ▷  ▷  ▷  -fuse 1 -fname_GEO 19_3_g4 \
16395 ▷  ▷  ▷  -output_to_inc_file \
16396 ▷  ▷  ▷  -girth 4 \
16397 ▷  ▷  ▷  -special_test_not_orderly \
16398 ▷  ▷  ▷  -test Test_lines \
16399 ▷  ▷  -end
16400
16401 # 14 sol, 10 sec on Mac
16402
16403 geo_20_3_g4:
16404 ▷  $(ORBITER) -v 2 \
16405 ▷  ▷  -define Test_lines -set -loop 4 21 1 -end \
16406 ▷  ▷  -define Geo -geometry_builder \
16407 ▷  ▷  ▷  -V 20 -B 20 -TDO 3 \
16408 ▷  ▷  ▷  -fuse 1 -fname_GEO 20_3_g4 \
16409 ▷  ▷  ▷  -output_to_inc_file \
16410 ▷  ▷  ▷  -girth 4 \
16411 ▷  ▷  ▷  -special_test_not_orderly \
16412 ▷  ▷  ▷  -test Test_lines \
16413 ▷  ▷  -end
16414
16415 # 162 sol, User time: 2:5 on Mac
16416
16417 geo_21_3_g4:
16418 ▷  $(ORBITER) -v 2 \
16419 ▷  ▷  -define Test_lines -set -loop 4 22 1 -end \
16420 ▷  ▷  -define Geo -geometry_builder \
16421 ▷  ▷  ▷  -V 21 -B 21 -TDO 3 \
16422 ▷  ▷  ▷  -fuse 1 -fname_GEO 21_3_g4 \
16423 ▷  ▷  ▷  -output_to_inc_file \
16424 ▷  ▷  ▷  -girth 4 \
16425 ▷  ▷  ▷  -special_test_not_orderly \
16426 ▷  ▷  ▷  -test Test_lines \
16427 ▷  ▷  -end
16428
16429
16430 geo_15_4:
16431 ▷  $(ORBITER) -v 2 \
16432 ▷  ▷  -define Test_lines -set -loop 4 16 1 -end \
16433 ▷  ▷  -define Geo -geometry_builder \
16434 ▷  ▷  ▷  -V 15 -B 15 -TDO 4 \
16435 ▷  ▷  ▷  -fuse 1 -fname_GEO 15_4 \
16436 ▷  ▷  ▷  -output_to_inc_file \
16437 ▷  ▷  ▷  -search_tree \
16438 ▷  ▷  ▷  -special_test_not_orderly \
16439 ▷  ▷  ▷  -test Test_lines \
16440 ▷  ▷  -end
16441 ▷  $(ORBITER) -v 2 \
16442 ▷  ▷  -draw_options -embedded -radius 50 \
16443 ▷  ▷  ▷  -nodes_empty \
16444 ▷  ▷  ▷  -scale 0.5 -line_width 0.3 -end \
16445 ▷  ▷  -tree_draw -file 15_4_tree.txt -end
16446 ▷  pdflatex 15_4_tree_draw.tex
16447 ▷  $(OPEN) 15_4_tree_draw.pdf
16448
```

```
16449 # 4 objects
16450 # ago=360, 30, 24, 15
16451 #User time: 0.15 of a second, dt=15 tps = 100
16452 #nb_calls_to_densenauty=6767
16453
16454
16455
16456 geo_16_4_g4:
16457 ▷  $(ORBITER) -v 2 \
16458 ▷  ▷  -define Test_lines -set -loop 4 17 1 -end \
16459 ▷  ▷  -define Geo -geometry_builder \
16460 ▷  ▷  ▷  -V 16 -B 16 -TDO 4 \
16461 ▷  ▷  ▷  -fuse 1 -fname_GEO 16_4_g4 \
16462 ▷  ▷  ▷  -output_to_inc_file \
16463 ▷  ▷  ▷  -special_test_not_orderly \
16464 ▷  ▷  ▷  -girth 4 \
16465 ▷  ▷  ▷  -test Test_lines \
16466 ▷  ▷  -end
16467
16468 # none
16469
16470
16471 geo_15_6:
16472 ▷  $(ORBITER) -v 2 \
16473 ▷  ▷  -define Test_lines -set -loop 4 16 1 -end \
16474 ▷  ▷  -define Geo -geometry_builder \
16475 ▷  ▷  ▷  -V 15 -B 15 -TDO 6 -fuse 1 \
16476 ▷  ▷  ▷  -fname_GEO 15_6 \
16477 ▷  ▷  ▷  -special_test_not_orderly \
16478 ▷  ▷  ▷  -output_to_inc_file \
16479 ▷  ▷  ▷  -test Test_lines \
16480 ▷  ▷  ▷  -output_to_sage_file \
16481 ▷  ▷  -end
16482
16483
16484
16485
16486 geo_16:
16487 ▷  $(ORBITER) -v 2 \
16488 ▷  ▷  -define Test_lines -set -loop 3 17 1 -end \
16489 ▷  ▷  -define Geo -geometry_builder \
16490 ▷  ▷  ▷  -V 16 -B 20 -TDO 5 \
16491 ▷  ▷  ▷  -fuse 1 -fname_GEO geo_16 \
16492 ▷  ▷  ▷  -special_test_not_orderly \
16493 ▷  ▷  ▷  -output_to_inc_file \
16494 ▷  ▷  ▷  -test Test_lines \
16495 ▷  ▷  -end
16496
16497
16498 geo_40_4_g4:
16499 ▷  $(ORBITER) -v 5 \
16500 ▷  ▷  -define Test_lines -set -loop 0 41 1 -end \
16501 ▷  ▷  -define Geo -geometry_builder \
16502 ▷  ▷  ▷  -V 40 -B 40 -TDO 4 \
16503 ▷  ▷  ▷  -fuse 1 \
16504 ▷  ▷  ▷  -fname_GEO 40_4_g4 \
16505 ▷  ▷  ▷  -girth 4 \
16506 ▷  ▷  ▷  -search_tree \
16507 ▷  ▷  ▷  -special_test_not_orderly \
```

```
16508 ▷  ▷  ▷  -test Test_lines \
16509 ▷  ▷  ▷  -output_to_sage_file \
16510 ▷  ▷  ▷  -output_to_inc_file \
16511 ▷  ▷  -end
16512 ▷  $(ORBITER) -v 2 \
16513 ▷  ▷  -draw_options -embedded -radius 50 \
16514 ▷  ▷  ▷  -xin 10000 -yin 10000 \
16515 ▷  ▷  ▷  -xout 1000000 -yout 1000000 \
16516 ▷  ▷  ▷  -nodes_empty \
16517 ▷  ▷  ▷  -scale 0.5 -line_width 0.3 -end \
16518 ▷  ▷  -tree_draw -file 40_4_g4_tree.txt -end
16519 ▷  pdflatex 40_4_g4_tree_draw.tex
16520 ▷  $(OPEN) 40_4_g4_tree_draw.pdf
16521
16522
16523 #-special_test_not_orderly is important for speed purposes
16524 # 2 geos, ago=51840^2
16525 # 40_4_g4.inc
16526
16527
16528 geo_63_3_g6:
16529 ▷  $(ORBITER) -v 2 \
16530 ▷  ▷  -define Test_lines -set -loop 4 64 1 -end \
16531 ▷  ▷  -define Geo -geometry_builder \
16532 ▷  ▷  ▷  -V 63 -B 63 -TDO 3 \
16533 ▷  ▷  ▷  -fuse 1 -fname_GEO 63_3_g6 \
16534 ▷  ▷  ▷  -girth 6 \
16535 ▷  ▷  ▷  -test Test_lines \
16536 ▷  ▷  ▷  -special_test_not_orderly \
16537 ▷  ▷  ▷  -output_to_sage_file \
16538 ▷  ▷  ▷  -output_to_inc_file \
16539 ▷  ▷  ▷  -search_tree \
16540 ▷  ▷  -end
16541
16542 # time 0:38 on Mac Oct 15, 2022
16543 # 2 geos, ago: 12096^2
16544 # 63_3_g6.inc
16545 # split Cayley hexagon H(2) and its dual
16546 # group = G_2(2) = PGU(3,3)
16547
16548
16549
16550 geo_LSQ6.inc:
16551 ▷  $(ORBITER) -v 2 \
16552 ▷  ▷  -define Test_lines -set -loop 1 19 1 -end \
16553 ▷  ▷  -define Geo -geometry_builder \
16554 ▷  ▷  ▷  -V 6,6,6 -B 1,1,1,36 -TDO \
16555 ▷  ▷  ▷  "1,0,0,6, 0,1,0,6, 0,0,1,6" \
16556 ▷  ▷  ▷  -fuse 3 \
16557 ▷  ▷  ▷  -fname_GEO LSQ6 \
16558 ▷  ▷  ▷  -output_to_inc_file \
16559 ▷  ▷  ▷  -test Test_lines \
16560 ▷  ▷  -end
16561
16562 #LSQ6.inc
16563
16564
16565
16566
```

```
16567
16568 ##############################################################################
16569 # Section 11.5: Tactical Decompositions
16570
16571 SECTION_TACTICAL_DECOMPOSITIONS:
16572
16573 test_11_5:
16574 ▷  make max_arc_16_4_start
16575 ▷  make max_arc_16_4_convert_stack_tdo
16576 ▷  make max_arc_16_4_refine
16577 ▷  make max_arc_16_4r_print
16578
16579
16580 max_arc_16_4_start:
16581 ▷  $(ORBITER) -v 4 -maximal_arc_parameters 16 4
16582
16583
16584 max_arc_16_4_convert_stack_tdo:
16585 ▷  $(ORBITER) -v 4 -convert_stack_to_tdo max_arc_q16_r4.stack
16586
16587 max_arc_16_4_refine:
16588 ▷  $(ORBITER) -v 4 -tdo_refinement \
16589 ▷  ▷  -input_file max_arc_q16_r4.tdo -dual_is_linear_space -end
16590 ▷
16591 max_arc_16_4r_print:
16592 ▷  $(ORBITER) -v 4 -tdo_print max_arc_q16_r4r.tdo
16593
16594
16595
16596
16597
16598 ##############################################################################
16599 # Chapter 12 - Graph Theory
16600 ##############################################################################
16601
16602
16603 test_12:
16604 ▷  make test_12_1
16605 ▷  make test_12_2
16606 ▷  make test_12_3
16607 ▷  make test_12_4
16608
16609
16610 ##############################################################################
16611 # Section 12.1: Creating Graphs
16612
16613
16614 SECTION_CREATING_GRAPHS:
16615
16616
16617 test_12_1:
16618 ▷  make Cycle_graph_13
16619 ▷  make make_triangle_graph
16620 ▷  make Chain_232
16621 ▷  make Paley_13_graph
16622 ▷  make Paley_13_graph_complement
16623 ▷  make Winnie_Li_graph_q16_index2
16624 ▷  make Sarnak_graph_29_5
16625 ▷  make tritangent_planes_graph
```

```
16626 ▷  make trihedral_pair_graph
16627 ▷  make small_graph
16628 ▷  make petersen
16629 ▷  make Johnson_6_2_0
16630 ▷  make Hamming_graph_3
16631 ▷  make Hamming_graph_7
16632 ▷  make Op_6_2_coll_graph
16633 ▷  make HJ_graph
16634 ▷  make Group_Perm315_Orbital_3.colored_graph
16635 ▷  make HJ_d2_graph
16636 ▷  make Cayley_Z11_1mod3
16637 ▷  make Cayley_Sym4_coxeter
16638 ▷  make Cayley_Sym4_star
16639 ▷  make Queens_graph
16640 ▷
16641
16642
16643 Cycle_graph_13:
16644 ▷  $(ORBITER) -v 2 \
16645 ▷  ▷  -define Gamma -graph \
16646 ▷  ▷  ▷  -cycle 13 \
16647 ▷  ▷  -end
16648
16649
16650
16651
16652 make_triangle_graph:
16653 ▷  echo $(TRIANGLE_GRAPH) >triangle_graph.csv
16654 ▷  $(ORBITER) -v 6 \
16655 ▷  ▷  -define G -graph \
16656 ▷  ▷  ▷  -load_csv_no_border \
16657 ▷  ▷  ▷  triangle_graph.csv \
16658 ▷  ▷  -end
16659
16660
16661
16662 Chain_232:
16663 ▷  $(ORBITER) -v 2 \
16664 ▷  ▷  -define P1 -vector -dense 2,3,2 -end \
16665 ▷  ▷  -define P2 -vector -dense 2,3,2 -end \
16666 ▷  ▷  -define Gamma -graph \
16667 ▷  ▷  ▷  -chain_graph P1 P2 \
16668 ▷  ▷  -end
16669
16670
16671
16672
16673 Paley_13_graph:
16674 ▷  $(ORBITER) -v 2 \
16675 ▷  ▷  -define F -finite_field -q 13 -end \
16676 ▷  ▷  -define Gamma -graph -Paley F -end \
16677
16678
16679 Paley_13_graph_complement:
16680 ▷  $(ORBITER) -v 2 \
16681 ▷  ▷  -define F -finite_field -q 13 -end \
16682 ▷  ▷  -define Gamma -graph -Paley F -complement -end \
16683
16684 Winnie_Li_graph_q16_index2:
```

```
16685 ▷  $(ORBITER) -v 2 \
16686 ▷  ▷  -define F -finite_field -q 16 -end \
16687 ▷  ▷  -define Gamma -graph -Winnie_Li F 2 -end \
16688
16689
16690 Sarnak_graph_29_5:
16691 ▷  $(ORBITER) -v 2 \
16692 ▷  ▷  -define Gamma -graph -Sarnak 29 5 -end \
16693
16694
16695 tritangent_planes_graph:
16696 ▷  $(ORBITER) -v 2 \
16697 ▷  ▷  -define Gamma \
16698 ▷  ▷  ▷  -graph -tritangent_planes_disjointness_graph \
16699 ▷  ▷  -end
16700
16701
16702 trihedral_pair_graph:
16703 ▷  $(ORBITER) -v 2 \
16704 ▷  ▷  -define Gamma \
16705 ▷  ▷  ▷  -graph -trihedral_pair_disjointness_graph \
16706 ▷  ▷  -end
16707
16708
16709
16710 small_graph:
16711 ▷  $(ORBITER) -v 2 \
16712 ▷  ▷  -define G -graph -edges_as_pairs \
16713 ▷  ▷  ▷  5 "0,1,0,2,0,3,0,4,1,3,1,4,2,4" \
16714 ▷  ▷  -end
16715
16716
16717
16718
16719 petersen:
16720 ▷  $(ORBITER) -v 2 \
16721 ▷  ▷  -define G -graph -Johnson 5 2 0 -end
16722
16723
16724
16725
16726 Johnson_6_2_0:
16727 ▷  $(ORBITER) -v 2 \
16728 ▷  ▷  -define G -graph -Johnson 6 2 0 -end
16729
16730 Hamming_graph_3:
16731 ▷  $(ORBITER) -v 2 \
16732 ▷  ▷  -define G -graph -Hamming 3 2 -end
16733
16734
16735 Hamming_graph_7:
16736 ▷  $(ORBITER) -v 2 \
16737 ▷  ▷  -define G -graph -Hamming 7 2 -end
16738
16739 # needs halljanko315.csv
16740 # from https://www.win.tue.nl/~aeb/drg/graphs/HJ315.html
16741 #There is a unique distance-regular graph Gamma with intersection array {10,8,8,2
      ; 1,1,4,5}. It was constructed in Cohen (1981), and uniqueness (given the interse
      ction array) was proved in Cohen & Tits (1985).
```

```
16742
16743
16744
16745 Op_6_2_coll_graph:
16746 ▷  $(ORBITER) -v 2 \
16747 ▷  ▷  -define F -finite_field -q 13 -end \
16748 ▷  ▷  -define O -orthogonal_space 1 6 F -end \
16749 ▷  ▷  -define v -vector -loop 0 35 1 -end \
16750 ▷  ▷  -define Gamma -graph -coll_orthogonal O v -end \
16751 ▷  ▷  -with Gamma -do \
16752 ▷  ▷  ▷  -graph_theoretic_activity -save \
16753 ▷  ▷  -end
16754
16755
16756 #O_1_6_13_coll_v.colored_graph
16757
16758
16759 HJ_graph:
16760 ▷  cp $(MY_PATH)/examples/users_guide/halljanko315.csv .
16761 ▷  $(ORBITER) -v 6 \
16762 ▷  ▷  -define G -graph \
16763 ▷  ▷  ▷  -load_csv_no_border \
16764 ▷  ▷  ▷  halljanko315.csv \
16765 ▷  ▷  -end
16766
16767
16768 Group_Perm315_Orbital_3.colored_graph: halljanko315_gens.csv
16769 ▷  $(ORBITER) -v 2 \
16770 ▷  ▷  -define gens -vector -format 5 -file \
16771 ▷  ▷  ▷  halljanko315_gens.csv -end \
16772 ▷  ▷  -define G -permutation_group \
16773 ▷  ▷  ▷  -bsgs halljanko315 "File\_halljanko315" \
16774 ▷  ▷  ▷  315 1209600 "0,1,2" 5 gens \
16775 ▷  ▷  -end \
16776 ▷  ▷  -define Gamma -graph \
16777 ▷  ▷  ▷  -orbital_graph G 3 \
16778 ▷  ▷  -end \
16779 ▷  ▷  -with Gamma -do \
16780 ▷  ▷  ▷  -graph_theoretic_activity -save \
16781 ▷  ▷  -end
16782
16783 HJ_d2_graph:
16784 ▷  $(ORBITER) -v 6 \
16785 ▷  ▷  -define G -graph \
16786 ▷  ▷  ▷  -load_csv_no_border \
16787 ▷  ▷  ▷  halljanko315.csv \
16788 ▷  ▷  ▷  -distance_2 \
16789 ▷  ▷  -end
16790
16791
16792
16793 Cayley_Z11_1mod3:
16794 ▷  $(ORBITER) -v 2 \
16795 ▷  ▷  -define F -finite_field -q 11 -end \
16796 ▷  ▷  -define S -vector -dense \
16797 ▷  ▷  ▷  "1,1, 1,4, 1,7, 1,10" -end \
16798 ▷  ▷  -define G -linear_group -AGL 1 F \
16799 ▷  ▷  ▷  -subgroup_by_generators "Z11" 11 1 "1,1" \
16800 ▷  ▷  -end \
```

```
16801 ▷  ▷  -define Gamma -graph \
16802 ▷  ▷  ▷  -Cayley_graph G S \
16803 ▷  ▷  -end
16804
16805
16806 Cayley_Sym4_coxeter:
16807 ▷  $(ORBITER) -v 2 \
16808 ▷  ▷  -define S -vector -dense "1,0,2,3, 0,2,1,3, 0,1,3,2" -end \
16809 ▷  ▷  -define G -permutation_group -symmetric_group 4 \
16810 ▷  ▷  -end \
16811 ▷  ▷  -define Gamma -graph \
16812 ▷  ▷  ▷  -Cayley_graph G S \
16813 ▷  ▷  -end
16814
16815
16816 Cayley_Sym4_star:
16817 ▷  $(ORBITER) -v 2 \
16818 ▷  ▷  -define S -vector -dense "1,0,2,3, 2,1,0,3, 3,1,2,0" -end \
16819 ▷  ▷  -define G -permutation_group -symmetric_group 4 \
16820 ▷  ▷  -end \
16821 ▷  ▷  -define Gamma -graph \
16822 ▷  ▷  ▷  -Cayley_graph G S \
16823 ▷  ▷  -end
16824
16825 Queens_graph:
16826 ▷  $(ORBITER) -v 2 \
16827 ▷  ▷  -define G -graph -non_attacking_queens_graph 8 -end \
16828
16829
16830 ##############################################################################
16831 # Section 12.2: Graphs Theoretic Activities
16832
16833
16834 SECTION_GRAPH_THEORETIC_ACTIVITIES:
16835
16836
16837 test_12_2:
16838 ▷  make triangle_graph_properties
16839 ▷  make Cycle_13_draw
16840 ▷  make Cycle_9_eigenvalues
16841 ▷  make Paley_13_draw
16842 ▷  make Paley_13_eigenvalues
16843 ▷  make tritangent_planes_graph_eigenvalues
16844 ▷  make Cayley_Z11_1mod3_eigenvalues_and_draw
16845 ▷  make Cayley_Sym4_coxeter_draw
16846 ▷  make Cayley_Sym5_coxeter_draw
16847 ▷  make Cayley_Sym4_star_eigenvalues_and_draw
16848 ▷  make graph_v5_e7.colored_graph
16849 ▷  make small_graph_draw
16850 ▷  make petersen_draw
16851 ▷  make Johnson_6_2_0_draw
16852 ▷  make Hamming_graph_3_draw
16853 ▷  make Hamming_graph_7_draw
16854 ▷  make Chain_232_properties
16855 ▷  make Chain_232_eigen
16856 ▷  make HJ_properties
16857 ▷  make HJ_d2_properties
16858 ▷  make PGO_5_2_collinearity_graph
16859 ▷  make trihedral_pair_graph_draw
```

```
16860
16861
16862 triangle_graph_properties:
16863 ▷  echo $(TRIANGLE_GRAPH) >triangle_graph.csv
16864 ▷  $(ORBITER) -v 6 \
16865 ▷  ▷  -define G -graph \
16866 ▷  ▷  ▷  -load_csv_no_border \
16867 ▷  ▷  ▷  triangle_graph.csv \
16868 ▷  ▷  -end \
16869 ▷  ▷  -with G -do \
16870 ▷  ▷  ▷  -graph_theoretic_activity -properties \
16871 ▷  ▷  -end
16872
16873
16874 Cycle_13_draw:
16875 ▷  $(ORBITER) -v 2 \
16876 ▷  ▷  -define Gamma -graph -cycle 13 -end \
16877 ▷  ▷  -with Gamma -do \
16878 ▷  ▷  -graph_theoretic_activity -export_csv -end \
16879 ▷  ▷  -with Gamma -do \
16880 ▷  ▷  -graph_theoretic_activity -export_graphviz -end
16881 ▷  $(ORBITER) -v 2 -draw_matrix \
16882 ▷  ▷  -input_csv_file Cycle_13.csv \
16883 ▷  ▷  -box_width 20 -bit_depth 8 -partition 4 13 13 -end
16884 ▷  dot -Tpng Cycle_13.gv >Cycle_13.png
16885 ▷  #twopi -Tpng Cycle_13.gv >Cycle_13.png
16886 ▷  #$(OPEN) Cycle_13_draw.bmp
16887 ▷  #pdflatex Cycle_13_report.tex
16888 ▷  #$(OPEN) Cycle_13_report.pdf
16889
16890
16891 Cycle_9_eigenvalues:
16892 ▷  $(ORBITER) -v 2 \
16893 ▷  ▷  -define Gamma -graph \
16894 ▷  ▷  ▷  -cycle 9 \
16895 ▷  ▷  -end \
16896 ▷  ▷  -with Gamma -do \
16897 ▷  ▷  -graph_theoretic_activity -eigenvalues -end
16898 ▷  pdflatex Cycle_9_eigenvalues.tex
16899 ▷  $(OPEN) Cycle_9_eigenvalues.pdf
16900
16901
16902 Paley_13_draw:
16903 ▷  $(ORBITER) -v 2 \
16904 ▷  ▷  -define F -finite_field -q 13 -end \
16905 ▷  ▷  -define Gamma -graph -Paley F -end \
16906 ▷  ▷  -with Gamma -do \
16907 ▷  ▷  -graph_theoretic_activity -export_csv -end \
16908 ▷  ▷  -with Gamma -do \
16909 ▷  ▷  -graph_theoretic_activity -export_graphviz -end
16910 ▷  $(ORBITER) -v 2 -draw_matrix \
16911 ▷  ▷  -input_csv_file Paley_13.csv \
16912 ▷  ▷  -box_width 20 -bit_depth 8 -partition 4 13 13 -end
16913 ▷  dot -Tpng Paley_13.gv >Paley_13.png
16914 ▷  $(OPEN) Paley_13_draw.bmp
16915
16916
16917 Paley_13_eigenvalues:
16918 ▷  $(ORBITER) -v 2 \
```

```
16919 ▷  ▷  -define F -finite_field -q 13 -end \
16920 ▷  ▷  -define Gamma -graph -Paley F -end \
16921 ▷  ▷  -with Gamma -do \
16922 ▷  ▷  -graph_theoretic_activity -eigenvalues -end
16923 ▷  pdflatex Paley_13_eigenvalues.tex
16924 ▷  $(OPEN) Paley_13_eigenvalues.pdf
16925
16926
16927 tritangent_planes_graph_eigenvalues:
16928 ▷  $(ORBITER) -v 2 \
16929 ▷  ▷  -define Gamma \
16930 ▷  ▷  ▷  -graph -tritangent_planes_disjointness_graph \
16931 ▷  ▷  -end \
16932 ▷  ▷  -with Gamma -do \
16933 ▷  ▷  -graph_theoretic_activity -eigenvalues -end
16934
16935
16936
16937 Cayley_Z11_1mod3_eigenvalues_and_draw:
16938 ▷  $(ORBITER) -v 2 \
16939 ▷  ▷  -draw_options -xin 2000000 \
16940 ▷  ▷  ▷  -yin 2000000 -embedded -radius 20000 -end \
16941 ▷  ▷  -define F -finite_field -q 11 -end \
16942 ▷  ▷  -define S -vector -dense \
16943 ▷  ▷  ▷  "1,1, 1,4, 1,7, 1,10" -end \
16944 ▷  ▷  -define G -linear_group -AGL 1 F \
16945 ▷  ▷  ▷  -subgroup_by_generators "Z11" 11 1 "1,1" \
16946 ▷  ▷  -end \
16947 ▷  ▷  -define Gamma -graph \
16948 ▷  ▷  ▷  -Cayley_graph G S \
16949 ▷  ▷  -end \
16950 ▷  ▷  -with Gamma -do \
16951 ▷  ▷  -graph_theoretic_activity -eigenvalues -end \
16952 ▷  ▷  -with Gamma -do \
16953 ▷  ▷  -graph_theoretic_activity -draw -end
16954 ▷  pdflatex Cayley_graph_AGL_1_11_draw.tex
16955 ▷  $(OPEN) Cayley_graph_AGL_1_11_draw.pdf
16956
16957
16958 Cayley_Sym4_coxeter_draw:
16959 ▷  $(ORBITER) -v 2 \
16960 ▷  ▷  -draw_options -xin 2000000 -yin 2000000 \
16961 ▷  ▷  ▷  -radius 20000 -embedded -nodes_empty -end \
16962 ▷  ▷  -define S -vector -dense \
16963 ▷  ▷  ▷  "1,0,2,3, 0,2,1,3, 0,1,3,2" -end \
16964 ▷  ▷  -define G -permutation_group -symmetric_group 4 \
16965 ▷  ▷  -end \
16966 ▷  ▷  -define Gamma -graph \
16967 ▷  ▷  ▷  -Cayley_graph G S \
16968 ▷  ▷  -end \
16969 ▷  ▷  -with Gamma -do \
16970 ▷  ▷  -graph_theoretic_activity -draw -end
16971 ▷  pdflatex Cayley_graph_Sym_4_draw.tex
16972 ▷  $(OPEN) Cayley_graph_Sym_4_draw.pdf
16973
16974 Cayley_Sym5_coxeter_draw:
16975 ▷  $(ORBITER) -v 2 \
16976 ▷  ▷  -draw_options -xin 1000000 -yin 1000000 \
16977 ▷  ▷  ▷  -embedded -radius 10000 -nodes_empty -end \
```

```
16978 ▷  ▷  -define S -vector -dense \
16979 ▷  ▷  ▷  "1,0,2,3,4, 0,2,1,3,4, 0,1,3,2,4, 0,1,2,4,3" -end \
16980 ▷  ▷  -define G -permutation_group -symmetric_group 5 \
16981 ▷  ▷  -end \
16982 ▷  ▷  -define Gamma -graph \
16983 ▷  ▷  ▷  -Cayley_graph G S \
16984 ▷  ▷  -end \
16985 ▷  ▷  -with Gamma -do \
16986 ▷  ▷  -graph_theoretic_activity -draw -end
16987 ▷  pdflatex Cayley_graph_Sym_5_draw.tex
16988 ▷  $(OPEN) Cayley_graph_Sym_5_draw.pdf
16989
16990
16991 Cayley_Sym4_star_eigenvalues_and_draw:
16992 ▷  $(ORBITER) -v 2 \
16993 ▷  ▷  -draw_options -xin 1000000 -yin 1000000 -embedded -end \
16994 ▷  ▷  -define S -vector -dense "1,0,2,3, 2,1,0,3, 3,1,2,0" -end \
16995 ▷  ▷  -define G -permutation_group -symmetric_group 4 \
16996 ▷  ▷  -end \
16997 ▷  ▷  -define Gamma -graph \
16998 ▷  ▷  ▷  -Cayley_graph G S \
16999 ▷  ▷  -end \
17000 ▷  ▷  -with Gamma -do \
17001 ▷  ▷  -graph_theoretic_activity -eigenvalues -end \
17002 ▷  ▷  -with Gamma -do \
17003 ▷  ▷  -graph_theoretic_activity -draw -end
17004 ▷  pdflatex Cayley_graph_Sym_4_draw.tex
17005 ▷  $(OPEN) Cayley_graph_Sym_4_draw.pdf
17006 ▷  pdflatex Cayley_graph_Sym_4_eigenvalues.tex
17007 ▷  $(OPEN) Cayley_graph_Sym_4_eigenvalues.pdf
17008
17009
17010
17011
17012 graph_v5_e7.colored_graph:
17013 ▷  $(ORBITER) -v 2 \
17014 ▷  ▷  -define G -graph -edges_as_pairs 5 \
17015 ▷  ▷  ▷  "0,1,0,2,0,3,0,4,1,3,1,4,2,4" \
17016 ▷  ▷  -end \
17017 ▷  ▷  -with G -do \
17018 ▷  ▷  -graph_theoretic_activity -save -end
17019
17020
17021
17022 small_graph_draw:
17023 ▷  $(ORBITER) -v 2 \
17024 ▷  ▷  -define G -graph -edges_as_pairs 5 \
17025 ▷  ▷  ▷  "0,1,0,2,0,3,0,4,1,3,1,4,2,4" \
17026 ▷  ▷  -end \
17027 ▷  ▷  -with G -do \
17028 ▷  ▷  -graph_theoretic_activity -export_csv -end \
17029 ▷  ▷  -with G -do \
17030 ▷  ▷  -graph_theoretic_activity -export_graphviz -end \
17031 ▷  ▷  -with G -do \
17032 ▷  ▷  -graph_theoretic_activity -save -end
17033 ▷  $(ORBITER) -v 2 -draw_matrix \
17034 ▷  ▷  -input_csv_file graph_v5_e7.csv \
17035 ▷  ▷  -box_width 40 -bit_depth 24 \
17036 ▷  ▷  -partition 4 "1,1,1,1,1" "1,1,1,1,1" -end
```

```
17037 ▷  dot -Tpng graph_v5_e7.gv >graph_v5_e7.png
17038
17039 # creates graph_v5_e7.csv
17040 # creates graph_v5_e7.colored_graph
17041
17042
17043 petersen_draw:
17044 ▷  $(ORBITER) -v 2 \
17045 ▷  ▷  -define G -graph -Johnson 5 2 0 -end \
17046 ▷  ▷  -with G -do \
17047 ▷  ▷  -graph_theoretic_activity -export_csv -end \
17048 ▷  ▷  -with G -do \
17049 ▷  ▷  -graph_theoretic_activity -export_graphviz -end \
17050 ▷  ▷  -with G -do \
17051 ▷  ▷  -graph_theoretic_activity -save -end
17052 ▷  $(ORBITER) -v 2 -draw_matrix \
17053 ▷  ▷  -input_csv_file Johnson_5_2_0.csv \
17054 ▷  ▷  -box_width 40 -bit_depth 24 -partition 4 "10" "10" -end
17055 ▷  dot -Tpng Johnson_5_2_0.gv >Johnson_5_2_0.png
17056
17057
17058 Johnson_6_2_0_draw:
17059 ▷  $(ORBITER) -v 2 \
17060 ▷  ▷  -define G -graph -Johnson 6 2 0 -end \
17061 ▷  ▷  -with G -do \
17062 ▷  ▷  -graph_theoretic_activity -export_csv -end \
17063 ▷  ▷  -with G -do \
17064 ▷  ▷  -graph_theoretic_activity -export_graphviz -end \
17065 ▷  ▷  -with G -do \
17066 ▷  ▷  -graph_theoretic_activity -save -end
17067 ▷  $(ORBITER) -v 2 -draw_matrix \
17068 ▷  ▷  -input_csv_file Johnson_6_2_0.csv \
17069 ▷  ▷  -box_width 40 -bit_depth 24 -partition 4 "10" "10" -end
17070 ▷  dot -Tpng Johnson_6_2_0.gv >Johnson_6_2_0.png
17071
17072
17073
17074 Hamming_graph_3_draw:
17075 ▷  $(ORBITER) -v 2 \
17076 ▷  ▷  -define G -graph -Hamming 3 2 -end \
17077 ▷  ▷  -with G -do \
17078 ▷  ▷  -graph_theoretic_activity -export_csv -end \
17079 ▷  ▷  -with G -do \
17080 ▷  ▷  -graph_theoretic_activity -export_graphviz -end \
17081 ▷  ▷  -with G -do \
17082 ▷  ▷  -graph_theoretic_activity -save -end
17083 ▷  $(ORBITER) -v 2 -draw_matrix \
17084 ▷  ▷  -input_csv_file Hamming_3_2.csv \
17085 ▷  ▷  -box_width 40 -bit_depth 24 \
17086 ▷  ▷  -partition 4 "1,1,1,1,1,1,1,1" "1,1,1,1,1,1,1,1" -end
17087 ▷  dot -Tpng Hamming_3_2.gv >Hamming_3_2.png
17088
17089
17090 Hamming_graph_7_draw:
17091 ▷  $(ORBITER) -v 2 \
17092 ▷  ▷  -define G -graph -Hamming 7 2 -end \
17093 ▷  ▷  -with G -do \
17094 ▷  ▷  -graph_theoretic_activity -export_csv -end \
17095 ▷  ▷  -with G -do \
```

```
17096 ▷  ▷  -graph_theoretic_activity -export_graphviz -end \
17097 ▷  ▷  -with G -do \
17098 ▷  ▷  -graph_theoretic_activity -save -end
17099 ▷  $(ORBITER) -v 2 -draw_matrix \
17100 ▷  ▷  -input_csv_file Hamming_7_2.csv \
17101 ▷  ▷  -box_width 8 -bit_depth 24 -partition 4 128 128 -end
17102 ▷  dot -Tpng Hamming_7_2.gv >Hamming_7_2.png
17103
17104
17105
17106
17107
17108
17109 Chain_232_properties:
17110 ▷  $(ORBITER) -v 2 \
17111 ▷  ▷  -define P1 -vector -dense 2,3,2 -end \
17112 ▷  ▷  -define P2 -vector -dense 2,3,2 -end \
17113 ▷  ▷  -define Gamma -graph \
17114 ▷  ▷  ▷  -chain_graph P1 P2 \
17115 ▷  ▷  -end \
17116 ▷  ▷  -with Gamma -do \
17117 ▷  ▷  ▷  -graph_theoretic_activity -export_csv \
17118 ▷  ▷  -end \
17119 ▷  ▷  -with Gamma -do \
17120 ▷  ▷  ▷  -graph_theoretic_activity -properties \
17121 ▷  ▷  -end
17122
17123 Chain_232_eigen:
17124 ▷  $(ORBITER) -v 2 \
17125 ▷  ▷  -define P1 -vector -dense 2,3,2 -end \
17126 ▷  ▷  -define P2 -vector -dense 2,3,2 -end \
17127 ▷  ▷  -define Gamma -graph \
17128 ▷  ▷  ▷  -chain_graph P1 P2 \
17129 ▷  ▷  -end \
17130 ▷  ▷  -with Gamma -do \
17131 ▷  ▷  ▷  -graph_theoretic_activity \
17132 ▷  ▷  ▷  -eigenvalues \
17133 ▷  ▷  -end
17134 ▷  pdflatex chain_graph_eigenvalues.tex
17135 ▷  $(OPEN) chain_graph_eigenvalues.pdf
17136
17137
17138
17139 # need the file halljanko315.csv
17140
17141 HJ_properties:
17142 ▷  $(ORBITER) -v 6 \
17143 ▷  ▷  -define G -graph \
17144 ▷  ▷  ▷  -load_csv_no_border \
17145 ▷  ▷  ▷  halljanko315.csv \
17146 ▷  ▷  -end \
17147 ▷  ▷  -with G -do \
17148 ▷  ▷  ▷  -graph_theoretic_activity -properties \
17149 ▷  ▷  -end
17150
17151 #Degree type: (10^{315} )
17152
17153
17154
```

```
17155 HJ_d2_properties:
17156 ▷  $(ORBITER) -v 6 \
17157 ▷  ▷  -define G -graph \
17158 ▷  ▷  ▷  -load_csv_no_border \
17159 ▷  ▷  ▷  halljanko315.csv \
17160 ▷  ▷  ▷  -distance_2 \
17161 ▷  ▷  -end \
17162 ▷  ▷  -with G -do \
17163 ▷  ▷  ▷  -graph_theoretic_activity \
17164 ▷  ▷  ▷  -properties \
17165 ▷  ▷  -end
17166
17167
17168 #Degree type: (80^{315} )
17169
17170
17171
17172
17173 PGO_5_2_collinearity_graph: O_5_2_incidence_matrix.csv
17174 ▷  $(ORBITER) -v 3 \
17175 ▷  ▷  -define Inc -vector -file O_5_2_incidence_matrix.csv -end \
17176 ▷  ▷  -define Gamma -graph -collinearity_graph Inc -end \
17177 ▷  ▷  -with Gamma -do \
17178 ▷  ▷  -graph_theoretic_activity \
17179 ▷  ▷  ▷  -properties \
17180 ▷  ▷  -end
17181
17182
17183 trihedral_pair_graph_draw:
17184 ▷  $(ORBITER) -v 2 -define Gamma \
17185 ▷  ▷  -graph -trihedral_pair_disjointness_graph -end \
17186 ▷  ▷  -with Gamma -do \
17187 ▷  ▷  -graph_theoretic_activity -export_csv -end
17188 ▷  $(ORBITER) -v 2 -draw_matrix \
17189 ▷  ▷  -input_csv_file trihedral_pair_disjointness.csv \
17190 ▷  ▷  -box_width 20 -bit_depth 8 -end
17191 ▷  $(OPEN) trihedral_pair_disjointness_draw.bmp
17192
17193
17194
17195
17196 ##############################################################################
17197 # Section 12.3: Graph Theory: Classification
17198
17199
17200 SECTION_GRAPH_THEORY_CLASSIFICATION:
17201
17202
17203 test_12_3:
17204 ▷  make graph_classify_5
17205 ▷  make tournament_classify_4
17206 ▷  make graph_classify_8_r3
17207 ▷  make Symmetric_4_inversion_graph_recognize
17208 ▷  make Symmetric_5_inversion_graph_recognize
17209
17210
17211
17212 graph_classify_5:
17213 ▷  $(ORBITER) -v 2 \
```

```
17214 ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
17215 ▷  ▷  -define GC -graph_classification \
17216 ▷  ▷  ▷  -n 5 \
17217 ▷  ▷  ▷  -poset_classification_control \
17218 ▷  ▷  ▷  ▷  -problem_label graphs_v5 \
17219 ▷  ▷  ▷  ▷  -depth 10 \
17220 ▷  ▷  ▷  ▷  -draw_options -radius 250 \
17221 ▷  ▷  ▷  ▷  -embedded \
17222 ▷  ▷  ▷  ▷  -end \
17223 ▷  ▷  ▷  -end \
17224 ▷  ▷  -end \
17225 ▷  ▷  -with GC -do \
17226 ▷  ▷  -graph_classification_activity \
17227 ▷  ▷  ▷  -list_graphs_at_level 5 5 \
17228 ▷  ▷  -end \
17229 ▷  ▷  -with GC -do \
17230 ▷  ▷  -graph_classification_activity \
17231 ▷  ▷  ▷  -draw_options \
17232 ▷  ▷  ▷  ▷  -radius 300 -nodes_empty \
17233 ▷  ▷  ▷  ▷  -line_width 1.5  \
17234 ▷  ▷  ▷  ▷  -scale 0.1 \
17235 ▷  ▷  ▷  -end \
17236 ▷  ▷  ▷  -draw_graphs_at_level 5 \
17237 ▷  ▷  -end \
17238 ▷  ▷  -print_symbols
17239 ▷  pdflatex graphs_v5_level_5_reps.tex
17240 ▷  $(OPEN) graphs_v5_level_5_reps.pdf
17241 ▷  #pdflatex graphs_v5_poset.tex
17242 ▷  #$(OPEN) graphs_v5_poset.pdf
17243
17244
17245 tournament_classify_4:
17246 ▷  $(ORBITER) -v 2 \
17247 ▷  ▷  -define GC -graph_classification \
17248 ▷  ▷  ▷  -n 4 -tournament \
17249 ▷  ▷  ▷  -poset_classification_control \
17250 ▷  ▷  ▷  ▷  -problem_label tournament_4 \
17251 ▷  ▷  ▷  ▷  -depth 6 \
17252 ▷  ▷  ▷  ▷  -draw_options \
17253 ▷  ▷  ▷  ▷  ▷  -radius 250 -embedded \
17254 ▷  ▷  ▷  ▷  -end \
17255 ▷  ▷  ▷  -end \
17256 ▷  ▷  -end \
17257 ▷  ▷  -with GC -do \
17258 ▷  ▷  -graph_classification_activity \
17259 ▷  ▷  ▷  -draw_options \
17260 ▷  ▷  ▷  ▷  -radius 400 \
17261 ▷  ▷  ▷  ▷  -line_width 2 -scale 0.10 \
17262 ▷  ▷  ▷  -end \
17263 ▷  ▷  ▷  -draw_graphs_at_level 6 \
17264 ▷  ▷  -end \
17265 ▷  ▷  -print_symbols
17266 ▷  pdflatex tournament_4_level_6_reps.tex
17267 ▷  $(OPEN) tournament_4_level_6_reps.pdf
17268 ▷
17269
17270
17271
17272
```

```
17273 graph_classify_8_r3:
17274 ▷  $(ORBITER) -v 3 \
17275 ▷  ▷  -define GC -graph_classification \
17276 ▷  ▷  ▷  -n 8 -regular 3 \
17277 ▷  ▷  ▷  -poset_classification_control \
17278 ▷  ▷  ▷  ▷  -problem_label graphs_v8_r3 \
17279 ▷  ▷  ▷  ▷  -depth 12 \
17280 ▷  ▷  ▷  ▷  -draw_options -radius 250 \
17281 ▷  ▷  ▷  ▷  ▷  -line_width 0.2 -embedded \
17282 ▷  ▷  ▷  ▷  -end \
17283 ▷  ▷  ▷  -end \
17284 ▷  ▷  -end \
17285 ▷  ▷  -with GC -do \
17286 ▷  ▷  -graph_classification_activity \
17287 ▷  ▷  ▷  -draw_options \
17288 ▷  ▷  ▷  ▷  -radius 400 \
17289 ▷  ▷  ▷  ▷  -line_width 2 -scale 0.10 \
17290 ▷  ▷  ▷  -end \
17291 ▷  ▷  ▷  -draw_graphs_at_level 12 \
17292 ▷  ▷  -end \
17293 ▷  ▷  -print_symbols
17294 ▷  #pdflatex graphs_v8_r3_poset_lvl_12.tex
17295 ▷  #$(OPEN) graphs_v8_r3_poset_lvl_12.pdf
17296
17297
17298
17299
17300 Symmetric_4_inversion_graph_recognize:
17301 ▷  $(ORBITER) -v 10 \
17302 ▷  ▷  -define G -permutation_group -symmetric_group 4 -end \
17303 ▷  ▷  -with G -do \
17304 ▷  ▷  -group_theoretic_activity \
17305 ▷  ▷  ▷  -export_inversion_graphs "Symmetric4_inversion_graphs.csv" \
17306 ▷  ▷  -end
17307 ▷  $(ORBITER) -v 2 \
17308 ▷  ▷  -define GC -graph_classification \
17309 ▷  ▷  ▷  -n 4 \
17310 ▷  ▷  ▷  -poset_classification_control \
17311 ▷  ▷  ▷  ▷  -problem_label graphs_v4 -depth 6 \
17312 ▷  ▷  ▷  ▷  -draw_options -radius 250 -embedded -end \
17313 ▷  ▷  ▷  -end \
17314 ▷  ▷  -end \
17315 ▷  ▷  -with GC -do \
17316 ▷  ▷  -graph_classification_activity \
17317 ▷  ▷  ▷  -recognize_graphs_from_adjacency_matrix_csv \
17318 ▷  ▷  ▷  ▷  Symmetric4_inversion_graphs.csv \
17319 ▷  ▷  -end \
17320 ▷  ▷  -print_symbols
17321
17322
17323 Symmetric_5_inversion_graph_recognize:
17324 ▷  $(ORBITER) -v 10 \
17325 ▷  ▷  -define G -permutation_group -symmetric_group 5 -end \
17326 ▷  ▷  -with G -do \
17327 ▷  ▷  -group_theoretic_activity \
17328 ▷  ▷  ▷  -export_inversion_graphs \
17329 ▷  ▷  ▷  ▷  "Symmetric5_inversion_graphs.csv" \
17330 ▷  ▷  -end
17331 ▷  $(ORBITER) -v 2 \
```

