User’s Guide
Build Number 3006

Anton Betten

December 1, 2022
# Contents

1 Introduction 7  
   1.1 What is Orbiter .................................................. 7

2 Getting Started 9  
   2.1 Running and Installing Orbiter .................................. 9
   2.2 The Orbiter Session .............................................. 14
   2.3 Makefiles and Shell Scripts ..................................... 17
   2.4 Objects and Activities ........................................... 20
   2.5 Mathematical Data ................................................ 25
   2.6 Set Builder ....................................................... 27
   2.7 Vector Builders .................................................. 28
   2.8 Formula Builders ............................................... 32

3 Basic Algebra 35  
   3.1 Basic Number Theory ............................................ 35
   3.2 Finite Fields ..................................................... 42
   3.3 Extension Fields ................................................ 47
   3.4 Linear Algebra Over Finite Fields ............................. 54
   3.5 Advanced Topics in Finite Fields ............................... 60
   3.6 Basic Ring Theory ............................................ 69

4 Geometry 71  
   4.1 Finite Projective Spaces ......................................... 71
   4.2 Indexing Points and Lines ...................................... 78
   4.3 Finite Desarguesian Projective Planes ......................... 83
   4.4 The Grassmannian ............................................... 87
   4.5 Algebraic Sets .................................................. 90
   4.6 The Klein Quadric and the Plücker Map ....................... 93
   4.7 Orthogonal Spaces ............................................. 95
   4.8 Hermitian Varieties ........................................... 104
   4.9 Advanced Topics ............................................... 106
   4.10 Geometric Objects ............................................ 113
5 **Group Theory**

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Permutation Groups</td>
<td>119</td>
</tr>
<tr>
<td>5.2</td>
<td>Linear Groups</td>
<td>125</td>
</tr>
<tr>
<td>5.3</td>
<td>Subgroups</td>
<td>143</td>
</tr>
<tr>
<td>5.4</td>
<td>Linear Groups, Advanced Topics</td>
<td>155</td>
</tr>
<tr>
<td>5.5</td>
<td>Induced Actions</td>
<td>159</td>
</tr>
<tr>
<td>5.6</td>
<td>Group Theoretic Activities</td>
<td>170</td>
</tr>
<tr>
<td>5.7</td>
<td>Group Theoretic Activities Based on Magma</td>
<td>180</td>
</tr>
</tbody>
</table>

6 **Orbit Algorithms**

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>Schreier Trees</td>
<td>187</td>
</tr>
<tr>
<td>6.2</td>
<td>Poset Classification</td>
<td>195</td>
</tr>
<tr>
<td>6.3</td>
<td>Orbits on Subsets</td>
<td>198</td>
</tr>
<tr>
<td>6.4</td>
<td>Orbits on Subspaces</td>
<td>203</td>
</tr>
<tr>
<td>6.5</td>
<td>Orbits on Set-Partitions</td>
<td>205</td>
</tr>
<tr>
<td>6.6</td>
<td>Arcs and Caps in Projective Spaces</td>
<td>207</td>
</tr>
<tr>
<td>6.7</td>
<td>Cubic Curves</td>
<td>212</td>
</tr>
</tbody>
</table>

7 **Cubic Surfaces**

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1</td>
<td>Creation</td>
<td>213</td>
</tr>
<tr>
<td>7.2</td>
<td>Quartic Curves</td>
<td>221</td>
</tr>
<tr>
<td>7.3</td>
<td>Classification</td>
<td>235</td>
</tr>
<tr>
<td>7.4</td>
<td>Isomorphism Testing and Recognition</td>
<td>250</td>
</tr>
<tr>
<td>7.5</td>
<td>Dickson Surfaces</td>
<td>254</td>
</tr>
<tr>
<td>7.6</td>
<td>ATLAS and Tables</td>
<td>256</td>
</tr>
</tbody>
</table>

8 **Ring Theory**

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1</td>
<td>Polynomials Over Finite Fields</td>
<td>257</td>
</tr>
<tr>
<td>8.2</td>
<td>Multivariate Polynomial Rings</td>
<td>266</td>
</tr>
</tbody>
</table>

9 **Applications**

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.1</td>
<td>Number Theory</td>
<td>275</td>
</tr>
<tr>
<td>9.2</td>
<td>Representation Theory</td>
<td>278</td>
</tr>
<tr>
<td>9.3</td>
<td>Cryptography</td>
<td>280</td>
</tr>
</tbody>
</table>

10 **Coding Theory**

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1</td>
<td>Introduction</td>
<td>289</td>
</tr>
<tr>
<td>10.2</td>
<td>Linear Codes</td>
<td>296</td>
</tr>
<tr>
<td>10.3</td>
<td>Golay Codes</td>
<td>304</td>
</tr>
<tr>
<td>10.4</td>
<td>CRC Codes</td>
<td>305</td>
</tr>
<tr>
<td>10.5</td>
<td>Reed-Muller Codes</td>
<td>313</td>
</tr>
<tr>
<td>10.6</td>
<td>BCH Codes</td>
<td>315</td>
</tr>
<tr>
<td>10.7</td>
<td>Reed-Solomon Codes</td>
<td>322</td>
</tr>
<tr>
<td>10.8</td>
<td>Bounds</td>
<td>327</td>
</tr>
<tr>
<td>10.9</td>
<td>Classification of Optimal Linear Codes</td>
<td>329</td>
</tr>
</tbody>
</table>
11 Combinatorics ........................................ 333
  11.1 Introduction .................................. 333
  11.2 Diophantine Systems ......................... 340
  11.3 Combinatorial Linear Spaces .................. 344
  11.4 Classification of Configurations and Geometries 347
  11.5 Design Theory ................................ 356
  11.6 Design Theory – Large Sets .................... 361
  11.7 Design Theory – Delandtsheer-Doyen .......... 363
  11.8 Tactical Decompositions ....................... 365