```
17332 ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
17333 ▷  ▷  -define GC -graph_classification \
17334 ▷  ▷  ▷  -n 5 \
17335 ▷  ▷  ▷  -poset_classification_control \
17336 ▷  ▷  ▷  ▷  -problem_label graphs_v5 \
17337 ▷  ▷  ▷  ▷  -depth 10 \
17338 ▷  ▷  ▷  ▷  -draw_options \
17339 ▷  ▷  ▷  ▷  ▷  -radius 250 -embedded \
17340 ▷  ▷  ▷  ▷  -end \
17341 ▷  ▷  ▷  -end \
17342 ▷  ▷  -end \
17343 ▷  ▷  -with GC -do \
17344 ▷  ▷  -graph_classification_activity \
17345 ▷  ▷  ▷  -recognize_graphs_from_adjacency_matrix_csv \
17346 ▷  ▷  ▷  Symmetric5_inversion_graphs.csv \
17347 ▷  ▷  -end \
17348 ▷  ▷  -print_symbols
17349 ▷  #pdflatex graphs_v5_poset.tex
17350 ▷  #$(OPEN) graphs_v5_poset.pdf
17351
17352
17353
17354
17355 ##############################################################################
17356 # Chapter 13 - Design Theory
17357 ##############################################################################
17358
17359
17360
17361 test_13:
17362 ▷  make test_13_1 # slow
17363 ▷  make test_13_2
17364 ▷  make test_13_3
17365 ▷  make test_13_4
17366 ▷  make test_13_5
17367
17368
17369
17370
17371 ##############################################################################
17372 # Section 13.1: Design Theory
17373
17374 SECTION_DESIGN_THEORY:
17375
17376
17377 test_11_5:
17378 ▷  make design_PG_2_3
17379 ▷  make design_PG_2_4
17380 ▷  make wreath_product_designs_n4_k2_inc.txt
17381 ▷  make wreath_product_designs_n8_k6_inc.txt
17382 ▷  make design_20
17383 ▷  make design_600
17384 ▷  make LSQ_5A
17385 ▷  make LSQ_5B
17386 ▷  make LSQ_5_c
17387
17388
17389
17390 design_PG_2_3:
```

```
17391 ▷  $(ORBITER) -v 8 \
17392 ▷  ▷  -define F -finite_field -q 3 -end \
17393 ▷  ▷  -define D -design -field F -family PG_2_q -end \
17394 ▷  ▷  -with D -do \
17395 ▷  ▷  ▷  -design_activity \
17396 ▷  ▷  ▷  ▷  -export_inc \
17397 ▷  ▷  -end \
17398 ▷  ▷  -with D -do \
17399 ▷  ▷  ▷  -design_activity \
17400 ▷  ▷  ▷  ▷  -tactical_decomposition \
17401 ▷  ▷  -end
17402
17403 # writes PG_2_q3_inc.txt
17404
17405 design_PG_2_4:
17406 ▷  $(ORBITER) -v 8 \
17407 ▷  ▷  -define F -finite_field -q 4 -end \
17408 ▷  ▷  -define D -design -field F -family PG_2_q -end \
17409 ▷  ▷  -with D -do \
17410 ▷  ▷  ▷  -design_activity \
17411 ▷  ▷  ▷  ▷  -export_inc \
17412 ▷  ▷  -end
17413
17414
17415
17416 wreath_product_designs_n4_k2_inc.txt:
17417 ▷  $(ORBITER) -v 8 \
17418 ▷  ▷  -define D -design -wreath_product_designs 4 2 -end \
17419 ▷  ▷  -with D -do \
17420 ▷  ▷  ▷  -design_activity \
17421 ▷  ▷  ▷  ▷  -export_inc \
17422 ▷  ▷  -end
17423
17424
17425
17426 wreath_product_designs_n8_k6_inc.txt:
17427 ▷  $(ORBITER) -v 8 \
17428 ▷  ▷  -define D -design -wreath_product_designs 8 6 -end \
17429 ▷  ▷  -with D -do \
17430 ▷  ▷  ▷  -design_activity \
17431 ▷  ▷  ▷  ▷  -export_inc \
17432 ▷  ▷  -end
17433
17434
17435 # wreath_product_designs_n8_k6_inc.txt
17436 # The design has 16 points and 3920 blocks of size 6.
17437
17438
17439 design_20:
17440 ▷  $(ORBITER) -v 8 \
17441 ▷  ▷  -define D -design -list_of_blocks_coded 15 3 \
17442 ▷  ▷  "61, 67, 72, 76, 129, 147, 152, 156, 197, 204, 215, 224, 249, 261, 267, 276,
      296, 303, 309, 319" \
17443 ▷  ▷  -end \
17444 ▷  ▷  -with D -do \
17445 ▷  ▷  ▷  -design_activity \
17446 ▷  ▷  ▷  ▷  -export_inc \
17447 ▷  ▷  -end
17448
```

```
17449
17450 design_600:
17451 ▷  echo $(GEO_BLOCKS_600) | sed 's/ //' >geo_600_blocks.csv
17452 ▷  $(ORBITER) -v 8 \
17453 ▷  ▷  -define D -design -list_of_blocks_from_file \
17454 ▷  ▷  ▷  15 geo_600_blocks.csv \
17455 ▷  ▷  -end \
17456 ▷  ▷  -with D -do \
17457 ▷  ▷  ▷  -design_activity \
17458 ▷  ▷  ▷  ▷  -export_inc \
17459 ▷  ▷  -end \
17460 ▷  ▷  -with D -do \
17461 ▷  ▷  ▷  -design_activity \
17462 ▷  ▷  ▷  ▷  -intersection_matrix \
17463 ▷  ▷  -end
17464
17465 # see this for why we need to use sed:
17466 #https://stackoverflow.com/questions/2003536/bash-why-is-echo-adding-extra-space
17467
17468 LSQ_5A:
17469 ▷  $(ORBITER) -v 8 \
17470 ▷  ▷  -define LSQ5A -vector -format 5 -dense $(LSQ_5A_TABLE) -end \
17471 ▷  ▷  -define D -design -linear_space_from_latin_square \
17472 ▷  ▷  ▷  LSQ5A \
17473 ▷  ▷  -end \
17474 ▷  ▷  -with D -do \
17475 ▷  ▷  ▷  -design_activity \
17476 ▷  ▷  ▷  ▷  -export_inc \
17477 ▷  ▷  -end
17478
17479 LSQ_5B:
17480 ▷  $(ORBITER) -v 8 \
17481 ▷  ▷  -define LSQ5B -vector -format 5 -dense $(LSQ_5B_TABLE) -end \
17482 ▷  ▷  -define D -design -linear_space_from_latin_square \
17483 ▷  ▷  ▷  LSQ5B \
17484 ▷  ▷  -end \
17485 ▷  ▷  -with D -do \
17486 ▷  ▷  ▷  -design_activity \
17487 ▷  ▷  ▷  ▷  -export_inc \
17488 ▷  ▷  -end
17489
17490
17491 #latin_square_order5_LSQ5B_inc.txt
17492
17493
17494 LSQ_5_c:
17495 ▷  $(ORBITER) -v 3 \
17496 ▷  ▷  -define C -combinatorial_object \
17497 ▷  ▷  ▷  -label LSQ_5 LSQ\_5 \
17498 ▷  ▷  ▷  -file_of_incidence_geometries latin_square_order5_LSQ5A_inc.txt 15 28 90 \
17499 ▷  ▷  ▷  -file_of_incidence_geometries latin_square_order5_LSQ5B_inc.txt 15 28 90 \
17500 ▷  ▷  -end \
17501 ▷  ▷  -with C -do \
17502 ▷  ▷  -combinatorial_object_activity \
17503 ▷  ▷  ▷  -canonical_form \
17504 ▷  ▷  ▷  ▷  -save_ago \
17505 ▷  ▷  ▷  ▷  -max_TDO_depth 4 \
17506 ▷  ▷  ▷  -end \
17507 ▷  ▷  -end \
```

```
17508 ▷  ▷  -with C -do \
17509 ▷  ▷  -combinatorial_object_activity \
17510 ▷  ▷  ▷  -report \
17511 ▷  ▷  ▷  ▷  -export_flag_orbits \
17512 ▷  ▷  ▷  ▷  -show_TDO \
17513 ▷  ▷  ▷  ▷  -show_TDA \
17514 ▷  ▷  ▷  ▷  -dont_show_incidence_matrices \
17515 ▷  ▷  ▷  -end \
17516 ▷  ▷  -end
17517 ▷  pdflatex LSQ_5_classification.tex
17518 ▷  $(OPEN) LSQ_5_classification.pdf
17519
17520
17521
17522
17523
17524 ###############################################################################
17525 # Section 13.2: Assuming Symmetry
17526
17527 SECTION_ASSUMING_SYMMETRY:
17528
17529 test_11_6:
17530
17531 ▷  make KM_cyclic_7
17532 ▷  make KM_PGGL_2_32
17533 ▷  #make KM_PGGL_2_32_solve
17534 ▷  #make KM_PSL_3_5
17535 ▷  make C5_KM_2_3
17536 ▷  make C5_KM_1_3
17537 ▷  make C5_concatenate
17538 ▷  make C5_solve
17539 ▷  make C5_design_classify
17540 ▷
17541
17542
17543
17544
17545
17546
17547 KM_cyclic_7:
17548 ▷  $(ORBITER) -v 3 \
17549 ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
17550 ▷  ▷  -define gens -vector -dense "1,2,3,4,5,6,0" -end \
17551 ▷  ▷  -define G -permutation_group -symmetric_group 7 \
17552 ▷  ▷  ▷  -subgroup_by_generators "C7" 7 1 gens \
17553 ▷  ▷  -end \
17554 ▷  ▷  -define Control -poset_classification_control \
17555 ▷  ▷  ▷  -problem_label C7 -W -depth 3 \
17556 ▷  ▷  ▷  -draw_options -embedded -sideways -radius 50 \
17557 ▷  ▷  ▷  ▷  -scale 0.5 -line_width 0.3 -end \
17558 ▷  ▷  -end \
17559 ▷  ▷  -define Orb -orbits -group G \
17560 ▷  ▷  ▷  -on_subsets 3 Control \
17561 ▷  ▷  -end \
17562 ▷  ▷  -with Orb -do -orbits_activity \
17563 ▷  ▷  ▷  -Kramer_Mesner_matrix 2 3 \
17564 ▷  ▷  -end \
17565 ▷  ▷  -with Orb -do -orbits_activity \
17566 ▷  ▷  ▷  -report \
```

```
17567 ▷  ▷  ▷  -report_options \
17568 ▷  ▷  ▷  ▷  -draw_poset -type_ordinary \
17569 ▷  ▷  ▷  -end \
17570 ▷  ▷  -end
17571 ▷  $(ORBITER) -v 4 \
17572 ▷  ▷  -define A -vector -file C7_KM_2_3.csv -end \
17573 ▷  ▷  -define D -diophant \
17574 ▷  ▷  -label "C7_KM_2_3_system" \
17575 ▷  ▷  -coefficient_matrix A \
17576 ▷  ▷  -RHS_constant "EQ=1" \
17577 ▷  ▷  -x_min_global 0 -x_max_global 1 \
17578 ▷  ▷  -end \
17579 ▷  ▷  -with D -do \
17580 ▷  ▷  ▷  -diophant_activity -solve_mckay \
17581 ▷  ▷  -end
17582
17583
17584 # to create simple 7-designs on 33 points with block size 8 and lambda = 10 invar
      iant under PGGL(2,32):
17585
17586 KM_PGGL_2_32:
17587 ▷  $(ORBITER) -v 3 \
17588 ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
17589 ▷  ▷  -define Control -poset_classification_control \
17590 ▷  ▷  ▷  -problem_label KM_PGGL_2_32 \
17591 ▷  ▷  ▷  -W -depth 8 \
17592 ▷  ▷  ▷  -draw_options -embedded -sideways -radius 50 \
17593 ▷  ▷  ▷  ▷  -scale 0.5 -line_width 0.3 -end \
17594 ▷  ▷  -end \
17595 ▷  ▷  -define G -linear_group -PGGL 2 32 -end \
17596 ▷  ▷  -define Orb -orbits -group G \
17597 ▷  ▷  ▷  -on_subsets 8 Control \
17598 ▷  ▷  -end \
17599 ▷  ▷  -with Orb -do -orbits_activity \
17600 ▷  ▷  ▷  -Kramer_Mesner_matrix 7 8 \
17601 ▷  ▷  -end \
17602 ▷  ▷  -with Orb -do -orbits_activity \
17603 ▷  ▷  ▷  -report \
17604 ▷  ▷  ▷  -report_options \
17605 ▷  ▷  ▷  ▷  -draw_poset -type_ordinary \
17606 ▷  ▷  ▷  -end \
17607 ▷  ▷  -end
17608 ▷  $(ORBITER) -v 2 -draw_matrix \
17609 ▷  ▷  -input_csv_file KM_PGGL_2_32_KM_7_8.csv \
17610 ▷  ▷  -box_width 20 -bit_depth 24 \
17611 ▷  ▷  -partition 3 32 97 -end
17612 ▷  pdflatex KM_PGGL_2_32_poset_lvl_8.tex
17613 ▷  $(OPEN) KM_PGGL_2_32_poset_lvl_8.pdf
17614 ▷  $(OPEN) KM_PGGL_2_32_KM_7_8_draw.bmp
17615
17616 KM_PGGL_2_32_solve:
17617 ▷  $(ORBITER) -v 4 \
17618 ▷  ▷  -define A -vector -file KM_PGGL_2_32_KM_7_8.csv -end \
17619 ▷  ▷  -define D -diophant \
17620 ▷  ▷  -label "KM_PGGL_2_32_KM_7_8_system" \
17621 ▷  ▷  -coefficient_matrix A \
17622 ▷  ▷  -RHS_constant "EQ=10" \
17623 ▷  ▷  -x_min_global 0 -x_max_global 1 \
17624 ▷  ▷  -end \
```

```
17625 ▷  ▷  -with D -do \
17626 ▷  ▷  ▷  -diophant_activity -solve_mckay \
17627 ▷  ▷  -end
17628
17629
17630
17631
17632
17633 KM_PSL_3_5:
17634 ▷  $(ORBITER) -v 3 \
17635 ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
17636 ▷  ▷  -define Control -poset_classification_control \
17637 ▷  ▷  ▷  -problem_label KM_PSL_3_5 \
17638 ▷  ▷  ▷  -W -depth 10 \
17639 ▷  ▷  ▷  -draw_options -embedded -sideways \
17640 ▷  ▷  ▷  ▷  -radius 50 -scale 0.5 -line_width 0.3 -end \
17641 ▷  ▷  -end \
17642 ▷  ▷  -define G -linear_group -PSL 3 5 -end \
17643 ▷  ▷  -define Orb -orbits -group G \
17644 ▷  ▷  ▷  -on_subsets 10 Control \
17645 ▷  ▷  -end \
17646 ▷  ▷  -with Orb -do -orbits_activity \
17647 ▷  ▷  ▷  -Kramer_Mesner_matrix 8 10 \
17648 ▷  ▷  -end \
17649 ▷  ▷  -with Orb -do -orbits_activity \
17650 ▷  ▷  ▷  -report \
17651 ▷  ▷  ▷  -report_options \
17652 ▷  ▷  ▷  ▷  -draw_poset -type_ordinary \
17653 ▷  ▷  ▷  -end \
17654 ▷  ▷  -end
17655 ▷  $(ORBITER) -v 2 -draw_matrix \
17656 ▷  ▷  -input_csv_file KM_PSL_3_5_KM_8_10.csv \
17657 ▷  ▷  -box_width 10 -bit_depth 8 -partition 3 42 174 -end
17658 ▷  $(ORBITER) -v 4 \
17659 ▷  ▷  -define A -vector -file KM_PSL_3_5_KM_8_10.csv -end \
17660 ▷  ▷  -define D -diophant \
17661 ▷  ▷  ▷  -label "KM_PSL_3_5_KM_8_10_system" \
17662 ▷  ▷  ▷  -coefficient_matrix A \
17663 ▷  ▷  ▷  -RHS_constant "EQ=93" \
17664 ▷  ▷  ▷  -x_min_global 0 -x_max_global 1 \
17665 ▷  ▷  -end \
17666 ▷  ▷  -with D -do \
17667 ▷  ▷  ▷  -diophant_activity -solve_mckay \
17668 ▷  ▷  -end
17669
17670
17671
17672
17673
17674 C5_KM_2_3:
17675 ▷  $(ORBITER) -v 3 \
17676 ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
17677 ▷  ▷  -define gens -vector -dense $(GEN_C5) -end \
17678 ▷  ▷  -define G -permutation_group -symmetric_group 15 \
17679 ▷  ▷  ▷  -subgroup_by_generators "C5" 5 1 gens \
17680 ▷  ▷  -end \
17681 ▷  ▷  -define Control -poset_classification_control \
17682 ▷  ▷  ▷  -problem_label C5 -W -depth 3 \
17683 ▷  ▷  ▷  -draw_options -embedded -sideways -radius 50 \
```

```
17684 ▷  ▷  ▷  ▷  -scale 0.5 -line_width 0.3 -end \
17685 ▷  ▷  -end \
17686 ▷  ▷  -define Orb -orbits -group G \
17687 ▷  ▷  ▷  -on_subsets 3 Control \
17688 ▷  ▷  -end \
17689 ▷  ▷  -with Orb -do -orbits_activity \
17690 ▷  ▷  ▷  -Kramer_Mesner_matrix 2 3 \
17691 ▷  ▷  -end
17692
17693 #C5_KM_2_3.csv
17694
17695
17696 C5_KM_1_3:
17697 ▷  $(ORBITER) -v 3 \
17698 ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
17699 ▷  ▷  -define gens -vector -dense $(GEN_C5) -end \
17700 ▷  ▷  -define G -permutation_group -symmetric_group 15 \
17701 ▷  ▷  ▷  -subgroup_by_generators "C5" 5 1 gens \
17702 ▷  ▷  -end \
17703 ▷  ▷  -define Control -poset_classification_control \
17704 ▷  ▷  ▷  -problem_label C5 -W -depth 3 \
17705 ▷  ▷  ▷  -draw_options -embedded -sideways -radius 50 \
17706 ▷  ▷  ▷  ▷  -scale 0.5 -line_width 0.3 -end \
17707 ▷  ▷  -end \
17708 ▷  ▷  -define Orb -orbits -group G \
17709 ▷  ▷  ▷  -on_subsets 3 Control \
17710 ▷  ▷  -end \
17711 ▷  ▷  -with Orb -do -orbits_activity \
17712 ▷  ▷  ▷  -Kramer_Mesner_matrix 1 3 \
17713 ▷  ▷  -end \
17714 ▷  ▷  -with Orb -do -orbits_activity \
17715 ▷  ▷  ▷  -export_something set_orbits 3 \
17716 ▷  ▷  -end \
17717
17718
17719 # 91 orbits at level 3
17720
17721 #C5_KM_1_3.csv
17722 #C5_set_orbits_level_3.csv
17723
17724
17725 C5_concatenate:
17726 ▷  $(ORBITER) -v 3 \
17727 ▷  ▷  -csv_file_concatenate C5_KM_combined.csv 2 C5_KM_1_3.csv C5_KM_2_3.csv
17728
17729 #C5_KM_combined.csv
17730 # in the combined system, there are three rows
17731 # from the first system and 21 rows from the second system
17732
17733
17734 C5_solve:
17735 ▷  $(ORBITER) -v 4 \
17736 ▷  ▷  -define A -vector -file C5_KM_combined.csv -end \
17737 ▷  ▷  -define D -diophant \
17738 ▷  ▷  ▷  -label "C5_KM_combined_system" \
17739 ▷  ▷  ▷  -coefficient_matrix A \
17740 ▷  ▷  ▷  -x_min_global 0 \
17741 ▷  ▷  ▷  -x_max_global 1 \
17742 ▷  ▷  ▷  -RHS "mult=3,EQ=5" \
```

```
17743 ▷  ▷  ▷  -RHS "mult=21,LE=1" \
17744 ▷  ▷  -end \
17745 ▷  ▷  -with D -do \
17746 ▷  ▷  ▷  -diophant_activity -solve_mckay \
17747 ▷  ▷  -end
17748
17749 #diophant::solve_all_mckay found 21540 solutions in 61001 backtrack nodes
17750 #Found 21540 solutions with 61001 backtrack steps
17751
17752 #C5_KM_combined_system_sol.csv
17753
17754
17755 C5_design_classify:
17756 ▷  $(ORBITER) -v 2 \
17757 ▷  ▷  -define C -combinatorial_object \
17758 ▷  ▷  ▷  -label C5_config C5\_config \
17759 ▷  ▷  ▷  -file_of_designs_through_block_orbits \
17760 ▷  ▷  ▷  ▷  C5_KM_combined_system_sol.csv \
17761 ▷  ▷  ▷  ▷  C5_set_orbits_level_3.csv \
17762 ▷  ▷  ▷  ▷  15 3 \
17763 ▷  ▷  -end \
17764 ▷  ▷  -with C -do \
17765 ▷  ▷  -combinatorial_object_activity \
17766 ▷  ▷  ▷  -canonical_form \
17767 ▷  ▷  ▷  -end \
17768 ▷  ▷  -end \
17769 ▷  ▷  -with C -do \
17770 ▷  ▷  -combinatorial_object_activity \
17771 ▷  ▷  ▷  -report \
17772 ▷  ▷  ▷  ▷  -show_incidence_matrices \
17773 ▷  ▷  ▷  -end \
17774 ▷  ▷  -end
17775 ▷  pdflatex C5_config_classification.tex
17776 ▷  $(OPEN) C5_config_classification.pdf
17777
17778
17779 # finds 42 isomorphism types
17780
17781 # number 0 has group order 600
17782 # it is flag transitive and anti flag transitive
17783 # number 14 has group order 120
17784 # flag orbits: 15+60
17785 # anti flag orbits: 60+120
17786
17787
17788
17789
17790
17791 ################################################################################
17792 # Section 13.3: Design Theory - Large Sets
17793
17794 SECTION_DESIGN_THEORY_LARGE_SETS:
17795
17796 test_11_7:
17797 ▷  make AG_2_3.inc
17798 ▷  make LS_AG_2_3_design_table_create
17799 ▷  make AG_2_3_on_designs
17800 ▷  make AG_2_3_stab_orb_0
17801 ▷  make AG_2_3_stab_orb_0_Perm840_res192
```

```
17802 ▷  make LS_AG_2_3_disjoint_sets_graph_and_cliques
17803 ▷  make LS_AG_2_3_disjoint_sets_split
17804 ▷  make LS_AG_2_3_export_solutions
17805 ▷  make design_PG_2_3_table_create
17806 ▷  make design_PG_2_3_group_5
17807 ▷  make design_PG_2_3_group_5_sol_0
17808
17809
17810
17811 AG_2_3.inc:
17812 ▷  $(ORBITER) -v 2 \
17813 ▷  ▷  -define Geo -geometry_builder \
17814 ▷  ▷  ▷  -V 9 -B 12 \
17815 ▷  ▷  ▷  -TDO 4 -fuse 1 \
17816 ▷  ▷  ▷  -fname_GEO AG_2_3 \
17817 ▷  ▷  ▷  -test 3,4,5,6,7,8,9 \
17818 ▷  ▷  ▷  -output_to_inc_file \
17819 ▷  ▷  -end
17820
17821 #9 12 3
17822 #0 13 22 27 35 41 47 53 55 59 71 76
17823 #-1 1
17824 #432
17825
17826
17827
17828
17829
17830 LS_AG_2_3_design_table_create:
17831 ▷  $(ORBITER) -v 5 \
17832 ▷  ▷  -define B -vector -dense $(AG_2_3_BLOCKS) -end \
17833 ▷  ▷  -define D -design -list_of_blocks_coded 9 3 B -end \
17834 ▷  ▷  -define Sym9 -permutation_group -symmetric_group 9 -end \
17835 ▷  ▷  -define T -design_table D "AG_2_3" Sym9 -end
17836
17837 # creates AG_2_3_design_table.csv
17838 # and AG_2_3.makefile
17839
17840 #0,0,13,22,27,35,41,47,53,55,59,71,76
17841 # is the first design in AG_2_3_design_table.csv
17842
17843 #poset_orbit_node::init_root_node storing strong generators for a group of order
      362880
17844 # stabilizer order 432
17845 # 840 designs
17846
17847 #User time: 0.13 of a second, dt=13 tps = 100
17848
17849
17850 AG_2_3_on_designs:
17851 ▷  $(ORBITER) -v 2 \
17852 ▷  ▷  -define gens -vector -file AG_2_3_gens.csv -end \
17853 ▷  ▷  -define G -permutation_group \
17854 ▷  ▷  -bsgs AG_2_3 "AG_2_3" 840 362880 "0,1,2,3,4,5,6,7" 8 gens -end \
17855 ▷  ▷  -define Orb -orbits -group G \
17856 ▷  ▷  ▷  -on_points \
17857 ▷  ▷  -end \
17858 ▷  ▷  -with Orb -do -orbits_activity \
17859 ▷  ▷  ▷  -stabilizer 0 \
```

```
17860 ▷  ▷  -end
17861
17862 #Written file AG_2_3_stab_pt_0.makefile of size 239
17863
17864
17865
17866
17867
17868 # the stabilizer of the first design:
17869
17870
17871 #ToDo AG_2_3_stab_orb_0_gens.csv does not exist or is empty
17872
17873 AG_2_3_stab_orb_0:
17874 ▷  $(ORBITER) -v 2 \
17875 ▷  ▷  -define gens -vector -file AG_2_3_stab_pt_0_gens.csv -end \
17876 ▷  ▷  -define G -permutation_group \
17877 ▷  ▷  -bsgs AG_2_3_stab_orb_0 "AG_2_3_stab_pt_0" \
17878 ▷  ▷  ▷  840 432 "0,1,2,3,4,5,6,7,8" 5 gens \
17879 ▷  ▷  -end \
17880 ▷  ▷  -define Gr -modified_group -from G \
17881 ▷  ▷  ▷  -restricted_action $(LARGE_SET_AG_2_3_NEIGHBOR_SET) \\_neighbor\\_set \
17882 ▷  ▷  -end \
17883 ▷  ▷  -with Gr -do \
17884 ▷  ▷  -group_theoretic_activity \
17885 ▷  ▷  ▷  -export_orbiter \
17886 ▷  ▷  -end
17887 ▷  ▷
17888
17889 AG_2_3_stab_orb_0_Perm840_res192:
17890 ▷  $(ORBITER) -v 2 \
17891 ▷  ▷  -define gens -vector -file Perm840_res192_gens.csv -end \
17892 ▷  ▷  -define G -permutation_group \
17893 ▷  ▷  -bsgs Perm840_res192 "Perm840 {\rm res192}" \
17894 ▷  ▷  ▷  192 432 "0,1,2,3,4,5,6,7,8" 4 gens \
17895 ▷  ▷  -end \
17896 ▷  ▷  -with G -do \
17897 ▷  ▷  -group_theoretic_activity \
17898 ▷  ▷  ▷  -report \
17899 ▷  ▷  -end
17900 ▷  pdflatex Perm840_res192_report.tex
17901 ▷  $(OPEN) Perm840_res192_report.pdf
17902
17903
17904
17905 LS_AG_2_3_disjoint_sets_graph_and_cliques:
17906 ▷  $(ORBITER) -v 2 \
17907 ▷  ▷  -define Gamma -graph \
17908 ▷  ▷  ▷  -disjoint_sets_graph \
17909 ▷  ▷  ▷  AG_2_3_design_table.csv \
17910 ▷  ▷  -end \
17911 ▷  ▷  -with Gamma -do \
17912 ▷  ▷  -graph_theoretic_activity \
17913 ▷  ▷  ▷  -save \
17914 ▷  ▷  -end \
17915 ▷  ▷  -with Gamma -do \
17916 ▷  ▷  -graph_theoretic_activity \
17917 ▷  ▷  ▷  -find_cliques -target_size 7 -end \
17918 ▷  ▷  -end \
```

```
17919 ▷  ▷  -print_symbols
17920
17921
17922 #AG_2_3_design_table_disjoint_sets.colored_graph
17923 #User time: 0.66 of a second, dt=66 tps = 100
17924 #AG_2_3_design_table_disjoint_sets_sol.txt
17925 #AG_2_3_design_table_disjoint_sets_sol.csv
17926
17927 #15360 solutions
17928
17929 LS_AG_2_3_disjoint_sets_split:
17930 ▷  $(ORBITER) -v 4 \
17931 ▷  ▷  -define Gamma -graph -load \
17932 ▷  ▷  ▷  AG_2_3_design_table_disjoint_sets.colored_graph \
17933 ▷  ▷  -end \
17934 ▷  ▷  -with Gamma -do \
17935 ▷  ▷  -graph_theoretic_activity \
17936 ▷  ▷  ▷  -split_by_clique "0" "0" \
17937 ▷  ▷  -end
17938
17939
17940 #AG_2_3_design_table_disjoint_sets_0.graph
17941 #AG_2_3_design_table_disjoint_sets_0_subset.txt
17942
17943
17944
17945 LS_AG_2_3_export_solutions:
17946 ▷  $(ORBITER) -v 20 \
17947 ▷  ▷  -define B -vector -dense $(AG_2_3_BLOCKS) -end \
17948 ▷  ▷  -define D -design -list_of_blocks 9 3 B -end \
17949 ▷  ▷  -define Sym9 -permutation_group -symmetric_group 9 -end \
17950 ▷  ▷  -define T -design_table D "AG_2_3" Sym9 -end \
17951 ▷  ▷  -with D -do \
17952 ▷  ▷  -design_activity \
17953 ▷  ▷  ▷  -extract_solutions_by_index "AG_2_3" Sym9 \
17954 ▷  ▷  ▷  ▷  AG_2_3_design_table_disjoint_sets_sol.csv \
17955 ▷  ▷  ▷  ▷  solutions.csv \
17956 ▷  ▷  ▷  ▷  "" \
17957 ▷  ▷  -end
17958
17959
17960 #User time: 0.39 of a second, dt=39 tps = 100
17961 # solutions.csv
17962
17963
17964
17965
17966 design_PG_2_3_table_create:
17967 ▷  $(ORBITER) -v 2 \
17968 ▷  ▷  -define F -finite_field -q 3 -end \
17969 ▷  ▷  -define D -design -field F -family PG_2_q -end \
17970 ▷  ▷  -define Sym13 -permutation_group -symmetric_group 13 -end \
17971 ▷  ▷  -define T -design_table D "PG_2_13" Sym13 -end
17972
17973 # written file PG_2_13_design_table.csv
17974 # 1108800 designs
17975 #User time: 7:30
17976
17977 design_PG_2_3_group_5:
```

```
17978 ▷  $(ORBITER) -v 2 \
17979 ▷  ▷  -define F -finite_field -q 3 -end \
17980 ▷  ▷  -define D -design -field F -family PG_2_q -end \
17981 ▷  ▷  -define T -design_table D "PG_2_13" Sym13 -end \
17982 ▷  ▷  -define LSW -large_set_with_symmetry_assumption T \
17983 ▷  ▷  ▷  ▷  -H "5" $(GENERATORS_H5) \
17984 ▷  ▷  ▷  ▷  -N "1200" $(GENERATORS_N5) \
17985 ▷  ▷  ▷  ▷  -prefix "H5" \
17986 ▷  ▷  ▷  ▷  -selected_orbit_length 5 \
17987 ▷  ▷  ▷  -end \
17988 ▷  ▷  -with LSW -do \
17989 ▷  ▷  ▷  -large_set_with_symmetry_assumption_activity \
17990 ▷  ▷  ▷  ▷  -normalizer_on_orbits_of_a_given_length 5 \
17991 ▷  ▷  ▷  -end
17992
17993 #H5_N_orbit_reps.csv
17994 # 678 orbits
17995 #User time: 2:39
17996
17997
17998 design_PG_2_3_group_5_sol_0:
17999 ▷  $(ORBITER) -v 2 \
18000 ▷  ▷  -define F -finite_field -q 3 -end \
18001 ▷  ▷  -define D -design -field F -family PG_2_q -end \
18002 ▷  ▷  -define T -design_table D "PG_2_13" Sym13 -end \
18003 ▷  ▷  -define LSW -large_set_with_symmetry_assumption T \
18004 ▷  ▷  ▷  ▷  -H "5" $(GENERATORS_H5) \
18005 ▷  ▷  ▷  ▷  -N "1200" $(GENERATORS_N5) \
18006 ▷  ▷  ▷  ▷  -prefix "H5" \
18007 ▷  ▷  ▷  ▷  -selected_orbit_length 5 \
18008 ▷  ▷  ▷  -end \
18009 ▷  ▷  -with LSW -do \
18010 ▷  ▷  ▷  -large_set_with_symmetry_assumption_activity \
18011 ▷  ▷  ▷  ▷  -read_solution_file 5 case_0_sol.txt \
18012 ▷  ▷  ▷  -end
18013
18014
18015
18016
18017 ##############################################################################
18018 # Section 13.4: Design Theory - Delandtsheer Doyen
18019
18020 SECTION_DESIGN_THEORY_DELANDTHSHEER_DOYEN:
18021
18022 test_11_8:
18023 ▷  make DD_PP4
18024 ▷  make DD_PP4_span_design
18025 ▷  make DD_PP4_design_orbit
18026 ▷  #make PG_2_27_special
18027
18028
18029 DD_PP4:
18030 ▷  $(ORBITER) -v 6 \
18031 ▷  ▷  -define pair_search_control -poset_classification_control \
18032 ▷  ▷  ▷  -problem_label PP4_pairs -W \
18033 ▷  ▷  ▷  -draw_options -end \
18034 ▷  ▷  -end \
18035 ▷  ▷  -define search_control -poset_classification_control \
18036 ▷  ▷  ▷  -problem_label PP4_search -W \
```

```
18037 ▷  ▷  ▷  -draw_options -end \
18038 ▷  ▷  ▷  -depth 5 \
18039 ▷  ▷  -end \
18040 ▷  ▷  -Delandtsheer_Doyen \
18041 ▷  ▷  ▷  -pair_search_control pair_search_control \
18042 ▷  ▷  ▷  -search_control search_control \
18043 ▷  ▷  ▷  -search_partial_base_lines \
18044 ▷  ▷  ▷  $(PP4) $(PP4_GROUP1) $(PP4_MASK1) \
18045 ▷  ▷  ▷  -end \
18046
18047
18048 DD_PP4_span_design:
18049 ▷  $(ORBITER) -v 6 \
18050 ▷  ▷  -define D -design -list_of_base_blocks \
18051 ▷  ▷  ▷  group \
18052 ▷  ▷  ▷  PP4_no_mask_cyclic21_cases.csv
18053 ▷  ▷  ▷  Starter \
18054 ▷  ▷  -end
18055
18056
18057 DD_PP4_design_orbit:
18058 ▷  $(ORBITER) -v 4 \
18059 ▷  ▷  -define G1 -linear_group -AGL 1 3 \
18060 ▷  ▷  ▷  -end \
18061 ▷  ▷  -define G2 -linear_group -AGL 1 7 \
18062 ▷  ▷  ▷  -end \
18063 ▷  ▷  -define G -modified_group -direct_product "G1,G2" \
18064 ▷  ▷  ▷  "21" "1,1,1,1" \
18065 ▷  ▷  -end \
18066 ▷  ▷  -with G -do \
18067 ▷  ▷  -group_theoretic_activity \
18068 ▷  ▷  ▷  -report \
18069 ▷  ▷  -end \
18070 ▷  ▷  -define D -design -list_of_base_blocks \
18071 ▷  ▷  ▷  G \
18072 ▷  ▷  ▷  PP4_no_mask_cyclic21_cases.csv \
18073 ▷  ▷  ▷  Starter \
18074 ▷  ▷  -end
18075
18076
18077
18078
18079
18080
18081
18082 PG_2_27_special:
18083 ▷  $(ORBITER) -v 2 \
18084 ▷  ▷  -define F -finite_field -q 27 -override_polynomial 46 -end \
18085 ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
18086 ▷  ▷  -with P -do -projective_space_activity \
18087 ▷  ▷  ▷  -cheat_sheet \
18088 ▷  ▷  -end
18089 ▷  pdflatex PG_2_27.tex
18090 ▷  $(OPEN) PG_2_27.pdf
18091
18092
18093
18094
18095
```

```
18096
18097 ################################################################################
18098 # Chapter 14 - Finite Geometry
18099 ################################################################################
18100
18101
18102 test_14:
18103 ▷  make test_14_1
18104 ▷  make test_14_2
18105 ▷  make test_14_3
18106 ▷  make test_14_4
18107
18108
18109
18110
18111 ################################################################################
18112 # Section 14.1: Spreads
18113
18114 SECTION_SPREADS:
18115
18116 test_14_1:
18117 ▷  make create_spread_9a
18118 ▷  make create_spread_9b
18119 ▷  make create_spread_25_7
18120 ▷  make create_spread_Rao_Rao_27
18121 ▷  make desarguesian_spread_in_PG_3_2
18122 ▷  make desarguesian_spread_in_PG_5_2
18123 ▷  make desarguesian_spread_in_PG_3_4
18124 ▷  make desarguesian_spread_in_PG_3_5
18125 ▷  make classify_spreads_4
18126 ▷  make classify_spreads_16_4
18127 ▷  make classify_spreads_25_starter_lift_case_0
18128 ▷  make spreads_25_starter_0_cliques
18129 ▷  make classify_spreads_25_starter_lift_all_cases
18130 ▷  make spreads_25_starter_cliques
18131 ▷  make classify_spreads_25_isomorph
18132 ▷  make classify_spreads_27_3_starter
18133 ▷  make classify_spreads_27_starter_lift_all_cases
18134 ▷  make spreads_27_starter_cliques
18135 ▷  make classify_spreads_27_isomorph_and_recognize
18136 ▷  make create_spread_27_0
18137 ▷  make create_spread_27_1
18138 ▷  make create_spread_27_2
18139 ▷  make create_spread_27_3
18140 ▷  make create_spread_27_4
18141 ▷  make create_spread_27_5
18142 ▷  make create_spread_27_6
18143
18144
18145
18146 create_spread_9a:
18147 ▷  $(ORBITER) -v 3 \
18148 ▷  ▷  -define F -finite_field -q 3 -end \
18149 ▷  ▷  -define G -linear_group -PGL 4 F -end \
18150 ▷  ▷  -define S -spread -kernel_field F \
18151 ▷  ▷  ▷  -group G -k 2 -catalogue 0 \
18152 ▷  ▷  ▷  -end
18153
18154 # desarguesian spread, ago = 5760
```

```
18155
18156 create_spread_9b:
18157 ▷  $(ORBITER) -v 3 \
18158 ▷  ▷  -define F -finite_field -q 3 -end \
18159 ▷  ▷  -define G -linear_group -PGL 4 F -end \
18160 ▷  ▷  -define S -spread -kernel_field F \
18161 ▷  ▷  ▷  -group G -k 2 -catalogue 1 \
18162 ▷  ▷  ▷  -end
18163
18164
18165 # Hall spread, ago = 1920
18166
18167
18168 create_spread_25_7:
18169 ▷  $(ORBITER) -v 3 \
18170 ▷  ▷  -define F -finite_field -q 5 -end \
18171 ▷  ▷  -define G -linear_group -PGL 4 F -end \
18172 ▷  ▷  -define S -spread -kernel_field F \
18173 ▷  ▷  ▷  -group G -k 2 -catalogue 7 \
18174 ▷  ▷  ▷  -end
18175
18176
18177
18178 create_spread_Rao_Rao_27:
18179 ▷  $(ORBITER) -v 3 \
18180 ▷  ▷  -define F -finite_field -q 3 -end \
18181 ▷  ▷  -define SS -vector -dense $(SPREAD_SET_27_RAO_RAO) -end \
18182 ▷  ▷  -define G -linear_group -PGL 6 F -end \
18183 ▷  ▷  -define S -spread -kernel_field F \
18184 ▷  ▷  ▷  -group G -k 3 -spread_set "Rao_Rao" "Rao\_Rao" SS \
18185 ▷  ▷  ▷  -end
18186
18187
18188 desarguesian_spread_in_PG_3_2:
18189 ▷  $(ORBITER) -v 3 \
18190 ▷  ▷  -define FQ -finite_field -q 4 -end \
18191 ▷  ▷  -define Fq -finite_field -q 2 -end \
18192 ▷  ▷  -with FQ -and Fq -do -finite_field_activity \
18193 ▷  ▷  ▷  -cheat_sheet_desarguesian_spread 2 -end
18194 ▷  pdflatex Desarguesian_Spread_3_2.tex
18195 ▷  $(OPEN) Desarguesian_Spread_3_2.pdf
18196
18197 desarguesian_spread_in_PG_5_2:
18198 ▷  $(ORBITER) -v 3 \
18199 ▷  ▷  -define FQ -finite_field -q 8 -end \
18200 ▷  ▷  -define Fq -finite_field -q 2 -end \
18201 ▷  ▷  -with FQ -and Fq -do -finite_field_activity \
18202 ▷  ▷  ▷  -cheat_sheet_desarguesian_spread 2 -end
18203 ▷  pdflatex Desarguesian_Spread_5_2.tex
18204 ▷  $(OPEN) Desarguesian_Spread_5_2.pdf
18205
18206 desarguesian_spread_in_PG_3_4:
18207 ▷  $(ORBITER) -v 3 \
18208 ▷  ▷  -define FQ -finite_field -q 16 -end \
18209 ▷  ▷  -define Fq -finite_field -q 4 -end \
18210 ▷  ▷  -with FQ -and Fq -do -finite_field_activity \
18211 ▷  ▷  ▷  -cheat_sheet_desarguesian_spread 2 -end
18212 ▷  pdflatex Desarguesian_Spread_3_4.tex
18213 ▷  $(OPEN) Desarguesian_Spread_3_4.pdf
```

```
18214
18215 desarguesian_spread_in_PG_3_5:
18216 ▷  $(ORBITER) -v 3 \
18217 ▷  ▷  -define FQ -finite_field -q 25 -end \
18218 ▷  ▷  -define Fq -finite_field -q 5 -end \
18219 ▷  ▷  -with FQ -and Fq -do -finite_field_activity \
18220 ▷  ▷  ▷  -cheat_sheet_desarguesian_spread 2 -end
18221 ▷  pdflatex Desarguesian_Spread_3_5.tex
18222 ▷  $(OPEN) Desarguesian_Spread_3_5.pdf
18223
18224
18225 # ToDo -report
18226
18227 classify_spreads_4:
18228 ▷  $(ORBITER) -v 3 \
18229 ▷  ▷  -define Control -poset_classification_control \
18230 ▷  ▷  ▷  -draw_options \
18231 ▷  ▷  ▷  ▷  -embedded \
18232 ▷  ▷  ▷  -end \
18233 ▷  ▷  ▷  -W \
18234 ▷  ▷  ▷  -problem_label spreads_2_2 \
18235 ▷  ▷  -end \
18236 ▷  ▷  -define F -finite_field -q 2 -end \
18237 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
18238 ▷  ▷  -define C -spread_classifier \
18239 ▷  ▷  ▷  -projective_space P \
18240 ▷  ▷  ▷  -poset_classification_control Control \
18241 ▷  ▷  ▷  -k 2 \
18242 ▷  ▷  ▷  -starter_size 5 \
18243 ▷  ▷  ▷  -output_prefix "." \
18244 ▷  ▷  -end \
18245 ▷  ▷  -with C -do -spread_classify_activity \
18246 ▷  ▷  ▷  -compute_starter \
18247 ▷  ▷  -end
18248 ▷  #pdflatex spreads_2_2_poset_lvl_5.tex
18249 ▷  #$(OPEN) spreads_2_2_poset_lvl_5.pdf
18250
18251
18252 classify_spreads_16_4:
18253 ▷  $(ORBITER) -v 4 \
18254 ▷  ▷  -define Control -poset_classification_control \
18255 ▷  ▷  ▷  -draw_options \
18256 ▷  ▷  ▷  ▷  -radius 20 \
18257 ▷  ▷  ▷  ▷  -nodes_empty \
18258 ▷  ▷  ▷  ▷  -line_width 0.2 \
18259 ▷  ▷  ▷  ▷  -embedded \
18260 ▷  ▷  ▷  -end \
18261 ▷  ▷  ▷  -problem_label spreads_16_4 \
18262 ▷  ▷  -end \
18263 ▷  ▷  -define F -finite_field -q 4 -end \
18264 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
18265 ▷  ▷  -define C -spread_classifier \
18266 ▷  ▷  ▷  -projective_space P \
18267 ▷  ▷  ▷  -poset_classification_control Control \
18268 ▷  ▷  ▷  -k 2 \
18269 ▷  ▷  ▷  -starter_size 17 \
18270 ▷  ▷  ▷  -output_prefix "." \
18271 ▷  ▷  -end \
18272 ▷  ▷  -with C -do -spread_classify_activity \
```

```
18273 ▷  ▷  ▷  -compute_starter \
18274 ▷  ▷  -end
18275 ▷  #pdflatex spreads_16_4_poset_lvl_17.tex
18276 ▷  #$(OPEN) spreads_16_4_poset_lvl_17.pdf
18277
18278
18279
18280
18281
18282 classify_spreads_25_starter_lift_case_0:
18283 ▷  $(ORBITER) -v 3 \
18284 ▷  ▷  -define Control -poset_classification_control \
18285 ▷  ▷  ▷  -draw_options \
18286 ▷  ▷  ▷  ▷  -radius 20 \
18287 ▷  ▷  ▷  ▷  -nodes_empty \
18288 ▷  ▷  ▷  ▷  -line_width 0.2 \
18289 ▷  ▷  ▷  ▷  -embedded \
18290 ▷  ▷  ▷  -end \
18291 ▷  ▷  ▷  -W \
18292 ▷  ▷  ▷  -problem_label spreads_25 \
18293 ▷  ▷  -end \
18294 ▷  ▷  -define F -finite_field -q 5 -end \
18295 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
18296 ▷  ▷  -define C -spread_classifier \
18297 ▷  ▷  ▷  -projective_space P \
18298 ▷  ▷  ▷  -k 2 \
18299 ▷  ▷  ▷  -starter_size 5 \
18300 ▷  ▷  ▷  -recoordinatize \
18301 ▷  ▷  ▷  -poset_classification_control Control \
18302 ▷  ▷  ▷  -output_prefix "" \
18303 ▷  ▷  -end \
18304 ▷  ▷  -with C -do -spread_classify_activity \
18305 ▷  ▷  ▷  -compute_starter \
18306 ▷  ▷  -end \
18307 ▷  ▷  -with C -do -spread_classify_activity \
18308 ▷  ▷  ▷  -prepare_lifting_single_case 0 \
18309 ▷  ▷  -end
18310
18311
18312 #save_colored_graph fname=spreads_25_graph_0.bin
18313 #save_colored_graph nb_vertices=225
18314 #save_colored_graph nb_colors=21
18315 #save_colored_graph nb_colors_per_vertex=1
18316 #save_colored_graph done
18317 #colored_graph::save done
18318 #Written file spreads_25_graph_0.bin of size 5914
18319
18320
18321 spreads_25_starter_0_cliques:
18322 ▷  $(ORBITER) -v 2 \
18323 ▷  ▷  ▷  -define G -graph -load spreads_25_graph_0.bin -end \
18324 ▷  ▷  ▷  -with G -do \
18325 ▷  ▷  ▷  -graph_theoretic_activity \
18326 ▷  ▷  ▷  ▷  -find_cliques -rainbow -target_size 21 -end \
18327 ▷  ▷  ▷  -end \
18328
18329 #graph_theoretic_activity::perform_activity Gr->label=spreads_25_graph_0 nb_sol =
       7680
18330
```

```
18331
18332 classify_spreads_25_starter_lift_all_cases:
18333 ▷  $(ORBITER) -v 3 \
18334 ▷  ▷  -define Control -poset_classification_control \
18335 ▷  ▷  ▷  -draw_options \
18336 ▷  ▷  ▷  ▷  -radius 20 \
18337 ▷  ▷  ▷  ▷  -nodes_empty \
18338 ▷  ▷  ▷  ▷  -line_width 0.2 \
18339 ▷  ▷  ▷  ▷  -embedded \
18340 ▷  ▷  ▷  -end \
18341 ▷  ▷  ▷  -W \
18342 ▷  ▷  ▷  -problem_label spreads_25 \
18343 ▷  ▷  -end \
18344 ▷  ▷  -define F -finite_field -q 5 -end \
18345 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
18346 ▷  ▷  -define C -spread_classifier \
18347 ▷  ▷  ▷  -projective_space P \
18348 ▷  ▷  ▷  -k 2 \
18349 ▷  ▷  ▷  -starter_size 5 \
18350 ▷  ▷  ▷  -recoordinatize \
18351 ▷  ▷  ▷  -poset_classification_control Control \
18352 ▷  ▷  ▷  -output_prefix "" \
18353 ▷  ▷  -end \
18354 ▷  ▷  -with C -do -spread_classify_activity \
18355 ▷  ▷  ▷  -compute_starter \
18356 ▷  ▷  -end \
18357 ▷  ▷  -with C -do -spread_classify_activity \
18358 ▷  ▷  ▷  -prepare_lifting_all_cases \
18359 ▷  ▷  -end
18360
18361 spreads_25_starter_cliques:
18362 ▷  $(ORBITER) -v 2 \
18363 ▷  ▷  -loop L 0 29 1 \
18364 ▷  ▷  ▷  -define G -graph -load spreads_25_graph_%L.bin -end \
18365 ▷  ▷  ▷  -with G -do \
18366 ▷  ▷  ▷  -graph_theoretic_activity \
18367 ▷  ▷  ▷  ▷  -find_cliques -rainbow -target_size 21 -end \
18368 ▷  ▷  ▷  -end \
18369 ▷  ▷  -end_loop L
18370
18371
18372 classify_spreads_25_isomorph:
18373 ▷  $(ORBITER) -v 4 \
18374 ▷  ▷  -define Control -poset_classification_control \
18375 ▷  ▷  ▷  -draw_options \
18376 ▷  ▷  ▷  ▷  -radius 20 \
18377 ▷  ▷  ▷  ▷  -nodes_empty \
18378 ▷  ▷  ▷  ▷  -line_width 0.2 \
18379 ▷  ▷  ▷  ▷  -embedded \
18380 ▷  ▷  ▷  -end \
18381 ▷  ▷  ▷  -W \
18382 ▷  ▷  ▷  -problem_label spreads_25 \
18383 ▷  ▷  -end \
18384 ▷  ▷  -define F -finite_field -q 5 -end \
18385 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
18386 ▷  ▷  -define C -spread_classifier \
18387 ▷  ▷  ▷  -projective_space P \
18388 ▷  ▷  ▷  -k 2 \
18389 ▷  ▷  ▷  -starter_size 5 \
```

```
18390 ▷  ▷  ▷  -recoordinatize \
18391 ▷  ▷  ▷  -poset_classification_control Control \
18392 ▷  ▷  ▷  -output_prefix "" \
18393 ▷  ▷  -end \
18394 ▷  ▷  -with C -do -spread_classify_activity \
18395 ▷  ▷  ▷  -compute_starter \
18396 ▷  ▷  -end \
18397 ▷  ▷  -with C -do -spread_classify_activity \
18398 ▷  ▷  ▷  -isomorph \
18399 ▷  ▷  ▷  ▷  -prefix_iso "./spreads_25" \
18400 ▷  ▷  ▷  ▷  -use_database_for_starter \
18401 ▷  ▷  ▷  ▷  -build_db \
18402 ▷  ▷  ▷  ▷  -solution_prefix "" \
18403 ▷  ▷  ▷  ▷  -base_fname "" \
18404 ▷  ▷  ▷  -end \
18405 ▷  ▷  -end \
18406 ▷  ▷  -with C -do -spread_classify_activity \
18407 ▷  ▷  ▷  -isomorph \
18408 ▷  ▷  ▷  ▷  -prefix_iso "./spreads_25" \
18409 ▷  ▷  ▷  ▷  -use_database_for_starter \
18410 ▷  ▷  ▷  ▷  -read_solutions \
18411 ▷  ▷  ▷  ▷  -solution_prefix "" \
18412 ▷  ▷  ▷  ▷  -base_fname "spreads_25_graph" \
18413 ▷  ▷  ▷  -end \
18414 ▷  ▷  -end \
18415 ▷  ▷  -with C -do -spread_classify_activity \
18416 ▷  ▷  ▷  -isomorph \
18417 ▷  ▷  ▷  ▷  -prefix_iso "./spreads_25" \
18418 ▷  ▷  ▷  ▷  -use_database_for_starter \
18419 ▷  ▷  ▷  ▷  -compute_orbits \
18420 ▷  ▷  ▷  ▷  -solution_prefix "" \
18421 ▷  ▷  ▷  ▷  -base_fname "spreads_25_graph" \
18422 ▷  ▷  ▷  -end \
18423 ▷  ▷  -end \
18424 ▷  ▷  -with C -do -spread_classify_activity \
18425 ▷  ▷  ▷  -isomorph \
18426 ▷  ▷  ▷  ▷  -prefix_iso "./spreads_25" \
18427 ▷  ▷  ▷  ▷  -use_database_for_starter \
18428 ▷  ▷  ▷  ▷  -isomorph_testing \
18429 ▷  ▷  ▷  ▷  -solution_prefix "" \
18430 ▷  ▷  ▷  ▷  -base_fname "spreads_25_graph" \
18431 ▷  ▷  ▷  -end \
18432 ▷  ▷  -end \
18433 ▷  ▷  -with C -do -spread_classify_activity \
18434 ▷  ▷  ▷  -isomorph \
18435 ▷  ▷  ▷  ▷  -prefix_iso "./spreads_25" \
18436 ▷  ▷  ▷  ▷  -use_database_for_starter \
18437 ▷  ▷  ▷  ▷  -isomorph_report \
18438 ▷  ▷  ▷  ▷  -solution_prefix "" \
18439 ▷  ▷  ▷  ▷  -base_fname "spreads_25_graph" \
18440 ▷  ▷  ▷  -end \
18441 ▷  ▷  -end
18442 ▷  pdflatex spreads_25_isomorphism_types.tex
18443 ▷  $(OPEN) spreads_25_isomorphism_types.pdf
18444
18445 #We found 21 isomorphism types
18446 #1:33
18447
18448
```

```
18449
18450 classify_spreads_27_3_starter:
18451 ▷  $(ORBITER) -v 10 \
18452 ▷  ▷  -define Control -poset_classification_control \
18453 ▷  ▷  ▷  -draw_options \
18454 ▷  ▷  ▷  ▷  -radius 20 \
18455 ▷  ▷  ▷  ▷  -nodes_empty \
18456 ▷  ▷  ▷  ▷  -line_width 0.2 \
18457 ▷  ▷  ▷  ▷  -embedded \
18458 ▷  ▷  ▷  -end \
18459 ▷  ▷  ▷  -W \
18460 ▷  ▷  ▷  -problem_label spreads_27_3 \
18461 ▷  ▷  -end \
18462 ▷  ▷  -define F -finite_field -q 3 -end \
18463 ▷  ▷  -define P -projective_space -n 5 -field F -v 0 -end \
18464 ▷  ▷  -define C -spread_classifier \
18465 ▷  ▷  ▷  -projective_space P \
18466 ▷  ▷  ▷  -poset_classification_control Control \
18467 ▷  ▷  ▷  -k 3 \
18468 ▷  ▷  ▷  -starter_size 5 \
18469 ▷  ▷  ▷  -recoordinatize \
18470 ▷  ▷  ▷  -output_prefix "." \
18471 ▷  ▷  -end \
18472 ▷  ▷  -with C -do -spread_classify_activity \
18473 ▷  ▷  ▷  -compute_starter \
18474 ▷  ▷  -end \
18475 ▷  ▷  -with C -do -spread_classify_activity \
18476 ▷  ▷  ▷  -prepare_lifting_single_case 0 \
18477 ▷  ▷  -end
18478 ▷  #pdflatex spreads_27_3_poset_detailed_lvl_5.tex
18479 ▷  #$(OPEN) spreads_27_3_poset_detailed_lvl_5.pdf
18480
18481 # 50 orbits at level 5:
18482 #5 : 50 orbits
18483 #total: 60
18484 #(39^2, 26^2, 10, 6^2, 5, 3^9, 2^9, 1^{24}) average is 4 + 26 / 50
18485
18486 # time 4:31
18487
18488
18489
18490 classify_spreads_27_starter_lift_all_cases:
18491 ▷  $(ORBITER) -v 3 \
18492 ▷  ▷  -define Control -poset_classification_control \
18493 ▷  ▷  ▷  -draw_options \
18494 ▷  ▷  ▷  ▷  -radius 20 \
18495 ▷  ▷  ▷  ▷  -nodes_empty \
18496 ▷  ▷  ▷  ▷  -line_width 0.2 \
18497 ▷  ▷  ▷  ▷  -embedded \
18498 ▷  ▷  ▷  -end \
18499 ▷  ▷  ▷  -W \
18500 ▷  ▷  ▷  -problem_label spreads_27_3 \
18501 ▷  ▷  -end \
18502 ▷  ▷  -define F -finite_field -q 3 -end \
18503 ▷  ▷  -define P -projective_space -n 5 -field F -v 0 -end \
18504 ▷  ▷  -define C -spread_classifier \
18505 ▷  ▷  ▷  -projective_space P \
18506 ▷  ▷  ▷  -k 3 \
18507 ▷  ▷  ▷  -starter_size 5 \
```

```
18508 ▷  ▷  ▷  -recoordinatize \
18509 ▷  ▷  ▷  -poset_classification_control Control \
18510 ▷  ▷  ▷  -output_prefix "" \
18511 ▷  ▷  -end \
18512 ▷  ▷  -with C -do -spread_classify_activity \
18513 ▷  ▷  ▷  -compute_starter \
18514 ▷  ▷  -end \
18515 ▷  ▷  -with C -do -spread_classify_activity \
18516 ▷  ▷  ▷  -prepare_lifting_all_cases \
18517 ▷  ▷  -end
18518
18519
18520 # 50 graphs
18521
18522
18523
18524 spreads_27_starter_cliques:
18525 ▷  $(ORBITER) -v 2 \
18526 ▷  ▷  -loop L 0 50 1 \
18527 ▷  ▷  ▷  -define G -graph -load spreads_27_3_graph_%L.bin -end \
18528 ▷  ▷  ▷  -with G -do \
18529 ▷  ▷  ▷  -graph_theoretic_activity \
18530 ▷  ▷  ▷  ▷  -find_cliques -rainbow -target_size 23 -end \
18531 ▷  ▷  ▷  -end \
18532 ▷  ▷  -end_loop L
18533
18534 #4:41
18535
18536
18537
18538 classify_spreads_27_isomorph_and_recognize:
18539 ▷  $(ORBITER) -v 3 \
18540 ▷  ▷  -define Control -poset_classification_control \
18541 ▷  ▷  ▷  -draw_options \
18542 ▷  ▷  ▷  ▷  -radius 20 \
18543 ▷  ▷  ▷  ▷  -nodes_empty \
18544 ▷  ▷  ▷  ▷  -line_width 0.2 \
18545 ▷  ▷  ▷  ▷  -embedded \
18546 ▷  ▷  ▷  -end \
18547 ▷  ▷  ▷  -W \
18548 ▷  ▷  ▷  -problem_label spreads_27_3 \
18549 ▷  ▷  -end \
18550 ▷  ▷  -define F -finite_field -q 3 -end \
18551 ▷  ▷  -define P -projective_space -n 5 -field F -v 0 -end \
18552 ▷  ▷  -define C -spread_classifier \
18553 ▷  ▷  ▷  -projective_space P \
18554 ▷  ▷  ▷  -k 3 \
18555 ▷  ▷  ▷  -starter_size 5 \
18556 ▷  ▷  ▷  -recoordinatize \
18557 ▷  ▷  ▷  -poset_classification_control Control \
18558 ▷  ▷  ▷  -output_prefix "" \
18559 ▷  ▷  -end \
18560 ▷  ▷  -with C -do -spread_classify_activity \
18561 ▷  ▷  ▷  -compute_starter \
18562 ▷  ▷  -end \
18563 ▷  ▷  -with C -do -spread_classify_activity \
18564 ▷  ▷  ▷  -isomorph \
18565 ▷  ▷  ▷  ▷  -prefix_iso "./spreads_27_3" \
18566 ▷  ▷  ▷  ▷  -use_database_for_starter \
```

```
18567 ▷  ▷  ▷  ▷  -build_db \
18568 ▷  ▷  ▷  ▷  -solution_prefix "" \
18569 ▷  ▷  ▷  ▷  -base_fname "" \
18570 ▷  ▷  ▷  -end \
18571 ▷  ▷  -end \
18572 ▷  ▷  -with C -do -spread_classify_activity \
18573 ▷  ▷  ▷  -isomorph \
18574 ▷  ▷  ▷  ▷  -prefix_iso "./spreads_27_3" \
18575 ▷  ▷  ▷  ▷  -use_database_for_starter \
18576 ▷  ▷  ▷  ▷  -read_solutions \
18577 ▷  ▷  ▷  ▷  -solution_prefix "" \
18578 ▷  ▷  ▷  ▷  -base_fname "spreads_27_3_graph" \
18579 ▷  ▷  ▷  -end \
18580 ▷  ▷  -end \
18581 ▷  ▷  -with C -do -spread_classify_activity \
18582 ▷  ▷  ▷  -isomorph \
18583 ▷  ▷  ▷  ▷  -prefix_iso "./spreads_27_3" \
18584 ▷  ▷  ▷  ▷  -use_database_for_starter \
18585 ▷  ▷  ▷  ▷  -compute_orbits \
18586 ▷  ▷  ▷  ▷  -solution_prefix "" \
18587 ▷  ▷  ▷  ▷  -base_fname "spreads_27_3_graph" \
18588 ▷  ▷  ▷  -end \
18589 ▷  ▷  -end \
18590 ▷  ▷  -with C -do -spread_classify_activity \
18591 ▷  ▷  ▷  -isomorph \
18592 ▷  ▷  ▷  ▷  -prefix_iso "./spreads_27_3" \
18593 ▷  ▷  ▷  ▷  -use_database_for_starter \
18594 ▷  ▷  ▷  ▷  -isomorph_testing \
18595 ▷  ▷  ▷  ▷  -solution_prefix "" \
18596 ▷  ▷  ▷  ▷  -base_fname "spreads_27_3_graph" \
18597 ▷  ▷  ▷  -end \
18598 ▷  ▷  -end \
18599 ▷  ▷  -with C -do -spread_classify_activity \
18600 ▷  ▷  ▷  -isomorph \
18601 ▷  ▷  ▷  ▷  -prefix_iso "./spreads_27_3" \
18602 ▷  ▷  ▷  ▷  -use_database_for_starter \
18603 ▷  ▷  ▷  ▷  -isomorph_report \
18604 ▷  ▷  ▷  ▷  -solution_prefix "" \
18605 ▷  ▷  ▷  ▷  -base_fname "spreads_27_3_graph" \
18606 ▷  ▷  ▷  -end \
18607 ▷  ▷  -end \
18608 ▷  ▷  -with C -do -spread_classify_activity \
18609 ▷  ▷  ▷  -isomorph \
18610 ▷  ▷  ▷  ▷  -prefix_iso "./spreads_27_3" \
18611 ▷  ▷  ▷  ▷  -use_database_for_starter \
18612 ▷  ▷  ▷  ▷  -recognize $(SPREAD_S27_RAO_RAO) \
18613 ▷  ▷  ▷  ▷  -solution_prefix "" \
18614 ▷  ▷  ▷  ▷  -base_fname "spreads_27_3_graph" \
18615 ▷  ▷  ▷  -end \
18616 ▷  ▷  -end
18617 ▷  #pdflatex spreads_27_isomorphism_types.tex
18618 ▷  #$(OPEN) spreads_27_isomorphism_types.pdf
18619 ▷  #pdflatex spreads_27_aut_group.tex
18620 ▷  #$(OPEN) spreads_27_aut_group.pdf
18621
18622
18623 # SPREAD_S27_RAO_RAO is isomorphic to spread 0 in the list
18624 # (which is different from the ordering of the Orbiter catalogue)
18625 # the stabilizer of the spread has order 84.
```

```
18626
18627
18628 #substructure_lifting_data::write_hash_and_datref_file id_to_hash tallied:
18629 #( 1^6076, 2^289, 3^35, 4^5 )
18630 # using 64 bit hash values, based on a modified version of Paul Hsieh´s SuperFast
      Hash
18631
18632 #We found 7 isomorphism types
18633 #0:36
18634
18635 #generators for the stabilizer of the Rao/Rao spread:
18636 #1,2,2,2,0,2,1,1,2,2,2,1,2,1,2,1,1,0,2,2,0,2,0,0,1,1,2,2,2,0,0,1,1,2,0,1, #1,0,1,
      0,1,2,0,1,2,2,2,2,1,1,1,0,1,1,2,0,1,1,2,1,1,1,0,2,2,2,0,2,0,1,1,0,
18637
18638
18639
18640 create_spread_27_0:
18641 ▷  $(ORBITER) -v 3 \
18642 ▷  ▷  -define F -finite_field -q 3 -end \
18643 ▷  ▷  -define G -linear_group -PGL 6 F -end \
18644 ▷  ▷  -define S -spread -kernel_field F \
18645 ▷  ▷  ▷  -group G -k 3 -catalogue 0 \
18646 ▷  ▷  ▷  -end \
18647 ▷  ▷  -with S -do -spread_activity \
18648 ▷  ▷  ▷  -report \
18649 ▷  ▷  -end
18650 ▷  pdflatex catalogue_q3_k3_0_report.tex
18651 ▷  $(OPEN) catalogue_q3_k3_0_report.pdf
18652
18653 create_spread_27_1:
18654 ▷  $(ORBITER) -v 3 \
18655 ▷  ▷  -define F -finite_field -q 3 -end \
18656 ▷  ▷  -define G -linear_group -PGL 6 F -end \
18657 ▷  ▷  -define S -spread -kernel_field F \
18658 ▷  ▷  ▷  -group G -k 3 -catalogue 1 \
18659 ▷  ▷  ▷  -end \
18660 ▷  ▷  -with S -do -spread_activity \
18661 ▷  ▷  ▷  -report \
18662 ▷  ▷  -end
18663 ▷  pdflatex catalogue_q3_k3_1_report.tex
18664 ▷  $(OPEN) catalogue_q3_k3_1_report.pdf
18665
18666 create_spread_27_2:
18667 ▷  $(ORBITER) -v 3 \
18668 ▷  ▷  -define F -finite_field -q 3 -end \
18669 ▷  ▷  -define G -linear_group -PGL 6 F -end \
18670 ▷  ▷  -define S -spread -kernel_field F \
18671 ▷  ▷  ▷  -group G -k 3 -catalogue 2 \
18672 ▷  ▷  ▷  -end \
18673 ▷  ▷  -with S -do -spread_activity \
18674 ▷  ▷  ▷  -report \
18675 ▷  ▷  -end
18676 ▷  pdflatex catalogue_q3_k3_2_report.tex
18677 ▷  $(OPEN) catalogue_q3_k3_2_report.pdf
18678
18679 create_spread_27_3:
18680 ▷  $(ORBITER) -v 3 \
18681 ▷  ▷  -define F -finite_field -q 3 -end \
18682 ▷  ▷  -define G -linear_group -PGL 6 F -end \
```

```
18683 ▷  ▷  -define S -spread -kernel_field F \
18684 ▷  ▷  ▷  -group G -k 3 -catalogue 3 \
18685 ▷  ▷  ▷  -end \
18686 ▷  ▷  -with S -do -spread_activity \
18687 ▷  ▷  ▷  -report \
18688 ▷  ▷  -end
18689 ▷  pdflatex catalogue_q3_k3_3_report.tex
18690 ▷  $(OPEN) catalogue_q3_k3_3_report.pdf
18691
18692 create_spread_27_4:
18693 ▷  $(ORBITER) -v 3 \
18694 ▷  ▷  -define F -finite_field -q 3 -end \
18695 ▷  ▷  -define G -linear_group -PGL 6 F -end \
18696 ▷  ▷  -define S -spread -kernel_field F \
18697 ▷  ▷  ▷  -group G -k 3 -catalogue 4 \
18698 ▷  ▷  ▷  -end \
18699 ▷  ▷  -with S -do -spread_activity \
18700 ▷  ▷  ▷  -report \
18701 ▷  ▷  -end
18702 ▷  pdflatex catalogue_q3_k3_4_report.tex
18703 ▷  $(OPEN) catalogue_q3_k3_4_report.pdf
18704
18705 create_spread_27_5:
18706 ▷  $(ORBITER) -v 3 \
18707 ▷  ▷  -define F -finite_field -q 3 -end \
18708 ▷  ▷  -define G -linear_group -PGL 6 F -end \
18709 ▷  ▷  -define S -spread -kernel_field F \
18710 ▷  ▷  ▷  -group G -k 3 -catalogue 5 \
18711 ▷  ▷  ▷  -end \
18712 ▷  ▷  -with S -do -spread_activity \
18713 ▷  ▷  ▷  -report \
18714 ▷  ▷  -end
18715 ▷  pdflatex catalogue_q3_k3_5_report.tex
18716 ▷  $(OPEN) catalogue_q3_k3_5_report.pdf
18717
18718 create_spread_27_6:
18719 ▷  $(ORBITER) -v 3 \
18720 ▷  ▷  -define F -finite_field -q 3 -end \
18721 ▷  ▷  -define G -linear_group -PGL 6 F -end \
18722 ▷  ▷  -define S -spread -kernel_field F \
18723 ▷  ▷  ▷  -group G -k 3 -catalogue 6 \
18724 ▷  ▷  ▷  -end \
18725 ▷  ▷  -with S -do -spread_activity \
18726 ▷  ▷  ▷  -report \
18727 ▷  ▷  -end
18728 ▷  pdflatex catalogue_q3_k3_6_report.tex
18729 ▷  $(OPEN) catalogue_q3_k3_6_report.pdf
18730
18731
18732
18733
18734
18735
18736
18737
18738
18739 ###############################################################################
18740 # Section 14.2: Translation planes
18741
```

```
18742
18743 SECTION_TRANSLATION_PLANES:
18744
18745
18746 test_14_2:
18747 ▷  make create_translation_plane_9b
18748 ▷  make create_translation_plane_16_4_0
18749 ▷  make create_translation_plane_16_2_0
18750 ▷  make RREF_plane_16_2_0_rank_of_incma
18751 ▷  make create_translation_plane_25_14_rank
18752 ▷  make create_translation_plane_27_Rao_Rao
18753 ▷  make RREF_Rao_Rao_plane_incma_rank
18754 ▷  make create_translation_plane_27_p_rank_of_incidence_matrix
18755 ▷  make create_translation_plane_27_5_Rao_Rao
18756 ▷  make translation_plane_27_4_orbits
18757 ▷  make translation_plane_27_5_orbits
18758 ▷  make translation_plane_27_6_orbits
18759
18760
18761
18762
18763 create_translation_plane_9b:
18764 ▷  $(ORBITER) -v 3 \
18765 ▷  ▷  -define F -finite_field -q 3 -end \
18766 ▷  ▷  -define G -linear_group -PGL 4 F -end \
18767 ▷  ▷  -define G1 -linear_group -PGL 5 F -end \
18768 ▷  ▷  -define S -spread -kernel_field F \
18769 ▷  ▷  ▷  -group G -k 2 -catalogue 1 \
18770 ▷  ▷  ▷  -end \
18771 ▷  ▷  -define T -translation_plane S G G1 -end \
18772 ▷  ▷  -with T -do -translation_plane_activity \
18773 ▷  ▷  ▷  -export_incma \
18774 ▷  ▷  -end \
18775 ▷  ▷  -with T -do -translation_plane_activity \
18776 ▷  ▷  ▷  -report \
18777 ▷  ▷  -end \
18778 ▷  ▷  -define A -linear_group -import_group_of_plane T -end \
18779 ▷  ▷  -define Orb -orbits -group A \
18780 ▷  ▷  ▷  -on_points \
18781 ▷  ▷  -end \
18782 ▷  ▷  -with Orb -do -orbits_activity \
18783 ▷  ▷  ▷  -report \
18784 ▷  ▷  -end \
18785 ▷  ▷  -with Orb -do -orbits_activity \
18786 ▷  ▷  ▷  -stabilizer 92 \
18787 ▷  ▷  -end \
18788 ▷  ▷  -with Orb -do -orbits_activity \
18789 ▷  ▷  ▷  -export_trees \
18790 ▷  ▷  -end
18791 ▷  $(ORBITER) -v 2 \
18792 ▷  ▷  -draw_matrix \
18793 ▷  ▷  ▷  -input_csv_file plane_catalogue_q3_k2_1_incma.csv \
18794 ▷  ▷  ▷  -box_width 6 -bit_depth 8 \
18795 ▷  ▷  ▷  -partition 2 91 91 \
18796 ▷  ▷  -end
18797 ▷  $(ORBITER) -v 3 \
18798 ▷  ▷  -draw_layered_graph \
18799 ▷  ▷  ▷  orbit_PGL_5_3_on_andre_3.layered_graph \
18800 ▷  ▷  ▷  -radius 250 -spanning_tree -embedded -nodes_empty \
```

```
18801 ▷  ▷  ▷  -line_width 1.1 -x_stretch 2.4 -scale 0.15 \
18802 ▷  ▷  -end
18803 ▷  pdflatex orbit_PGL_5_3_on_andre_3_draw.tex
18804 ▷  $(OPEN) orbit_PGL_5_3_on_andre_3_draw.pdf
18805 ▷  pdflatex group_of_plane_plane_catalogue_q3_k2_1_orbits_report.tex
18806 ▷  $(OPEN) group_of_plane_plane_catalogue_q3_k2_1_orbits_report.pdf
18807 ▷  pdflatex group_of_plane_plane_catalogue_q3_k2_1_stab_pt_92_report.tex
18808 ▷  $(OPEN) group_of_plane_plane_catalogue_q3_k2_1_stab_pt_92_report.pdf
18809
18810
18811
18812
18813 create_translation_plane_16_4_0:
18814 ▷  $(ORBITER) -v 3 \
18815 ▷  ▷  -define F -finite_field -q 4 -end \
18816 ▷  ▷  -define G -linear_group -PGGL 4 F -end \
18817 ▷  ▷  -define G1 -linear_group -PGGL 5 F -end \
18818 ▷  ▷  -define S -spread -kernel_field F \
18819 ▷  ▷  ▷  -group G -k 2 -catalogue 0 \
18820 ▷  ▷  ▷  -end \
18821 ▷  ▷  -define T -translation_plane S G G1 -end \
18822 ▷  ▷  -with T -do -translation_plane_activity \
18823 ▷  ▷  ▷  -export_incma \
18824 ▷  ▷  -end
18825 ▷  $(ORBITER) -v 2 \
18826 ▷  ▷  -draw_matrix \
18827 ▷  ▷  ▷  -input_csv_file plane_catalogue_q4_k2_0_incma.csv \
18828 ▷  ▷  ▷  -box_width 6 -bit_depth 8 \
18829 ▷  ▷  ▷  -partition 4 273 273 \
18830 ▷  ▷  -end
18831 ▷  $(OPEN) plane_catalogue_q4_k2_0_incma_draw.bmp
18832
18833
18834 #0 : "1200", // Hall spread
18835 #1 : "81600", // Desarguesian spread
18836 #2 : "576", // Semifield spread
18837
18838
18839 create_translation_plane_16_2_0:
18840 ▷  $(ORBITER) -v 3 \
18841 ▷  ▷  -define F -finite_field -q 2 -end \
18842 ▷  ▷  -define G -linear_group -PGL 8 F -end \
18843 ▷  ▷  -define G1 -linear_group -PGL 9 F -end \
18844 ▷  ▷  -define S -spread -kernel_field F \
18845 ▷  ▷  ▷  -group G -k 4 -catalogue 0 \
18846 ▷  ▷  ▷  -end \
18847 ▷  ▷  -define T -translation_plane S G G1 -end \
18848 ▷  ▷  -with T -do -translation_plane_activity \
18849 ▷  ▷  ▷  -export_incma \
18850 ▷  ▷  -end
18851 ▷  $(ORBITER) -v 2 \
18852 ▷  ▷  -draw_matrix \
18853 ▷  ▷  ▷  -input_csv_file plane_catalogue_q2_k4_0_incma.csv \
18854 ▷  ▷  ▷  -box_width 6 -bit_depth 8 \
18855 ▷  ▷  ▷  -partition 4 273 273 \
18856 ▷  ▷  -end
18857 ▷  $(OPEN) plane_catalogue_q2_k4_0_incma_draw.bmp
18858
18859 #0 : "1008",
```

```
18860 #1 : "1008",
18861 #2 : "1728",
18862 #3 : "216",
18863 #4 : "360",
18864 #5 : "288",
18865 #6 : "3600",
18866 #7 : "244800",
18867
18868 RREF_plane_16_2_0_rank_of_incma:
18869 ▷  $(ORBITER) -v 2 \
18870 ▷  ▷  -define F -finite_field -q 2 -end \
18871 ▷  ▷  -define v -vector -field F  \
18872 ▷  ▷  ▷  -file plane_catalogue_q2_k4_0_incma.csv \
18873 ▷  ▷  -end \
18874 ▷  ▷  -with F -do -finite_field_activity \
18875 ▷  ▷  -RREF v \
18876 ▷  ▷  -end
18877
18878 # 2-rank is 106, so the plane is Lorimer-Rahilly
18879
18880
18881 create_translation_plane_25_14_rank:
18882 ▷  $(ORBITER) -v 3 \
18883 ▷  ▷  -define F -finite_field -q 5 -end \
18884 ▷  ▷  -define G -linear_group -PGL 4 F -end \
18885 ▷  ▷  -define G1 -linear_group -PGL 5 F -end \
18886 ▷  ▷  -define S -spread -kernel_field F \
18887 ▷  ▷  ▷  -group G -k 2 -catalogue 14 \
18888 ▷  ▷  ▷  -end \
18889 ▷  ▷  -define T -translation_plane S G G1 -end \
18890 ▷  ▷  -with T -do -translation_plane_activity \
18891 ▷  ▷  ▷  -export_incma \
18892 ▷  ▷  -end
18893 ▷  $(ORBITER) -v 2 \
18894 ▷  ▷  -define F -finite_field -q 2 -end \
18895 ▷  ▷  -define v -vector -field F  \
18896 ▷  ▷  ▷  -file plane_catalogue_q5_k2_14_incma.csv \
18897 ▷  ▷  -end \
18898 ▷  ▷  -with F -do -finite_field_activity \
18899 ▷  ▷  ▷  -RREF v \
18900 ▷  ▷  -end
18901
18902
18903
18904 create_translation_plane_27_Rao_Rao:
18905 ▷  $(ORBITER) -v 3 \
18906 ▷  ▷  -define F -finite_field -q 3 -end \
18907 ▷  ▷  -define SS -vector -dense $(SPREAD_SET_27_RAO_RAO) -end \
18908 ▷  ▷  -define G -linear_group -PGL 6 F -end \
18909 ▷  ▷  -define G1 -linear_group -PGL 7 F -end \
18910 ▷  ▷  -define S -spread -kernel_field F \
18911 ▷  ▷  ▷  -group G -k 3 -spread_set "Rao_Rao" "Rao\_Rao" SS \
18912 ▷  ▷  ▷  -end \
18913 ▷  ▷  -define T -translation_plane S G G1 -end \
18914 ▷  ▷  -with T -do -translation_plane_activity \
18915 ▷  ▷  ▷  -export_incma \
18916 ▷  ▷  -end
18917
18918
```

```
18919 # creates plane_Rao_Rao_incma.csv
18920
18921
18922 RREF_Rao_Rao_plane_incma_rank:
18923 ▷  $(ORBITER) -v 2 \
18924 ▷  ▷  -define F -finite_field -q 3 -end \
18925 ▷  ▷  -define v -vector -field F  \
18926 ▷  ▷  ▷  -file plane_Rao_Rao_incma.csv \
18927 ▷  ▷  -end \
18928 ▷  ▷  -with F -do -finite_field_activity \
18929 ▷  ▷  ▷  -RREF v \
18930 ▷  ▷  -end
18931
18932 # 3-rank is 271, so the Rao / Rao plane is Moorhouse IV.
18933
18934
18935 create_translation_plane_27_p_rank_of_incidence_matrix:
18936 ▷  $(ORBITER) -v 3 \
18937 ▷  ▷  -define F -finite_field -q 3 -end \
18938 ▷  ▷  -define G -linear_group -PGL 6 F -end \
18939 ▷  ▷  -define G1 -linear_group -PGL 7 F -end \
18940 ▷  ▷  -define S -spread -kernel_field F \
18941 ▷  ▷  ▷  -group G -k 3 -catalogue 6 \
18942 ▷  ▷  ▷  -end \
18943 ▷  ▷  -define T -translation_plane S G G1 -end \
18944 ▷  ▷  -with T -do -translation_plane_activity \
18945 ▷  ▷  ▷  -p_rank 3 \
18946 ▷  ▷  -end
18947
18948 # OCN : 3-rank : spread stab : pt-orb : line-orb : Moorhouse list
18949 # 0 : 217 : 766584 : 757 : 757 : flag trans : desarguesian = Moorhouse I
18950 # 1 : 262 :   2106 : 1,27,729 : 1,27,729 : generalized twisted field = Moorhouse
      II
18951 # 2 : 268 :   1014 : 2,26,729 : 1,54,702 : Andre = Moorhouse VII
18952 # 3 : 273 :     81 : 1,27,729 : 1,27,729 : Sherk = Moorhouse V
18953 # 4 : 274 :   1092 : 28,729 : 1,756 : Hering = Moorhouse III
18954 # 5 : 271 :     84 : 28,729 : 1,756 : flag trans : Moorhouse IV = Rao Rao
18955 # 6 : 265 :     84 : 28,729 : 1,756 : flag trans :  Moorhouse VI
18956
18957 # so, Rao / Rao is OCN=5
18958
18959
18960
18961 create_translation_plane_27_5_Rao_Rao:
18962 ▷  $(ORBITER) -v 3 \
18963 ▷  ▷  -define F -finite_field -q 3 -end \
18964 ▷  ▷  -define G -linear_group -PGL 6 F -end \
18965 ▷  ▷  -define G1 -linear_group -PGL 7 F -end \
18966 ▷  ▷  -define S -spread -kernel_field F \
18967 ▷  ▷  ▷  -group G -k 3 -catalogue 5 \
18968 ▷  ▷  ▷  -end \
18969 ▷  ▷  -define T -translation_plane S G G1 -end \
18970 ▷  ▷  -with T -do -translation_plane_activity \
18971 ▷  ▷  ▷  -export_incma \
18972 ▷  ▷  -end \
18973 ▷  ▷  -with T -do -translation_plane_activity \
18974 ▷  ▷  ▷  -report \
18975 ▷  ▷  -end \
18976 ▷  ▷  -define A -linear_group -import_group_of_plane T -end \
```

```
18977 ▷  ▷  -define Orb -orbits -group A \
18978 ▷  ▷  ▷  -on_points \
18979 ▷  ▷  -end
18980 ▷  pdflatex plane_catalogue_q3_k3_5_report.tex
18981 ▷  $(OPEN) plane_catalogue_q3_k3_5_report.pdf
18982
18983
18984
18985
18986 translation_plane_27_4_orbits:
18987 ▷  $(ORBITER) -v 3 \
18988 ▷  ▷  -define F -finite_field -q 3 -end \
18989 ▷  ▷  -define G -linear_group -PGL 6 F -end \
18990 ▷  ▷  -define G1 -linear_group -PGL 7 F -end \
18991 ▷  ▷  -define S -spread -kernel_field F \
18992 ▷  ▷  ▷  -group G -k 3 -catalogue 4 \
18993 ▷  ▷  ▷  -end \
18994 ▷  ▷  -define T -translation_plane S G G1 -end \
18995 ▷  ▷  -with T -do -translation_plane_activity \
18996 ▷  ▷  ▷  -export_incma \
18997 ▷  ▷  -end \
18998 ▷  ▷  -with T -do -translation_plane_activity \
18999 ▷  ▷  ▷  -report \
19000 ▷  ▷  -end \
19001 ▷  ▷  -define A -linear_group -import_group_of_plane T -end \
19002 ▷  ▷  -define Orb -orbits -group A \
19003 ▷  ▷  ▷  -on_points \
19004 ▷  ▷  -end
19005
19006
19007
19008 translation_plane_27_5_orbits:
19009 ▷  $(ORBITER) -v 3 \
19010 ▷  ▷  -define F -finite_field -q 3 -end \
19011 ▷  ▷  -define G -linear_group -PGL 6 F -end \
19012 ▷  ▷  -define G1 -linear_group -PGL 7 F -end \
19013 ▷  ▷  -define S -spread -kernel_field F \
19014 ▷  ▷  ▷  -group G -k 3 -catalogue 5 \
19015 ▷  ▷  ▷  -end \
19016 ▷  ▷  -define T -translation_plane S G G1 -end \
19017 ▷  ▷  -with T -do -translation_plane_activity \
19018 ▷  ▷  ▷  -export_incma \
19019 ▷  ▷  -end \
19020 ▷  ▷  -with T -do -translation_plane_activity \
19021 ▷  ▷  ▷  -report \
19022 ▷  ▷  -end \
19023 ▷  ▷  -define A -linear_group -import_group_of_plane T -end \
19024 ▷  ▷  -define Orb -orbits -group A \
19025 ▷  ▷  ▷  -on_points \
19026 ▷  ▷  -end
19027
19028
19029 translation_plane_27_6_orbits:
19030 ▷  $(ORBITER) -v 3 \
19031 ▷  ▷  -define F -finite_field -q 3 -end \
19032 ▷  ▷  -define G -linear_group -PGL 6 F -end \
19033 ▷  ▷  -define G1 -linear_group -PGL 7 F -end \
19034 ▷  ▷  -define S -spread -kernel_field F \
19035 ▷  ▷  ▷  -group G -k 3 -catalogue 6 \
```

```
19036 ▷  ▷  ▷  -end \
19037 ▷  ▷  -define T -translation_plane S G G1 -end \
19038 ▷  ▷  -with T -do -translation_plane_activity \
19039 ▷  ▷  ▷  -export_incma \
19040 ▷  ▷  -end \
19041 ▷  ▷  -with T -do -translation_plane_activity \
19042 ▷  ▷  ▷  -report \
19043 ▷  ▷  -end \
19044 ▷  ▷  -define A -linear_group -import_group_of_plane T -end \
19045 ▷  ▷  -define Orb -orbits -group A \
19046 ▷  ▷  ▷  -on_points \
19047 ▷  ▷  -end
19048
19049
19050
19051
19052
19053
19054
19055 ###############################################################################
19056 # Section 14.3: Packings
19057
19058
19059 SECTION_PACKINGS:
19060
19061
19062 test_14_3:
19063 ▷  #make spread_table_PG_3_4 # slow
19064 ▷  make spread_table_PG_3_5_regular
19065 ▷  make PG_3_5_element_of_order_31_GL_normalizer
19066 ▷  make PG_3_5_element_of_order_31_ME_normalizer
19067 ▷  make PG_3_5_assume_31_graph
19068
19069
19070
19071
19072 spread_table_PG_3_4:
19073 ▷  - mkdir SPREAD_TABLES_4
19074 ▷  $(ORBITER) -v 6 \
19075 ▷  ▷  -define Control -poset_classification_control \
19076 ▷  ▷  ▷  -problem_label spreads_PG_3_4 \
19077 ▷  ▷  -end \
19078 ▷  ▷  -define F -finite_field -q 4 -end \
19079 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
19080 ▷  ▷  -define T -spread_table P 2 "0,1,2" "SPREAD_TABLES_4/" Control
19081
19082
19083 # 5096448 spreads
19084 # 1020 self dual spreads
19085 # User time: 56:38 on Mac
19086 # User time: 17:53 on Mac 1/11/2024
19087
19088 #spread_tables::save writing file SPREAD_TABLES_4/spread_16_spreads.csv
19089 #spread_tables::save written file SPREAD_TABLES_4/spread_16_spreads.csv
19090 #spread_tables::save, writing file SPREAD_TABLES_4/spread_16_spreads_iso.csv
19091 #spread_tables::save, written file SPREAD_TABLES_4/spread_16_spreads_iso.csv
19092 #spread_tables::save, writing file SPREAD_TABLES_4/spread_16_dual_spread_idx.csv
19093 #spread_tables::save, written file SPREAD_TABLES_4/spread_16_dual_spread_idx.csv
19094 #spread_tables::save, writing file SPREAD_TABLES_4/spread_16_self_dual_spreads.cs
```

```
      v
19095 #spread_tables::save, written file SPREAD_TABLES_4/spread_16_self_dual_spreads.cs
      v
19096 #spread_tables::save writing file SPREAD_TABLES_4/spread_16_schreier_table.csv
19097 #spread_tables::save written file SPREAD_TABLES_4/spread_16_schreier_table.csv
19098
19099
19100
19101 spread_table_PG_3_5_regular:
19102 ▷  - mkdir SPREAD_TABLES_5_REG
19103 ▷  $(ORBITER) -v 6 \
19104 ▷  ▷  -define Control -poset_classification_control \
19105 ▷  ▷  ▷  -problem_label spreads_PG_3_5 \
19106 ▷  ▷  -end \
19107 ▷  ▷  -define F -finite_field -q 5 -end \
19108 ▷  ▷  -define P -projective_space -n 3 -field F -end \
19109 ▷  ▷  -define T -spread_table P 2 "12" "SPREAD_TABLES_5_REG/" Control \
19110 ▷  ▷  -print_symbols
19111
19112
19113 # 21 isomorphism types of spreads in PG(3,5)
19114 # 12 is the index of the regular spread in the classification of spreads
19115 # 12/9/2020: 34 sec on Mac
19116 # time: 0:19  1/11/2024 on Mac
19117 # 155000 spreads
19118
19119 #spread_tables::save writing file SPREAD_TABLES_5_REG/spread_25_spreads.csv
19120 #spread_tables::save written file SPREAD_TABLES_5_REG/spread_25_spreads.csv
19121 #spread_tables::save, writing file SPREAD_TABLES_5_REG/spread_25_spreads_iso.csv
19122 #spread_tables::save, written file SPREAD_TABLES_5_REG/spread_25_spreads_iso.csv
19123 #spread_tables::save, writing file SPREAD_TABLES_5_REG/spread_25_dual_spread_idx.
      csv
19124 #spread_tables::save, written file SPREAD_TABLES_5_REG/spread_25_dual_spread_idx.
      csv
19125 #spread_tables::save, writing file SPREAD_TABLES_5_REG/spread_25_self_dual_spread
      s.csv
19126 #spread_tables::save, written file SPREAD_TABLES_5_REG/spread_25_self_dual_spread
      s.csv
19127 #spread_tables::save writing file SPREAD_TABLES_5_REG/spread_25_schreier_table.cs
      v
19128 #spread_tables::save written file SPREAD_TABLES_5_REG/spread_25_schreier_table.cs
      v
19129
19130
19131 PG_3_5_element_of_order_31_GL_normalizer:
19132 ▷  $(ORBITER) -v 6 -define G \
19133 ▷  ▷  -linear_group -GL 4 5 -end \
19134 ▷  ▷  -with G -do \
19135 ▷  ▷  -group_theoretic_activity \
19136 ▷  ▷  ▷  -normalizer_of_cyclic_subgroup "124" \
19137 ▷  ▷  ▷  "1,0,0,0, 0,3,4,3, 0,3,3,4, 0,3,2,3" \
19138 ▷  ▷  -end
19139 ▷  pdflatex normalizer_of_124_in_GL_4_5.tex
19140 ▷  $(OPEN) normalizer_of_124_in_GL_4_5.pdf
19141
19142 # needs magma
19143 # the group has order 124.
19144 # the normalizer has order 1488
19145 # normalizer has order 1488=4*372=4*4*3*31
```

```
19146
19147 PG_3_5_element_of_order_31_ME_normalizer:
19148 ▷  $(ORBITER) -v 6 -define G \
19149 ▷  ▷  -linear_group -PGL 4 5 -end \
19150 ▷  ▷  -with G -do \
19151 ▷  ▷  -group_theoretic_activity \
19152 ▷  ▷  ▷  -normalizer_of_cyclic_subgroup "31" \
19153 ▷  ▷  ▷  "1,0,0,0, 0,3,4,3, 0,3,3,4, 0,3,2,3" \
19154 ▷  ▷  -end
19155 ▷  mv normalizer_of_31_in_PGL_4_5.tex normalizer_of_31_ME_in_PGL_4_5.tex
19156 ▷  pdflatex normalizer_of_31_ME_in_PGL_4_5.tex
19157 ▷  $(OPEN) normalizer_of_31_ME_in_PGL_4_5.pdf
19158
19159
19160 # group has order 31
19161 # normalizer has order 372
19162
19163
19164
19165 PG_3_5_assume_31_graph:
19166 ▷  $(ORBITER) -v 5 \
19167 ▷  ▷  -define Control -poset_classification_control \
19168 ▷  ▷  ▷  -problem_label spreads_PG_3_5 \
19169 ▷  ▷  -end \
19170 ▷  ▷  -define F -finite_field -q 5 -end \
19171 ▷  ▷  -define P3 -projective_space -n 3 -field F -v 0 -end \
19172 ▷  ▷  -define P5 -projective_space -n 5 -field F -v 0 -end \
19173 ▷  ▷  -define T -spread_table P3 2 "12" "SPREAD_TABLES_5_REG/" Control \
19174 ▷  ▷  -define PC -packing_classify P3 P5 T \
19175 ▷  ▷  -define PW -packing_with_symmetry_assumption PC \
19176 ▷  ▷  ▷  ▷  -H "H31" $(PGL_4_5_SUBGROUP_31_ME) -end \
19177 ▷  ▷  ▷  ▷  -N "N31" $(PGL_4_5_SUBGROUP_31_ME_NORMALIZER) -end \
19178 ▷  ▷  ▷  -end \
19179 ▷  ▷  -define PWF -packing_choose_fixed_points PW 0 -end \
19180 ▷  ▷  -define L -packing_long_orbits PWF \
19181 ▷  ▷  ▷  -orbit_length 31 -create_graphs \
19182 ▷  ▷  ▷  -end \
19183 ▷  ▷  -print_symbols
19184
19185
19186 no:
19187 ▷  pdflatex H31_reduced_spread_orbits_orbits_report.tex
19188 ▷  $(OPEN) H31_reduced_spread_orbits_orbits_report.pdf
19189 ▷  pdflatex H31_line_orbits_orbits_report.tex
19190 ▷  $(OPEN) H31_line_orbits_orbits_report.pdf
19191 ▷  pdflatex H31_line_orbits_orbits_report.tex
19192 ▷  $(OPEN) H31_line_orbits_orbits_report.pdf
19193 ▷  pdflatex N31_line_orbits_orbits_report.tex
19194 ▷  $(OPEN) N31_line_orbits_orbits_report.pdf
19195 ▷  pdflatex H31_point_orbits_orbits_report.tex
19196 ▷  $(OPEN) H31_point_orbits_orbits_report.pdf
19197 ▷  pdflatex N31_point_orbits_orbits_report.tex
19198 ▷  $(OPEN) N31_point_orbits_orbits_report.pdf
19199
19200 ▷  #pdflatex H31_spread_orbits_orbits_report.tex
19201 ▷  #$(OPEN) H31_spread_orbits_orbits_report.pdf
19202 #H31_line_orbits_orbits.bin
19203 #H31_line_orbits_orbits_report.tex
19204 #H_spread_orbits_orbit_types_report.tex
```

```
19205 #H31_spread_orbits_orbits.bin
19206 #H31_good_orbits
19207 #H31_spread_types_reduced_orbit_types_report.tex
19208 #H31_reduced_spread_orbits_orbits.bin
19209 #H31_fpc0_lo.graph
19210
19211
19212 # number of points = 156
19213 # number of lines = 856
19214 #spread_classify::init The order of PGGL(4,5) is 29016000000
19215 #spread orbit 12 has group order 187200 orbit_length = 155000
19216 #spread_tables::init nb_lines=806
19217 #spread_tables::init spread_size=26
19218 #spread_tables::init nb_iso_types_of_spreads=21
19219 #spread_tables::init prefix=SPREAD_TABLES_5_REG/spread_25
19220 #packing_classify::init_P3_and_P5_and_Gr P3->N_points=156
19221 #packing_classify::init_P3_and_P5_and_Gr P3->N_lines=806
19222 #packing_classify::init_P3_and_P5_and_Gr P5->N_points=3906
19223 #packing_classify::init_P3_and_P5_and_Gr P5->N_lines=508431
19224 #-H H31
19225 #-PGL 4 5
19226 #-subgroup_by_generators 31 1 1,0,0,0, 0,3,4,3, 0,3,3,4, 0,3,2,3
19227 #-N N31
19228 #-PGL 4 5
19229 #-subgroup_by_generators normalizer_31 4 1,0,0,0,0,4,0,0,0,0,4,0,0,0,0,4, 1,0,0,0
      ,0,3,0,0,0,0,3,0,0,0,0,3, 1,0,0,0,0,4,0,0,0,0,2,1,0,3,2,4, 1,0,0,0,0,0,1,0,0,0,0,
      1,0,1,1,3,
19230 #packing_was::init_H H_goi=31
19231 #packing_was::init_N N_goi=372
19232 #orbits_on_something::init after reading orbits from file H31_point_orbits_orbits
      .bin
19233 #orbits_on_something::init orbit length distribution:
19234 #1, 31^5
19235 #orbits_on_something::init after reading orbits from file N31_point_orbits_orbits
      .bin
19236 #orbits_on_something::init orbit length distribution:
19237 #1, 31, 124
19238 #orbits_on_something::init after reading orbits from file H31_line_orbits_orbits.
      bin
19239 #orbits_on_something::init orbit length distribution:
19240 #31^{26}
19241 #orbits_on_something::init after reading orbits from file H31_spread_orbits_orbit
      s.bin
19242 #orbits_on_something::init orbit length distribution:
19243 #31^{5000}
19244 #packing_was::reduce_spreads nb_good_spreads = 248
19245 #packing_was::compute_H_orbits_on_lines after reduced_spread_orbits_under_H->crea
      te_latex_report
19246 #orbits_on_something::classify_orbits_by_length
19247 #orbits_on_something::classify_orbits_by_length The distribution of orbit lengths
       is: 31^8
19248 #orbits_on_something::init after reading orbits from file N31_line_orbits_orbits.
      bin
19249 #orbits_on_something::init orbit length distribution:
19250 #31^2, 372^2
19251
19252
19253
19254 #packing_was::compute_reduced_spread_types_wrt_H
```

```
19255 #orbit_type_repository::init nb_sets = 248 goi=31
19256
19257
19258 #orbits_on_something::init after reading orbits from file H31_reduced_spread_orbi
      ts_orbits.bin
19259 #orbits_on_something::init orbit length distribution:
19260 #31^8
19261
19262
19263 #orbits_on_something::init after reading orbits from file N31_line_orbits_orbits.
      bin
19264 #orbits_on_something::init orbit length distribution:
19265 #31^2, 372^2
19266
19267
19268 ################################################################################
19269 # Section 14.4: BLT-sets
19270
19271 SECTION_BLT_SETS:
19272
19273
19274 test_14_4:
19275 ▷  make BLT_5_1
19276 ▷  make BLT_5_1_export_gap
19277 ▷  make BLT_5_Linear
19278 ▷  make BLT_9_0
19279 ▷  make BLT_9_Kantor1
19280 ▷  make BLT_11_0
19281 ▷  make BLT_11_Fisher
19282 ▷  make BLT_11_Mondello
19283 ▷  make BLT_11_FTWKB
19284 ▷  make BLT_13_Kantor2
19285 ▷  make BLT_67_4
19286 ▷  make BLT_67_4_export_gap
19287 ▷  make BLT_9_deep_search
19288 ▷  make BLT_11_deep_search
19289 ▷  make BLT_13_deep_search
19290 ▷  make BLT_13_classify_starter
19291 ▷  make BLT_13_cliques
19292 ▷  make BLT_13_isomorph_read_DB
19293 ▷  make BLT_13_isomorph_read_solutions
19294 ▷  make BLT_13_isomorph_stabilizer_orbits
19295 ▷  make BLT_13_isomorph_testing
19296 ▷  make blt_set_export_gap_all
19297
19298
19299
19300 BLT_5_1:
19301 ▷  $(ORBITER) -v 10 \
19302 ▷  ▷  -define F -finite_field -q 5 -end \
19303 ▷  ▷  -define O -orthogonal_space 0 5 F -end \
19304 ▷  ▷  -define BLT -BLT_set \
19305 ▷  ▷  ▷  -space O -catalogue 1 -invariants \
19306 ▷  ▷  -end \
19307 ▷  ▷  -with BLT -do -blt_set_activity \
19308 ▷  ▷  ▷  -report \
19309 ▷  ▷  -end
19310 ▷  pdflatex BLT_catalogue_q5_iso1.tex
19311 ▷  $(OPEN) BLT_catalogue_q5_iso1.pdf
```

```
19312
19313
19314 BLT_5_1_export_gap:
19315 ▷  $(ORBITER) -v 2 \
19316 ▷  ▷  -define F -finite_field -q 5 -end \
19317 ▷  ▷  -define O -orthogonal_space 0 5 F -end \
19318 ▷  ▷  -define BLT -BLT_set \
19319 ▷  ▷  ▷  -space O -catalogue 1 \
19320 ▷  ▷  -end \
19321 ▷  ▷  -with BLT -do -blt_set_activity \
19322 ▷  ▷  ▷  -export_gap \
19323 ▷  ▷  -end
19324
19325
19326 BLT_5_Linear:
19327 ▷  $(ORBITER) -v 2 \
19328 ▷  ▷  -define F -finite_field -q 5 -end \
19329 ▷  ▷  -define O -orthogonal_space 0 5 F -end \
19330 ▷  ▷  -define BLT -BLT_set \
19331 ▷  ▷  ▷  -space O -family "Linear" -invariants \
19332 ▷  ▷  -end \
19333 ▷  ▷  -with BLT -do -blt_set_activity \
19334 ▷  ▷  ▷  -report \
19335 ▷  ▷  -end
19336 ▷  pdflatex BLT_Linear_q5.tex
19337 ▷  $(OPEN) BLT_Linear_q5.pdf
19338
19339
19340 BLT_9_0:
19341 ▷  $(ORBITER) -v 10 \
19342 ▷  ▷  -define F -finite_field -q 9 -end \
19343 ▷  ▷  -define O -orthogonal_space 0 5 F -end \
19344 ▷  ▷  -define BLT -BLT_set \
19345 ▷  ▷  ▷  -space O -catalogue 0 -invariants \
19346 ▷  ▷  -end \
19347 ▷  ▷  -with BLT -do -blt_set_activity \
19348 ▷  ▷  ▷  -report \
19349 ▷  ▷  -end
19350 ▷  pdflatex BLT_catalogue_q9_iso0.tex
19351 ▷  $(OPEN) BLT_catalogue_q9_iso0.pdf
19352
19353
19354
19355
19356 BLT_9_Kantor1:
19357 ▷  $(ORBITER) -v 2 \
19358 ▷  ▷  -define F -finite_field -q 9 -end \
19359 ▷  ▷  -define O -orthogonal_space 0 5 F -end \
19360 ▷  ▷  -define BLT -BLT_set \
19361 ▷  ▷  ▷  -space O -family "Kantor1" -invariants \
19362 ▷  ▷  -end \
19363 ▷  ▷  -with BLT -do -blt_set_activity \
19364 ▷  ▷  ▷  -report \
19365 ▷  ▷  -end
19366 ▷  pdflatex BLT_Kantor1_q9.tex
19367 ▷  $(OPEN) BLT_Kantor1_q9.pdf
19368
19369
19370
```

```
19371
19372 BLT_11_0:
19373 ▷  $(ORBITER) -v 2 \
19374 ▷  ▷  -define F -finite_field -q 11 -end \
19375 ▷  ▷  -define O -orthogonal_space 0 5 F -end \
19376 ▷  ▷  -define BLT -BLT_set \
19377 ▷  ▷  ▷  -space O -catalogue 0 -invariants \
19378 ▷  ▷  -end \
19379 ▷  ▷  -with BLT -do -blt_set_activity \
19380 ▷  ▷  ▷  -report \
19381 ▷  ▷  -end
19382 ▷  pdflatex BLT_catalogue_q11_iso0.tex
19383 ▷  $(OPEN) BLT_catalogue_q11_iso0.pdf
19384
19385
19386 BLT_11_Fisher:
19387 ▷  $(ORBITER) -v 2 \
19388 ▷  ▷  -define F -finite_field -q 11 -end \
19389 ▷  ▷  -define O -orthogonal_space 0 5 F -end \
19390 ▷  ▷  -define BLT -BLT_set \
19391 ▷  ▷  ▷  -space O -family "Fisher" -invariants \
19392 ▷  ▷  -end \
19393 ▷  ▷  -with BLT -do -blt_set_activity \
19394 ▷  ▷  ▷  -report \
19395 ▷  ▷  -end
19396 ▷  pdflatex BLT_Fisher_q11.tex
19397 ▷  $(OPEN) BLT_Fisher_q11.pdf
19398
19399 BLT_11_Mondello:
19400 ▷  $(ORBITER) -v 2 \
19401 ▷  ▷  -define F -finite_field -q 11 -end \
19402 ▷  ▷  -define O -orthogonal_space 0 5 F -end \
19403 ▷  ▷  -define BLT -BLT_set \
19404 ▷  ▷  ▷  -space O -family "Mondello" -invariants \
19405 ▷  ▷  -end \
19406 ▷  ▷  -with BLT -do -blt_set_activity \
19407 ▷  ▷  ▷  -report \
19408 ▷  ▷  -end
19409 ▷  pdflatex BLT_Mondello_q11.tex
19410 ▷  $(OPEN) BLT_Mondello_q11.pdf
19411
19412
19413 BLT_11_FTWKB:
19414 ▷  $(ORBITER) -v 2 \
19415 ▷  ▷  -define F -finite_field -q 11 -end \
19416 ▷  ▷  -define O -orthogonal_space 0 5 F -end \
19417 ▷  ▷  -define BLT -BLT_set \
19418 ▷  ▷  ▷  -space O -family "FTWKB" -invariants \
19419 ▷  ▷  -end \
19420 ▷  ▷  -with BLT -do -blt_set_activity \
19421 ▷  ▷  ▷  -report \
19422 ▷  ▷  -end
19423 ▷  pdflatex BLT_FTWKB_q11.tex
19424 ▷  $(OPEN) BLT_FTWKB_q11.pdf
19425
19426
19427 # for K2, q must be congruent to 2 or 3 mod 5
19428 BLT_13_Kantor2:
19429 ▷  $(ORBITER) -v 2 \
```

```
19430 ▷  ▷  -define F -finite_field -q 13 -end \
19431 ▷  ▷  -define O -orthogonal_space 0 5 F -end \
19432 ▷  ▷  -define BLT -BLT_set \
19433 ▷  ▷  ▷  -space O -family "Kantor2" -invariants \
19434 ▷  ▷  -end \
19435 ▷  ▷  -with BLT -do -blt_set_activity \
19436 ▷  ▷  ▷  -report \
19437 ▷  ▷  -end
19438 ▷  pdflatex BLT_Kantor2_q13.tex
19439 ▷  $(OPEN) BLT_Kantor2_q13.pdf
19440
19441
19442 BLT_67_4:
19443 ▷  $(ORBITER) -v 4 \
19444 ▷  ▷  -define F -finite_field -q 67 -end \
19445 ▷  ▷  -define O -orthogonal_space 0 5 F -end \
19446 ▷  ▷  -define BLT -BLT_set \
19447 ▷  ▷  ▷  -space O -catalogue 4 -invariants \
19448 ▷  ▷  -end \
19449 ▷  ▷  -with BLT -do -blt_set_activity \
19450 ▷  ▷  ▷  -report \
19451 ▷  ▷  -end
19452
19453
19454 BLT_67_4_export_gap:
19455 ▷  $(ORBITER) -v 2 \
19456 ▷  ▷  -define F -finite_field -q 67 -end \
19457 ▷  ▷  -define O -orthogonal_space 0 5 F -end \
19458 ▷  ▷  -define BLT -BLT_set \
19459 ▷  ▷  ▷  -space O -catalogue 4 \
19460 ▷  ▷  -end \
19461 ▷  ▷  -with BLT -do -blt_set_activity \
19462 ▷  ▷  ▷  -export_gap \
19463 ▷  ▷  -end
19464
19465
19466
19467
19468 BLT_9_deep_search:
19469 ▷  $(ORBITER) -v 2 \
19470 ▷  ▷  -define F -finite_field -q 9 -end \
19471 ▷  ▷  -define O -orthogonal_space 0 5 F -end \
19472 ▷  ▷  -define C -BLT_set_classifier O -starter_size 10 -end \
19473 ▷  ▷  -define Activity -poset_classification_activity -report -end -end \
19474 ▷  ▷  -define Activity_export -poset_classification_activity \
19475 ▷  ▷  ▷  -export_level_to_cpp 10 -end \
19476 ▷  ▷  -with C -do -BLT_set_classify_activity \
19477 ▷  ▷  ▷  -compute_starter \
19478 ▷  ▷  ▷  ▷  -problem_label BLT_q9 \
19479 ▷  ▷  ▷  ▷  -W -depth 10 \
19480 ▷  ▷  ▷  -end \
19481 ▷  ▷  -end \
19482 ▷  ▷  -with C -do -BLT_set_classify_activity \
19483 ▷  ▷  ▷  -poset_classification_activity \
19484 ▷  ▷  ▷  ▷  Activity \
19485 ▷  ▷  -end \
19486 ▷  ▷  -with C -do -BLT_set_classify_activity \
19487 ▷  ▷  ▷  -poset_classification_activity \
19488 ▷  ▷  ▷  ▷  Activity_export \
```

```
19489 ▷  ▷  -end
19490 ▷  pdflatex BLT_q9_poset.tex
19491 ▷  $(OPEN) BLT_q9_poset.pdf
19492
19493
19494
19495 BLT_11_deep_search:
19496 ▷  $(ORBITER) -v 2 \
19497 ▷  ▷  -define F -finite_field -q 11 -end \
19498 ▷  ▷  -define O -orthogonal_space 0 5 F -end \
19499 ▷  ▷  -define C -BLT_set_classifier O -starter_size 12 -end \
19500 ▷  ▷  -with C -do -BLT_set_classify_activity \
19501 ▷  ▷  ▷  -compute_starter \
19502 ▷  ▷  ▷  ▷  -problem_label BLT_q11 \
19503 ▷  ▷  ▷  ▷  -W -depth 12 \
19504 ▷  ▷  ▷  -end \
19505 ▷  ▷  -end
19506 ▷  #pdflatex BLT_q11_poset.tex
19507 ▷  #$(OPEN) BLT_q11_poset.pdf
19508
19509
19510
19511
19512 BLT_13_deep_search:
19513 ▷  $(ORBITER) -v 2 \
19514 ▷  ▷  -define F -finite_field -q 13 -end \
19515 ▷  ▷  -define O -orthogonal_space 0 5 F -end \
19516 ▷  ▷  -define C -BLT_set_classifier O -starter_size 14 -end \
19517 ▷  ▷  -with C -do -BLT_set_classify_activity \
19518 ▷  ▷  ▷  -compute_starter \
19519 ▷  ▷  ▷  ▷  -problem_label BLT_q13 \
19520 ▷  ▷  ▷  ▷  -W -depth 14 \
19521 ▷  ▷  ▷  -end \
19522 ▷  ▷  -end
19523 ▷  #pdflatex BLT_q13_poset.tex
19524 ▷  #$(OPEN) BLT_q13_poset.pdf
19525
19526
19527
19528
19529 BLT_13_classify_starter:
19530 ▷  $(ORBITER) -v 2 \
19531 ▷  ▷  -define F -finite_field -q 13 -end \
19532 ▷  ▷  -define O -orthogonal_space 0 5 F -end \
19533 ▷  ▷  -define C -BLT_set_classifier O -starter_size 5 -end \
19534 ▷  ▷  -with C -do -BLT_set_classify_activity \
19535 ▷  ▷  ▷  -compute_starter \
19536 ▷  ▷  ▷  ▷  -problem_label BLT_q13 \
19537 ▷  ▷  ▷  ▷  -W -depth 5 \
19538 ▷  ▷  ▷  -end \
19539 ▷  ▷  -end \
19540 ▷  ▷  -with C -do -BLT_set_classify_activity \
19541 ▷  ▷  ▷  -create_graphs \
19542 ▷  ▷  -end
19543
19544
19545
19546
19547 BLT_13_cliques:
```

```
19548 ▷  $(ORBITER) -v 2 \
19549 ▷  ▷  -loop L 0 38 1 \
19550 ▷  ▷  ▷  -define G -graph \
19551 ▷  ▷  ▷  ▷  -load BLT_q13_graph_5_%L.bin \
19552 ▷  ▷  ▷  -end \
19553 ▷  ▷  ▷  -with G -do \
19554 ▷  ▷  ▷  -graph_theoretic_activity \
19555 ▷  ▷  ▷  ▷  -find_cliques -rainbow -target_size 9 -end \
19556 ▷  ▷  ▷  -end \
19557 ▷  ▷  -end_loop L
19558
19559
19560 # 3 solutions:
19561 #BLT_q13_graph_5_0_sol.txt
19562 #BLT_q13_graph_5_0_sol.csv
19563
19564
19565
19566 BLT_13_isomorph_read_DB:
19567 ▷  $(ORBITER) -v 2 \
19568 ▷  ▷  -define F -finite_field -q 13 -end \
19569 ▷  ▷  -define O -orthogonal_space 0 5 F -end \
19570 ▷  ▷  -define C -BLT_set_classifier O -starter_size 5 -end \
19571 ▷  ▷  -with C -do -BLT_set_classify_activity \
19572 ▷  ▷  ▷  -compute_starter \
19573 ▷  ▷  ▷  ▷  -problem_label BLT_q13 \
19574 ▷  ▷  ▷  ▷  -W -depth 5 \
19575 ▷  ▷  ▷  -end \
19576 ▷  ▷  -end \
19577 ▷  ▷  -with C -do -BLT_set_classify_activity \
19578 ▷  ▷  ▷  -isomorph \
19579 ▷  ▷  ▷  ▷  -prefix_iso "./BLT_q13" \
19580 ▷  ▷  ▷  ▷  -use_database_for_starter \
19581 ▷  ▷  ▷  ▷  -build_db \
19582 ▷  ▷  ▷  ▷  -solution_prefix "" \
19583 ▷  ▷  ▷  ▷  -base_fname "" \
19584 ▷  ▷  ▷  -end \
19585 ▷  ▷  -end
19586
19587
19588
19589 BLT_13_isomorph_read_solutions:
19590 ▷  $(ORBITER) -v 2 \
19591 ▷  ▷  -define F -finite_field -q 13 -end \
19592 ▷  ▷  -define O -orthogonal_space 0 5 F -end \
19593 ▷  ▷  -define C -BLT_set_classifier O -starter_size 5 -end \
19594 ▷  ▷  -with C -do -BLT_set_classify_activity \
19595 ▷  ▷  ▷  -compute_starter \
19596 ▷  ▷  ▷  ▷  -problem_label BLT_q13 \
19597 ▷  ▷  ▷  ▷  -W -depth 5 \
19598 ▷  ▷  ▷  -end \
19599 ▷  ▷  -end \
19600 ▷  ▷  -with C -do -BLT_set_classify_activity \
19601 ▷  ▷  ▷  -isomorph \
19602 ▷  ▷  ▷  ▷  -prefix_iso "./BLT_q13" \
19603 ▷  ▷  ▷  ▷  -use_database_for_starter \
19604 ▷  ▷  ▷  ▷  -read_solutions \
19605 ▷  ▷  ▷  ▷  -list_of_cases BLT_q13_list_of_cases_5_0_1.csv \
19606 ▷  ▷  ▷  ▷  -solution_prefix "" \
```

```
19607 ▷  ▷  ▷  ▷  -base_fname "BLT_q13_graph" \
19608 ▷  ▷  ▷  -end \
19609 ▷  ▷  -end
19610
19611
19612 BLT_13_isomorph_stabilizer_orbits:
19613 ▷  $(ORBITER) -v 2 \
19614 ▷  ▷  -define F -finite_field -q 13 -end \
19615 ▷  ▷  -define O -orthogonal_space 0 5 F -end \
19616 ▷  ▷  -define C -BLT_set_classifier O -starter_size 5 -end \
19617 ▷  ▷  -with C -do -BLT_set_classify_activity \
19618 ▷  ▷  ▷  -compute_starter \
19619 ▷  ▷  ▷  ▷  -problem_label BLT_q13 \
19620 ▷  ▷  ▷  ▷  -W -depth 5 \
19621 ▷  ▷  ▷  -end \
19622 ▷  ▷  -end \
19623 ▷  ▷  -with C -do -BLT_set_classify_activity \
19624 ▷  ▷  ▷  -isomorph \
19625 ▷  ▷  ▷  ▷  -prefix_iso "./BLT_q13" \
19626 ▷  ▷  ▷  ▷  -use_database_for_starter \
19627 ▷  ▷  ▷  ▷  -compute_orbits \
19628 ▷  ▷  ▷  ▷  -list_of_cases BLT_q13_list_of_cases_5_0_1.csv \
19629 ▷  ▷  ▷  ▷  -solution_prefix "" \
19630 ▷  ▷  ▷  ▷  -base_fname "BLT_q13_graph" \
19631 ▷  ▷  ▷  -end \
19632 ▷  ▷  -end
19633
19634 BLT_13_isomorph_testing:
19635 ▷  $(ORBITER) -v 4 \
19636 ▷  ▷  -define F -finite_field -q 13 -end \
19637 ▷  ▷  -define O -orthogonal_space 0 5 F -end \
19638 ▷  ▷  -define C -BLT_set_classifier O -starter_size 5 -end \
19639 ▷  ▷  -with C -do -BLT_set_classify_activity \
19640 ▷  ▷  ▷  -compute_starter \
19641 ▷  ▷  ▷  ▷  -problem_label BLT_q13 \
19642 ▷  ▷  ▷  ▷  -W -depth 5 \
19643 ▷  ▷  ▷  -end \
19644 ▷  ▷  -end \
19645 ▷  ▷  -with C -do -BLT_set_classify_activity \
19646 ▷  ▷  ▷  -isomorph \
19647 ▷  ▷  ▷  ▷  -prefix_iso "./BLT_q13" \
19648 ▷  ▷  ▷  ▷  -use_database_for_starter \
19649 ▷  ▷  ▷  ▷  -isomorph_testing \
19650 ▷  ▷  ▷  ▷  -solution_prefix "" \
19651 ▷  ▷  ▷  ▷  -base_fname "BLT_q13_graph" \
19652 ▷  ▷  ▷  -end \
19653 ▷  ▷  -end \
19654 ▷  ▷  -with C -do -BLT_set_classify_activity \
19655 ▷  ▷  ▷  -isomorph \
19656 ▷  ▷  ▷  ▷  -prefix_iso "./BLT_q13" \
19657 ▷  ▷  ▷  ▷  -use_database_for_starter \
19658 ▷  ▷  ▷  ▷  -isomorph_report \
19659 ▷  ▷  ▷  ▷  -solution_prefix "" \
19660 ▷  ▷  ▷  ▷  -base_fname "BLT_q13_graph" \
19661 ▷  ▷  ▷  -end \
19662 ▷  ▷  -end
19663 ▷  pdflatex BLT_q13_isomorphism_types.tex
19664 ▷  $(OPEN) BLT_q13_isomorphism_types.pdf
19665
```

```
19666
19667
19668
19669
19670 blt_set_export_gap_all:
19671 ▷  $(ORBITER) -v 3 \
19672 ▷  ▷  -define Q -vector -dense \
19673 ▷  ▷  ▷  $(BLT_ORDER_Q) \
19674 ▷  ▷  -end \
19675 ▷  ▷  -define NB -vector -dense \
19676 ▷  ▷  ▷  $(BLT_NUMBER_ISO) \
19677 ▷  ▷  -end \
19678 ▷  ▷  -loop_over i Q \
19679 ▷  ▷  ▷  -define F%i -finite_field -q "%i[Q]" -end \
19680 ▷  ▷  ▷  -define O%i -orthogonal_space 0 5 F%i -end \
19681 ▷  ▷  ▷  -loop j 0 %i[NB] 1 \
19682 ▷  ▷  ▷  ▷  -define BLT_%i_%j -BLT_set \
19683 ▷  ▷  ▷  ▷  ▷  -space O%i -catalogue %j \
19684 ▷  ▷  ▷  ▷  -end \
19685 ▷  ▷  ▷  ▷  -with BLT_%i_%j -do -blt_set_activity \
19686 ▷  ▷  ▷  ▷  ▷  -export_gap \
19687 ▷  ▷  ▷  ▷  -end \
19688 ▷  ▷  ▷  -end_loop j \
19689 ▷  ▷  -end_loop_over i \
19690 ▷  ▷  -print_symbols
19691
19692
19693
19694
19695
19696
19697
19698
19699 ################################################################################
19700 # Chapter 15 - Computer Science Primitives
19701 ################################################################################
19702
19703
19704
19705 test_15:
19706 ▷  make test_15_1
19707 ▷  make test_15_2
19708 ▷  make test_15_3
19709
19710
19711
19712
19713 ################################################################################
19714 # Section 15.1: Clique finding
19715
19716
19717 SECTION_GRAPH_THEORY_CLIQUE_FINDING:
19718
19719
19720 test_15_1:
19721 ▷  make small_graph_cliques
19722 ▷  make BLT_q13_graph_5_0_cliques_bw
19723 ▷  make BLT_q13_graph_5_0_cliques_rainbow
19724 ▷  make small_graph_cliques_Sajeeb
```

```
19725 ▷  make Paley_13_aut
19726 ▷  make Paley_13
19727 ▷  make Paley_13_cliques_classify
19728 ▷  make Paley_13_cliques_all
19729 ▷  make tritangent_planes_graph_aut
19730 ▷  make tritangent_planes_graph_cliques_classify
19731 ▷  make tritangent_planes_graph_cliques_all
19732 ▷  make PGO_5_2_cliques
19733 ▷  make HJ_d2_c5
19734 ▷  make HJ64_cliques5
19735 ▷  make HJ64_cliques5_classify
19736
19737
19738
19739 small_graph_cliques: graph_v5_e7.colored_graph
19740 ▷  $(ORBITER) -v 2 \
19741 ▷  ▷  -define G -graph -load graph_v5_e7.colored_graph -end \
19742 ▷  ▷  -with G -do \
19743 ▷  ▷  -graph_theoretic_activity \
19744 ▷  ▷  ▷  -find_cliques -target_size 3 \
19745 ▷  ▷  -end
19746
19747 # nb_sol = 3
19748
19749
19750
19751
19752 BLT_q13_graph_5_0_cliques_bw:
19753 ▷  $(ORBITER) -v 2 \
19754 ▷  ▷  -define G -graph -load BLT_q13_graph_5_0.bin -end \
19755 ▷  ▷  -with G -do \
19756 ▷  ▷  -graph_theoretic_activity \
19757 ▷  ▷  ▷  -find_cliques -target_size 9 \
19758 ▷  ▷  -end
19759
19760 # all_cliques_black_and_white
19761
19762
19763 BLT_q13_graph_5_0_cliques_rainbow:
19764 ▷  $(ORBITER) -v 2 \
19765 ▷  ▷  -define G -graph -load BLT_q13_graph_5_0.bin -end \
19766 ▷  ▷  -with G -do \
19767 ▷  ▷  -graph_theoretic_activity \
19768 ▷  ▷  ▷  -find_cliques -rainbow -target_size 9 \
19769 ▷  ▷  -end
19770
19771 # all_rainbow_cliques
19772
19773
19774 small_graph_cliques_Sajeeb:
19775 ▷  $(ORBITER) -v 2 \
19776 ▷  ▷  -define G -graph -load graph_v5_e7.colored_graph -end \
19777 ▷  ▷  -with G -do \
19778 ▷  ▷  -graph_theoretic_activity \
19779 ▷  ▷  ▷  -find_cliques -Sajeeb -target_size 3 \
19780 ▷  ▷  -end
19781
19782 # nb_sol = 3
19783
```

```
19784 Paley_13_aut:
19785 ▷  $(ORBITER) -v 2 \
19786 ▷  ▷  -define F -finite_field -q 13 -end \
19787 ▷  ▷  -define Gamma -graph -Paley F -end \
19788 ▷  ▷  -with Gamma -do \
19789 ▷  ▷  -graph_theoretic_activity \
19790 ▷  ▷  ▷  -automorphism_group \
19791 ▷  ▷  -end
19792
19793 # writes Paley_13_group.makefile
19794 #User time: 0 of a second, dt=0 tps = 100
19795 #nb_calls_to_densenauty=1
19796
19797
19798
19799 Paley_13:
19800 ▷  $(ORBITER) -v 2 \
19801 ▷  ▷  -define gens -vector -file Paley_13_gens.csv -end \
19802 ▷  ▷  -define G -permutation_group \
19803 ▷  ▷  -bsgs Paley_13 "Paley\_13" 13 78 "0,1" 3 gens -end \
19804
19805
19806 Paley_13_cliques_classify:
19807 ▷  $(ORBITER) -v 4 \
19808 ▷  ▷  -define Control -poset_classification_control \
19809 ▷  ▷  ▷  -W \
19810 ▷  ▷  ▷  -problem_label Paley13_cliques \
19811 ▷  ▷  ▷  -clique_test Gamma \
19812 ▷  ▷  ▷  -depth 5 \
19813 ▷  ▷  -end \
19814 ▷  ▷  -define F -finite_field -q 13 -end \
19815 ▷  ▷  -define gens -vector -file Paley_13_gens.csv -end \
19816 ▷  ▷  -define G -permutation_group \
19817 ▷  ▷  ▷  -bsgs Paley_13 "Paley\_13" 13 78 "0,1" 3 gens -end \
19818 ▷  ▷  -define Gamma -graph -Paley F -end \
19819 ▷  ▷  -define Orb -orbits -group G \
19820 ▷  ▷  ▷  -on_subsets 5 Control \
19821 ▷  ▷  -end
19822
19823 #User time: 0.01 of a second, dt=1 tps = 100
19824
19825
19826 Paley_13_cliques_all:
19827 ▷  $(ORBITER) -v 10 \
19828 ▷  ▷  -define F -finite_field -q 13 -end \
19829 ▷  ▷  -define Gamma -graph -Paley F -end \
19830 ▷  ▷  -with Gamma -do \
19831 ▷  ▷  -graph_theoretic_activity \
19832 ▷  ▷  ▷  -find_cliques -target_size 3 \
19833 ▷  ▷  -end
19834
19835
19836 tritangent_planes_graph_aut:
19837 ▷  $(ORBITER) -v 2 \
19838 ▷  ▷  -define Gamma \
19839 ▷  ▷  ▷  -graph -tritangent_planes_disjointness_graph \
19840 ▷  ▷  -end \
19841 ▷  ▷  -with Gamma -do \
19842 ▷  ▷  -graph_theoretic_activity \
```

```
19843 ▷  ▷  ▷  -automorphism_group \
19844 ▷  ▷  -end
19845 ▷  pdflatex tritangent_planes_disjointness_report.tex
19846 ▷  $(OPEN) tritangent_planes_disjointness_report.pdf
19847
19848 #tritangent_planes_disjointness_report.tex
19849 #tritangent_planes_disjointness_gens.csv
19850 #tritangent_planes_disjointness_group.makefile
19851
19852 tritangent_planes_graph_cliques_classify:
19853 ▷  $(ORBITER) -v 4 \
19854 ▷  ▷  -define Control -poset_classification_control \
19855 ▷  ▷  ▷  -W \
19856 ▷  ▷  ▷  -problem_label tri \
19857 ▷  ▷  ▷  -clique_test Gamma \
19858 ▷  ▷  ▷  -depth 9 \
19859 ▷  ▷  -end \
19860 ▷  ▷  -define gens -vector -file \
19861 ▷  ▷  ▷  tritangent_planes_disjointness_gens.csv \
19862 ▷  ▷  -end \
19863 ▷  ▷  -define G -permutation_group \
19864 ▷  ▷  -bsgs tritangent_planes_disjointness \
19865 ▷  ▷  ▷  "tritangent\_planes\_disjointness" 45 51840 \
19866 ▷  ▷  ▷  "0,3,12,24" 6 gens -end \
19867 ▷  ▷  -define Gamma \
19868 ▷  ▷  ▷  -graph -tritangent_planes_disjointness_graph \
19869 ▷  ▷  -end \
19870 ▷  ▷  -define Orb -orbits -group G \
19871 ▷  ▷  ▷  -on_subsets 9 Control \
19872 ▷  ▷  -end \
19873 ▷  ▷  -with Orb -do -orbits_activity \
19874 ▷  ▷  ▷  -recognize "4,9,11,14,21,26,30,35,40" \
19875 ▷  ▷  -end
19876
19877 #tri_reps_lvl_9.csv
19878
19879
19880 tritangent_planes_graph_cliques_all:
19881 ▷  $(ORBITER) -v 2 \
19882 ▷  ▷  -define Gamma \
19883 ▷  ▷  ▷  -graph -tritangent_planes_disjointness_graph \
19884 ▷  ▷  -end \
19885 ▷  ▷  -with Gamma -do \
19886 ▷  ▷  -graph_theoretic_activity \
19887 ▷  ▷  ▷  -find_cliques -target_size 9 \
19888 ▷  ▷  -end
19889
19890
19891 #tritangent_planes_disjointness_sol.csv
19892
19893
19894
19895 PGO_5_2_cliques: O_5_2_incidence_matrix.csv
19896 ▷  $(ORBITER) -v 3 \
19897 ▷  ▷  -define Inc -vector -file O_5_2_incidence_matrix.csv -end \
19898 ▷  ▷  -define Gamma -graph -collinearity_graph Inc -end \
19899 ▷  ▷  -with Gamma -do \
19900 ▷  ▷  -graph_theoretic_activity \
19901 ▷  ▷  ▷  -find_cliques -target_size 3 -end \
```

```
19902 ▷  ▷  -end
19903
19904
19905 HJ_d2_c5:
19906 ▷  $(ORBITER) -v 6 \
19907 ▷  ▷  -define G -graph \
19908 ▷  ▷  ▷  -load_csv_no_border \
19909 ▷  ▷  ▷  halljanko315.csv \
19910 ▷  ▷  ▷  -distance_2 \
19911 ▷  ▷  -end \
19912 ▷  ▷  -with G -do \
19913 ▷  ▷  -graph_theoretic_activity \
19914 ▷  ▷  ▷  -find_cliques -target_size 5 -end \
19915 ▷  ▷  -end
19916
19917
19918
19919 #graph_theoretic_activity::perform_activity Gr->label=halljanko315 nb_sol = 26208
      0
19920
19921
19922 HJ64_cliques5:
19923 ▷  $(ORBITER) -v 6 \
19924 ▷  ▷  -define Gamma -graph \
19925 ▷  ▷  ▷  -load \
19926 ▷  ▷  ▷  Group_Perm315_Orbital_3.colored_graph \
19927 ▷  ▷  -end \
19928 ▷  ▷  -with Gamma -do \
19929 ▷  ▷  -graph_theoretic_activity \
19930 ▷  ▷  ▷  -find_cliques -target_size 5 -end \
19931 ▷  ▷  -end
19932
19933 #graph_theoretic_activity::perform_activity Gr->label=Group_Perm315_Orbital_3 nb_
      sol = 1008
19934 #Group_Perm315_Orbital_3_sol.csv
19935
19936
19937
19938 HJ64_cliques5_classify:
19939 ▷  $(ORBITER) -v 6 \
19940 ▷  ▷  -define Control -poset_classification_control \
19941 ▷  ▷  ▷  -W \
19942 ▷  ▷  ▷  -problem_label HJ64_cliques \
19943 ▷  ▷  ▷  -clique_test Gamma \
19944 ▷  ▷  ▷  -depth 5 \
19945 ▷  ▷  -end \
19946 ▷  ▷  -define Gamma -graph \
19947 ▷  ▷  ▷  -load \
19948 ▷  ▷  ▷  Group_Perm315_Orbital_3.colored_graph \
19949 ▷  ▷  -end \
19950 ▷  ▷  -define gens -vector \
19951 ▷  ▷  ▷  -file halljanko315_gens.csv \
19952 ▷  ▷  -end \
19953 ▷  ▷  -define G -permutation_group \
19954 ▷  ▷  -bsgs halljanko315 "File\_halljanko315" \
19955 ▷  ▷  ▷  315 1209600 "0,1,42,95" 5 gens -end \
19956 ▷  ▷  -define Orb -orbits -group G \
19957 ▷  ▷  ▷  -on_subsets 5 Control \
19958 ▷  ▷  -end
```

```
19959
19960 #HJ64_cliques_reps_lvl_5.csv
19961
19962 # 1 orbit
19963 #ROW,REP,AGO,OL
19964 #0,"0,8,31,110,283",1200,1008
19965 #END
19966
19967
19968 ################################################################################
19969 # Section 15.2: Rainbow cliques
19970
19971
19972 SECTION_GRAPH_THEORY_RAINBOW_CLIQUES:
19973
19974
19975 test_15_2:
19976 ▷  make BLT_11_classify_starter
19977 ▷  make BLT_11_clique_0
19978 ▷  make BLT_11_clique_0_Sajeeb
19979 ▷  make BLT_11_0_sort
19980 ▷  make BLT_11_0_sorted_draw
19981
19982
19983 BLT_11_classify_starter:
19984 ▷  $(ORBITER) -v 2 \
19985 ▷  ▷  -define F -finite_field -q 11 -end \
19986 ▷  ▷  -define O -orthogonal_space 0 5 F -end \
19987 ▷  ▷  -define C -BLT_set_classifier O -starter_size 5 -end \
19988 ▷  ▷  -with C -do -BLT_set_classify_activity \
19989 ▷  ▷  ▷  -compute_starter \
19990 ▷  ▷  ▷  ▷  -problem_label BLT_q11 \
19991 ▷  ▷  ▷  ▷  -W -depth 5 \
19992 ▷  ▷  ▷  -end \
19993 ▷  ▷  -end \
19994 ▷  ▷  -with C -do -BLT_set_classify_activity \
19995 ▷  ▷  ▷  -create_graphs \
19996 ▷  ▷  -end
19997
19998 #NB_CASES_Q11=
19999
20000
20001 BLT_11_clique_0:
20002 ▷  $(ORBITER) -v 2 \
20003 ▷  ▷  -define G -graph \
20004 ▷  ▷  ▷  -load BLT_q11_graph_5_0.bin \
20005 ▷  ▷  -end \
20006 ▷  ▷  -with G -do \
20007 ▷  ▷  -graph_theoretic_activity \
20008 ▷  ▷  ▷  -find_cliques -rainbow -target_size 12 -end \
20009 ▷  ▷  -end
20010
20011 # nb_sol = 3
20012
20013 BLT_11_clique_0_Sajeeb:
20014 ▷  $(ORBITER) -v 2 \
20015 ▷  ▷  -define G -graph \
20016 ▷  ▷  ▷  -load BLT_q11_graph_5_0.bin \
20017 ▷  ▷  -end \
```

```
20018 ▷  ▷  -with G -do \
20019 ▷  ▷  -graph_theoretic_activity \
20020 ▷  ▷  ▷  -find_cliques -rainbow -Sajeeb -end \
20021 ▷  ▷  -end
20022
20023 # nb_sol = 3
20024
20025
20026 BLT_11_0_sort:
20027 ▷  $(ORBITER) -v 2 \
20028 ▷  ▷  -define G -graph \
20029 ▷  ▷  ▷  -load BLT_q11_graph_5_0.bin \
20030 ▷  ▷  -end \
20031 ▷  ▷  -with G -do \
20032 ▷  ▷  -graph_theoretic_activity \
20033 ▷  ▷  ▷  -sort_by_colors \
20034 ▷  ▷  -end
20035
20036 #BLT_q11_graph_5_0_sorted.bin
20037
20038
20039 BLT_11_0_sorted_draw:
20040 ▷  $(ORBITER) -v 2 \
20041 ▷  ▷  -draw_options \
20042 ▷  ▷  ▷  -radius 100 \
20043 ▷  ▷  ▷  -scale 0.4 \
20044 ▷  ▷  ▷  -line_width 1.0 -embedded \
20045 ▷  ▷  -end \
20046 ▷  ▷  -define G -graph \
20047 ▷  ▷  ▷  -load BLT_q11_graph_5_0_sorted.bin \
20048 ▷  ▷  -end \
20049 ▷  ▷  -with G -do \
20050 ▷  ▷  -graph_theoretic_activity \
20051 ▷  ▷  ▷  -draw \
20052 ▷  ▷  -end
20053 ▷  pdflatex BLT_q11_graph_5_0_sorted_draw.tex
20054 ▷  $(OPEN) BLT_q11_graph_5_0_sorted_draw.pdf
20055
20056
20057
20058
20059 ##############################################################################
20060 # Section 15.3: Diophantine Systems
20061
20062 SECTION_DIOPHANT:
20063
20064
20065 test_15_3:
20066 ▷  make part10
20067 ▷  make octic_monomials
20068 ▷  make solve_test_system
20069 ▷  make McKay_test
20070 ▷  make DLX_test
20071
20072
20073
20074 part10:
20075 ▷  $(ORBITER) -v 4 \
20076 ▷  ▷  -define A -vector -format 1 -dense "10,9,8,7,6,5,4,3,2,1" -end \
```

```
20077 ▷  ▷  -define D -diophant \
20078 ▷  ▷  ▷  -label part10 \
20079 ▷  ▷  ▷  -coefficient_matrix A \
20080 ▷  ▷  ▷  -RHS "mult=1,EQ=10" \
20081 ▷  ▷  ▷  -x_min_global 0 -x_max_global 10 \
20082 ▷  ▷  -end \
20083 ▷  ▷  -with D -do \
20084 ▷  ▷  ▷  -diophant_activity -solve_mckay \
20085 ▷  ▷  -end
20086 ▷  ▷
20087
20088
20089 ▷
20090 # Finds 42 solutions with 67 backtrack steps
20091
20092
20093 octic_monomials:
20094 ▷  $(ORBITER) -v 4 \
20095 ▷  ▷  -define A -vector -format 1 -dense "1,1,1,1" -end \
20096 ▷  ▷  -define D -diophant \
20097 ▷  ▷  ▷  -label octic_monomials \
20098 ▷  ▷  ▷  -coefficient_matrix A \
20099 ▷  ▷  ▷  -RHS "mult=1,EQ=9" \
20100 ▷  ▷  ▷  -x_min_global 0 -x_max_global 8 \
20101 ▷  ▷  -end \
20102 ▷  ▷  -with D -do \
20103 ▷  ▷  ▷  -diophant_activity -solve_mckay \
20104 ▷  ▷  -end
20105 ▷  sort -r octic_monomials_sol.csv >octic_monomials_sorted.csv
20106
20107 #Found 165 solutions with 210 backtrack steps
20108 # 165=binomial(11,3)
20109
20110
20111 solve_test_system:
20112 ▷  $(ORBITER) -v 4 \
20113 ▷  ▷  -define A -vector -format 7 -dense $(TEST_SYSTEM) -end \
20114 ▷  ▷  -define D -diophant \
20115 ▷  ▷  ▷  -label test_system \
20116 ▷  ▷  ▷  -coefficient_matrix A \
20117 ▷  ▷  ▷  -RHS "mult=7,EQ=1" \
20118 ▷  ▷  ▷  -x_min_global 0 -x_max_global 1 \
20119 ▷  ▷  -end
20120
20121
20122 McKay_test:
20123 ▷  $(ORBITER) -v 4 \
20124 ▷  ▷  -define A -vector -format 7 -dense $(TEST_SYSTEM) -end \
20125 ▷  ▷  -define D -diophant \
20126 ▷  ▷  ▷  -label test_system \
20127 ▷  ▷  ▷  -coefficient_matrix A \
20128 ▷  ▷  ▷  -RHS "mult=7,EQ=1" \
20129 ▷  ▷  ▷  -x_min_global 0 -x_max_global 1 \
20130 ▷  ▷  -end \
20131 ▷  ▷  -with D -do \
20132 ▷  ▷  ▷  -diophant_activity -solve_mckay \
20133 ▷  ▷  -end
20134
20135 DLX_test:
```

```
20136 ▷  $(ORBITER) -v 4 \
20137 ▷  ▷  -define A -vector -format 7 -dense $(TEST_SYSTEM) -end \
20138 ▷  ▷  -define D \
20139 ▷  ▷  -diophant -label test_system \
20140 ▷  ▷  ▷  -coefficient_matrix A \
20141 ▷  ▷  ▷  -RHS "mult=7,EQ=1" \
20142 ▷  ▷  ▷  -x_min_global 0 -x_max_global 1 \
20143 ▷  ▷  -end \
20144 ▷  ▷  -with D -do \
20145 ▷  ▷  ▷  -diophant_activity -solve_DLX \
20146 ▷  ▷  -end
20147
20148 #DLX_test.sol
20149 # 1 solution in 6 backtrack steps
20150
20151
20152
20153
20154 ##############################################################################
20155 # Chapter 16 - Canonical Forms with Nauty
20156 ##############################################################################
20157
20158
20159
20160 test_16:
20161 ▷  make test_16_2
20162 ▷  make test_16_3
20163 ▷  make test_16_4
20164 ▷  make test_16_6
20165 ▷  make test_16_7
20166 ▷  make test_16_8
20167
20168
20169
20170 ##############################################################################
20171 # Section 16.1: Overview of Canonical Forms
20172
20173
20174 SECTION_OVERVIEW_CANONICAL_FORMS:
20175
20176
20177
20178
20179
20180 ##############################################################################
20181 # Section 16.2: Objects in projective Space
20182
20183
20184 SECTION_OBJECTS_IN_PROJECTIVE_SPACE:
20185
20186
20187 test_16_2:
20188 ▷  make EC_canon
20189 ▷  make Hirschfeld_q4_c
20190 ▷  make Hirschfeld_stab_group
20191 ▷  make Hirschfeld_stab_subgroup
20192 ▷  make Hirschfeld_stab_orbits
20193 ▷  make Hirschfeld_q4_set_c
20194 ▷  make Dickson_sets_stabilizer
```

```
20195 ▷  make Endrass_7c
20196 ▷  make hyperoval_16_canonical_form
20197 ▷  make cubic_curves_PG_2_8_canon
20198 ▷  make ovoid_q8_canon
20199 ▷  make ovoid_ST_q8_canon
20200 ▷  make Suzuki_8
20201 ▷  make quartic_curve_25_0_0_canonical
20202 ▷  make CRC_BCH_Q16_N51_D3_canonical
20203 ▷  make Veronese_1_3_q7_c
20204
20205
20206 EC_canon: elliptic_curve_b1_c3_q11.txt
20207 ▷  $(ORBITER) -v 3 \
20208 ▷  ▷  -define C -combinatorial_object \
20209 ▷  ▷  ▷  -label EC EC \
20210 ▷  ▷  ▷  -file_of_points elliptic_curve_b1_c3_q11.txt \
20211 ▷  ▷  -end \
20212 ▷  ▷  -define F -finite_field -q 11 -end \
20213 ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
20214 ▷  ▷  -with C -do \
20215 ▷  ▷  -combinatorial_object_activity \
20216 ▷  ▷  ▷  -canonical_form_PG P \
20217 ▷  ▷  ▷  ▷  -save_ago \
20218 ▷  ▷  ▷  ▷  -save_transversal \
20219 ▷  ▷  ▷  ▷  -max_TDO_depth 400 \
20220 ▷  ▷  ▷  -end \
20221 ▷  ▷  -end \
20222 ▷  ▷  -with C -do \
20223 ▷  ▷  -combinatorial_object_activity \
20224 ▷  ▷  ▷  -report \
20225 ▷  ▷  ▷  ▷  -export_flag_orbits \
20226 ▷  ▷  ▷  ▷  -show_TDO \
20227 ▷  ▷  ▷  ▷  -show_TDA \
20228 ▷  ▷  ▷  ▷  -dont_show_incidence_matrices \
20229 ▷  ▷  ▷  ▷  -export_group_GAP \
20230 ▷  ▷  ▷  -end \
20231 ▷  ▷  -end
20232 ▷  pdflatex EC_classification.tex
20233 ▷  $(OPEN) EC_classification.pdf
20234 ▷  $(ORBITER) -v 2 -draw_matrix \
20235 ▷  ▷  -input_csv_file EC_object0_TDA_flag_orbits.csv \
20236 ▷  ▷  -secondary_input_csv_file EC_object0_TDA.csv \
20237 ▷  ▷  -box_width 20 -bit_depth 24 \
20238 ▷  ▷  -end
20239 ▷  $(OPEN) EC_object0_TDA_flag_orbits_draw.bmp
20240
20241
20242
20243
20244 Hirschfeld_q4_c:
20245 ▷  $(ORBITER) -v 6 \
20246 ▷  ▷  -define C -combinatorial_object \
20247 ▷  ▷  ▷  -label Hirschfeld_surface_q4 Hirschfeld\_surface\_q4 \
20248 ▷  ▷  ▷  -file_of_points Hirschfeld_surface_q4.txt \
20249 ▷  ▷  -end \
20250 ▷  ▷  -define F -finite_field -q 4 -end \
20251 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
20252 ▷  ▷  -with C -do \
20253 ▷  ▷  -combinatorial_object_activity \
```

```
20254 ▷  ▷  ▷  -canonical_form_PG P \
20255 ▷  ▷  ▷  ▷  -save_ago \
20256 ▷  ▷  ▷  ▷  -max_TDO_depth 10 \
20257 ▷  ▷  ▷  -end \
20258 ▷  ▷  -end \
20259 ▷  ▷  -with C -do \
20260 ▷  ▷  -combinatorial_object_activity \
20261 ▷  ▷  ▷  -report \
20262 ▷  ▷  ▷  ▷  -export_flag_orbits \
20263 ▷  ▷  ▷  ▷  -show_TDO \
20264 ▷  ▷  ▷  ▷  -show_TDA \
20265 ▷  ▷  ▷  ▷  -export_labels \
20266 ▷  ▷  ▷  ▷  -dont_show_incidence_matrices \
20267 ▷  ▷  ▷  ▷  -export_group_GAP \
20268 ▷  ▷  ▷  -end \
20269 ▷  ▷  -end
20270 ▷  pdflatex Hirschfeld_surface_q4_classification.tex
20271 ▷  $(OPEN) Hirschfeld_surface_q4_classification.pdf
20272
20273
20274 # group order is 51840
20275
20276
20277 Hirschfeld_stab_group:
20278 ▷  $(ORBITER) -v 9 \
20279 ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
20280 ▷  ▷  -define G -linear_group -PGGL 4 4 \
20281 ▷  ▷  ▷  -subgroup_by_generators "Hirschfeld_Stab" \
20282 ▷  ▷  ▷  51840 6 $(HIRSCHFELD_STAB_GENERATORS) \
20283 ▷  ▷  ▷  -end \
20284 ▷  ▷  -with G -do \
20285 ▷  ▷  -group_theoretic_activity \
20286 ▷  ▷  ▷  -report \
20287 ▷  ▷  -end
20288 ▷  pdflatex PGGL_4_4_Subgroup_Hirschfeld_Stab_51840_report.tex
20289 ▷  $(OPEN) PGGL_4_4_Subgroup_Hirschfeld_Stab_51840_report.pdf
20290
20291 Hirschfeld_stab_subgroup:
20292 ▷  $(ORBITER) -v 9 \
20293 ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
20294 ▷  ▷  -define G -linear_group -PGGL 4 4 \
20295 ▷  ▷  ▷  -subgroup_by_generators "Hirschfeld_Stab" \
20296 ▷  ▷  ▷  51840 6 $(HIRSCHFELD_STAB_GENERATORS) \
20297 ▷  ▷  ▷  -end \
20298 ▷  ▷  -define Gsp -modified_group -from G \
20299 ▷  ▷  ▷  -create_special_subgroup \
20300 ▷  ▷  -end \
20301 ▷  ▷  -with Gsp -do \
20302 ▷  ▷  -group_theoretic_activity \
20303 ▷  ▷  ▷  -report \
20304 ▷  ▷  -end \
20305 ▷  ▷  -define Orb -orbits -group Gsp \
20306 ▷  ▷  ▷  -on_points \
20307 ▷  ▷  -end
20308
20309 Hirschfeld_stab_orbits:
20310 ▷  $(ORBITER) -v 9 \
20311 ▷  ▷  -orbiter_path $(ORBITER_EXE_PATH) \
20312 ▷  ▷  -define G -linear_group -PGGL 4 4 \
```

```
20313 ▷  ▷  ▷  -subgroup_by_generators "Hirschfeld_Stab" \
20314 ▷  ▷  ▷  51840 6 $(HIRSCHFELD_STAB_GENERATORS) \
20315 ▷  ▷  ▷  -end \
20316 ▷  ▷  -with G -do \
20317 ▷  ▷  -group_theoretic_activity \
20318 ▷  ▷  ▷  -report \
20319 ▷  ▷  -end \
20320 ▷  ▷  -define Orb -orbits -group G \
20321 ▷  ▷  ▷  -on_points \
20322 ▷  ▷  -end \
20323 ▷  ▷  -with Orb -do -orbits_activity \
20324 ▷  ▷  ▷  -export_something "orbit" 0 \
20325 ▷  ▷  -end
20326
20327 #orbits_PGGL_4_4_Subgroup_Hirschfeld_Stab_51840_orbit_0.csv
20328
20329
20330 Hirschfeld_q4_set_c:
20331 ▷  $(ORBITER) -v 4 \
20332 ▷  ▷  -define H -set -here \
20333 ▷  ▷  ▷  $(HIRSCHFELD_SURFACE_Q4_SET_OF_POINTS) \
20334 ▷  ▷  -end \
20335 ▷  ▷  -define C -combinatorial_object \
20336 ▷  ▷  ▷  -label Hirschfeld_surface_q4 Hirschfeld\_surface\_q4 \
20337 ▷  ▷  ▷  -set_of_points H \
20338 ▷  ▷  -end \
20339 ▷  ▷  -define F -finite_field -q 4 -end \
20340 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
20341 ▷  ▷  -with C -do \
20342 ▷  ▷  -combinatorial_object_activity \
20343 ▷  ▷  ▷  -canonical_form_PG P \
20344 ▷  ▷  ▷  ▷  -save_ago \
20345 ▷  ▷  ▷  ▷  -max_TDO_depth 4 \
20346 ▷  ▷  ▷  -end \
20347 ▷  ▷  -end \
20348 ▷  ▷  -with C -do \
20349 ▷  ▷  -combinatorial_object_activity \
20350 ▷  ▷  ▷  -report \
20351 ▷  ▷  ▷  ▷  -show_TDO \
20352 ▷  ▷  ▷  ▷  -show_TDA \
20353 ▷  ▷  ▷  -end \
20354 ▷  ▷  -end
20355 ▷  pdflatex Hirschfeld_surface_q4_classification.tex
20356 ▷  $(OPEN) Hirschfeld_surface_q4_classification.pdf
20357
20358
20359
20360 Dickson_sets_stabilizer:
20361 ▷  $(ORBITER) -v 3 \
20362 ▷  ▷  -define C -combinatorial_object \
20363 ▷  ▷  ▷  -label Dickson_sets Dickson\_sets \
20364 ▷  ▷  ▷  -set_of_points "0,1,2,5,6" \
20365 ▷  ▷  ▷  -set_of_points "0,1,2,3,6" \
20366 ▷  ▷  ▷  -set_of_points "0,1,2,3,4" \
20367 ▷  ▷  ▷  -set_of_points "0,1,2,3,8" \
20368 ▷  ▷  ▷  -set_of_points "0,1,2,5,6,7,8" \
20369 ▷  ▷  ▷  -set_of_points "0,1,2,3,5,6,7" \
20370 ▷  ▷  ▷  -set_of_points "0,1,2,3,5,6,9" \
20371 ▷  ▷  ▷  -set_of_points "0,1,2,3,5,6,10" \
```

```
20372 ▷  ▷  ▷  -set_of_points "0,1,2,3,5,6,4" \
20373 ▷  ▷  ▷  -set_of_points "0,1,2,3,8,11,13" \
20374 ▷  ▷  ▷  -set_of_points "3,6,9,7,10,12,8,11,13,14,4" \
20375 ▷  ▷  ▷  -set_of_points "3,5,6,9,7,10,12,11,13,14,4" \
20376 ▷  ▷  ▷  -set_of_points "0,1,2,3,5,6,9,7,10,12,4" \
20377 ▷  ▷  -end \
20378 ▷  ▷  -define F -finite_field -q 2 -end \
20379 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
20380 ▷  ▷  -with C -do \
20381 ▷  ▷  -combinatorial_object_activity \
20382 ▷  ▷  ▷  -canonical_form_PG P \
20383 ▷  ▷  ▷  ▷  -save_ago \
20384 ▷  ▷  ▷  ▷  -max_TDO_depth 4 \
20385 ▷  ▷  ▷  -end \
20386 ▷  ▷  -end \
20387 ▷  ▷  -with C -do \
20388 ▷  ▷  -combinatorial_object_activity \
20389 ▷  ▷  ▷  -report \
20390 ▷  ▷  ▷  ▷  -show_TDO \
20391 ▷  ▷  ▷  ▷  -show_TDA \
20392 ▷  ▷  ▷  -end \
20393 ▷  ▷  -end
20394 ▷  pdflatex Dickson_sets_classification.tex
20395 ▷  $(OPEN) Dickson_sets_classification.pdf
20396
20397
20398
20399 Endrass_7c: Endrass_F7.txt
20400 ▷  $(ORBITER) -v 2 \
20401 ▷  ▷  -define C -combinatorial_object \
20402 ▷  ▷  ▷  -label Endrass_F7 Endrass\_F7 \
20403 ▷  ▷  ▷  -file_of_points Endrass_F7.txt \
20404 ▷  ▷  -end \
20405 ▷  ▷  -define F -finite_field -q 7 -end \
20406 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
20407 ▷  ▷  -with C -do \
20408 ▷  ▷  -combinatorial_object_activity \
20409 ▷  ▷  ▷  -canonical_form_PG P \
20410 ▷  ▷  ▷  ▷  -save_ago \
20411 ▷  ▷  ▷  ▷  -max_TDO_depth 4 \
20412 ▷  ▷  ▷  -end \
20413 ▷  ▷  -end \
20414 ▷  ▷  -with C -do \
20415 ▷  ▷  -combinatorial_object_activity \
20416 ▷  ▷  ▷  -report \
20417 ▷  ▷  ▷  ▷  -show_TDO \
20418 ▷  ▷  ▷  ▷  -show_TDA \
20419 ▷  ▷  ▷  -end \
20420 ▷  ▷  -end
20421 ▷  #pdflatex Endrass_F7_classification.tex
20422 ▷  #$(OPEN) Endrass_F7_classification.pdf
20423
20424 # ToDo latex file is too big
20425 # ToDo group order is computed wrong
20426 # should be: group order is 32
20427
20428
20429 hyperoval_16_canonical_form:
20430 ▷  $(ORBITER) -v 2 \
```

```
20431 ▷  ▷  -define C -combinatorial_object \
20432 ▷  ▷  ▷  -label hyperoval_q16 hyperoval\_q16 \
20433 ▷  ▷  ▷  -set_of_points $(HYPEROVAL_16_16320) \
20434 ▷  ▷  ▷  -set_of_points $(HYPEROVAL_16_144) \
20435 ▷  ▷  -end \
20436 ▷  ▷  -define F -finite_field -q 16 -end \
20437 ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
20438 ▷  ▷  -with C -do \
20439 ▷  ▷  -combinatorial_object_activity \
20440 ▷  ▷  ▷  -canonical_form_PG P \
20441 ▷  ▷  ▷  ▷  -save_ago \
20442 ▷  ▷  ▷  ▷  -save_transversal \
20443 ▷  ▷  ▷  ▷  -max_TDO_depth 10 \
20444 ▷  ▷  ▷  -end \
20445 ▷  ▷  -end \
20446 ▷  ▷  -with C -do \
20447 ▷  ▷  -combinatorial_object_activity \
20448 ▷  ▷  ▷  -report \
20449 ▷  ▷  ▷  ▷  -export_flag_orbits \
20450 ▷  ▷  ▷  ▷  -show_TDO \
20451 ▷  ▷  ▷  ▷  -show_TDA \
20452 ▷  ▷  ▷  ▷  -dont_show_incidence_matrices \
20453 ▷  ▷  ▷  ▷  -export_group_GAP \
20454 ▷  ▷  ▷  -end \
20455 ▷  ▷  -end
20456 ▷  pdflatex hyperoval_q16_classification.tex
20457 ▷  $(OPEN) hyperoval_q16_classification.pdf
20458 ▷  $(ORBITER) -v 2 -draw_matrix \
20459 ▷  ▷  -input_csv_file hyperoval_q16_object0_TDA_flag_orbits.csv \
20460 ▷  ▷  -secondary_input_csv_file hyperoval_q16_object0_TDA.csv \
20461 ▷  ▷  -box_width 4 -bit_depth 24 \
20462 ▷  ▷  -end
20463 ▷  $(OPEN) hyperoval_q16_object0_TDA_flag_orbits_draw.bmp
20464 ▷  $(ORBITER) -v 2 -draw_matrix \
20465 ▷  ▷  -input_csv_file hyperoval_q16_object1_TDA_flag_orbits.csv \
20466 ▷  ▷  -secondary_input_csv_file hyperoval_q16_object1_TDA.csv \
20467 ▷  ▷  -box_width 4 -bit_depth 24 \
20468 ▷  ▷  -end
20469 ▷  $(OPEN) hyperoval_q16_object1_TDA_flag_orbits_draw.bmp
20470
20471
20472
20473
20474
20475 cubic_curves_PG_2_8_canon:
20476 ▷  $(ORBITER) -v 6 \
20477 ▷  ▷  -define C -combinatorial_object \
20478 ▷  ▷  ▷  -label cc_8 cc\_8 \
20479 ▷  ▷  ▷  -set_of_points "2,3,28,46,51,61,40,71" \
20480 ▷  ▷  -end \
20481 ▷  ▷  -define F -finite_field -q 8 -end \
20482 ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
20483 ▷  ▷  -with C -do \
20484 ▷  ▷  -combinatorial_object_activity \
20485 ▷  ▷  ▷  -canonical_form_PG P \
20486 ▷  ▷  ▷  ▷  -save_ago \
20487 ▷  ▷  ▷  ▷  -max_TDO_depth 10 \
20488 ▷  ▷  ▷  -end \
20489 ▷  ▷  -end \
```

```
20490 ▷  ▷  -with C -do \
20491 ▷  ▷  -combinatorial_object_activity \
20492 ▷  ▷  ▷  -report \
20493 ▷  ▷  ▷  ▷  -export_flag_orbits \
20494 ▷  ▷  ▷  ▷  -show_TDO \
20495 ▷  ▷  ▷  ▷  -show_TDA \
20496 ▷  ▷  ▷  ▷  -dont_show_incidence_matrices \
20497 ▷  ▷  ▷  ▷  -export_group_GAP \
20498 ▷  ▷  ▷  -end \
20499 ▷  ▷  -end
20500 ▷  pdflatex cc_8_classification.tex
20501 ▷  $(OPEN) cc_8_classification.pdf
20502
20503
20504
20505
20506 ovoid_q8_canon: ovoid_q8.txt
20507 ▷  $(ORBITER) -v 6 \
20508 ▷  ▷  -define C -combinatorial_object \
20509 ▷  ▷  ▷  -label ovoid ovoid \
20510 ▷  ▷  ▷  -file_of_points ovoid_q8.txt \
20511 ▷  ▷  -end \
20512 ▷  ▷  -define F -finite_field -q 8 -end \
20513 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
20514 ▷  ▷  -with C -do \
20515 ▷  ▷  -combinatorial_object_activity \
20516 ▷  ▷  ▷  -canonical_form_PG P \
20517 ▷  ▷  ▷  ▷  -save_ago \
20518 ▷  ▷  ▷  ▷  -max_TDO_depth 4 \
20519 ▷  ▷  ▷  -end \
20520 ▷  ▷  -end \
20521 ▷  ▷  -with C -do \
20522 ▷  ▷  -combinatorial_object_activity \
20523 ▷  ▷  ▷  -report \
20524 ▷  ▷  ▷  ▷  -show_TDO \
20525 ▷  ▷  ▷  ▷  -show_TDA \
20526 ▷  ▷  ▷  ▷  -dont_show_incidence_matrices \
20527 ▷  ▷  ▷  ▷  -export_group_GAP \
20528 ▷  ▷  ▷  -end \
20529 ▷  ▷  -end
20530 ▷  #pdflatex ovoid_classification.tex
20531 ▷  #$(OPEN) ovoid_classification.pdf
20532
20533 #150 10k u 0 0 6 1 5 5 4 0 f 1 5 w 94 99.1% ▷
20534
20535
20536
20537
20538 # group order 1572480
20539
20540 ovoid_ST_q8_canon: ovoid_ST_q8.txt
20541 ▷  $(ORBITER) -v 6 \
20542 ▷  ▷  -define C -combinatorial_object \
20543 ▷  ▷  ▷  -label ovoid_ST ovoid\_ST \
20544 ▷  ▷  ▷  -file_of_points ovoid_ST_q8.txt \
20545 ▷  ▷  -end \
20546 ▷  ▷  -define F -finite_field -q 8 -end \
20547 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
20548 ▷  ▷  -with C -do \
```

```
20549 ▷  ▷  -combinatorial_object_activity \
20550 ▷  ▷  ▷  -canonical_form_PG P \
20551 ▷  ▷  ▷  ▷  -save_ago \
20552 ▷  ▷  ▷  ▷  -max_TDO_depth 4 \
20553 ▷  ▷  ▷  -end \
20554 ▷  ▷  -end \
20555 ▷  ▷  -with C -do \
20556 ▷  ▷  -combinatorial_object_activity \
20557 ▷  ▷  ▷  -report \
20558 ▷  ▷  ▷  ▷  -show_TDO \
20559 ▷  ▷  ▷  ▷  -show_TDA \
20560 ▷  ▷  ▷  ▷  -dont_show_incidence_matrices \
20561 ▷  ▷  ▷  ▷  -export_group_GAP \
20562 ▷  ▷  ▷  -end \
20563 ▷  ▷  -end
20564 ▷  #pdflatex ovoid_ST_classification.tex
20565 ▷  #$(OPEN) ovoid_ST_classification.pdf
20566
20567 # very slow
20568 #130+0810+0i7528pf+07.22 99.8%▷
20569
20570
20571
20572 Suzuki_8:
20573 ▷  $(ORBITER) -v 6 \
20574 ▷  ▷  -define F -finite_field -q 8 -end \
20575 ▷  ▷  -define gens -vector -field F \
20576 ▷  ▷  ▷  -compact $(SUZUKI_8_GENERATORS) -end \
20577 ▷  ▷  -define G -linear_group -PGGL 4 8 \
20578 ▷  ▷  -subgroup_by_generators "Sz8" "87360" 5 gens \
20579 ▷  ▷  ▷  -end \
20580 ▷  ▷  -with G -do \
20581 ▷  ▷  -group_theoretic_activity \
20582 ▷  ▷  ▷  -report \
20583 ▷  ▷  -end
20584 ▷  pdflatex PGGL_4_8_Subgroup_Sz8_87360_report.tex
20585 ▷  $(OPEN) PGGL_4_8_Subgroup_Sz8_87360_report.pdf
20586
20587
20588
20589 quartic_curve_25_0_0_canonical:
20590 ▷  $(ORBITER) -v 3 \
20591 ▷  ▷  -define F -finite_field -q 25 -end \
20592 ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
20593 ▷  ▷  -define C -combinatorial_object \
20594 ▷  ▷  ▷  -label quartic_25_0_0 quartic\_25\_0\_0 \
20595 ▷  ▷  ▷  -set_of_points "10,11,59,63,124,135,136,170,206,257,275,284,285,367,378,393
      ,433,619,641,644" \
20596 ▷  ▷  ▷  -set_of_points "9, 24, 62, 67, 77, 84, 87, 89, 125, 130, 158, 172, 197, 219
      , 266, 271, 325, 356, 391, 392, 400, 429, 454, 458, 470, 503, 531, 553, 605, 625,
       627, 646" \
20597 ▷  ▷  ▷  -set_of_points "9, 24, 55, 75, 79, 88, 93, 94, 112, 142, 186, 213, 230, 249
      , 316, 329, 337, 371, 381, 383, 392, 405, 417, 439, 464, 466, 470, 524, 530, 567,
       583, 594" \
20598 ▷  ▷  ▷  -set_of_points "10, 23, 34, 51, 65, 83, 117, 126, 144, 146, 147, 159, 172,
      181, 185, 197, 198, 207, 240, 281, 283, 293, 300, 301, 305, 346, 357, 384, 409, 4
      19, 444, 459, 463, 465, 468, 514, 533, 543, 547, 549, 586, 615, 618, 628" \
20599 ▷  ▷  ▷  -set_of_points "2, 12, 48, 65, 87, 120, 189, 246, 305, 323, 354, 375, 434,
      435, 455, 482, 496, 557, 586, 595" \
```

```
20600 ▷  ▷  -end \
20601 ▷  ▷  -with C -do \
20602 ▷  ▷  -combinatorial_object_activity \
20603 ▷  ▷  ▷  -canonical_form_PG P \
20604 ▷  ▷  ▷  ▷  -save_ago \
20605 ▷  ▷  ▷  ▷  -max_TDO_depth 4 \
20606 ▷  ▷  ▷  -end \
20607 ▷  ▷  -end \
20608 ▷  ▷  -with C -do \
20609 ▷  ▷  -combinatorial_object_activity \
20610 ▷  ▷  ▷  -report \
20611 ▷  ▷  ▷  ▷  -show_TDO \
20612 ▷  ▷  ▷  ▷  -show_TDA \
20613 ▷  ▷  ▷  ▷  -dont_show_incidence_matrices \
20614 ▷  ▷  ▷  ▷  -export_group_GAP \
20615 ▷  ▷  ▷  -end \
20616 ▷  ▷  -end
20617 ▷  pdflatex quartic_25_0_0_classification.tex
20618 ▷  $(OPEN) quartic_25_0_0_classification.pdf
20619
20620
20621
20622 CRC_BCH_Q16_N51_D3_canonical:
20623 ▷  $(ORBITER) -v 12 \
20624 ▷  ▷  -define F -finite_field -q 16 -override_polynomial 19 -end \
20625 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
20626 ▷  ▷  -define C -combinatorial_object \
20627 ▷  ▷  ▷  -label BCH_Q16_N51 BCH\_Q16\_N51 \
20628 ▷  ▷  ▷  -set_of_points $(CRC_BCH_Q16_N51_D3_PROJECTIVE_SET) \
20629 ▷  ▷  -end \
20630 ▷  ▷  -with C -do \
20631 ▷  ▷  -combinatorial_object_activity \
20632 ▷  ▷  ▷  -canonical_form_PG P \
20633 ▷  ▷  ▷  ▷  -save_ago \
20634 ▷  ▷  ▷  ▷  -max_TDO_depth 4 \
20635 ▷  ▷  ▷  -end \
20636 ▷  ▷  -end \
20637 ▷  ▷  -with C -do \
20638 ▷  ▷  -combinatorial_object_activity \
20639 ▷  ▷  ▷  -report \
20640 ▷  ▷  ▷  ▷  -show_TDO \
20641 ▷  ▷  ▷  ▷  -show_TDA \
20642 ▷  ▷  ▷  ▷  -dont_show_incidence_matrices \
20643 ▷  ▷  ▷  ▷  -export_group_GAP \
20644 ▷  ▷  ▷  -end \
20645 ▷  ▷  -end
20646 ▷  pdflatex BCH_Q16_N51_classification.tex
20647 ▷  $(OPEN) BCH_Q16_N51_classification.pdf
20648
20649
20650 # stabilizer of order 408 is generated by:
20651 #1,0,0,0,0,0,1,0,1,5,9,3,12,10,8,10,1, 0,1,0,0,0,0,1,0,0,0,0,1,1,2,13,5,0,
20652
20653
20654 Veronese_1_3_q7_c:
20655 ▷  $(ORBITER) -v 6 \
20656 ▷  ▷  -define C -combinatorial_object \
20657 ▷  ▷  ▷  -label veronese_1_3_q7 veronese\_1\_3\_q7 \
20658 ▷  ▷  ▷  -set_of_points "3, 0, 4, 268, 336, 184, 224, 364" \
```

```
20659 ▷  ▷  -end \
20660 ▷  ▷  -define F -finite_field -q 7 -end \
20661 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
20662 ▷  ▷  -with C -do \
20663 ▷  ▷  -combinatorial_object_activity \
20664 ▷  ▷  ▷  -canonical_form_PG P \
20665 ▷  ▷  ▷  ▷  -save_ago \
20666 ▷  ▷  ▷  ▷  -max_TDO_depth 4 \
20667 ▷  ▷  ▷  -end \
20668 ▷  ▷  -end \
20669 ▷  ▷  -with C -do \
20670 ▷  ▷  -combinatorial_object_activity \
20671 ▷  ▷  ▷  -report \
20672 ▷  ▷  ▷  ▷  -show_TDO \
20673 ▷  ▷  ▷  ▷  -show_TDA \
20674 ▷  ▷  ▷  ▷  -dont_show_incidence_matrices \
20675 ▷  ▷  ▷  ▷  -export_group_GAP \
20676 ▷  ▷  ▷  -end \
20677 ▷  ▷  -end
20678 ▷  pdflatex veronese_1_3_q7_classification.tex
20679 ▷  $(OPEN) veronese_1_3_q7_classification.pdf
20680
20681
20682 ################################################################################
20683 # Section 16.3: Incidence Geometries
20684
20685
20686 SECTION_INCIDENCE_GEOMETRIES:
20687
20688
20689
20690 test_16_3:
20691 ▷  make geo_7_3_c
20692 ▷  make geo_10_3_c
20693 ▷  make geo_10_3_c_lex_least
20694 ▷  #make geo_14_3_c
20695 ▷  #make geo_15_3_c
20696 ▷  make TFC_24_3_c
20697 ▷  #make geo_40_4_g4_c
20698 ▷  #make geo_17_3_g4_c
20699 ▷  make AG_2_3_c
20700 ▷  make geo_LSQ6_c
20701
20702
20703 geo_7_3_c:
20704 ▷  $(ORBITER) -v 10 \
20705 ▷  ▷  -draw_incidence_structure_description \
20706 ▷  ▷  ▷  -width 60 -width_10 6 -end \
20707 ▷  ▷  -define C -combinatorial_object \
20708 ▷  ▷  ▷  -label geo_7_3 geo\_7\_3 \
20709 ▷  ▷  ▷  -file_of_incidence_geometries 7_3.inc 7 7 21 \
20710 ▷  ▷  -end \
20711 ▷  ▷  -with C -do \
20712 ▷  ▷  -combinatorial_object_activity \
20713 ▷  ▷  ▷  -canonical_form \
20714 ▷  ▷  ▷  ▷  -save_ago \
20715 ▷  ▷  ▷  ▷  -save_transversal \
20716 ▷  ▷  ▷  -end \
20717 ▷  ▷  -end \
```

```
20718 ▷  ▷  -with C -do \
20719 ▷  ▷  -combinatorial_object_activity \
20720 ▷  ▷  ▷  -report \
20721 ▷  ▷  ▷  ▷  -export_flag_orbits \
20722 ▷  ▷  ▷  ▷  -show_incidence_matrices \
20723 ▷  ▷  ▷  ▷  -export_group_GAP \
20724 ▷  ▷  ▷  -end \
20725 ▷  ▷  -end
20726 ▷  pdflatex 7_3_classification.tex
20727 ▷  $(OPEN) 7_3_classification.pdf
20728 ▷  $(ORBITER) -v 2 -draw_matrix \
20729 ▷  ▷  -input_csv_file 7_3_object0_TDA_flag_orbits.csv \
20730 ▷  ▷  -secondary_input_csv_file 7_3_object0_TDA.csv \
20731 ▷  ▷  -box_width 32 -bit_depth 24 \
20732 ▷  ▷  -end
20733 ▷  $(ORBITER) -v 2 -draw_matrix \
20734 ▷  ▷  -input_csv_file 7_3_object0_INP_flag_orbits.csv \
20735 ▷  ▷  -secondary_input_csv_file 7_3_object0_INP.csv \
20736 ▷  ▷  -box_width 32 -bit_depth 24 \
20737 ▷  ▷  -end
20738 ▷  $(OPEN) 7_3_object0_INP_flag_orbits_draw.bmp
20739
20740
20741
20742 geo_10_3_c:
20743 ▷  $(ORBITER) -v 10 \
20744 ▷  ▷  -draw_incidence_structure_description \
20745 ▷  ▷  ▷  -width 60 -width_10 6 -end \
20746 ▷  ▷  -define C -combinatorial_object \
20747 ▷  ▷  ▷  -label geo_10_3 geo\_10\_3 \
20748 ▷  ▷  ▷  -file_of_incidence_geometries 10_3.inc 10 10 30 \
20749 ▷  ▷  -end \
20750 ▷  ▷  -with C -do \
20751 ▷  ▷  -combinatorial_object_activity \
20752 ▷  ▷  ▷  -canonical_form \
20753 ▷  ▷  ▷  ▷  -save_ago \
20754 ▷  ▷  ▷  ▷  -save_transversal \
20755 ▷  ▷  ▷  -end \
20756 ▷  ▷  -end \
20757 ▷  ▷  -with C -do \
20758 ▷  ▷  -combinatorial_object_activity \
20759 ▷  ▷  ▷  -report \
20760 ▷  ▷  ▷  ▷  -export_flag_orbits \
20761 ▷  ▷  ▷  ▷  -show_incidence_matrices \
20762 ▷  ▷  ▷  ▷  -export_group_GAP \
20763 ▷  ▷  ▷  -end \
20764 ▷  ▷  -end
20765 ▷  pdflatex 10_3_classification.tex
20766 ▷  $(OPEN) 10_3_classification.pdf
20767 ▷  $(ORBITER) -v 2 -draw_matrix \
20768 ▷  ▷  -input_csv_file 10_3_object7_TDA_flag_orbits.csv \
20769 ▷  ▷  -secondary_input_csv_file 10_3_object7_TDA.csv \
20770 ▷  ▷  -box_width 16 -bit_depth 24 \
20771 ▷  ▷  -end
20772 ▷  $(ORBITER) -v 2 -draw_matrix \
20773 ▷  ▷  -input_csv_file 10_3_object7_INP_flag_orbits.csv \
20774 ▷  ▷  -secondary_input_csv_file 10_3_object7_INP.csv \
20775 ▷  ▷  -box_width 16 -bit_depth 24 \
20776 ▷  ▷  -end
```

```
20777
20778
20779
20780
20781
20782 geo_10_3_c_lex_least:
20783 ▷  $(ORBITER) -v 10 \
20784 ▷  ▷  -draw_incidence_structure_description \
20785 ▷  ▷  ▷  -width 60 -width_10 6 -end \
20786 ▷  ▷  -define Test_lines -set -loop 4 11 1 -end \
20787 ▷  ▷  -define Geo -geometry_builder \
20788 ▷  ▷  ▷  -V 10 -B 10 -TDO 3 -fuse 1 \
20789 ▷  ▷  ▷  -fname_GEO 10_3 \
20790 ▷  ▷  ▷  -test Test_lines \
20791 ▷  ▷  -end \
20792 ▷  ▷  -define C -combinatorial_object \
20793 ▷  ▷  ▷  -label geo_10_3 geo\_10\_3 \
20794 ▷  ▷  ▷  -file_of_incidence_geometries 10_3.inc 10 10 30 \
20795 ▷  ▷  -end \
20796 ▷  ▷  -with C -do \
20797 ▷  ▷  -combinatorial_object_activity \
20798 ▷  ▷  ▷  -canonical_form \
20799 ▷  ▷  ▷  ▷  -save_ago \
20800 ▷  ▷  ▷  ▷  -save_transversal \
20801 ▷  ▷  ▷  -end \
20802 ▷  ▷  -end \
20803 ▷  ▷  -with C -do \
20804 ▷  ▷  -combinatorial_object_activity \
20805 ▷  ▷  ▷  -report \
20806 ▷  ▷  ▷  ▷  -export_flag_orbits \
20807 ▷  ▷  ▷  ▷  -show_incidence_matrices \
20808 ▷  ▷  ▷  ▷  -export_group_GAP \
20809 ▷  ▷  ▷  ▷  -show_TDO \
20810 ▷  ▷  ▷  ▷  -show_TDA \
20811 ▷  ▷  ▷  ▷  -lex_least Geo \
20812 ▷  ▷  ▷  -end \
20813 ▷  ▷  -end
20814 ▷  pdflatex geo_10_3_classification.tex
20815 ▷  $(OPEN) geo_10_3_classification.pdf
20816 ▷  $(ORBITER) -v 2 -draw_matrix \
20817 ▷  ▷  -input_csv_file geo_10_3_object7_TDA_flag_orbits.csv \
20818 ▷  ▷  -secondary_input_csv_file geo_10_3_object7_TDA.csv \
20819 ▷  ▷  -box_width 16 -bit_depth 24 \
20820 ▷  ▷  -end
20821 ▷  $(ORBITER) -v 2 -draw_matrix \
20822 ▷  ▷  -input_csv_file geo_10_3_object7_INP_flag_orbits.csv \
20823 ▷  ▷  -secondary_input_csv_file geo_10_3_object7_INP.csv \
20824 ▷  ▷  -box_width 16 -bit_depth 24 \
20825 ▷  ▷  -end
20826
20827 #10_3_object7_TDA_flag_orbits.csv
20828
20829 #0 1 2 10 13 14 20 25 26 31 33 35 41 44 46 52 53 56 62 67 68 74 77 79 85 88 89
20830
20831
20832
20833 geo_14_3_c:
20834 ▷  $(ORBITER) -v 2 \
20835 ▷  ▷  -draw_incidence_structure_description \
```

```
20836 ▷  ▷  ▷  -width 60 -width_10 6 -end \
20837 ▷  ▷  -define Test_lines -set -loop 4 15 1 -end \
20838 ▷  ▷  -define C -combinatorial_object \
20839 ▷  ▷  ▷  -label geo_14_3 geo\_14\_3 \
20840 ▷  ▷  ▷  -file_of_incidence_geometries 14_3.inc 14 14 42 \
20841 ▷  ▷  -end \
20842 ▷  ▷  -with C -do \
20843 ▷  ▷  -combinatorial_object_activity \
20844 ▷  ▷  ▷  -canonical_form \
20845 ▷  ▷  ▷  ▷  -save_ago \
20846 ▷  ▷  ▷  ▷  -save_transversal \
20847 ▷  ▷  ▷  -end \
20848 ▷  ▷  -end
20849
20850
20851 geo_15_3_c:
20852 ▷  $(ORBITER) -v 2 \
20853 ▷  ▷  -draw_incidence_structure_description \
20854 ▷  ▷  ▷  -width 50 -width_10 5 -end \
20855 ▷  ▷  -define C -combinatorial_object \
20856 ▷  ▷  ▷  -label geo_15_3 geo\_15\_3 \
20857 ▷  ▷  ▷  -file_of_incidence_geometries 15_3.inc 15 15 45 \
20858 ▷  ▷  -end \
20859 ▷  ▷  -with C -do \
20860 ▷  ▷  -combinatorial_object_activity \
20861 ▷  ▷  ▷  -canonical_form \
20862 ▷  ▷  ▷  -save_ago \
20863 ▷  ▷  -end
20864 ▷  pdflatex 15_3_classification.tex
20865 ▷  $(OPEN) 15_3_classification.pdf
20866
20867 TFC_24_3_c:
20868 ▷  echo $(FILE_24_3_TFC_INC) >24_3_TFC.inc
20869 ▷  $(ORBITER) -v 20 \
20870 ▷  ▷  -define C -combinatorial_object \
20871 ▷  ▷  ▷  -label 24_3_TFC 24\_3\_TFC \
20872 ▷  ▷  ▷  -file_of_incidence_geometries 24_3_TFC.inc 24 24 72 \
20873 ▷  ▷  -end \
20874 ▷  ▷  -with C -do \
20875 ▷  ▷  -combinatorial_object_activity \
20876 ▷  ▷  ▷  -canonical_form \
20877 ▷  ▷  ▷  -save_ago \
20878 ▷  ▷  ▷  -end \
20879 ▷  ▷  -end \
20880 ▷  ▷  -with C -do \
20881 ▷  ▷  -combinatorial_object_activity \
20882 ▷  ▷  ▷  -report \
20883 ▷  ▷  ▷  ▷  -export_flag_orbits \
20884 ▷  ▷  ▷  ▷  -show_TDO \
20885 ▷  ▷  ▷  ▷  -show_TDA \
20886 ▷  ▷  ▷  ▷  -show_incidence_matrices \
20887 ▷  ▷  ▷  -end \
20888 ▷  ▷  -end
20889 ▷  pdflatex 24_3_TFC_classification.tex
20890 ▷  $(OPEN) 24_3_TFC_classification.pdf
20891 ▷  $(ORBITER) -v 2 -draw_matrix \
20892 ▷  ▷  -input_csv_file 24_3_TFC_object2_TDA_flag_orbits.csv \
20893 ▷  ▷  -secondary_input_csv_file 24_3_TFC_object2_TDA.csv \
20894 ▷  ▷  -box_width 40 -bit_depth 24 \
```

```
20895 ▷  ▷  -end
20896 ▷  $(OPEN) 24_3_TFC_object2_TDA_flag_orbits_draw.bmp
20897
20898
20899 geo_40_4_g4_c:
20900 ▷  $(ORBITER) -v 2 \
20901 ▷  ▷  -draw_incidence_structure_description \
20902 ▷  ▷  ▷  -width 50 -width_10 5 -end \
20903 ▷  ▷  -define C -combinatorial_object \
20904 ▷  ▷  ▷  -label 40_4_g4 40\_4\_g4 \
20905 ▷  ▷  ▷  -file_of_incidence_geometries \
20906 ▷  ▷  ▷  ▷  40_4_g4.inc 40 40 160 \
20907 ▷  ▷  -end \
20908 ▷  ▷  -with C -do \
20909 ▷  ▷  -combinatorial_object_activity \
20910 ▷  ▷  ▷  -canonical_form \
20911 ▷  ▷  ▷  ▷  -save_ago \
20912 ▷  ▷  ▷  -end \
20913 ▷  ▷  -end \
20914 ▷  ▷  -with C -do \
20915 ▷  ▷  -combinatorial_object_activity \
20916 ▷  ▷  ▷  -report \
20917 ▷  ▷  ▷  ▷  -export_flag_orbits \
20918 ▷  ▷  ▷  ▷  -show_TDO \
20919 ▷  ▷  ▷  ▷  -show_TDA \
20920 ▷  ▷  ▷  ▷  -show_incidence_matrices \
20921 ▷  ▷  ▷  -end \
20922 ▷  ▷  -end
20923 ▷  $(ORBITER) -v 2 \
20924 ▷  ▷  -loop L 0 2 1 \
20925 ▷  ▷  ▷  -draw_matrix \
20926 ▷  ▷  ▷  -input_csv_file 40_4_g4_object%L_TDA_flag_orbits.csv \
20927 ▷  ▷  ▷  -secondary_input_csv_file 40_4_g4_object%L_TDA.csv \
20928 ▷  ▷  ▷  -box_width 32 -bit_depth 24 \
20929 ▷  ▷  ▷  -end \
20930 ▷  ▷  ▷  -system "convert \
20931 ▷  ▷  ▷  ▷  40_4_g4_object%L_TDA_flag_orbits_draw.bmp \
20932 ▷  ▷  ▷  ▷  40_4_g4_object%L_TDA_flag_orbits_draw.png" \
20933 ▷  ▷  -end_loop L
20934 ▷  pdflatex 40_4_g4_classification.tex
20935 ▷  $(OPEN) 40_4_g4_classification.pdf
20936
20937 geo_17_3_g4_c:
20938 ▷  $(ORBITER) -v 2 \
20939 ▷  ▷  -draw_incidence_structure_description \
20940 ▷  ▷  ▷  -width 50 -width_10 5 -end \
20941 ▷  ▷  -define C -combinatorial_object \
20942 ▷  ▷  ▷  -label 17_3_g4 17\_3\_g4 \
20943 ▷  ▷  ▷  -file_of_incidence_geometries 17_3_g4.inc 17 17 51 \
20944 ▷  ▷  -end \
20945 ▷  ▷  -with C -do \
20946 ▷  ▷  -combinatorial_object_activity \
20947 ▷  ▷  ▷  -canonical_form \
20948 ▷  ▷  ▷  ▷  -save_ago \
20949 ▷  ▷  ▷  -end \
20950 ▷  ▷  -end \
20951 ▷  ▷  -with C -do \
20952 ▷  ▷  -combinatorial_object_activity \
20953 ▷  ▷  ▷  -report \
```

```
20954 ▷  ▷  ▷  ▷  -export_flag_orbits \
20955 ▷  ▷  ▷  ▷  -show_TDO \
20956 ▷  ▷  ▷  ▷  -show_TDA \
20957 ▷  ▷  ▷  ▷  -show_incidence_matrices \
20958 ▷  ▷  ▷  -end \
20959 ▷  ▷  -end
20960 ▷  pdflatex 17_3_g4_classification.tex
20961 ▷  $(OPEN) 17_3_g4_classification.pdf
20962
20963
20964 AG_2_3_c: AG_2_3.inc
20965 ▷  $(ORBITER) -v 2 \
20966 ▷  ▷  -define C -combinatorial_object \
20967 ▷  ▷  ▷  -label AG_2_3 AG\_2\_3 \
20968 ▷  ▷  ▷  -file_of_incidence_geometries \
20969 ▷  ▷  ▷  AG_2_3.inc 9 12 36 \
20970 ▷  ▷  -end \
20971 ▷  ▷  -with C -do \
20972 ▷  ▷  -combinatorial_object_activity \
20973 ▷  ▷  ▷  -canonical_form \
20974 ▷  ▷  ▷  ▷  -save_ago \
20975 ▷  ▷  ▷  ▷  -max_TDO_depth 10 \
20976 ▷  ▷  ▷  -end \
20977 ▷  ▷  -end \
20978 ▷  ▷  -with C -do \
20979 ▷  ▷  -combinatorial_object_activity \
20980 ▷  ▷  ▷  -report \
20981 ▷  ▷  ▷  ▷  -export_flag_orbits \
20982 ▷  ▷  ▷  ▷  -show_TDO \
20983 ▷  ▷  ▷  ▷  -show_TDA \
20984 ▷  ▷  ▷  ▷  -show_incidence_matrices \
20985 ▷  ▷  ▷  -end \
20986 ▷  ▷  -end
20987 ▷  pdflatex AG_2_3_classification.tex
20988 ▷  $(OPEN) AG_2_3_classification.pdf
20989 ▷  $(ORBITER) -v 2 -draw_matrix \
20990 ▷  ▷  -input_csv_file AG_2_3_object0_INP_flag_orbits.csv \
20991 ▷  ▷  -secondary_input_csv_file AG_2_3_object0_INP.csv \
20992 ▷  ▷  -box_width 40 -bit_depth 24 \
20993 ▷  ▷  -end
20994 ▷  $(OPEN) AG_2_3_object0_INP_flag_orbits_draw.bmp
20995
20996
20997
20998
20999 geo_LSQ6_c: LSQ6.inc
21000 ▷  $(ORBITER) -v 3 \
21001 ▷  ▷  -draw_incidence_structure_description \
21002 ▷  ▷  ▷  -width 60 -width_10 6 -end \
21003 ▷  ▷  -define C -combinatorial_object \
21004 ▷  ▷  ▷  -label LSQ6 LSQ6 \
21005 ▷  ▷  ▷  -file_of_incidence_geometries \
21006 ▷  ▷  ▷  ▷  LSQ6.inc 18 39 126 \
21007 ▷  ▷  -end \
21008 ▷  ▷  -with C -do \
21009 ▷  ▷  -combinatorial_object_activity \
21010 ▷  ▷  ▷  -canonical_form \
21011 ▷  ▷  ▷  ▷  -save_ago \
21012 ▷  ▷  ▷  ▷  -save_transversal \
```

```
21013 ▷  ▷  ▷  -end \
21014 ▷  ▷  -end \
21015 ▷  ▷  -with C -do \
21016 ▷  ▷  -combinatorial_object_activity \
21017 ▷  ▷  ▷  -report \
21018 ▷  ▷  ▷  ▷  -export_flag_orbits \
21019 ▷  ▷  ▷  ▷  -show_incidence_matrices \
21020 ▷  ▷  ▷  ▷  -export_group_GAP \
21021 ▷  ▷  ▷  -end \
21022 ▷  ▷  -end
21023 ▷  pdflatex LSQ6_classification.tex
21024 ▷  $(OPEN) LSQ6_classification.pdf
21025 ▷  $(ORBITER) -v 2 \
21026 ▷  ▷  -loop L 0 12 1 \
21027 ▷  ▷  ▷  -draw_matrix \
21028 ▷  ▷  ▷  -input_csv_file LSQ6_object%L_TDA_flag_orbits.csv \
21029 ▷  ▷  ▷  -secondary_input_csv_file LSQ6_object%L_TDA.csv \
21030 ▷  ▷  ▷  -box_width 32 -bit_depth 24 \
21031 ▷  ▷  ▷  -end \
21032 ▷  ▷  ▷  -system "convert \
21033 ▷  ▷  ▷  ▷  LSQ6_object%L_TDA_flag_orbits_draw.bmp \
21034 ▷  ▷  ▷  ▷  LSQ6_object%L_TDA_flag_orbits_draw.png" \
21035 ▷  ▷  -end_loop L
21036 ▷  $(ORBITER) -v 2 \
21037 ▷  ▷  -loop L 0 12 1 \
21038 ▷  ▷  ▷  -draw_matrix \
21039 ▷  ▷  ▷  -input_csv_file LSQ6_object%L_INP_flag_orbits.csv \
21040 ▷  ▷  ▷  -secondary_input_csv_file LSQ6_object%L_INP.csv \
21041 ▷  ▷  ▷  -box_width 32 -bit_depth 24 \
21042 ▷  ▷  ▷  -end \
21043 ▷  ▷  -end_loop L
21044
21045
21046
21047
21048
21049
21050
21051
21052
21053
21054
21055
21056
21057 ###############################################################################
21058 # Section 16.4: Objects from Design Theory
21059
21060
21061 SECTION_OBJECTS_FROM_DESIGN_THEORY:
21062
21063
21064 test_16_4:
21065 ▷  make LS_AG_2_3_solutions_classify
21066 ▷  make design_27c
21067 ▷  make design_PG_2_3_canonical
21068 ▷  make wreath_product_designs_n8_k6_c
21069 ▷  make wreath_product_designs_n4_k2_c
21070
21071
```

```
21072
21073 LS_AG_2_3_solutions_classify:
21074 ▷  $(ORBITER) -v 2 \
21075 ▷  ▷  -draw_incidence_structure_description \
21076 ▷  ▷  ▷  -width 20 -width_10 2 -end \
21077 ▷  ▷  -define C -combinatorial_object \
21078 ▷  ▷  ▷  -label LS_AG_2_3 LS\_AG\_2\_3 \
21079 ▷  ▷  ▷  -file_of_designs \
21080 ▷  ▷  ▷  solutions.csv 9 84 3 12 \
21081 ▷  ▷  -end \
21082 ▷  ▷  -with C -do \
21083 ▷  ▷  -combinatorial_object_activity \
21084 ▷  ▷  ▷  -canonical_form \
21085 ▷  ▷  ▷  ▷  -save_ago \
21086 ▷  ▷  ▷  ▷  -save_transversal \
21087 ▷  ▷  ▷  ▷  -max_TDO_depth 10 \
21088 ▷  ▷  ▷  -end \
21089 ▷  ▷  -end \
21090 ▷  ▷  -with C -do \
21091 ▷  ▷  -combinatorial_object_activity \
21092 ▷  ▷  ▷  -report \
21093 ▷  ▷  ▷  ▷  -export_flag_orbits \
21094 ▷  ▷  ▷  ▷  -show_TDO \
21095 ▷  ▷  ▷  -end \
21096 ▷  ▷  -end
21097 ▷  pdflatex LS_AG_2_3_classification.tex
21098 ▷  $(OPEN) LS_AG_2_3_classification.pdf
21099 ▷  $(ORBITER) -v 2 -draw_matrix \
21100 ▷  ▷  -input_csv_file LS_AG_2_3_object0_INP_flag_orbits.csv \
21101 ▷  ▷  -secondary_input_csv_file LS_AG_2_3_object0_INP.csv \
21102 ▷  ▷  -box_width 12 -bit_depth 24 \
21103 ▷  ▷  -end
21104 ▷  $(OPEN) LS_AG_2_3_object0_INP_flag_orbits_draw.bmp
21105 ▷  $(ORBITER) -v 2 -draw_matrix \
21106 ▷  ▷  -input_csv_file LS_AG_2_3_object1_INP_flag_orbits.csv \
21107 ▷  ▷  -secondary_input_csv_file LS_AG_2_3_object1_INP.csv \
21108 ▷  ▷  -box_width 12 -bit_depth 24 \
21109 ▷  ▷  -end
21110 ▷  $(OPEN) LS_AG_2_3_object1_INP_flag_orbits_draw.bmp
21111
21112
21113
21114
21115
21116 design_27c:
21117 ▷  $(ORBITER) -v 4 \
21118 ▷  ▷  -define C -combinatorial_object \
21119 ▷  ▷  ▷  -label design design \
21120 ▷  ▷  ▷  -set_of_points "2,56,30,112,253,90,440,508" \
21121 ▷  ▷  -end \
21122 ▷  ▷  -define F -finite_field -q 27 \
21123 ▷  ▷  ▷  -override_polynomial 46 \
21124 ▷  ▷  -end \
21125 ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
21126 ▷  ▷  -with C -do \
21127 ▷  ▷  -combinatorial_object_activity \
21128 ▷  ▷  ▷  -canonical_form_PG P \
21129 ▷  ▷  ▷  -end \
21130 ▷  ▷  -end \
```

```
21131 ▷  ▷  -with C -do \
21132 ▷  ▷  -combinatorial_object_activity \
21133 ▷  ▷  ▷  -report \
21134 ▷  ▷  -end
21135 ▷  pdflatex design_classification.tex
21136 ▷  $(OPEN) design_classification.pdf
21137
21138
21139
21140 design_PG_2_3_canonical:
21141 ▷  $(ORBITER) -v 3 \
21142 ▷  ▷  -define F -finite_field -q 3 -end \
21143 ▷  ▷  -define D -design -field F -family PG_2_q -end \
21144 ▷  ▷  -with D -do \
21145 ▷  ▷  ▷  -design_activity \
21146 ▷  ▷  ▷  ▷  -export_inc \
21147 ▷  ▷  ▷  -end \
21148 ▷  ▷  -end
21149 ▷  $(ORBITER) -v 3 \
21150 ▷  ▷  -draw_incidence_structure_description \
21151 ▷  ▷  ▷  -width 60 -width_10 6 -end \
21152 ▷  ▷  -define C -combinatorial_object \
21153 ▷  ▷  ▷  -label PG_2_3 PG\_2\_3 \
21154 ▷  ▷  ▷  -file_of_incidence_geometries \
21155 ▷  ▷  ▷  ▷  PG_2_3_inc.txt 13 13 52 \
21156 ▷  ▷  -end \
21157 ▷  ▷  -with C -do \
21158 ▷  ▷  -combinatorial_object_activity \
21159 ▷  ▷  ▷  -canonical_form \
21160 ▷  ▷  ▷  ▷  -save_ago \
21161 ▷  ▷  ▷  ▷  -save_transversal \
21162 ▷  ▷  ▷  -end \
21163 ▷  ▷  -end \
21164 ▷  ▷  -with C -do \
21165 ▷  ▷  -combinatorial_object_activity \
21166 ▷  ▷  ▷  -report \
21167 ▷  ▷  ▷  ▷  -export_flag_orbits \
21168 ▷  ▷  ▷  ▷  -show_incidence_matrices \
21169 ▷  ▷  ▷  ▷  -export_group_GAP \
21170 ▷  ▷  ▷  -end \
21171 ▷  ▷  -end
21172 ▷  pdflatex PG_2_3_classification.tex
21173 ▷  $(OPEN) PG_2_3_classification.pdf
21174 ▷  $(ORBITER) -v 2 -draw_matrix \
21175 ▷  ▷  -input_csv_file PG_2_3_object0_TDA_flag_orbits.csv \
21176 ▷  ▷  -secondary_input_csv_file PG_2_3_object0_TDA.csv \
21177 ▷  ▷  -box_width 32 -bit_depth 24 \
21178 ▷  ▷  -end
21179 ▷  $(OPEN) PG_2_3_object0_TDA_flag_orbits_draw.bmp
21180 ▷
21181
21182
21183 wreath_product_designs_n4_k2_c: wreath_product_designs_n4_k2_inc.txt
21184 ▷  $(ORBITER) -v 10 \
21185 ▷  ▷  -draw_incidence_structure_description \
21186 ▷  ▷  ▷  -width 60 -width_10 6 -end \
21187 ▷  ▷  -define C -combinatorial_object \
21188 ▷  ▷  ▷  -label wreath_4_2 wreath\_4\_2 \
21189 ▷  ▷  ▷  -file_of_incidence_geometries \
```

```
21190 ▷  ▷  ▷  wreath_product_designs_n4_k2_inc.txt \
21191 ▷  ▷  ▷  8 12 24 \
21192 ▷  ▷  -end \
21193 ▷  ▷  -with C -do \
21194 ▷  ▷  -combinatorial_object_activity \
21195 ▷  ▷  ▷  -canonical_form \
21196 ▷  ▷  ▷  ▷  -save_ago \
21197 ▷  ▷  ▷  ▷  -save_transversal \
21198 ▷  ▷  ▷  -end \
21199 ▷  ▷  -end \
21200 ▷  ▷  -with C -do \
21201 ▷  ▷  -combinatorial_object_activity \
21202 ▷  ▷  ▷  -report \
21203 ▷  ▷  ▷  ▷  -export_flag_orbits \
21204 ▷  ▷  ▷  ▷  -show_incidence_matrices \
21205 ▷  ▷  ▷  ▷  -export_group_GAP \
21206 ▷  ▷  ▷  -end \
21207 ▷  ▷  -end
21208 ▷  pdflatex wreath_4_2_classification.tex
21209 ▷  $(OPEN) wreath_4_2_classification.pdf
21210
21211
21212 wreath_product_designs_n8_k6_c: wreath_product_designs_n8_k6_inc.txt
21213 ▷  $(ORBITER) -v 10 \
21214 ▷  ▷  -draw_incidence_structure_description \
21215 ▷  ▷  ▷  -width 60 -width_10 6 -end \
21216 ▷  ▷  -define C -combinatorial_object \
21217 ▷  ▷  ▷  -label wreath_8_6 wreath\_8\_6 \
21218 ▷  ▷  ▷  -file_of_incidence_geometries \
21219 ▷  ▷  ▷  wreath_product_designs_n8_k6_inc.txt \
21220 ▷  ▷  ▷  16 3920 23520 \
21221 ▷  ▷  -end \
21222 ▷  ▷  -with C -do \
21223 ▷  ▷  -combinatorial_object_activity \
21224 ▷  ▷  ▷  -canonical_form \
21225 ▷  ▷  ▷  ▷  -save_ago \
21226 ▷  ▷  ▷  ▷  -save_transversal \
21227 ▷  ▷  ▷  -end \
21228 ▷  ▷  -end \
21229 ▷  ▷  -with C -do \
21230 ▷  ▷  -combinatorial_object_activity \
21231 ▷  ▷  ▷  -report \
21232 ▷  ▷  ▷  ▷  -export_flag_orbits \
21233 ▷  ▷  ▷  ▷  -export_group_GAP \
21234 ▷  ▷  ▷  -end \
21235 ▷  ▷  -end
21236 ▷  pdflatex wreath_8_6_classification.tex
21237 ▷  $(OPEN) wreath_8_6_classification.pdf
21238
21239
21240
21241
21242
21243
21244 ###############################################################################
21245 # Section 16.5: Linear Codes
21246
21247
21248 SECTION_CANONICAL_FORMS_OF_LINEAR_CODES:
```

```
21249
21250
21251 test_16_5:
21252 ▷  make code_3_2_aut
21253 ▷  make code_6_3_aut
21254 ▷  make RM_3_1_group
21255 ▷  make RM_3_1_group_and_diagram
21256 ▷  make RM_4_1_group
21257 ▷  make RS_6_4_7_group
21258 ▷  make GV_n15_k6_d5_group
21259 ▷  make code_n15_k6_d6_a_group
21260 ▷  make code_n15_k6_d6_b_group
21261
21262
21263
21264 code_3_2_aut:
21265 ▷  $(ORBITER) -v 2 \
21266 ▷  ▷  -define F -finite_field -q 2 -end \
21267 ▷  ▷  -define genma -vector -field F -format 2 \
21268 ▷  ▷  ▷  -dense $(CODE_N3_K2_Q2_GENMA) \
21269 ▷  ▷  -end \
21270 ▷  ▷  -define P -projective_space -n 1 -field F -v 0 -end \
21271 ▷  ▷  -with P -do \
21272 ▷  ▷  -projective_space_activity \
21273 ▷  ▷  ▷  -canonical_form_of_code \
21274 ▷  ▷  ▷  ▷  "3_2" genma -save_ago -label "3_2" \
21275 ▷  ▷  ▷  ▷  -classification_prefix "3_2" \
21276 ▷  ▷  ▷  -end \
21277 ▷  ▷  -end
21278 ▷  pdflatex 3_2_classification.tex
21279 ▷  $(OPEN) 3_2_classification.pdf
21280 ▷  $(ORBITER) -v 2 -draw_matrix \
21281 ▷  ▷  -input_csv_file 3_2_object0_TDA_flag_orbits.csv \
21282 ▷  ▷  -secondary_input_csv_file 3_2_object0_TDA.csv \
21283 ▷  ▷  -box_width 16 -bit_depth 24 \
21284 ▷  ▷  -end
21285 ▷  $(OPEN) 3_2_object0_TDA_flag_orbits_draw.bmp
21286
21287
21288
21289
21290 code_6_3_aut:
21291 ▷  $(ORBITER) -v 2 \
21292 ▷  ▷  -define F -finite_field -q 2 -end \
21293 ▷  ▷  -define genma -vector -field F -format 3 \
21294 ▷  ▷  ▷  -compact $(CODE_N6_K3_Q2_GENMA) \
21295 ▷  ▷  -end \
21296 ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
21297 ▷  ▷  -with P -do \
21298 ▷  ▷  -projective_space_activity \
21299 ▷  ▷  ▷  -canonical_form_of_code \
21300 ▷  ▷  ▷  ▷  "6_3" genma -save_ago -label "6_3" \
21301 ▷  ▷  ▷  ▷  -classification_prefix "6_3" \
21302 ▷  ▷  ▷  -end \
21303 ▷  ▷  -end
21304 ▷  pdflatex 6_3_classification.tex
21305 ▷  $(OPEN) 6_3_classification.pdf
21306 ▷  $(ORBITER) -v 2 -draw_matrix \
21307 ▷  ▷  -input_csv_file 6_3_object0_TDA_flag_orbits.csv \
```

```
21308 ▷  ▷  -secondary_input_csv_file 6_3_object0_TDA.csv \
21309 ▷  ▷  -box_width 16 -bit_depth 24 \
21310 ▷  ▷  -end
21311 ▷  $(OPEN) 6_3_object0_TDA_flag_orbits_draw.bmp
21312
21313 # group of order 24
21314
21315
21316 RM_3_1_group:
21317 ▷  $(ORBITER) -v 2 \
21318 ▷  ▷  -define F -finite_field -q 2 -end \
21319 ▷  ▷  -define genma -vector -field F -format 4 \
21320 ▷  ▷  ▷  -compact $(CODE_RM_3_1_GENMA) \
21321 ▷  ▷  -end \
21322 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
21323 ▷  ▷  -with P -do \
21324 ▷  ▷  -projective_space_activity \
21325 ▷  ▷  ▷  -canonical_form_of_code \
21326 ▷  ▷  ▷  ▷  "RM_3_1" genma -save_ago -label "RM_3_1" \
21327 ▷  ▷  ▷  ▷  -classification_prefix "RM_3_1" \
21328 ▷  ▷  ▷  -end \
21329 ▷  ▷  -end
21330 ▷  pdflatex RM_3_1_classification.tex
21331 ▷  $(OPEN) RM_3_1_classification.pdf
21332
21333 # group order 1344
21334 #RM_3_1_object0_INP_flag_orbits.csv
21335
21336 RM_3_1_group_and_diagram:
21337 ▷  $(ORBITER) -v 2 \
21338 ▷  ▷  -define F -finite_field -q 2 -end \
21339 ▷  ▷  -define genma -vector -field F -format 4 \
21340 ▷  ▷  ▷  -compact $(CODE_RM_3_1_GENMA) \
21341 ▷  ▷  -end \
21342 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
21343 ▷  ▷  -with P -do \
21344 ▷  ▷  -projective_space_activity \
21345 ▷  ▷  ▷  -canonical_form_of_code \
21346 ▷  ▷  ▷  ▷  "RM_3_1" genma -save_ago -label "RM_3_1" \
21347 ▷  ▷  ▷  ▷  -classification_prefix "RM_3_1" \
21348 ▷  ▷  ▷  -end \
21349 ▷  ▷  -end
21350 ▷  pdflatex RM_3_1_classification.tex
21351 ▷  $(OPEN) RM_3_1_classification.pdf
21352 ▷  $(ORBITER) -v 2 -draw_matrix \
21353 ▷  ▷  -input_csv_file RM_3_1_object0_INP_flag_orbits.csv \
21354 ▷  ▷  -secondary_input_csv_file RM_3_1_object0_INP.csv \
21355 ▷  ▷  -box_width 16 -bit_depth 24 \
21356 ▷  ▷  -end
21357 ▷  $(ORBITER) -v 2 -draw_matrix \
21358 ▷  ▷  -input_csv_file RM_3_1_object0_TDA_flag_orbits.csv \
21359 ▷  ▷  -secondary_input_csv_file RM_3_1_object0_TDA.csv \
21360 ▷  ▷  -box_width 16 -bit_depth 24 \
21361 ▷  ▷  -end
21362 ▷  $(OPEN) RM_3_1_object0_INP_flag_orbits_draw.bmp
21363 ▷  $(OPEN) RM_3_1_object0_TDA_flag_orbits_draw.bmp
21364
21365
21366
```

```
21367 RM_4_1_group:
21368 ▷  $(ORBITER) -v 2 \
21369 ▷  ▷  -define F -finite_field -q 2 -end \
21370 ▷  ▷  -define genma -vector -field F -format 5 \
21371 ▷  ▷  ▷  -compact $(CODE_RM_4_1_GENMA) \
21372 ▷  ▷  -end \
21373 ▷  ▷  -define P -projective_space -n 4 -field F -v 0 -end \
21374 ▷  ▷  -with P -do \
21375 ▷  ▷  -projective_space_activity \
21376 ▷  ▷  ▷  -canonical_form_of_code \
21377 ▷  ▷  ▷  ▷  "RM_4_1" genma -save_ago -label "RM_4_1" \
21378 ▷  ▷  ▷  ▷  -classification_prefix "RM_4_1" \
21379 ▷  ▷  ▷  -end \
21380 ▷  ▷  -end
21381 ▷  pdflatex RM_4_1_classification.tex
21382 ▷  $(OPEN) RM_4_1_classification.pdf
21383 ▷  $(ORBITER) -v 2 -draw_matrix \
21384 ▷  ▷  -input_csv_file RM_4_1_object0_INP_flag_orbits.csv \
21385 ▷  ▷  -secondary_input_csv_file RM_4_1_object0_INP.csv \
21386 ▷  ▷  -box_width 16 -bit_depth 24 \
21387 ▷  ▷  -end
21388 ▷  $(ORBITER) -v 2 -draw_matrix \
21389 ▷  ▷  -input_csv_file RM_4_1_object0_TDA_flag_orbits.csv \
21390 ▷  ▷  -secondary_input_csv_file RM_4_1_object0_TDA.csv \
21391 ▷  ▷  -box_width 16 -bit_depth 24 \
21392 ▷  ▷  -end
21393 ▷  $(OPEN) RM_4_1_object0_INP_flag_orbits_draw.bmp
21394 ▷  $(OPEN) RM_4_1_object0_TDA_flag_orbits_draw.bmp
21395
21396
21397 # group order 322560 = 24*30*28*16
21398
21399
21400 RS_6_4_7_group:
21401 ▷  $(ORBITER) -v 2 \
21402 ▷  ▷  -define F -finite_field -q 7 -end \
21403 ▷  ▷  -define genma -vector -field F -format 4 \
21404 ▷  ▷  ▷  -compact $(CODE_RS_6_4_7) \
21405 ▷  ▷  -end \
21406 ▷  ▷  -define P -projective_space -n 3 -field F -v 0 -end \
21407 ▷  ▷  -with P -do \
21408 ▷  ▷  -projective_space_activity \
21409 ▷  ▷  ▷  -canonical_form_of_code \
21410 ▷  ▷  ▷  ▷  "RS_6" genma -save_ago -label "RS_6" \
21411 ▷  ▷  ▷  ▷  -classification_prefix "RS_6" \
21412 ▷  ▷  ▷  -end \
21413 ▷  ▷  -end
21414
21415
21416 GV_n15_k6_d5_group:
21417 ▷  $(ORBITER) -v 2 \
21418 ▷  ▷  -define F -finite_field -q 2 -end \
21419 ▷  ▷  -define genma -vector -field F -format 6 \
21420 ▷  ▷  ▷  -compact $(CODE_GV_N15_K6) \
21421 ▷  ▷  -end \
21422 ▷  ▷  -define P -projective_space -n 5 -field F -v 0 -end \
21423 ▷  ▷  -with P -do \
21424 ▷  ▷  -projective_space_activity \
21425 ▷  ▷  ▷  -canonical_form_of_code \
```

```
21426 ▷  ▷  ▷  ▷  "GV_n15_k6_d5" genma -save_ago -label "GV_n15_k6_d5" \
21427 ▷  ▷  ▷  ▷  -classification_prefix "GV_n15_k6_d5" \
21428 ▷  ▷  ▷  -end \
21429 ▷  ▷  -end
21430 ▷  #pdflatex GV_n15_k6_d5_classification.tex
21431 ▷  #$(OPEN) GV_n15_k6_d5_classification.pdf
21432
21433 # ToDo latex file too large
21434 #ago=12
21435
21436
21437
21438 code_n15_k6_d6_a_group:
21439 ▷  $(ORBITER) -v 2 \
21440 ▷  ▷  -define F -finite_field -q 2 -end \
21441 ▷  ▷  -define genma -vector -field F -format 6 \
21442 ▷  ▷  ▷  -compact $(CODE_15_6_6_A) \
21443 ▷  ▷  -end \
21444 ▷  ▷  -define P -projective_space -n 5 -field F -v 0 -end \
21445 ▷  ▷  -with P -do \
21446 ▷  ▷  -projective_space_activity \
21447 ▷  ▷  ▷  -canonical_form_of_code \
21448 ▷  ▷  ▷  ▷  "n15_k6_d6_a" genma -save_ago -label "n15_k6_d6_a" \
21449 ▷  ▷  ▷  ▷  -classification_prefix "n15_k6_d6_a" \
21450 ▷  ▷  ▷  -end \
21451 ▷  ▷  -end
21452 ▷  #pdflatex n15_k6_d6_a_classification.tex
21453 ▷  #$(OPEN) n15_k6_d6_a_classification.pdf
21454
21455
21456 code_n15_k6_d6_b_group:
21457 ▷  $(ORBITER) -v 2 \
21458 ▷  ▷  -define F -finite_field -q 2 -end \
21459 ▷  ▷  -define genma -vector -field F -format 6 \
21460 ▷  ▷  ▷  -compact $(CODE_15_6_6_B) \
21461 ▷  ▷  -end \
21462 ▷  ▷  -define P -projective_space -n 5 -field F -v 0 -end \
21463 ▷  ▷  -with P -do \
21464 ▷  ▷  -projective_space_activity \
21465 ▷  ▷  ▷  -canonical_form_of_code \
21466 ▷  ▷  ▷  ▷  "n15_k6_d6_b" genma -save_ago -label "n15_k6_d6_b" \
21467 ▷  ▷  ▷  ▷  -classification_prefix "n15_k6_d6_b" \
21468 ▷  ▷  ▷  -end \
21469 ▷  ▷  -end
21470 ▷  #pdflatex n15_k6_d6_b_classification.tex
21471 ▷  #$(OPEN) n15_k6_d6_b_classification.pdf
21472
21473
21474
21475
21476 ##############################################################################
21477 # Section 16.6: General Codes
21478
21479
21480 SECTION_CANONICAL_FORMS_OF_GENERAL_CODES:
21481
21482 test_16_6:
21483 ▷  make Hamming_graph_7_with_Hamming_code
21484
```

```
21485
21486
21487 Hamming_graph_7_with_Hamming_code:
21488 ▷  $(ORBITER) -v 2 \
21489 ▷  ▷  -define G -graph -Hamming 7 2 \
21490 ▷  ▷  ▷  -subset "_Hamming_code" "\\_with\\_Hamming\\_code" \
21491 ▷  ▷  ▷  $(HAMMING_CODE_CODEWORDS) -end \
21492 ▷  ▷  -with G -do \
21493 ▷  ▷  -graph_theoretic_activity -export_csv -end \
21494 ▷  ▷  -with G -do \
21495 ▷  ▷  -graph_theoretic_activity -export_graphviz -end \
21496 ▷  ▷  -with G -do \
21497 ▷  ▷  -graph_theoretic_activity -save -end \
21498 ▷  ▷  -with G -do \
21499 ▷  ▷  -graph_theoretic_activity -automorphism_group -end
21500 ▷  pdflatex Hamming_7_2_Hamming_code_report.tex
21501 ▷  $(OPEN) Hamming_7_2_Hamming_code_report.pdf
21502
21503 # group of order 2688 = 16 * 168
21504
21505
21506
21507
21508 ##############################################################################
21509 # Section 16.7: Graphs
21510
21511
21512 SECTION_CANONICAL_FORMS_OF_GRAPHS:
21513
21514
21515 test_16_7:
21516 ▷  make Cycle_13_aut
21517 ▷  make Queens_graph_aut
21518 ▷  make Queens_graph_aut_orbits_draw
21519 ▷  make inversion_graph
21520 ▷  make Chain_232_aut
21521 ▷  #make JK_graph_pp16_1
21522 ▷  #make JK_graph_pp16_2
21523 ▷  #make JK_graph_pp16_9
21524 ▷  #make JK_graph_grid_3_3
21525 ▷  #make JK_graph_sts_13
21526 ▷  make halljanko315_gens.csv
21527 ▷  make HJ_group_and_orbits
21528 ▷  make HJ_orbital_graph_3
21529 ▷  make PGO_5_2_graph_group
21530
21531
21532 Cycle_13_aut:
21533 ▷  $(ORBITER) -v 2 \
21534 ▷  ▷  -define Gamma -graph -cycle 13 -end \
21535 ▷  ▷  -with Gamma -do \
21536 ▷  ▷  ▷  -graph_theoretic_activity -automorphism_group \
21537 ▷  ▷  -end \
21538
21539
21540 Queens_graph_aut:
21541 ▷  $(ORBITER) -v 2 \
21542 ▷  ▷  -define Gamma -graph -non_attacking_queens_graph 8 -end \
21543 ▷  ▷  -with Gamma -do \
```

```
21544 ▷  ▷  ▷  -graph_theoretic_activity -automorphism_group \
21545 ▷  ▷  -end
21546 ▷  pdflatex non_attacking_queens_graph_8_report.tex
21547 ▷  $(OPEN) non_attacking_queens_graph_8_report.pdf
21548
21549 #non_attacking_queens_graph_8_orbits.csv
21550
21551
21552 Queens_graph_aut_orbits_draw:
21553 ▷  $(ORBITER) -v 2 -reformat \
21554 ▷  ▷  non_attacking_queens_graph_8_orbits.csv \
21555 ▷  ▷  Queens_orbits_on_points_8x8.csv 8
21556 ▷  $(ORBITER) -v 2 \
21557 ▷  ▷  -define all_one_r -vector -repeat 1 8 -end \
21558 ▷  ▷  -define all_one_c -vector -repeat 1 8 -end \
21559 ▷  ▷  -draw_matrix \
21560 ▷  ▷  -input_csv_file Queens_orbits_on_points_8x8.csv \
21561 ▷  ▷  -box_width 40 -bit_depth 24 \
21562 ▷  ▷  ▷  -partition 1 \
21563 ▷  ▷  ▷  ▷  all_one_r all_one_c \
21564 ▷  ▷  -end
21565 ▷  $(OPEN) Queens_orbits_on_points_8x8_draw.bmp
21566
21567
21568
21569 inversion_graph:
21570 ▷  $(ORBITER) -v 6 \
21571 ▷  ▷  -define G -graph \
21572 ▷  ▷  ▷  -inversion_graph "1,0,2,3" \
21573 ▷  ▷  -end \
21574 ▷  ▷  -with G -do \
21575 ▷  ▷  ▷  -graph_theoretic_activity -properties \
21576 ▷  ▷  -end \
21577 ▷  ▷  -with G -do \
21578 ▷  ▷  ▷  -graph_theoretic_activity -automorphism_group \
21579 ▷  ▷  -end
21580
21581
21582
21583 Chain_232_aut:
21584 ▷  $(ORBITER) -v 2 \
21585 ▷  ▷  -define P1 -vector -dense 2,3,2 -end \
21586 ▷  ▷  -define P2 -vector -dense 2,3,2 -end \
21587 ▷  ▷  -define Gamma -graph \
21588 ▷  ▷  ▷  -chain_graph P1 P2 \
21589 ▷  ▷  -end \
21590 ▷  ▷  -with Gamma -do \
21591 ▷  ▷  ▷  -graph_theoretic_activity -automorphism_group \
21592 ▷  ▷  -end
21593 ▷  pdflatex chain_graph_report.tex
21594 ▷  $(OPEN) chain_graph_report.pdf
21595
21596
21597
21598
21599 JK_graph_pp16_1:
21600 ▷  $(ORBITER) -v 2 \
21601 ▷  ▷  -define Gamma -graph -load_dimacs \
21602 ▷  ▷  ../JUNTTILA_KASKI/benchmarks/pp/pp16-1 \
```

```
21603 ▷  ▷  -end \
21604 ▷  ▷  -with Gamma -do \
21605 ▷  ▷  -graph_theoretic_activity -save -end \
21606 ▷  ▷  -with Gamma -do \
21607 ▷  ▷  -graph_theoretic_activity -automorphism_group -end \
21608
21609 # go=34217164800
21610 #nauty_interface_with_group::create_automorphism_group_of_graph_with_partition_an
      d_labeling: nb_backtrack1 = 6
21611 #nauty_interface_with_group::create_automorphism_group_of_graph_with_partition_an
      d_labeling: nb_backtrack2 = 134
21612
21613 JK_graph_pp16_2:
21614 ▷  $(ORBITER) -v 2 \
21615 ▷  ▷  -define Gamma -graph -load_dimacs \
21616 ▷  ▷  ../JUNTTILA_KASKI/benchmarks/pp/pp16-2 \
21617 ▷  ▷  -end \
21618 ▷  ▷  -with Gamma -do \
21619 ▷  ▷  -graph_theoretic_activity -save -end \
21620 ▷  ▷  -with Gamma -do \
21621 ▷  ▷  -graph_theoretic_activity -automorphism_group -end \
21622
21623 # does not finish
21624
21625 JK_graph_pp16_9:
21626 ▷  $(ORBITER) -v 2 \
21627 ▷  ▷  -define Gamma -graph -load_dimacs \
21628 ▷  ▷  ../JUNTTILA_KASKI/benchmarks/pp/pp16-9 \
21629 ▷  ▷  -end \
21630 ▷  ▷  -with Gamma -do \
21631 ▷  ▷  -graph_theoretic_activity -save -end \
21632 ▷  ▷  -with Gamma -do \
21633 ▷  ▷  -graph_theoretic_activity -automorphism_group -end \
21634
21635
21636 JK_graph_grid_3_3:
21637 ▷  $(ORBITER) -v 2 \
21638 ▷  ▷  -define Gamma -graph -load_dimacs \
21639 ▷  ▷  ../JUNTTILA_KASKI/benchmarks/grid/grid-w-3-3 \
21640 ▷  ▷  -end \
21641 ▷  ▷  -with Gamma -do \
21642 ▷  ▷  -graph_theoretic_activity -save -end \
21643 ▷  ▷  -with Gamma -do \
21644 ▷  ▷  -graph_theoretic_activity -automorphism_group -end \
21645
21646
21647
21648
21649 JK_graph_sts_13:
21650 ▷  $(ORBITER) -v 2 \
21651 ▷  ▷  -define Gamma -graph -load_dimacs \
21652 ▷  ▷  ▷  ../JUNTTILA_KASKI/benchmarks/srg/sts-13 \
21653 ▷  ▷  -end \
21654 ▷  ▷  -with Gamma -do \
21655 ▷  ▷  -graph_theoretic_activity -save -end \
21656 ▷  ▷  -with Gamma -do \
21657 ▷  ▷  -graph_theoretic_activity -automorphism_group -end
21658 ▷  make ORBITER_PATH=$(ORBITER_EXE_PATH) -f sts-13_group.makefile sts-13
21659
```

```
21660
21661
21662 halljanko315_gens.csv:
21663 ▷  $(ORBITER) -v 6 \
21664 ▷  ▷  -define G -graph \
21665 ▷  ▷  ▷  -load_csv_no_border \
21666 ▷  ▷  ▷  halljanko315.csv \
21667 ▷  ▷  -end \
21668 ▷  ▷  -with G -do \
21669 ▷  ▷  ▷  -graph_theoretic_activity -automorphism_group \
21670 ▷  ▷  -end \
21671 ▷  ▷  -with G -do \
21672 ▷  ▷  ▷  -graph_theoretic_activity -properties \
21673 ▷  ▷  -end
21674
21675
21676
21677 HJ_group_and_orbits:
21678 ▷  $(ORBITER) -v 2 \
21679 ▷  ▷  -define Control -poset_classification_control \
21680 ▷  ▷  ▷  -W \
21681 ▷  ▷  ▷  -problem_label HJ_orbits \
21682 ▷  ▷  ▷  -depth 2 \
21683 ▷  ▷  -end \
21684 ▷  ▷  -define gens -vector -file \
21685 ▷  ▷  ▷  halljanko315_gens.csv -end \
21686 ▷  ▷  -define G -permutation_group \
21687 ▷  ▷  ▷  -bsgs halljanko315 "File\_halljanko315" \
21688 ▷  ▷  ▷  315 1209600 "0,1,2" 6 gens \
21689 ▷  ▷  -end \
21690 ▷  ▷  -define Orb -orbits -group G \
21691 ▷  ▷  ▷  -on_subsets 2 Control \
21692 ▷  ▷  -end
21693
21694 #ROW,REP,AGO,OL
21695 #0,"0,1",96,12600
21696 #1,"0,2",48,25200
21697 #2,"0,4",768,1575
21698 #3,"0,8",120,10080
21699 #END
21700
21701 HJ_orbital_graph_3:
21702 ▷  $(ORBITER) -v 2 \
21703 ▷  ▷  -define gens -vector -file \
21704 ▷  ▷  ▷  halljanko315_gens.csv -end \
21705 ▷  ▷  -define G -permutation_group \
21706 ▷  ▷  ▷  -bsgs halljanko315 "File\_halljanko315" \
21707 ▷  ▷  ▷  315 1209600 "0,1,2" 6 gens \
21708 ▷  ▷  -end \
21709 ▷  ▷  -define Gamma -graph \
21710 ▷  ▷  ▷  -orbital_graph G 3 \
21711 ▷  ▷  -end \
21712 ▷  ▷  -with Gamma -do \
21713 ▷  ▷  ▷  -graph_theoretic_activity \
21714 ▷  ▷  ▷  -properties \
21715 ▷  ▷  -end \
21716 ▷  ▷  -with Gamma -do \
21717 ▷  ▷  ▷  -graph_theoretic_activity \
21718 ▷  ▷  ▷  -save \
```

```
21719 ▷  ▷  -end
21720
21721 #Group_Perm315_Orbital_3.colored_graph
21722 #Degree type: (64^{315} )
21723
21724
21725
21726
21727
21728 PGO_5_2_graph_group: O_5_2_incidence_matrix.csv
21729 ▷  $(ORBITER) -v 3 \
21730 ▷  ▷  -define Inc -vector -file O_5_2_incidence_matrix.csv -end \
21731 ▷  ▷  -define Gamma -graph -collinearity_graph Inc -end \
21732 ▷  ▷  -with Gamma -do \
21733 ▷  ▷  -graph_theoretic_activity \
21734 ▷  ▷  ▷  -automorphism_group \
21735 ▷  ▷  -end \
21736 ▷  ▷  -with Gamma -do \
21737 ▷  ▷  -graph_theoretic_activity \
21738 ▷  ▷  ▷  -eigenvalues \
21739 ▷  ▷  -end
21740 ▷  pdflatex collinearity_graph_eigenvalues.tex
21741 ▷  $(OPEN) collinearity_graph_eigenvalues.pdf
21742
21743
21744
21745
21746 ################################################################################
21747 # Section 16.8: Quartic Curves
21748
21749
21750 SECTION_CANONICAL_FORMS_OF_QUARTIC_CURVES:
21751
21752
21753 test_16_8:
21754 ▷  make F_17_edge
21755 ▷  make Edge_curve_17_nauty
21756 ▷  make Edge_curve_17_group
21757
21758
21759
21760 F_17_edge:
21761 ▷  $(ORBITER) -v 3 \
21762 ▷  ▷  -define F -finite_field -q 17 -end \
21763 ▷  ▷  -with F -do -finite_field_activity \
21764 ▷  ▷  ▷  -cheat_sheet_GF -end
21765 ▷  pdflatex GF_17.tex
21766 ▷  $(OPEN) GF_17.pdf
21767
21768
21769
21770
21771
21772 Edge_curve_17_nauty:
21773 ▷  $(ORBITER) -v 3 \
21774 ▷  ▷  -define C -combinatorial_object \
21775 ▷  ▷  ▷  -label Edge_curve_q17 Edge\_curve\_q17 \
21776 ▷  ▷  ▷  -file_of_points Edge_q17.txt \
21777 ▷  ▷  -end \
```

```
21778 ▷  ▷  -define F -finite_field -q 17 -end \
21779 ▷  ▷  -define P -projective_space -n 2 -field F -v 0 -end \
21780 ▷  ▷  -with C -do \
21781 ▷  ▷  -combinatorial_object_activity \
21782 ▷  ▷  ▷  -canonical_form_PG P \
21783 ▷  ▷  ▷  ▷  -save_ago \
21784 ▷  ▷  ▷  ▷  -save_transversal \
21785 ▷  ▷  ▷  ▷  -max_TDO_depth 10 \
21786 ▷  ▷  ▷  ▷  -end \
21787 ▷  ▷  ▷  -end \
21788 ▷  ▷  -with C -do \
21789 ▷  ▷  -combinatorial_object_activity \
21790 ▷  ▷  ▷  -report \
21791 ▷  ▷  ▷  ▷  -export_flag_orbits \
21792 ▷  ▷  ▷  ▷  -show_TDO \
21793 ▷  ▷  ▷  ▷  -show_TDA \
21794 ▷  ▷  ▷  ▷  -dont_show_incidence_matrices \
21795 ▷  ▷  ▷  ▷  -export_group_GAP \
21796 ▷  ▷  ▷  -end \
21797 ▷  ▷  -end
21798 ▷  pdflatex Edge_curve_q17_classification.tex
21799 ▷  $(OPEN) Edge_curve_q17_classification.pdf
21800 ▷  $(ORBITER) -v 2 -draw_matrix \
21801 ▷  ▷  -input_csv_file Edge_curve_q17_object0_TDA_flag_orbits.csv \
21802 ▷  ▷  -secondary_input_csv_file Edge_curve_q17_object0_TDA.csv \
21803 ▷  ▷  -box_width 4 -bit_depth 24 \
21804 ▷  ▷  -end
21805 ▷  $(OPEN) Edge_curve_q17_object0_TDA_flag_orbits_draw.bmp
21806
21807 # 9 backtrack nodes total
21808
21809
21810 # aut = 24
21811 # User time: 0.04 of a second, dt=4 tps = 100
21812
21813
21814 # generators for a group of order 24:
21815 #1,0,0,0,13,0,0,0,4,
21816 #1,0,0,0,0,16,0,16,0,
21817 #0,1,16,2,4,4,15,4,4,
21818
21819
21820 Edge_curve_17_group:
21821 ▷  $(ORBITER) -v 3 \
21822 ▷  ▷  -define G -linear_group -PGL 3 17 \
21823 ▷  ▷  -subgroup_by_generators "Stab_Edge" "24" 3 \
21824 ▷  ▷  ▷  "1,0,0,0,13,0,0,0,4, \
21825 ▷  ▷  ▷  1,0,0,0,0,16,0,16,0, \
21826 ▷  ▷  ▷  0,1,16,2,4,4,15,4,4" \
21827 ▷  ▷  -end \
21828 ▷  ▷  -with G -do \
21829 ▷  ▷  ▷  -group_theoretic_activity \
21830 ▷  ▷  ▷  ▷  -print_elements_tex \
21831 ▷  ▷  ▷  ▷  -report_group_table \
21832 ▷  ▷  ▷  ▷  -report \
21833 ▷  ▷  -end
21834 ▷  pdflatex PGL_3_17_Subgroup_Stab_Edge_24_report.tex
21835 ▷  $(OPEN) PGL_3_17_Subgroup_Stab_Edge_24_report.pdf
21836
```

```
21837
21838
21839
21840
21841
21842 ################################################################################
21843 # Chapter 17 - Interfaces
21844 ################################################################################
21845
21846
21847 test_17:
21848 ▷  make test_17_1
21849 ▷  make test_17_2
21850 ▷  make test_17_3
21851 ▷  make test_17_4
21852 ▷  make test_17_5
21853
21854
21855 ################################################################################
21856 # Section 17.1: Graphical Output
21857
21858 SECTION_GRAPHICAL_OUTPUT:
21859
21860
21861 test_17_1:
21862 ▷  make PGL_6_2_Wedge_4_2_detached_report.tex
21863 ▷  make schreier_tree_graphical_output
21864 ▷  make Queens_graph_again
21865
21866
21867
21868
21869
21870 PGL_6_2_Wedge_4_2_detached_report.tex:
21871 ▷  $(ORBITER) -v 4 \
21872 ▷  ▷  -define G -linear_group -PGL 4 2 \
21873 ▷  ▷  ▷  -wedge_detached \
21874 ▷  ▷  -end \
21875 ▷  ▷  -with G -do \
21876 ▷  ▷  -group_theoretic_activity \
21877 ▷  ▷  ▷  -report \
21878 ▷  ▷  -end
21879 ▷  pdflatex PGL_6_2_Wedge_4_2_detached_report.tex
21880 ▷  $(OPEN) PGL_6_2_Wedge_4_2_detached_report.pdf
21881
21882
21883 schreier_tree_graphical_output:
21884 ▷  $(ORBITER) -v 4 \
21885 ▷  ▷  -draw_options \
21886 ▷  ▷  ▷  -yout 500000 \
21887 ▷  ▷  ▷  -radius 15 -nodes_empty \
21888 ▷  ▷  ▷  -line_width 0.5 -y_stretch 0.25 \
21889 ▷  ▷  ▷  -embedded \
21890 ▷  ▷  -end \
21891 ▷  ▷  -define G -linear_group -PGL 4 2 -end \
21892 ▷  ▷  -define Orb -orbits -group G \
21893 ▷  ▷  ▷  -on_polynomials 3 \
21894 ▷  ▷  -end \
21895 ▷  ▷  -with Orb -do -orbits_activity \
```

```
21896 ▷  ▷  ▷  -export_something "orbit" 6 \
21897 ▷  ▷  -end
21898 ▷  ▷  -with Orb -do -orbits_activity \
21899 ▷  ▷  ▷  -draw_tree 6 \
21900 ▷  ▷  -end
21901 ▷
21902
21903 Queens_graph_again:
21904 ▷  $(ORBITER) -v 2 \
21905 ▷  ▷  -define G -graph -non_attacking_queens_graph 8 -end \
21906 ▷  ▷  -with G -do \
21907 ▷  ▷  -graph_theoretic_activity -export_csv -end \
21908 ▷  ▷  -with G -do \
21909 ▷  ▷  -graph_theoretic_activity -export_graphviz -end \
21910 ▷  ▷  -with G -do \
21911 ▷  ▷  -graph_theoretic_activity -save -end \
21912 ▷  ▷  -with G -do \
21913 ▷  ▷  -graph_theoretic_activity -automorphism_group -end \
21914 ▷  ▷  -with G -do \
21915 ▷  ▷  -graph_theoretic_activity -find_cliques \
21916 ▷  ▷  ▷  -target_size 8 -output_file 8queens -end \
21917 ▷  ▷  -end
21918
21919
21920
21921
21922 ################################################################################
21923 # Section 17.2: The Povray Interface
21924
21925
21926 SECTION_POVRAY:
21927
21928 test_17_2:
21929 ▷  make cube
21930 ▷  make math261_test
21931 ▷  make plane1
21932 ▷  make plane2
21933 ▷  make analytic_geo_1
21934 ▷  make analytic_geo_1_video
21935 ▷  make monkey
21936 ▷  make Eckardt
21937 ▷  make Eckardt_deform
21938 ▷  make Eckardt_deform_2
21939 ▷  make Clebsch
21940 ▷  make endrass8
21941
21942
21943 cube:
21944 ▷  $(ORBITER) -v 2 -povray \
21945 ▷  ▷  -round 0 -nb_frames_default 30 \
21946 ▷  ▷  -output_mask cube_%d_%03d.pov \
21947 ▷  ▷  -video_options -W 1024 -H 768 \
21948 ▷  ▷  -global_picture_scale 0.5 \
21949 ▷  ▷  -default_angle 75 \
21950 ▷  ▷  -clipping_radius 2.7 \
21951 ▷  ▷  -end \
21952 ▷  ▷  -scene_objects \
21953 ▷  ▷  ▷  -obj_file cube_centered.obj \
21954 ▷  ▷  ▷  -edge "0, 1" \
```

```
21955 ▷  ▷  ▷  -edge "0, 2" \
21956 ▷  ▷  ▷  -edge "0, 4" \
21957 ▷  ▷  ▷  -edge "1, 3" \
21958 ▷  ▷  ▷  -edge "1, 5" \
21959 ▷  ▷  ▷  -edge "2, 3" \
21960 ▷  ▷  ▷  -edge "2, 6" \
21961 ▷  ▷  ▷  -edge "3, 7" \
21962 ▷  ▷  ▷  -edge "4, 5" \
21963 ▷  ▷  ▷  -edge "4, 6" \
21964 ▷  ▷  ▷  -edge "5, 7" \
21965 ▷  ▷  ▷  -edge "6, 7" \
21966 ▷  ▷  ▷  -group_of_things_as_interval 0 8 \
21967 ▷  ▷  ▷  -spheres 0 0.3 $(POLISHED_CHROME_WHITE) \
21968 ▷  ▷  ▷  -group_of_things_as_interval 0 6 \
21969 ▷  ▷  ▷  -prisms 1 0.05 $(YELLOW_TRANSPARENT) \
21970 ▷  ▷  ▷  -group_of_things_as_interval 0 12 \
21971 ▷  ▷  ▷  -cylinders 2 0.15 $(COLOR_RED) \
21972 ▷  ▷  -scene_objects_end \
21973 ▷  ▷  -povray_end
21974 ▷  - rm -rf POV
21975 ▷  mkdir POV
21976 ▷  mv cube_0_*.pov POV
21977 ▷  mv makefile_animation POV
21978
21979
21980 math261_test:
21981 ▷  $(ORBITER) -v 2 -povray \
21982 ▷  ▷  -round 0 -nb_frames_default 30 \
21983 ▷  ▷  -output_mask math261_%d_%03d.pov \
21984 ▷  ▷  -video_options -W 1024 -H 768 \
21985 ▷  ▷  -global_picture_scale 0.1 \
21986 ▷  ▷  -default_angle 75 \
21987 ▷  ▷  -clipping_radius 2.7 \
21988 ▷  ▷  -end \
21989 ▷  ▷  -scene_objects \
21990 ▷  ▷  ▷  -point "0,0,0" \
21991 ▷  ▷  ▷  -point "5,0,0" \
21992 ▷  ▷  ▷  -point "0,5,0" \
21993 ▷  ▷  ▷  -point "0,0,5" \
21994 ▷  ▷  ▷  -point "1,2,3" \
21995 ▷  ▷  ▷  -point "4,5,6" \
21996 ▷  ▷  ▷  -point "5,7,9" \
21997 ▷  ▷  ▷  -edge "0,1" \
21998 ▷  ▷  ▷  -edge "0,2" \
21999 ▷  ▷  ▷  -edge "0,3" \
22000 ▷  ▷  ▷  -edge "0,4" \
22001 ▷  ▷  ▷  -edge "0,5" \
22002 ▷  ▷  ▷  -edge "4,6" \
22003 ▷  ▷  ▷  -edge "5,6" \
22004 ▷  ▷  ▷  -face "0,4,6,5" \
22005 ▷  ▷  ▷  -group_of_things_as_interval 0 7 \
22006 ▷  ▷  ▷  -spheres 0 0.1 $(POLISHED_CHROME_WHITE) \
22007 ▷  ▷  ▷  -group_of_things_as_interval 0 7 \
22008 ▷  ▷  ▷  -cylinders 1 0.05 $(COLOR_RED) \
22009 ▷  ▷  ▷  -prisms 2 0.05 $(YELLOW_TRANSPARENT) \
22010 ▷  ▷  ▷  -group_of_things_as_interval 0 1 \
22011 ▷  ▷  -scene_objects_end \
22012 ▷  ▷  -povray_end
22013 ▷  - rm -rf POV
```

```
22014 ▷  mkdir POV
22015 ▷  mv math261_0_*.pov POV
22016 ▷  mv makefile_animation POV
22017
22018
22019
22020 plane1:
22021 ▷  cp $(ORBITER_PATH)/examples/users_guide/coordinate_grid.csv .
22022 ▷  $(ORBITER) -v 2 -povray \
22023 ▷  ▷  -round 0 -nb_frames_default 30 \
22024 ▷  ▷  -output_mask plane1_%d_%03d.pov \
22025 ▷  ▷  -video_options -W 1024 -H 768 \
22026 ▷  ▷  -global_picture_scale 0.40 \
22027 ▷  ▷  -default_angle 75 \
22028 ▷  ▷  -clipping_radius 5 -omit_bottom_plane  \
22029 ▷  ▷  -camera 0 "0,0,1" "5,5,3" "0,0,0"  \
22030 ▷  ▷  -rotate_about_z_axis \
22031 ▷  ▷  -boundary_box \
22032 ▷  ▷  -end \
22033 ▷  ▷  -scene_objects \
22034 ▷  ▷  ▷  -line_through_two_points_recentered_from_csv_file coordinate_grid.csv \
22035 ▷  ▷  ▷  -plane_by_dual_coordinates "0,0,1,0" \
22036 ▷  ▷  ▷  -plane_by_dual_coordinates "0,1,0,0" \
22037 ▷  ▷  ▷  -plane_by_dual_coordinates "1,0,0,0" \
22038 ▷  ▷  ▷  -point "-2.25,0,0" \
22039 ▷  ▷  ▷  -point "0,-1.8,0" \
22040 ▷  ▷  ▷  -point "0,0,9" \
22041 ▷  ▷  ▷  -face "0,1,2,0" \
22042 ▷  ▷  ▷  -group_of_things "0" \
22043 ▷  ▷  ▷  -group_of_things "1" \
22044 ▷  ▷  ▷  -group_of_things "2" \
22045 ▷  ▷  ▷  -lines 0 0.15 $(COLOR_RED_SHINY) \
22046 ▷  ▷  ▷  -lines 1 0.15 $(COLOR_GREEN_SHINY) \
22047 ▷  ▷  ▷  -lines 2 0.15 $(COLOR_BLUE_SHINY) \
22048 ▷  ▷  ▷  -group_of_things_as_interval 3 39 \
22049 ▷  ▷  ▷  -lines 3 0.05 $(COLOR_BLACK_SHINY) \
22050 ▷  ▷  ▷  -group_of_things "0" \
22051 ▷  ▷  ▷  -planes 0 $(COLOR_BLUE_SEE_THROUGH) \
22052 ▷  ▷  ▷  -group_of_things "1" \
22053 ▷  ▷  ▷  -group_of_things "2" \
22054 ▷  ▷  ▷  -group_of_things "0" \
22055 ▷  ▷  ▷  -prisms 0 0.05 $(COLOR_YELLOW_THICK) \
22056 ▷  ▷  -scene_objects_end \
22057 ▷  ▷  -povray_end
22058 ▷  - rm -rf POV
22059 ▷  mkdir POV
22060 ▷  mv plane1_0_*.pov POV
22061 ▷  mv makefile_animation POV
22062
22063
22064
22065 plane2:
22066 ▷  cp $(ORBITER_PATH)/examples/users_guide/coordinate_grid.csv .
22067 ▷  $(ORBITER) -v 2 -povray \
22068 ▷  ▷  -round 0 -nb_frames_default 30 \
22069 ▷  ▷  -output_mask plane2_%d_%03d.pov \
22070 ▷  ▷  -video_options -W 2560 -H 1920 \
22071 ▷  ▷  -global_picture_scale 0.40 \
22072 ▷  ▷  -default_angle 75 \
```

```
22073 ▷  ▷  -clipping_radius 5 -omit_bottom_plane  \
22074 ▷  ▷  ▷  -camera 0 "0,0,1" "6,6,2" "0,0,0"  \
22075 ▷  ▷  ▷  -rotate_about_z_axis \
22076 ▷  ▷  ▷  -boundary_box \
22077 ▷  ▷  -end \
22078 ▷  ▷  -scene_objects \
22079 ▷  ▷  ▷  -line_through_two_points_recentered_from_csv_file coordinate_grid.csv \
22080 ▷  ▷  ▷  -plane_by_dual_coordinates "0,0,1,0" \
22081 ▷  ▷  ▷  -plane_by_dual_coordinates "0,1,0,0" \
22082 ▷  ▷  ▷  -plane_by_dual_coordinates "1,0,0,0" \
22083 ▷  ▷  ▷  -plane_by_dual_coordinates "4,5,-1,9" \
22084 ▷  ▷  ▷  -group_of_things "0" \
22085 ▷  ▷  ▷  -group_of_things "1" \
22086 ▷  ▷  ▷  -group_of_things "2" \
22087 ▷  ▷  ▷  -group_of_things_as_interval 3 39 \
22088 ▷  ▷  ▷  -lines 0 0.15 $(COLOR_RED_SHINY) \
22089 ▷  ▷  ▷  -lines 1 0.15 $(COLOR_GREEN_SHINY) \
22090 ▷  ▷  ▷  -lines 2 0.15 $(COLOR_BLUE_SHINY) \
22091 ▷  ▷  ▷  -lines 3 0.05 $(COLOR_BLACK_SHINY) \
22092 ▷  ▷  ▷  -group_of_things "0" \
22093 ▷  ▷  ▷  -planes 4 $(COLOR_BLUE_SEE_THROUGH) \
22094 ▷  ▷  ▷  -group_of_things "3" \
22095 ▷  ▷  -scene_objects_end \
22096 ▷  ▷  -povray_end
22097 ▷  - rm -rf POV
22098 ▷  mkdir POV
22099 ▷  mv plane2_0_*.pov POV
22100 ▷  mv makefile_animation POV
22101
22102
22103
22104
22105 analytic_geo_1:
22106 ▷  cp $(ORBITER_PATH)/examples/users_guide/coordinate_grid.csv .
22107 ▷  $(ORBITER) -v 2 -povray \
22108 ▷  ▷  -round 0 -nb_frames_default 30 \
22109 ▷  ▷  -output_mask analytic_geo_1_%d_%03d.pov \
22110 ▷  ▷  -video_options -W 2560 -H 1920 \
22111 ▷  ▷  -global_picture_scale 0.80 \
22112 ▷  ▷  -default_angle 75 \
22113 ▷  ▷  -clipping_radius 5 -omit_bottom_plane  \
22114 ▷  ▷  ▷  -camera 0 "0,0,1" "6,6,2" "0,0,0"  \
22115 ▷  ▷  ▷  -rotate_about_z_axis \
22116 ▷  ▷  ▷  -boundary_box \
22117 ▷  ▷  -end \
22118 ▷  ▷  -scene_objects \
22119 ▷  ▷  ▷  -line_through_two_points_recentered_from_csv_file coordinate_grid.csv \
22120 ▷  ▷  ▷  -plane_by_dual_coordinates "0,0,1,0" \
22121 ▷  ▷  ▷  -plane_by_dual_coordinates "0,1,0,0" \
22122 ▷  ▷  ▷  -plane_by_dual_coordinates "1,0,0,0" \
22123 ▷  ▷  ▷  -group_of_things "0" \
22124 ▷  ▷  ▷  -group_of_things "1" \
22125 ▷  ▷  ▷  -group_of_things "2" \
22126 ▷  ▷  ▷  -group_of_things_as_interval 3 39 \
22127 ▷  ▷  ▷  -lines 0 0.05 $(COLOR_RED_SHINY) \
22128 ▷  ▷  ▷  -lines 1 0.05 $(COLOR_GREEN_SHINY) \
22129 ▷  ▷  ▷  -lines 2 0.05 $(COLOR_BLUE_SHINY) \
22130 ▷  ▷  ▷  -lines 3 0.04 $(COLOR_BLACK_SHINY) \
22131 ▷  ▷  ▷  -group_of_things "0" \
```

```
22132 ▷  ▷  ▷  -group_of_things "1" \
22133 ▷  ▷  ▷  -group_of_things "2" \
22134 ▷  ▷  ▷  -planes 4 $(COLOR_BLUE_SEE_THROUGH) \
22135 ▷  ▷  ▷  -planes 5 $(COLOR_GREEN_SEE_THROUGH) \
22136 ▷  ▷  ▷  -planes 6 $(COLOR_RED_SEE_THROUGH) \
22137 ▷  ▷  ▷  -point "0,0,0" \
22138 ▷  ▷  ▷  -point "1,0,0" \
22139 ▷  ▷  ▷  -point "1,2,0" \
22140 ▷  ▷  ▷  -point "1,2,3" \
22141 ▷  ▷  ▷  -edge "84,85" \
22142 ▷  ▷  ▷  -edge "85,86" \
22143 ▷  ▷  ▷  -edge "86,87" \
22144 ▷  ▷  ▷  -edge "84,87" \
22145 ▷  ▷  ▷  -group_of_things "84,85,86" \
22146 ▷  ▷  ▷  -spheres 7 0.1 $(POLISHED_CHROME_WHITE) \
22147 ▷  ▷  ▷  -group_of_things "87" \
22148 ▷  ▷  ▷  -spheres 8 0.10 $(COLOR_YELLOW_SHINY) \
22149 ▷  ▷  ▷  -group_of_things "0,1,2" \
22150 ▷  ▷  ▷  -cylinders 9 0.075 $(POLISHED_CHROME_WHITE) \
22151 ▷  ▷  ▷  -group_of_things "3" \
22152 ▷  ▷  ▷  -cylinders 10 0.075 $(COLOR_YELLOW_SHINY) \
22153 ▷  ▷  -scene_objects_end \
22154 ▷  ▷  -povray_end
22155 ▷  - rm -rf POV
22156 ▷  mkdir POV
22157 ▷  mv analytic_geo_1_0_*.pov POV
22158 ▷  mv makefile_animation POV
22159
22160
22161 analytic_geo_1_video:
22162 ▷  - rm -r FRAMES
22163 ▷  - mkdir FRAMES
22164 ▷  - rm analytic_geo_1.mp4
22165 ▷  $(ORBITER) \
22166 ▷  ▷  -prepare_frames \
22167 ▷  ▷  ▷  -i 0 30 PNG/ANALYTIC_GEO_1/analytic_geo_1_0_%03d.png \
22168 ▷  ▷  ▷  -output_starts_at 0 \
22169 ▷  ▷  ▷  -o FRAMES/frame%04d.png \
22170 ▷  ▷  -end
22171 ▷  ffmpeg -r 5 -f image2 -i FRAMES/frame%04d.png \
22172 ▷  ▷  -f mp4 -q:v 0 -vcodec mpeg4 analytic_geo_1.mp4
22173
22174 monkey:
22175 ▷  $(ORBITER) -v 2 -povray \
22176 ▷  ▷  -round 0 -nb_frames_default 30 \
22177 ▷  ▷  -output_mask monkey_%d_%03d.pov \
22178 ▷  ▷  -video_options -W 1024 -H 768 \
22179 ▷  ▷  -global_picture_scale 0.8 \
22180 ▷  ▷  -default_angle 75 \
22181 ▷  ▷  -clipping_radius 0.8 \
22182 ▷  ▷  -camera 0 "0,0,1" "1,1,0.5" "0,0,0" \
22183 ▷  ▷  -rotate_about_z_axis \
22184 ▷  ▷  -end \
22185 ▷  ▷  -scene_objects \
22186 ▷  ▷  ▷  -cubic_lex $(MONKEY_SADDLE_CUBIC) \
22187 ▷  ▷  ▷  -plane_by_dual_coordinates "0,0,1,0" \
22188 ▷  ▷  ▷  -group_of_things "0" \
22189 ▷  ▷  ▷  -group_of_things "0" \
22190 ▷  ▷  ▷  -cubics 0 $(COLOR_GOLD) \
```

```
22191 ▷  ▷  ▷  -planes 1 $(COLOR_BLUE) \
22192 ▷  ▷  -scene_objects_end \
22193 ▷  ▷  -povray_end
22194 ▷  - rm -rf POV
22195 ▷  mkdir POV
22196 ▷  mv monkey_0_*.pov POV
22197 ▷  mv makefile_animation POV
22198
22199
22200
22201
22202 Eckardt:
22203 ▷  $(ORBITER) -v 2 -povray \
22204 ▷  ▷  -round 0 -nb_frames_default 30 \
22205 ▷  ▷  -output_mask Eckardt_%d_%03d.pov \
22206 ▷  ▷  -video_options -W 1024 -H 768 \
22207 ▷  ▷  -global_picture_scale 0.9 \
22208 ▷  ▷  -default_angle 75 \
22209 ▷  ▷  -clipping_radius 2.4 \
22210 ▷  ▷  -camera 0 "1,1,1" "-3,1,3" "0.12,0.12,0.12" \
22211 ▷  ▷  -end \
22212 ▷  ▷  -scene_objects \
22213 ▷  ▷  ▷  -Hilbert_Cohn_Vossen_surface \
22214 ▷  ▷  ▷  -group_of_things "0" \
22215 ▷  ▷  ▷  -cubics 0 $(SURFACE_COLOR) \
22216 ▷  ▷  ▷  -group_of_things_as_interval 0 6 \
22217 ▷  ▷  ▷  -group_of_things_as_interval 6 6 \
22218 ▷  ▷  ▷  -group_of_things_as_interval_with_exceptions 12 15 \
22219 ▷  ▷  ▷  ▷  "14,19,23" \
22220 ▷  ▷  ▷  -lines 1 0.02 $(COLOR_RED_SHINY) \
22221 ▷  ▷  ▷  -lines 2 0.02 $(COLOR_BLUE_SHINY) \
22222 ▷  ▷  ▷  -lines 3 0.02 $(COLOR_YELLOW_SHINY) \
22223 ▷  ▷  ▷  -label 0 "a1" \
22224 ▷  ▷  ▷  -label 2 "a2" \
22225 ▷  ▷  ▷  -label 4 "a3" \
22226 ▷  ▷  ▷  -label 6 "a4" \
22227 ▷  ▷  ▷  -label 8 "a5" \
22228 ▷  ▷  ▷  -label 10 "a6" \
22229 ▷  ▷  ▷  -label 12 "b1" \
22230 ▷  ▷  ▷  -label 14 "b2" \
22231 ▷  ▷  ▷  -label 16 "b3" \
22232 ▷  ▷  ▷  -label 18 "b4" \
22233 ▷  ▷  ▷  -label 20 "b5" \
22234 ▷  ▷  ▷  -label 22 "b6" \
22235 ▷  ▷  ▷  -label 24 "c12" \
22236 ▷  ▷  ▷  -label 26 "c13" \
22237 ▷  ▷  ▷  -label 30 "c15" \
22238 ▷  ▷  ▷  -label 32 "c16" \
22239 ▷  ▷  ▷  -label 34 "c23" \
22240 ▷  ▷  ▷  -label 36 "c24" \
22241 ▷  ▷  ▷  -label 40 "c26" \
22242 ▷  ▷  ▷  -label 42 "c34" \
22243 ▷  ▷  ▷  -label 44 "c35" \
22244 ▷  ▷  ▷  -label 48 "c45" \
22245 ▷  ▷  ▷  -label 50 "c46" \
22246 ▷  ▷  ▷  -label 52 "c56" \
22247 ▷  ▷  ▷  -group_of_things_as_interval 0 6 \
22248 ▷  ▷  ▷  -texts 4 0.2 0.15 $(COLOR_BLACK_NO_SHADOW) \
22249 ▷  ▷  ▷  -group_of_things_as_interval 6 6 \
```

```
22250 ▷  ▷  ▷  -texts 5 0.2 0.15 $(COLOR_BLACK_NO_SHADOW) \
22251 ▷  ▷  ▷  -group_of_things_as_interval 12 12 \
22252 ▷  ▷  ▷  -texts 6 0.2 0.15 $(COLOR_BLACK_NO_SHADOW) \
22253 ▷  ▷  -scene_objects_end \
22254 ▷  ▷  -povray_end
22255 ▷  - rm -rf POV
22256 ▷  mkdir POV
22257 ▷  mv Eckardt_0_*.pov POV
22258 ▷  mv makefile_animation POV
22259
22260
22261 #"-3,2.333,4"  * 1.5 = "-4.5,3.5,6"
22262 #M := Matrix([[-4.5, 3.5, 6], [1, 1, 1]])
22263 #NullSpace(M)
22264 #=0.186080731891197,-0.781539073943026,0.595458342051830
22265 #▷ ▷  -rotate_about_custom_axis "0.186080731891197,-0.781539073943026,0.5954583420
      51830" \
22266 #-W 1024 -H 768
22267 #-W 2560 -H 1920
22268 #-W 4096 -H 3072
22269
22270
22271
22272
22273 Eckardt_deform:
22274 ▷  $(ORBITER) -v 2 -povray \
22275 ▷  ▷  -round 0 -nb_frames_default 93 \
22276 ▷  ▷  -output_mask Eckardt_deform_%d_%03d.pov \
22277 ▷  ▷  -video_options -W 1024 -H 768 \
22278 ▷  ▷  -global_picture_scale 0.9 \
22279 ▷  ▷  -default_angle 75 \
22280 ▷  ▷  -clipping_radius 2.4 \
22281 ▷  ▷  -camera 0 "1,1,1" "-3,1,3" "0.12,0.12,0.12" \
22282 ▷  ▷  -end \
22283 ▷  ▷  -scene_objects \
22284 ▷  ▷  ▷  -Hilbert_Cohn_Vossen_surface \
22285 ▷  ▷  ▷  -group_of_things "0" \
22286 ▷  ▷  ▷  -deformation_of_cubic_lex 93 1.107148718 1.570796327 0 \
22287 ▷  ▷  ▷  ▷  $(ECKARDT_CUBIC_DEFORM1_LEX) \
22288 ▷  ▷  ▷  ▷  $(ECKARDT_CUBIC_DEFORM2_LEX) \
22289 ▷  ▷  ▷  -group_of_things_as_interval 0 93 \
22290 ▷  ▷  ▷  -group_is_animated 1 \
22291 ▷  ▷  ▷  -cubics 1 $(SURFACE_COLOR_SEETHROUGH) \
22292 ▷  ▷  -scene_objects_end \
22293 ▷  ▷  -povray_end
22294 ▷  - rm -rf POV
22295 ▷  mkdir POV
22296 ▷  mv Eckardt_deform_0_*.pov POV
22297 ▷  mv makefile_animation POV
22298
22299
22300
22301
22302
22303 Eckardt_deform_2:
22304 ▷  $(ORBITER) -v 2 -povray \
22305 ▷  ▷  -round 0 -nb_frames_default 30 \
22306 ▷  ▷  -output_mask Eckardt_deform_%d_%03d.pov \
22307 ▷  ▷  -video_options -W 1024 -H 768 \
```

```
22308 ▷  ▷  -global_picture_scale 0.9 \
22309 ▷  ▷  -default_angle 75 \
22310 ▷  ▷  -clipping_radius 2.4 \
22311 ▷  ▷  -camera 0 "1,1,1" "-3,1,3" "0.12,0.12,0.12" \
22312 ▷  ▷  -end \
22313 ▷  ▷  -scene_objects \
22314 ▷  ▷  ▷  -Hilbert_Cohn_Vossen_surface \
22315 ▷  ▷  ▷  -group_of_things "0" \
22316 ▷  ▷  ▷  -deformation_of_cubic_lex 93 1.107148718 1.570796327 0 \
22317 ▷  ▷  ▷  ▷  $(ECKARDT_CUBIC_DEFORM1_LEX) \
22318 ▷  ▷  ▷  ▷  $(ECKARDT_CUBIC_DEFORM2_LEX) \
22319 ▷  ▷  ▷  --group_of_things_as_interval 0 93 \
22320 ▷  ▷  ▷  --group_is_animated 1 \
22321 ▷  ▷  ▷  -group_of_things "0" \
22322 ▷  ▷  ▷  -cubics 1 $(SURFACE_COLOR_SEETHROUGH) \
22323 ▷  ▷  ▷  -group_of_things "24" \
22324 ▷  ▷  ▷  -cubics 2 $(COLOR_RED) \
22325 ▷  ▷  ▷  -group_of_things "70" \
22326 ▷  ▷  ▷  -cubics 3 $(COLOR_BLUE) \
22327 ▷  ▷  -scene_objects_end \
22328 ▷  ▷  -povray_end
22329 ▷  - rm -rf POV
22330 ▷  mkdir POV
22331 ▷  mv Eckardt_deform_0_*.pov POV
22332 ▷  mv makefile_animation POV
22333
22334
22335
22336 Clebsch:
22337 ▷  $(ORBITER) -v 2 -povray \
22338 ▷  ▷  -round 0 -nb_frames_default 30 \
22339 ▷  ▷  -output_mask Clebsch_%d_%03d.pov \
22340 ▷  ▷  -video_options -W 1024 -H 768 \
22341 ▷  ▷  -global_picture_scale 0.9 \
22342 ▷  ▷  -default_angle 80 \
22343 ▷  ▷  -clipping_radius 2.4 \
22344 ▷  ▷  -camera 0 "1,1,1" "-4.5,3.5,6" "0,0,0" \
22345 ▷  ▷  -end \
22346 ▷  ▷  -scene_objects \
22347 ▷  ▷  ▷  -Clebsch_surface \
22348 ▷  ▷  ▷  -group_of_things "0" \
22349 ▷  ▷  ▷  -cubics 0 $(SURFACE_COLOR) \
22350 ▷  ▷  ▷  -group_of_things_as_interval 0 6 \
22351 ▷  ▷  ▷  -group_of_things_as_interval 6 6 \
22352 ▷  ▷  ▷  -group_of_things_as_interval 12 15 \
22353 ▷  ▷  ▷  -lines 1 0.02 $(COLOR_RED_SHINY) \
22354 ▷  ▷  ▷  -lines 2 0.02 $(COLOR_BLUE_SHINY) \
22355 ▷  ▷  ▷  -lines 3 0.02 $(COLOR_YELLOW_SHINY) \
22356 ▷  ▷  ▷  -group_of_things_as_interval 0 12 \
22357 ▷  ▷  ▷  -spheres 4 0.08 $(COLOR_TURQUOISE) \
22358 ▷  ▷  -scene_objects_end \
22359 ▷  ▷  -povray_end
22360 ▷  - rm -rf POV
22361 ▷  mkdir POV
22362 ▷  mv Clebsch_0_*.pov POV
22363 ▷  mv makefile_animation POV
22364
22365
22366
```

```
22367 endrass8:
22368 ▷  cp $(ORBITER_PATH)/examples/users_guide/coordinate_grid.csv .
22369 ▷  $(ORBITER) -v 2 -povray \
22370 ▷  ▷  -round 0 -nb_frames_default 30 \
22371 ▷  ▷  -output_mask endrass_octic_%d_%03d.pov \
22372 ▷  ▷  -video_options -W 1024 -H 768 \
22373 ▷  ▷  -global_picture_scale 0.75 \
22374 ▷  ▷  -default_angle 75 \
22375 ▷  ▷  -clipping_radius 3.7 \
22376 ▷  ▷  -no_bottom_plane \
22377 ▷  ▷  -camera 0 "1,1,1" "6,6,3" "0,0,0" \
22378 ▷  ▷  -rotate_about_111 \
22379 ▷  ▷  -end \
22380 ▷  ▷  -scene_objects \
22381 ▷  ▷  ▷  -line_through_two_points_recentered_from_csv_file \
22382 ▷  ▷  ▷  ▷  coordinate_grid.csv \
22383 ▷  ▷  ▷  -group_of_things "0" \
22384 ▷  ▷  ▷  -group_of_things "1" \
22385 ▷  ▷  ▷  -group_of_things "2" \
22386 ▷  ▷  ▷  -group_of_things_as_interval 3 39 \
22387 ▷  ▷  ▷  -lines 0 0.15 $(COLOR_RED_SHINY) \
22388 ▷  ▷  ▷  -lines 1 0.15 $(COLOR_GREEN_SHINY) \
22389 ▷  ▷  ▷  -lines 2 0.15 $(COLOR_BLUE_SHINY) \
22390 ▷  ▷  ▷  -lines 3 0.05 $(COLOR_BLACK_SHINY) \
22391 ▷  ▷  ▷  -octic_lex_165 $(ENDRASS_OCTIC_LEX_165) \
22392 ▷  ▷  ▷  -plane_by_dual_coordinates "0,0,1,0" \
22393 ▷  ▷  ▷  -group_of_things "0" \
22394 ▷  ▷  ▷  -group_of_things "0" \
22395 ▷  ▷  ▷  -octics 4  $(SURFACE_COLOR_SEETHROUGH) \
22396 ▷  ▷  ▷  -planes 5 "texture{ pigment{ color Blue transmit 0.5 } \
22397 finish { diffuse 0.9 phong 1}}" \
22398 ▷  ▷  -scene_objects_end \
22399 ▷  ▷  -povray_end
22400 ▷  - rm -rf POV
22401 ▷  mkdir POV
22402 ▷  mv endrass_octic_0_*.pov POV
22403 ▷  mv makefile_animation POV
22404
22405
22406
22407
22408
22409 ################################################################################
22410 # Section 17.3: Creating Animations
22411
22412 SECTION_ANIMATIONS:
22413
22414
22415 test_17_3:
22416 ▷  make dode
22417 ▷  make dode_video
22418 ▷  make monkey_video
22419 ▷  make Eckardt_deform_video
22420 ▷  make Eckardt_surface
22421 ▷  make Kummer_surface
22422 ▷  make Kummer_video
22423 ▷  make Beauville_surface
22424 ▷  make Clebsch_up_create_points
22425 ▷  make Clebsch_surface
```

```
22426 ▷  make Clebsch_surface_defining_equation
22427 ▷  make Clebsch_surface_defining_equation_and_curves
22428 ▷  make F7_povray
22429 ▷  make F7_video
22430 ▷  make McKean_povray
22431 ▷  make McKean_video
22432
22433
22434
22435 dode:
22436 ▷  $(ORBITER) -v 2 \
22437 ▷  ▷  -povray \
22438 ▷  ▷  -round 0 -nb_frames_default 30 \
22439 ▷  ▷  -output_mask dode_%d_%03d.pov \
22440 ▷  ▷  -video_options -W 1024 -H 768 \
22441 ▷  ▷  -global_picture_scale 0.50 \
22442 ▷  ▷  -default_angle 45 \
22443 ▷  ▷  -clipping_radius 5   \
22444 ▷  ▷  -camera 0 "1,1,1" "-2,2,4" "0,0,0"  \
22445 ▷  ▷  -rotate_about_111 \
22446 ▷  ▷  -end \
22447 ▷  ▷  -scene_objects \
22448 ▷  ▷  ▷  -dodecahedron \
22449 ▷  ▷  ▷  -group_of_things_as_interval 0 20 \
22450 ▷  ▷  ▷  -spheres 0 0.075 $(POLISHED_CHROME_WHITE) \
22451 ▷  ▷  ▷  -group_of_things_as_interval 0 30\
22452 ▷  ▷  ▷  -cylinders 1 0.05 $(COLOR_RED_SHINY) \
22453 ▷  ▷  ▷  -group_of_things_as_interval 0 12\
22454 ▷  ▷  ▷  -prisms 2 0.02 $(YELLOW_TRANSPARENT) \
22455 ▷  ▷  -scene_objects_end \
22456 ▷  ▷  -povray_end
22457 ▷  - rm -rf POV
22458 ▷  mkdir POV
22459 ▷  mv dode_0_*.pov POV
22460 ▷  mv makefile_animation POV
22461
22462
22463 dode_video:
22464 ▷  - rm -r FRAMES
22465 ▷  - mkdir FRAMES
22466 ▷  - rm dode.mp4
22467 ▷  $(ORBITER) \
22468 ▷  ▷  -prepare_frames \
22469 ▷  ▷  ▷  -i 0 30 DODE/dode_0_%03d.png \
22470 ▷  ▷  ▷  -output_starts_at 0 \
22471 ▷  ▷  ▷  -o FRAMES/frame%04d.png \
22472 ▷  ▷  -end
22473 ▷  ffmpeg -r 5 -f image2 -i FRAMES/frame%04d.png \
22474 ▷  ▷  -f mp4 -q:v 0 -vcodec mpeg4 dode.mp4
22475
22476
22477 monkey_video:
22478 ▷  - rm -r FRAMES
22479 ▷  - mkdir FRAMES
22480 ▷  - rm monkey.mp4
22481 ▷  $(ORBITER) \
22482 ▷  ▷  -prepare_frames \
22483 ▷  ▷  ▷  -i 0 30 monkey_0_%03d.png \
22484 ▷  ▷  ▷  -output_starts_at 0 \
```

```
22485 ▷  ▷  ▷  -o FRAMES/frame%04d.png \
22486 ▷  ▷  -end
22487 ▷  ffmpeg -r 5 -f image2 -i FRAMES/frame%04d.png \
22488 ▷  ▷  -f mp4 -q:v 0 -vcodec mpeg4 monkey.mp4
22489
22490 Eckardt_deform_video:
22491 ▷  - rm -r FRAMES
22492 ▷  - mkdir FRAMES
22493 ▷  - rm Eckardt_deform.mp4
22494 ▷  $(ORBITER) \
22495 ▷  ▷  -prepare_frames \
22496 ▷  ▷  ▷  -i 0 93 Eckardt_deform_0/Eckardt_deform_0_%03d.png \
22497 ▷  ▷  ▷  -output_starts_at 0 \
22498 ▷  ▷  ▷  -o FRAMES/frame%04d.png \
22499 ▷  ▷  -end
22500 ▷  ffmpeg -r 5 -f image2 -i FRAMES/frame%04d.png \
22501 ▷  ▷  -f mp4 -q:v 0 -vcodec mpeg4 Eckardt_deform.mp4
22502
22503
22504 Eckardt_surface:
22505 ▷  $(ORBITER) -v 2 -povray \
22506 ▷  ▷  -round 0 -nb_frames_default 30 \
22507 ▷  ▷  -output_mask Eckardt_%d_%03d.pov \
22508 ▷  ▷  -video_options -W 1024 -H 768 \
22509 ▷  ▷  -global_picture_scale 0.9 \
22510 ▷  ▷  -default_angle 75 \
22511 ▷  ▷  -clipping_radius 2.4 \
22512 ▷  ▷  -camera 0 "1,1,1" "-3,1,3" "0.12,0.12,0.12" \
22513 ▷  ▷  -end \
22514 ▷  ▷  -scene_objects \
22515 ▷  ▷  ▷  -cubic_Goursat "6,3,-15" \
22516 ▷  ▷  ▷  -group_of_things "0" \
22517 ▷  ▷  ▷  -cubics 0 $(SURFACE_COLOR_SEETHROUGH) \
22518 ▷  ▷  -scene_objects_end \
22519 ▷  ▷  -povray_end
22520 ▷  - rm -rf POV
22521 ▷  mkdir POV
22522 ▷  mv Eckardt_0_*.pov POV
22523 ▷  mv makefile_animation POV
22524
22525
22526
22527 Kummer_surface:
22528 ▷  $(ORBITER) -v 2 -povray \
22529 ▷  ▷  -round 0 -nb_frames_default 30 \
22530 ▷  ▷  -output_mask Kummer_%d_%03d.pov \
22531 ▷  ▷  -video_options -W 1024 -H 768 \
22532 ▷  ▷  -global_picture_scale 0.9 \
22533 ▷  ▷  -default_angle 75 \
22534 ▷  ▷  -clipping_radius 2.4 \
22535 ▷  ▷  -camera 0 "1,1,1" "-3,1,3" "0.12,0.12,0.12" \
22536 ▷  ▷  -end \
22537 ▷  ▷  -scene_objects \
22538 ▷  ▷  ▷  -quartic_lex_35 $(KUMMER_QUARTIC_LEX_35) \
22539 ▷  ▷  ▷  -group_of_things "0" \
22540 ▷  ▷  ▷  -quartics 0 $(SURFACE_COLOR_SEETHROUGH) \
22541 ▷  ▷  -scene_objects_end \
22542 ▷  ▷  -povray_end
22543 ▷  - rm -rf POV
```

```
22544 ▷  mkdir POV
22545 ▷  mv Kummer_0_*.pov POV
22546 ▷  mv makefile_animation POV
22547
22548
22549 # Maple:
22550 #Kummer := expand((x0^2 + x1^2 + x2^2 + x3^2)^2 - 3*(x0^4 + x1^4 + x2^4 + x3^4))
22551
22552
22553 Kummer_video:
22554 ▷  - rm -r FRAMES
22555 ▷  - mkdir FRAMES
22556 ▷  - rm Kummer.mp4
22557 ▷  $(ORBITER) \
22558 ▷  ▷  -prepare_frames \
22559 ▷  ▷  ▷  -i 0 30 KUMMER/Kummer_0_%03d.png \
22560 ▷  ▷  ▷  -output_starts_at 0 \
22561 ▷  ▷  ▷  -o FRAMES/frame%04d.png \
22562 ▷  ▷  -end
22563 ▷  ffmpeg -r 5 -f image2 -i FRAMES/frame%04d.png \
22564 ▷  ▷  -f mp4 -q:v 0 -vcodec mpeg4 Kummer.mp4
22565
22566
22567
22568 Beauville_surface:
22569 ▷  $(ORBITER) -v 2 -povray \
22570 ▷  ▷  -round 0 -nb_frames_default 30 \
22571 ▷  ▷  -output_mask Beauville_%d_%03d.pov \
22572 ▷  ▷  -video_options -W 1024 -H 768 \
22573 ▷  ▷  -global_picture_scale 0.3 \
22574 ▷  ▷  -default_angle 75 \
22575 ▷  ▷  -clipping_radius 2.4 \
22576 ▷  ▷  -camera 0 "1,1,1" "-3,1,3" "0.12,0.12,0.12" \
22577 ▷  ▷  -end \
22578 ▷  ▷  -scene_objects \
22579 ▷  ▷  ▷  -quintic_lex_56 $(BEAUVILLE_QUINTIC_LEX_56) \
22580 ▷  ▷  ▷  -group_of_things "0" \
22581 ▷  ▷  ▷  -quintics 0 $(SURFACE_COLOR_SEETHROUGH) \
22582 ▷  ▷  -scene_objects_end \
22583 ▷  ▷  -povray_end
22584 ▷  - rm -rf POV
22585 ▷  mkdir POV
22586 ▷  mv Beauville_0_*.pov POV
22587 ▷  mv makefile_animation POV
22588
22589
22590
22591
22592
22593
22594
22595
22596 Clebsch_up_create_points:
22597 ▷  $(ORBITER) -v 2 \
22598 ▷  ▷  -smooth_curve "Clebsch_map_of_circle_to_defininig_eqn_r2" \
22599 ▷  ▷  ▷  0.07 1000 5 0 $(TWO_PI) \
22600 ▷  ▷  -const a $(CLEBSCH_A) b $(CLEBSCH_B) c $(CLEBSCH_C) d $(CLEBSCH_D) \
22601 ▷  ▷  ▷  t00 $(T00) t01 $(T01) t02 $(T02) t03 $(T03) \
22602 ▷  ▷  ▷  t10 $(T10) t11 $(T11) t12 $(T12) t13 $(T13) \
```

```
22603 ▷  ▷  ▷  t20 $(T20) t21 $(T21) t22 $(T22) t23 $(T23) \
22604 ▷  ▷  ▷  t30 $(T30) t31 $(T31) t32 $(T32) t33 $(T33) \
22605 ▷  ▷  ▷  r 2 one 1 m -1 \
22606 ▷  ▷  -const_end \
22607 ▷  ▷  -var t \
22608 ▷  ▷  ▷  c001 c002 c011 c012 \
22609 ▷  ▷  ▷  d001 d011 d012 d112 \
22610 ▷  ▷  ▷  m002 m012 m022 m122 \
22611 ▷  ▷  ▷  n002 n012 n112 n022 n122 \
22612 ▷  ▷  ▷  y0 y1 y2 \
22613 ▷  ▷  ▷  y001 y002 y011 y012 y022 y112 y122 \
22614 ▷  ▷  ▷  x0 x1 x2 x3 \
22615 ▷  ▷  ▷  -var_end \
22616 ▷  ▷  -code \
22617 ▷  ▷  ▷  push t cos push r mult store y0 \
22618 ▷  ▷  ▷  push t sin push r mult store y1 \
22619 ▷  ▷  ▷  push one store y2 \
22620 ▷  ▷  ▷  push y0 push y0 push y1 mult mult store y001 \
22621 ▷  ▷  ▷  push y0 push y0 push y2 mult mult store y002 \
22622 ▷  ▷  ▷  push y0 push y1 push y1 mult mult store y011 \
22623 ▷  ▷  ▷  push y0 push y1 push y2 mult mult store y012 \
22624 ▷  ▷  ▷  push y0 push y2 push y2 mult mult store y022 \
22625 ▷  ▷  ▷  push y1 push y1 push y2 mult mult store y112 \
22626 ▷  ▷  ▷  push y1 push y2 push y2 mult mult store y122 \
22627 ▷  ▷  ▷  $(CLEBSCH_CUBICS) \
22628 ▷  ▷  ▷  ▷  push c001 push y001 mult \
22629 ▷  ▷  ▷  ▷  push c002 push y002 mult add \
22630 ▷  ▷  ▷  ▷  push c011 push y011 mult add \
22631 ▷  ▷  ▷  ▷  push c012 push y012 mult add \
22632 ▷  ▷  ▷  ▷  store x0 \
22633 ▷  ▷  ▷  ▷  push d001 push y001 mult \
22634 ▷  ▷  ▷  ▷  push d011 push y011 mult add \
22635 ▷  ▷  ▷  ▷  push d012 push y012 mult add \
22636 ▷  ▷  ▷  ▷  push d112 push y112 mult add \
22637 ▷  ▷  ▷  ▷  store x1 \
22638 ▷  ▷  ▷  ▷  push m002 push y002 mult \
22639 ▷  ▷  ▷  ▷  push m012 push y012 mult add \
22640 ▷  ▷  ▷  ▷  push m022 push y022 mult add \
22641 ▷  ▷  ▷  ▷  push m122 push y122 mult add \
22642 ▷  ▷  ▷  ▷  store x2 \
22643 ▷  ▷  ▷  ▷  push n002 push y002 mult \
22644 ▷  ▷  ▷  ▷  push n012 push y012 mult add \
22645 ▷  ▷  ▷  ▷  push n022 push y022 mult add \
22646 ▷  ▷  ▷  ▷  push n112 push y112 mult add \
22647 ▷  ▷  ▷  ▷  push n122 push y122 mult add \
22648 ▷  ▷  ▷  ▷  store x3 \
22649 ▷  ▷  ▷  ▷  push x0 push t00 mult \
22650 ▷  ▷  ▷  ▷  push x1 push t10 mult add \
22651 ▷  ▷  ▷  ▷  push x2 push t20 mult add \
22652 ▷  ▷  ▷  ▷  push x3 push t30 mult add \
22653 ▷  ▷  ▷  ▷  return \
22654 ▷  ▷  ▷  ▷  push x0 push t01 mult \
22655 ▷  ▷  ▷  ▷  push x1 push t11 mult add \
22656 ▷  ▷  ▷  ▷  push x2 push t21 mult add \
22657 ▷  ▷  ▷  ▷  push x3 push t31 mult add \
22658 ▷  ▷  ▷  ▷  return \
22659 ▷  ▷  ▷  ▷  push x0 push t02 mult \
22660 ▷  ▷  ▷  ▷  push x1 push t12 mult add \
22661 ▷  ▷  ▷  ▷  push x2 push t22 mult add \
```

```
22662 ▷  ▷  ▷  ▷  push x3 push t32 mult add \
22663 ▷  ▷  ▷  ▷  return \
22664 ▷  ▷  ▷  ▷  push x0 push t03 mult \
22665 ▷  ▷  ▷  ▷  push x1 push t13 mult add \
22666 ▷  ▷  ▷  ▷  push x2 push t23 mult add \
22667 ▷  ▷  ▷  ▷  push x3 push t33 mult add \
22668 ▷  ▷  ▷  ▷  return \
22669 ▷  ▷  -code_end
22670
22671
22672 Clebsch_surface:
22673 ▷  $(ORBITER) -v 2 -povray \
22674 ▷  ▷  -round 0 -nb_frames_default 30 \
22675 ▷  ▷  -output_mask Clebsch_%d_%03d.pov \
22676 ▷  ▷  -video_options -W 1024 -H 768 \
22677 ▷  ▷  -global_picture_scale 0.9 \
22678 ▷  ▷  -default_angle 75 \
22679 ▷  ▷  -clipping_radius 2.4 \
22680 ▷  ▷  -camera 0 "1,1,1" "-3,1,3" "0.12,0.12,0.12" \
22681 ▷  ▷  -end \
22682 ▷  ▷  -scene_objects \
22683 ▷  ▷  ▷  -cubic_orbiter "0,0,0,0,0,-4.236067972,\
22684 0,0,4.236067972,4.236067972,17.94427188,\
22685 -17.94427188,0,0,- 9.472135941,0,0,5.236067971,\
22686 8.472135938,- 27.41640782" \
22687 ▷  ▷  ▷  -group_of_things "0" \
22688 ▷  ▷  ▷  -cubics 0 $(SURFACE_COLOR_SEETHROUGH) \
22689 ▷  ▷  ▷  -point_list_from_csv_file \
22690 ▷  ▷  ▷  function_Clebsch_map_of_circle_N1000_points.csv \
22691 ▷  ▷  ▷  -group_of_things_as_interval 0 954 \
22692 ▷  ▷  ▷  -spheres 1 0.07 $(COLOR_RED) \
22693 ▷  ▷  -scene_objects_end \
22694 ▷  ▷  -povray_end
22695 ▷  - rm -rf POV
22696 ▷  mkdir POV
22697 ▷  mv Clebsch_0_*.pov POV
22698 ▷  mv makefile_animation POV
22699
22700
22701 Clebsch_surface_defining_equation:
22702 ▷  $(ORBITER) -v 2 -povray \
22703 ▷  ▷  -round 0 -nb_frames_default 30 \
22704 ▷  ▷  -output_mask Clebsch_%d_%03d.pov \
22705 ▷  ▷  -video_options -W 1024 -H 768 \
22706 ▷  ▷  -global_picture_scale 0.6 \
22707 ▷  ▷  -default_angle 75 \
22708 ▷  ▷  -clipping_radius 1.6 \
22709 ▷  ▷  -camera 0 "1,1,1" "-2,0,2" "0,0,0" \
22710 ▷  ▷  -end \
22711 ▷  ▷  -scene_objects \
22712 ▷  ▷  ▷  -cubic_orbiter "0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,2,2,2,2" \
22713 ▷  ▷  ▷  -group_of_things "0" \
22714 ▷  ▷  ▷  -cubics 0 $(SURFACE_COLOR_SEETHROUGH) \
22715 ▷  ▷  -scene_objects_end \
22716 ▷  ▷  -povray_end
22717 ▷  - rm -rf POV
22718 ▷  mkdir POV
22719 ▷  mv Clebsch_0_*.pov POV
22720 ▷  mv makefile_animation POV
```

```
22721
22722
22723
22724 Clebsch_surface_defining_equation_and_curves:
22725 ▷  $(ORBITER) -v 2 -povray \
22726 ▷  ▷  -round 0 -nb_frames_default 30 \
22727 ▷  ▷  -output_mask Clebsch_2curves_%d_%03d.pov \
22728 ▷  ▷  -video_options -W 1024 -H 768 \
22729 ▷  ▷  -global_picture_scale 0.6 \
22730 ▷  ▷  -default_angle 75 \
22731 ▷  ▷  -clipping_radius 1.6 \
22732 ▷  ▷  -camera 0 "1,1,1" "-2,0,2" "0,0,0" \
22733 ▷  ▷  -end \
22734 ▷  ▷  -scene_objects \
22735 ▷  ▷  ▷  -cubic_orbiter "0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,2,2,2,2" \
22736 ▷  ▷  ▷  -group_of_things "0" \
22737 ▷  ▷  ▷  -cubics 0 $(SURFACE_COLOR_SEETHROUGH) \
22738 ▷  ▷  ▷  -point_list_from_csv_file \
22739 ▷  ▷  ▷  function_Clebsch_map_of_circle_to_defininig_eqn_N1000_points.csv \
22740 ▷  ▷  ▷  -group_of_things_as_interval 0 656 \
22741 ▷  ▷  ▷  -spheres 1 0.07 $(COLOR_RED) \
22742 ▷  ▷  ▷  -point_list_from_csv_file \
22743 ▷  ▷  ▷  function_Clebsch_map_of_circle_to_defininig_eqn_r2_N1000_points.csv \
22744 ▷  ▷  ▷  -group_of_things_as_interval 656 1042 \
22745 ▷  ▷  ▷  -spheres 2 0.07 "texture{ pigment{ color Blue } \
22746 finish { diffuse 0.9 phong 1}}" \
22747 ▷  ▷  -scene_objects_end \
22748 ▷  ▷  -povray_end
22749 ▷  - rm -rf POV
22750 ▷  mkdir POV
22751 ▷  mv Clebsch_2curves_0_*.pov POV
22752 ▷  mv makefile_animation POV
22753
22754 #▷ ▷  ▷  -point_list_from_csv_file function_Clebsch_map_of_circle_N1000_points.csv
      \
22755 #▷ ▷  ▷  -group_of_things_as_interval 0 954 \
22756 #▷ ▷  ▷  -spheres 1 0.07 "texture{ pigment{ color Red } finish { diffuse 0.9 phong
      1}}" \
22757
22758
22759
22760
22761 F7_povray:
22762 ▷  $(ORBITER) -v 2 -povray \
22763 ▷  ▷  -round 0 -nb_frames_default 30 \
22764 ▷  ▷  -output_mask F7_15_lines_%d_%03d.pov \
22765 ▷  ▷  -video_options -W 1024 -H 768 \
22766 ▷  ▷  -global_picture_scale 1.5 \
22767 ▷  ▷  -default_angle 80 \
22768 ▷  ▷  -clipping_radius 4.4 \
22769 ▷  ▷  -omit_bottom_plane \
22770 ▷  ▷  -camera 0 "1,1,1" "-4.5,3.5,6" "0,0,0" \
22771 ▷  ▷  -end \
22772 ▷  ▷  -scene_objects \
22773 ▷  ▷  ▷  -cubic_lex "0, 0, 6, 0, 0, -13.39014946, -3.341901346, -6.972931640, 5.8271
      82718, 0, 0, 7.390149464, 7.390149464, 6.972931640, -1.512349728, -8.485281372, 0
      , 0, 0, 0" \
22774 ▷  ▷  ▷  -group_of_things "0" \
22775 ▷  ▷  ▷  -cubics 0 $(SURFACE_COLOR_SEETHROUGH) \
```

```
22776 ▷  ▷  ▷  -line_through_point_with_direction "0, 0, 0, 1, 0, 0" \
22777 ▷  ▷  ▷  -line_through_point_with_direction "0, 0, -1, 0, 1, 0" \
22778 ▷  ▷  ▷  -line_through_point_with_direction "0, 0, 0, 0, 0, -1" \
22779 ▷  ▷  ▷  -line_through_point_with_direction "1, 0, 0, 1, 1, 1" \
22780 ▷  ▷  ▷  -line_through_point_with_direction "-1.414213562, 0, 0, 4.146264370, 1.7320
      50808, 1.732050808" \
22781 ▷  ▷  ▷  -line_through_point_with_direction "0, 1.732050808, -1, 2.414213562, -0.317
      837246, 2.414213562" \
22782 ▷  ▷  ▷  -line_through_point_with_direction "-2.133352390, 0, -1, 1.674708020, 1, 0"
       \
22783 ▷  ▷  ▷  -line_through_point_with_direction "-2.539058015, 0, 0, 2.211360755, 1, 0"
      \
22784 ▷  ▷  ▷  -line_through_point_with_direction "0, 1.148188060, 0, 0, -0.9435440612, 1"
       \
22785 ▷  ▷  ▷  -line_through_point_with_direction "-0.9711971171, 0, 0, 1.162155272, 0, 1"
       \
22786 ▷  ▷  ▷  -line_through_point_with_direction "2.096037870, 2.096037870, 0, -1.0658519
      05, -1.065851905, 1" \
22787 ▷  ▷  ▷  -line_through_point_with_direction "3.921555783, 2.921555781, 0, -1.7224565
      85, -1.722456585, 1" \
22788 ▷  ▷  ▷  -group_of_things_as_interval 0 12 \
22789 ▷  ▷  ▷  -lines 1 0.04 $(COLOR_YELLOW) \
22790 ▷  ▷  -scene_objects_end \
22791 ▷  ▷  -povray_end
22792 ▷  - rm -rf POV
22793 ▷  mkdir POV
22794 ▷  mv F7_15_lines_0_*.pov POV
22795 ▷  mv makefile_animation POV
22796
22797
22798
22799
22800
22801
22802 F7_video:
22803 ▷  - rm -r FRAMES
22804 ▷  - mkdir FRAMES
22805 ▷  - rm fifteen_with_lines.mp4
22806 ▷  $(ORBITER) \
22807 ▷  ▷  -prepare_frames \
22808 ▷  ▷  ▷  -i 0 30 F7b/F7_15_lines_0_%03d.png \
22809 ▷  ▷  ▷  -output_starts_at 0 \
22810 ▷  ▷  ▷  -o FRAMES/frame%04d.png \
22811 ▷  ▷  -end
22812 ▷  ffmpeg -r 5 -f image2 -i FRAMES/frame%04d.png \
22813 ▷  ▷  -f mp4 -q:v 0 -vcodec mpeg4 fifteen_with_lines.mp4
22814
22815
22816 McKean_povray:
22817 ▷  $(ORBITER) -v 2 -povray \
22818 ▷  ▷  -round 0 -nb_frames_default 30 \
22819 ▷  ▷  -output_mask McKean_%d_%03d.pov \
22820 ▷  ▷  -video_options -W 1024 -H 768 \
22821 ▷  ▷  -global_picture_scale 1.5 \
22822 ▷  ▷  -default_angle 80 \
22823 ▷  ▷  -clipping_radius 4.4 \
22824 ▷  ▷  -omit_bottom_plane \
22825 ▷  ▷  -camera 0 "1,1,1" "-4.5,3.5,6" "0,0,0" \
22826 ▷  ▷  -end \
```

```
22827 ▷  ▷  -scene_objects \
22828 ▷  ▷  ▷  -cubic_lex "0, 0, 1, 0, 0, -1, -2, 1, \
22829 2, 0, 0, 1, 1,-1, -1, -1, 0, 0, 0, 0" \
22830 ▷  ▷  ▷  -group_of_things "0" \
22831 ▷  ▷  ▷  -cubics 0 $(SURFACE_COLOR_SEETHROUGH) \
22832 ▷  ▷  -scene_objects_end \
22833 ▷  ▷  -povray_end
22834 ▷  - rm -rf POV
22835 ▷  mkdir POV
22836 ▷  mv McKean_0_*.pov POV
22837 ▷  mv makefile_animation POV
22838
22839 McKean_video:
22840 ▷  - rm -r FRAMES
22841 ▷  - mkdir FRAMES
22842 ▷  - rm McKean.mp4
22843 ▷  $(ORBITER) \
22844 ▷  ▷  -prepare_frames \
22845 ▷  ▷  ▷  -i 0 30 MCKEAN/McKean_0_%03d.png \
22846 ▷  ▷  ▷  -output_starts_at 0 \
22847 ▷  ▷  ▷  -o FRAMES/frame%04d.png \
22848 ▷  ▷  -end
22849 ▷  ffmpeg -r 5 -f image2 -i FRAMES/frame%04d.png \
22850 ▷  ▷  -f mp4 -q:v 0 -vcodec mpeg4 McKean.mp4
22851
22852
22853
22854
22855 ##############################################################################
22856 # Section 17.4: Continuous Function Plotter
22857
22858
22859
22860 SECTION_CONTINUOUS_FUNCTION_PLOTTER:
22861
22862
22863
22864 test_17_4:
22865 ▷  make lissajous
22866 ▷  make lissajous_plot
22867 ▷  make lissajous_3d
22868 ▷  make lissajous_3d_plot
22869
22870
22871
22872
22873 lissajous:
22874 ▷  $(ORBITER) -v 2 \
22875 ▷  ▷  -smooth_curve "lissajous" 0.07 2000 15 0 18.85 \
22876 ▷  ▷  -const a 3 b 2 c 1.57 r 7 -const_end \
22877 ▷  ▷  -var t -var_end \
22878 ▷  ▷  -code \
22879 ▷  ▷  ▷  push t push a mult push c add sin push r mult return \
22880 ▷  ▷  ▷  push t push b mult sin push r mult return \
22881 ▷  ▷  -code_end \
22882
22883 #function_lissajous_N2000_points.csv
22884
22885 lissajous_plot:
```

```
22886 ▷  cp $(ORBITER_PATH)/examples/users_guide/coordinate_grid.csv .
22887 ▷  $(ORBITER) -v 2 -povray \
22888 ▷  ▷  -round 0 -nb_frames_default 1 \
22889 ▷  ▷  -output_mask lissajous_%d_%03d.pov \
22890 ▷  ▷  -video_options -W 1024 -H 768 \
22891 ▷  ▷  -global_picture_scale 0.40 \
22892 ▷  ▷  -default_angle 45 \
22893 ▷  ▷  -clipping_radius 5 \
22894 ▷  ▷  -omit_bottom_plane \
22895 ▷  ▷  -camera 0 "0,-1,0" "0,0,12" "0,0,0" \
22896 ▷  ▷  -rotate_about_z_axis \
22897 ▷  ▷  -end \
22898 ▷  ▷  -scene_objects \
22899 ▷  ▷  ▷  -line_through_two_points_recentered_from_csv_file \
22900 ▷  ▷  ▷  coordinate_grid.csv \
22901 ▷  ▷  ▷  -group_of_things "0" \
22902 ▷  ▷  ▷  -group_of_things "1" \
22903 ▷  ▷  ▷  -group_of_things "2" \
22904 ▷  ▷  ▷  -lines 0 0.09 "texture{ pigment{ color Yellow } }" \
22905 ▷  ▷  ▷  -lines 1 0.09 "texture{ pigment{ color Yellow } }" \
22906 ▷  ▷  ▷  -lines 2 0.09 "texture{ pigment{ color Yellow } }" \
22907 ▷  ▷  ▷  -group_of_things_as_interval 3 39 \
22908 ▷  ▷  ▷  -lines 3 0.02 "texture{ pigment{ color Black } }" \
22909 ▷  ▷  ▷  -point_list_from_csv_file \
22910 ▷  ▷  ▷  function_lissajous_N2000_points.csv \
22911 ▷  ▷  ▷  -group_of_things_as_interval 0 6524\
22912 ▷  ▷  ▷  -spheres 4 0.1 "texture{ pigment{ color Red } \
22913 finish { diffuse 0.9 phong 1}}" \
22914 ▷  ▷  ▷  -plane_by_dual_coordinates "0,0,1,0" \
22915 ▷  ▷  ▷  -group_of_things "0" \
22916 ▷  ▷  ▷  -planes 5 "texture{ pigment{ color Blue*0.5 \
22917 transmit 0.5 } }" \
22918 ▷  ▷  -scene_objects_end \
22919 ▷  ▷  -povray_end
22920 ▷  - rm -rf POV
22921 ▷  mkdir POV
22922 ▷  mv lissajous_0_*.pov POV
22923 ▷  mv makefile_animation POV
22924
22925 lissajous_3d:
22926 ▷  $(ORBITER) -v 2 \
22927 ▷  ▷  -smooth_curve "lissajous_3d" 0.07 2000 50 0 18.85 \
22928 ▷  ▷  -const a 3 b 2 c 1.57 r 7 -const_end \
22929 ▷  ▷  -var t -var_end \
22930 ▷  ▷  -code \
22931 ▷  ▷  ▷  push t push a mult push c add sin push r mult return \
22932 ▷  ▷  ▷  push t push b mult sin push r mult return \
22933 ▷  ▷  ▷  push t return \
22934 ▷  ▷  -code_end \
22935
22936 lissajous_3d_plot:
22937 ▷  cp $(ORBITER_PATH)/examples/users_guide/coordinate_grid.csv .
22938 ▷  $(ORBITER) -v 2 -povray \
22939 ▷  ▷  -round 0 -nb_frames_default 30 \
22940 ▷  ▷  -output_mask lissajous_3d_%d_%03d.pov \
22941 ▷  ▷  -video_options -W 1024 -H 768 \
22942 ▷  ▷  -global_picture_scale 0.40 \
22943 ▷  ▷  -default_angle 45 \
22944 ▷  ▷  -clipping_radius 5 \
```

```
22945 ▷  ▷  -omit_bottom_plane \
22946 ▷  ▷  -camera 0 "0,0,1" "7,7,5" "0,0,1" \
22947 ▷  ▷  -rotate_about_z_axis \
22948 ▷  ▷  -end \
22949 ▷  ▷  -scene_objects \
22950 ▷  ▷  ▷  -line_through_two_points_recentered_from_csv_file \
22951 ▷  ▷  ▷  coordinate_grid.csv \
22952 ▷  ▷  ▷  -group_of_things "0" \
22953 ▷  ▷  ▷  -group_of_things "1" \
22954 ▷  ▷  ▷  -group_of_things "2" \
22955 ▷  ▷  ▷  -lines 0 0.09 "texture{ pigment{ color Yellow } }" \
22956 ▷  ▷  ▷  -lines 1 0.09 "texture{ pigment{ color Yellow } }" \
22957 ▷  ▷  ▷  -lines 2 0.09 "texture{ pigment{ color Yellow } }" \
22958 ▷  ▷  ▷  -group_of_things_as_interval 3 39 \
22959 ▷  ▷  ▷  -lines 3 0.02 "texture{ pigment{ color Black } }" \
22960 ▷  ▷  ▷  -point_list_from_csv_file \
22961 ▷  ▷  ▷  function_lissajous_3d_N2000_points.csv \
22962 ▷  ▷  ▷  -group_of_things_as_interval 0 6538\
22963 ▷  ▷  ▷  -spheres 4 0.1 "texture{ pigment{ color Red } \
22964 finish { diffuse 0.9 phong 1}}" \
22965 ▷  ▷  ▷  -plane_by_dual_coordinates "0,0,1,0" \
22966 ▷  ▷  ▷  -group_of_things "0" \
22967 ▷  ▷  -scene_objects_end \
22968 ▷  ▷  -povray_end
22969 ▷  - rm -rf POV
22970 ▷  mkdir POV
22971 ▷  mv lissajous_3d_0_*.pov POV
22972 ▷  mv makefile_animation POV
22973
22974
22975 ################################################################################
22976 # Section 17.5: Gnuplot interface
22977
22978
22979
22980 SECTION_GNUPLOT_INTERFACE:
22981
22982
22983
22984 test_17_5:
22985 ▷  make gnuplot_test_data
22986 ▷  make gnuplot_Eulerfunction_50
22987
22988 gnuplot_test_data:
22989 ▷  echo $(GNUPLOT_TEST_INPUT) >gnuplot_test_data.csv
22990 ▷  $(ORBITER) -v 3 \
22991 ▷  ▷  -gnuplot gnuplot_test_data.csv \
22992 ▷  ▷  "Gnuplot test data" \
22993 ▷  ▷  "Input" \
22994 ▷  ▷  "Output"
22995 ▷  $(OPEN) gnuplot_test_data_gnuplot.png
22996
22997
22998 gnuplot_Eulerfunction_50:
22999 ▷  $(ORBITER) -v 1 -eulerfunction_interval 1 50
23000 ▷  $(ORBITER) -v 3 \
23001 ▷  ▷  -gnuplot table_eulerfunction_1_50.csv \
23002 ▷  ▷  "Euler function" \
23003 ▷  ▷  "Input" \
```

```
23004 ▷  ▷  "Output"
23005
23006
23007 #Row,N,PHI,NBPF,NBDPF
23008
23009
23010 ###############################################################################
23011 # Chapter 18 - Mathematical Data in Orbiter
23012 ###############################################################################
23013
23014
23015 test_18:
23016 ▷  make test_18_1
23017 ▷  make test_18_2
23018
23019
23020 ###############################################################################
23021 # Section 18.1: Mathematical Data on Cubic Surfaces
23022
23023 SECTION_MATHEMATICAL_DATA:
23024
23025
23026 test_18_1:
23027 ▷  make make_table_of_surfaces
23028 ▷  make make_table_of_quartic_curves
23029 ▷  make cubic_surfaces_table_q17
23030 ▷  make quartic_curves_table_q17
23031
23032
23033 make_table_of_surfaces:
23034 ▷  $(ORBITER) -v 3 \
23035 ▷  ▷  -make_table_of_surfaces
23036 ▷  pdflatex surfaces_report.tex
23037 ▷  $(OPEN) surfaces_report.pdf
23038
23039
23040 make_table_of_quartic_curves:
23041 ▷  $(ORBITER) -v 3 \
23042 ▷  ▷  -make_table_of_quartic_curves
23043 ▷  pdflatex quartic_curves_report.tex
23044 ▷  $(OPEN) quartic_curves_report.pdf
23045
23046
23047
23048 cubic_surfaces_table_q17:
23049 ▷  $(ORBITER) -v 3 \
23050 ▷  ▷  -define F -finite_field -q 17 -end \
23051 ▷  ▷  -define P -projective_space \
23052 ▷  ▷  ▷  -n 3 -field F -v 0 \
23053 ▷  ▷  -end \
23054 ▷  ▷  -with P -do \
23055 ▷  ▷  ▷  -projective_space_activity \
23056 ▷  ▷  ▷  ▷  -table_of_cubic_surfaces \
23057 ▷  ▷  -end
23058
23059 quartic_curves_table_q17:
23060 ▷  $(ORBITER) -v 3 \
23061 ▷  ▷  -define F -finite_field -q 17 -end \
23062 ▷  ▷  -define P -projective_space \
```

```
23063 ▷  ▷  ▷  -n 2 -field F -v 0 \
23064 ▷  ▷  -end \
23065 ▷  ▷  -with P -do \
23066 ▷  ▷  ▷  -projective_space_activity \
23067 ▷  ▷  ▷  ▷  -table_of_quartic_curves \
23068 ▷  ▷  -end
23069
23070 # quartic_curves_q17_info.csv
23071
23072
23073
23074 ################################################################################
23075 # Section 18.2: Mathematical Data on BLT-Sets
23076
23077 SECTION_MATHEMATICAL_DATA:
23078
23079
23080 test_18_2:
23081 ▷  make blt_set_table_q7
23082 ▷  make blt_set_table_q9
23083 ▷  make blt_set_table_q13
23084 ▷  make blt_set_table
23085 ▷  make blt_set_export_gap
23086
23087
23088
23089 blt_set_table_q7:
23090 ▷  $(ORBITER) -v 3 \
23091 ▷  ▷  -define F -finite_field -q 7 -end \
23092 ▷  ▷  -define O -orthogonal_space 0 5 F -end \
23093 ▷  ▷  -with O -do \
23094 ▷  ▷  ▷  -orthogonal_space_activity \
23095 ▷  ▷  ▷  ▷  -table_of_blt_sets \
23096 ▷  ▷  -end
23097
23098 blt_set_table_q9:
23099 ▷  $(ORBITER) -v 3 \
23100 ▷  ▷  -define F -finite_field -q 9 -end \
23101 ▷  ▷  -define O -orthogonal_space 0 5 F -end \
23102 ▷  ▷  -with O -do \
23103 ▷  ▷  ▷  -orthogonal_space_activity \
23104 ▷  ▷  ▷  ▷  -table_of_blt_sets \
23105 ▷  ▷  -end
23106
23107
23108 blt_set_table_q13:
23109 ▷  $(ORBITER) -v 3 \
23110 ▷  ▷  -define F -finite_field -q 13 -end \
23111 ▷  ▷  -define O -orthogonal_space 0 5 F -end \
23112 ▷  ▷  -with O -do \
23113 ▷  ▷  ▷  -orthogonal_space_activity \
23114 ▷  ▷  ▷  ▷  -table_of_blt_sets \
23115 ▷  ▷  -end
23116
23117
23118 blt_set_table:
23119 ▷  $(ORBITER) -v 30 \
23120 ▷  ▷  -define Q -vector -dense "3,5,7,9,11,13,17,19,23,25,27,29,31" -end \
23121 ▷  ▷  -define NB -vector -dense "1,2,2,3,4,3,6,5,9,6,6,9,8" -end \
```

```
23122 ▷  ▷  -loop_over i Q \
23123 ▷  ▷  ▷  -define F%i -finite_field -q "%i[Q]" -end \
23124 ▷  ▷  ▷  -define O%i -orthogonal_space 0 5 F%i -end \
23125 ▷  ▷  ▷  -with O%i -do \
23126 ▷  ▷  ▷  ▷  -orthogonal_space_activity \
23127 ▷  ▷  ▷  ▷  ▷  -table_of_blt_sets \
23128 ▷  ▷  ▷  ▷  -end \
23129 ▷  ▷  ▷  -end \
23130 ▷  ▷  -end_loop_over i \
23131 ▷  ▷  -print_symbols
23132
23133
23134 blt_set_export_gap:
23135 ▷  $(ORBITER) -v 30 \
23136 ▷  ▷  -define Q -vector -dense \
23137 ▷  ▷  ▷  $(BLT_ORDER_Q) \
23138 ▷  ▷  -end \
23139 ▷  ▷  -define NB -vector -dense \
23140 ▷  ▷  ▷  $(BLT_NUMBER_ISO) \
23141 ▷  ▷  -end \
23142 ▷  ▷  -loop_over i Q \
23143 ▷  ▷  ▷  -define F%i -finite_field -q "%i[Q]" -end \
23144 ▷  ▷  ▷  -define O%i -orthogonal_space 0 5 F%i -end \
23145 ▷  ▷  ▷  -loop j 0 %i[NB] 1 \
23146 ▷  ▷  ▷  ▷  -define BLT_%i_%j -BLT_set \
23147 ▷  ▷  ▷  ▷  ▷  -space O%i -catalogue %j \
23148 ▷  ▷  ▷  ▷  -end \
23149 ▷  ▷  ▷  ▷  -with BLT_%i_%j -do -blt_set_activity \
23150 ▷  ▷  ▷  ▷  ▷  -export_gap \
23151 ▷  ▷  ▷  ▷  -end \
23152 ▷  ▷  ▷  -end_loop j \
23153 ▷  ▷  -end_loop_over i \
23154 ▷  ▷  -print_symbols
23155
23156
23157 ##############################################################################
23158 # Chapter 19 - Miscellaneous
23159 ##############################################################################
23160
23161
23162 test_19:
23163 ▷  make test_19_1
23164 ▷  make test_19_2
23165
23166
23167
23168 ##############################################################################
23169 # Section 19.1: Miscellaneous
23170
23171 SECTION_MISCELLANEOUS:
23172
23173
23174
23175 test_19_1:
23176 ▷  make misc_select
23177 ▷  make misc_join
23178 ▷  make table_mod_12
23179
23180
```

```
23181
23182 misc_select:
23183 ▷  $(ORBITER) -v 3 \
23184 ▷  ▷  -define F -finite_field -q 7 -end \
23185 ▷  ▷  -with F -do -finite_field_activity -cheat_sheet_GF -end
23186 ▷  $(ORBITER) -v 4 -csv_file_select_rows_and_cols \
23187 ▷  ▷  GF_q7_multiplication_table_reordered.csv \
23188 ▷  ▷  "0,2,4" "0,2,4"
23189
23190
23191 misc_join:
23192 ▷  $(ORBITER) -v 4 \
23193 ▷  ▷  -csv_file_join 5 \
23194 ▷  ▷  ▷  poly_orbits_d3_n3_q2_select_F2.csv Orbit_idx \
23195 ▷  ▷  ▷  poly_orbits_d3_n3_q2_select_F4.csv Orbit_idx \
23196 ▷  ▷  ▷  poly_orbits_d3_n3_q2_select_F8.csv Orbit_idx \
23197 ▷  ▷  ▷  poly_orbits_d3_n3_q2_select_F16.csv Orbit_idx \
23198 ▷  ▷  ▷  poly_orbits_d3_n3_q2_select_F32.csv Orbit_idx
23199
23200
23201 table_mod_12:
23202 ▷  $(ORBITER) -v 2 \
23203 ▷  ▷  -define M -vector -load_csv_no_border clock_mult_excel.csv -end \
23204 ▷  ▷  -define all_one_r -vector -repeat 1 12 -end \
23205 ▷  ▷  -define all_one_c -vector -repeat 1 12 -end \
23206 ▷  ▷  -draw_matrix \
23207 ▷  ▷  -input_object M \
23208 ▷  ▷  -box_width 50 -bit_depth 24 \
23209 ▷  ▷  ▷  -partition 3 \
23210 ▷  ▷  ▷  ▷  all_one_r all_one_c \
23211 ▷  ▷  -end
23212
23213
23214
23215
23216 ################################################################################
23217 # Section 19.2: Limitations
23218
23219 SECTION_LIMITATIONS:
23220
23221 test_19_2:
23222 ▷  make F_15bit
23223 ▷  make F_31bit
23224 ▷  #make F_32bit
23225 ▷  #make F_63bit
23226 ▷  #make F_64bit
23227
23228
23229 # 2^15 - 19 is prime
23230 # https://primes.utm.edu/lists/2small/0bit.html
23231
23232 F_15bit:
23233 ▷  $(ORBITER) -v 10 \
23234 ▷  ▷  -define F -finite_field -q 32749 -without_tables -end \
23235 ▷  ▷  -define v -vector -field F -allow_negatives \
23236 ▷  ▷  ▷  -dense "-1,-1" -end \
23237 ▷  ▷  -with F -do -finite_field_activity -product_of v -end
23238 ▷
23239
```

```
23240 F_31bit:
23241 ▷  $(ORBITER) -v 10 \
23242 ▷  ▷  -define F -finite_field -q 2147483647 -without_tables -end \
23243 ▷  ▷  -define v -vector -field F -allow_negatives \
23244 ▷  ▷  ▷  -dense "-1,-1" -end \
23245 ▷  ▷  -with F -do -finite_field_activity -product_of v -end
23246
23247
23248 # The next command fail on purpose. The goal is to
23249 # show the limits of the integer domain.
23250
23251 F_32bit:
23252 ▷  $(ORBITER) -v 10 \
23253 ▷  ▷  -define F -finite_field -q 4294967291 -without_tables -end \
23254
23255 # we get an error message. The order of the field is too large for 32 bit int:
23256 # 2^32 - 5 = 4294967291
23257
23258
23259 F_63bit:
23260 ▷  $(ORBITER) -v 10 \
23261 ▷  ▷  -define F -finite_field -q 9223372036854775783 -without_tables -end \
23262 ▷  ▷  -define v -vector -field F -allow_negatives \
23263 ▷  ▷  ▷  -dense "-1,-1" -end \
23264 ▷  ▷  -with F -do -finite_field_activity -product_of v -end
23265
23266 #2^63-25 = 9223372036854775783 is prime
23267
23268 F_64bit:
23269 ▷  $(ORBITER) -v 10 \
23270 ▷  ▷  -define F -finite_field -q 18446744073709551557 -without_tables -end \
23271 ▷  ▷  -define v -vector -field F -allow_negatives \
23272 ▷  ▷  ▷  -dense "-1,-1" -end \
23273 ▷  ▷  -with F -do -finite_field_activity -product_of v -end
23274
23275 #2^64-59 = 18446744073709551557 is prime
23276
23277
23278
23279
23280 ####
23281
23282
23283 # unclassified:
23284
23285
23286
23287
23288
23289
23290
23291 extract:
23292 ▷  $(ORBITER) -v 3 \
23293 ▷  ▷  -extract_from_file makefile Cremona_map make_Cremona_map.txt
23294 ▷  ~/bin/a2tex.out -numbers -text_width 80 <make_Cremona_map.txt >make_Cremona_map
      .tex
23295
23296 draw_eigenvalue_diag23:
23297 ▷  $(ORBITER) -v 2 \
```

```
23298 ▷  ▷  -draw_options \
23299 ▷  ▷  ▷  -radius 10 \
23300 ▷  ▷  ▷  -line_width 1.5 -embedded \
23301 ▷  ▷  -end \
23302 ▷  ▷  -draw_mod_n -n 20 \
23303 ▷  ▷  ▷  -file ev_diag23 \
23304 ▷  ▷  ▷  -eigenvalues 2 0 0 3 \
23305 ▷  ▷  -end
23306 ▷  pdflatex ev_diag23_draw.tex
23307 ▷  $(OPEN) ev_diag23_draw.pdf
23308
```

Bibliography

[1] Abdullah Al-Azemi, Anton Betten, and Sajeeb Roy Chowdhury. A rainbow-clique search algorithm for BLT-sets. In *ICMS 2018—Proceedings of the International Congress on Mathematical Software; James H. Davenport, Manuel Kauers, George Labahn, Josef Urban (ed.)*, pages 71–79. Springer, 2018.

[2] Awss Al-ogaidi and Anton Betten. Large Arcs in Small Planes, Congressus Numerantium 232 (2019), 119–136.

[3] Johannes André. Über nicht-Desarguessche Ebenen mit transitiver Translationsgruppe. *Math. Z.*, 60:156–186, 1954.

[4] Laura Bader, Guglielmo Lunardon, and Joseph A. Thas. Derivation of flocks of quadratic cones. *Forum Math.*, 2(2):163–174, 1990.

[5] John Bamberg, Anton Betten, Philippe Cara, Jan De Beule, Michel Lavrauw, and Max Neunhöffer. *FinInG – Finite Incidence Geometry, Version 1.4.1*, 2018.

[6] Susan Barwick and Gary Ebert. *Unitals in projective planes*. Springer Monographs in Mathematics. Springer, New York, 2008.

[7] R. Bayley. `https://www.distanceregular.org`.

[8] Anton Betten. Twisted tensor product codes. *Des. Codes Cryptogr.*, 47(1-3):191–219, 2008.

[9] Anton Betten. Rainbow cliques and the classification of small BLT-sets. In *ISSAC 2013—Proceedings of the 38th International Symposium on Symbolic and Algebraic Computation*, pages 53–60. ACM, New York, 2013.

[10] Anton Betten. The orbiter ecosystem for combinatorial objects. In *ISSAC 2020—Proceedings of the 45th International Symposium on Symbolic and Algebraic Computation*, pages 30–37. ACM, New York, 2020.

[11] Anton Betten, 2022. Orbiter – A program to classify discrete objects, 2022, `https://github.com/abetten/orbiter`.

[12] Anton Betten and Dieter Betten. There is no Drake/Larson linear space on 30 points. *J. Combin. Des.*, 18(1):48–70, 2010.

[13] Anton Betten and Fatma Karaoglu. Cubic surfaces over small finite fields. *Des. Codes Cryptogr.*, 87(4):931–953, 2019.

[14] Anton Betten and Fatma Karaoglu. The Eckardt point configuration of cubic surfaces revisited. *Des. Codes Cryptogr.*, 90(9):2159–2180, 2022.

[15] Dieter Betten. 4-dimensionale Translationsebenen mit 8-dimensionaler Kollineationsgruppe. *Geometriae Dedicata*, 2:327–339, 1973.

[16] Wieb Bosma, John Cannon, and Catherine Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997. Computational algebra and number theory (London, 1993).

[17] A. Brouwer. The Cohen-Tits near octagon on 315 points, https://www.win.tue.nl/~aeb/drg/graphs/HJ315.html.

[18] R. H. Bruck and R. C. Bose. The construction of translation planes from projective spaces. *J. Algebra*, 1:85–102, 1964.

[19] G. Butler. *Fundamental algorithms for permutation groups*, volume 559 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1991.

[20] Arjeh M. Cohen. Geometries originating from certain distance-regular graphs. In *Finite geometries and designs (Proc. Conf., Chelwood Gate, 1980)*, volume 49 of *London Math. Soc. Lecture Note Ser.*, pages 81–87. Cambridge Univ. Press, Cambridge-New York, 1981.

[21] Henri Cohen. *A course in computational algebraic number theory*, volume 138 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, 1993.

[22] Jenny Anne Cooley. Cubic Surfaces over Finite Fields, Ph.D. thesis, University of Warwick, 2014.

[23] Cygwin. https://www.cygwin.com.

[24] Terry Czerwinski and David Oakden. The translation planes of order twenty-five. *J. Combin. Theory Ser. A*, 59(2):193–217, 1992.

[25] Anne Delandtsheer and Jean Doyen. Most block-transitive t-designs are point-primitive. *Geom. Dedicata*, 29(3):307–310, 1989.

[26] L.E. Dickson. Projective classification of cubic surfaces modulo 2, Ann. of Math. 16 (1915), 139-157.

[27] Docker. https://www.docker.com.

[28] S. Endrass. A projective surface of degree eight with 168 nodes. *J. Algebraic Geom.*, 6(2):325–334, 1997.

[29] J. Chris Fisher and Joseph A. Thas. Flocks in PG(3, q). *Math. Z.*, 169(1):1–11, 1979.

[30] Free Software Foundation, 2021. GNU make https://www.gnu.org/software/make/manual/make.html.

[31] The GAP Group. *GAP – Groups, Algorithms, and Programming, Version 4.11.1*, 2021. http://www.gap-system.org.

[32] D. G. Glynn and J. W. P. Hirschfeld. On the classification of geometric codes by polynomial functions. *Des. Codes Cryptogr.*, 6(3):189–204, 1995.

[33] Gnuplot. Gnuplot http://www.gnuplot.info.

[34] R. Hill. On Pellegrino's 20-caps in $S_{4,3}$. In *Combinatorics '81 (Rome, 1981)*, volume 78 of *North-Holland Math. Stud.*, pages 433–447. North-Holland, Amsterdam, 1983.

[35] J. W. P. Hirschfeld. del Pezzo surfaces over finite fields. *Tensor (N.S.)*, 37(1):79–84, 1982.

[36] James W. P. Hirschfeld. The double-six of lines over $PG(3, 4)$. *J. Austral. Math. Soc.*, 4:83–89, 1964.

[37] John D. Hobby. Metapost, a user's manual. 2010, 2010.

[38] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. *An introduction to mathematical cryptography*. Undergraduate Texts in Mathematics. Springer, New York, second edition, 2014.

[39] Derek F. Holt, Bettina Eick, and Eamonn A. O'Brien. *Handbook of computational group theory*. Discrete Mathematics and its Applications (Boca Raton). Chapman & Hall/CRC, Boca Raton, FL, 2005.

[40] Christopher Jefferson, Markus Pfeiffer, Rebecca Waldecker, and Eliza Jonauskyte. GAP package images. 2019. https://www.gap-system.org/Packages/images.html.

[41] Fatma Karaoglu. The cubic surfaces with twenty-seven lines over finite fields, Ph.D. thesis, University of Sussex, 2018.

[42] Fatma Karaoglu and Anton Betten. The number of cubic surfaces with 27 lines over a finite field. *J. Algebraic Combin.*, 56(1):43–57, 2022.

[43] Brian Kernighan and Dennis Ritchie. The C Programming Language. Prentice Hall. 1978 and 1988.

[44] Earl S. Kramer and Dale M. Mesner. t-designs on hypergraphs. *Discrete Math.*, 15(3):263–296, 1976.

[45] Joseph P. S. Kung. The cycle structure of a linear transformation over a finite field. *Linear Algebra Appl.*, 36:141–155, 1981.

[46] Maska Law. Flocks, generalizes quadrangles and translation planes from BLT-sets, Ph.D. thesis, University of Western Australia, 2003.

[47] Maska Law and Tim Penttila. Classification of flocks of the quadratic cone over fields of order at most 29. Number suppl., pages S232–S244. 2003. Special issue dedicated to Adriano Barlotti.

[48] Maska Law and Tim Penttila. Construction of BLT-sets over small fields. *European J. Combin.*, 25(1):1–22, 2004.

[49] Wen-Ch'ing Winnie Li. Character sums and abelian Ramanujan graphs. *J. Number Theory*, 41(2):199–217, 1992.

[50] A. Lubotzky, R. Phillips, and P. Sarnak. Ramanujan graphs. *Combinatorica*, 8(3):261–277, 1988.

[51] Heinz Lüneburg. *On the rational normal form of endomorphisms.* Bibliographisches Institut, Mannheim, 1987. A primer to constructive algebra.

[52] L. Lunelli and M. Sce. *k-archi completi nei piani proiettivi desarguesiani di rango* 8 *e* 16. Centro di Calcoli Numerici, Politecnico di Milano, Milan, 1958.

[53] 2021. Maple 21. Maplesoft, a division of Waterloo Maple Inc., Waterloo, Ontario. 2021.

[54] Brendan McKay, 1995. Possolve, A program to solve positive integer systems, Australian National University, private communication.

[55] Brendan McKay, 2020. Nauty and Traces (Version 2.7r1), Australian National University, 2020.

[56] G. Eric Moorhouse. Projective Planes of Order 16. `http://ericmoorhouse.org/pub/planes16/`.

[57] G. Eric Moorhouse. Projective Planes of Order 27. `http://ericmoorhouse.org/pub/planes27/`.

[58] M. L. Narayana Rao and K. Kuppuswamy Rao. A new flag transitive affine plane of order 27. *Proc. Amer. Math. Soc.*, 59(2):337–345, 1976.

[59] T. Penttila. Regular cyclic BLT-sets. Number 53, pages 167–172. 1998. Combinatorics '98 (Mondello).

[60] Tim Penttila and Gordon F. Royle. BLT-sets over small fields. *Australas. J. Combin.*, 17:295–307, 1998.

[61] Wilhelm Plesken. Counting with groups and rings. *J. Reine Angew. Math.*, 334:40–68, 1982.

[62] 2021. POV-RAY Developers. POV-RAY, Persistence of Vision Raytracer Pty. Ltd. `http://povray.org`, accessed 1/23/2021.

[63] L. Schläfli, 1858. An attempt to determine the twenty-seven lines upon a surface of the third order and to divide such surfaces into species in reference to the reality of the lines upon the surface, *Quart. J. Math.* **2** (1858), 55–110.

[64] Bernd Schmalz. Verwendung von Untergruppenleitern zur Bestimmung von Doppelnebenklassen. *Bayreuth. Math. Schr.*, (31):109–143, 1990.

[65] Ákos Seress. *Permutation group algorithms*, volume 152 of *Cambridge Tracts in Mathematics*. Cambridge University Press, Cambridge, 2003.

[66] William Stein and David Joyner. SAGE: System for algebra and geometry experimentation. *ACM SIGSAM Bulletin*, 39(2):61–64, 2005. http://www.sagemath.org/files/sage_stein2005.pdf.

[67] Ibrahim A. I. Suleiman and Robert A. Wilson. The 2-modular characters of Conway's third group Co_3. *J. Symbolic Comput.*, 24(3-4):493–506, 1997.

[68] Magma system. Magma Calculator http://magma.maths.usyd.edu.au/calc/, accessed 11/24/2019.

[69] Till Tantau. The TikZ and PGF Packages. Manual for Version 3.1.9a – 2022 https://tikz.dev.

[70] D.E. Taylor. *The geometry of the classical groups*, volume 9 of *Sigma Series in Pure Mathematics*. Heldermann Verlag, Berlin, 1992.

[71] Michael Walker. A class of translation planes. *Geometriae Dedicata*, 5(2):135–146, 1976.

[72] Wikipedia. Bmp file format. https://en.wikipedia.org/wiki/BMP_file_format.

[73] Robert Wilson, Peter Walsh, Jonathan Tripp, Ibrahim Suleiman, Richard Parker, Simon Norton, Simon Nickerson, Steve Linton, John Bray, and Rachel Abbott, 2019. ATLAS of Finite Group Representations - Version 3, http://brauer.maths.qmul.ac.uk/Atlas/v3/, accessed 11/24/2019.