12 Finite Geometry .................................... 369
  12.1 Spreads .................................... 369
  12.2 Translation Planes ............................ 383
  12.3 Packings ................................... 388
  12.4 BLT-Sets .................................. 391

13 Graph Theory ...................................... 399
  13.1 Creating Graphs ............................... 399
  13.2 Graph Theoretic Activities ..................... 406
  13.3 Classification of Graphs and Tournaments .... 412
  13.4 Clique Finding ................................ 415

14 Combinatorial Objects ............................. 419
  14.1 Combinatorial Objects ......................... 419
  14.2 Encoding Combinatorial Objects ................. 423

15 Canonical Forms with Nauty ....................... 427
  15.1 Overview of Canonical Forms ................... 427
  15.2 Canonical Forms of Objects in Projective Space 429
  15.3 Canonical Forms of Incidence Geometries ...... 436
  15.4 Canonical Forms of Objects from Design Theory 449
  15.5 Canonical Forms of Linear Codes ............... 453
  15.6 Canonical Forms of General Codes ............. 460
  15.7 Canonical Forms of Graphs ..................... 461
  15.8 Canonical Forms of Quartic Curves ............ 466

16 Interfaces .......................................... 469
  16.1 Graphical Output ............................. 469
  16.2 The Povray Interface .......................... 476
  16.3 Creating Animations ......................... 489
  16.4 Continuous Function Plotter .................... 490

17 Miscellaneous ..................................... 495
  17.1 Miscellaneous ................................ 495
  17.2 Limitations .................................. 498
Chapter 1

Introduction

1.1 What is 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. Orbiter is optimized for efficiency in terms of memory and execution speed. Orbiter is a library of C++ classes, together with a command line driven front end. There is no graphical user interface. The system offers two modes of use, programming or command line interface. This manual is about the command line interface. Readers who are interested in the Orbiter C++ class library should consult the programmer’s guide. A makefile with all commands used in this guide can be found in the examples subdirectory. For background on Orbiter, see [9].
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. [10]). Docker [25] is a system to run preconfigured software in an encapsulated way on various platforms, including Windows. We describe using Orbiter through unix *makefiles*, which are run through the tool *make* (cf. [28]). This is a software tool that allows collecting short command snippets 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 [21]. 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:

```
    orbiter
    └── work
```

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 [28].

A second way to run Orbiter is through Docker [25]. 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 in Figure 2.1. 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).

```
MY_PATH=~/DEV.22/orbiter
#MY_PATH=/scratch/betten/COMPILE/orbiter

ORBITER_PATH=$(MY_PATH)/src/apps/orbiter/
ORBITER=$(ORBITER_PATH)orbiter.out
SANDBOX=$(MY_PATH)/src/apps/sandbox/sandbox.out
```

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](http://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

This will produce an output similar to the following:

sh-3.2$ docker run -it --volume $PWD:/mnt -w /mnt abetten/orbiter orbiter.out
Unable to find image 'abetten/orbiter:latest' locally
latest: Pulling from abetten/orbiter
004f1ee87df: Pull complete
5d6f1e8117db: Pull complete
48c2fa6f66abe: Pull complete
234b70d0479d: Pull complete
6fa07a00e2f0: Pull complete
9187bd98e241: Pull complete
ae87b7ef500b: Pull complete
260a2765fa99: Pull complete
27d6fff93a58: Pull complete
7a09ec574418: Pull complete
1336494f74e1: Pull complete
Digest: sha256:889099d7e0b0a9ee168b7cb261d2da8ff64bd7d861c357e1caec59580d629ee9
Status: Downloaded newer image for abetten/orbiter:latest
Welcome to Orbiter! Your build number is 1311.
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
SYSTEMUNIX is defined
sizeof(int)=4
sizeof(long int)=8
Orbiter session finished.
User time: 0:00

The first part is Docker downloading Orbiter as a container. This can take a while, depending on the Internet speed. The second part (Welcome to Orbiter!) is the actual Orbiter session. No specific commands were given, so Orbiter simply starts up and quits. The first part is done only once. 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 (gnc and make). Windows users may have to install cygwin (plus the extra packages git, make, gnc). 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 find 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 Figure 2.2). 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`. 

---

Figure 2.2: GitHub Orbiter Page
2.2 The Orbiter Session

The orbiter workflow is depicted in Figure 2.3. Commands are issued through the command line, which invokes Orbiter sessions, which in turn perform the required computations and read and write data to files. The commands are parsed and separated into three basic types. Commands that create objects, commands that apply to previously created objects, and all other commands. Objects are maintained in a symbol table. The command line calls to Orbiter may or may not be organized in the form of makefiles, as discussed in 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:

```makefile
 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
```
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-v</td>
<td>v</td>
<td>Set verbosity to ( v ). Larger values of ( v ) lead to more text output. ( v = 0 ) gives minimal output.</td>
</tr>
<tr>
<td>-list_arguments</td>
<td></td>
<td>Prints the command line arguments.</td>
</tr>
<tr>
<td>-seed</td>
<td>( s )</td>
<td>Seed the pseudo random number generator with the integer value ( s ).</td>
</tr>
<tr>
<td>-memory_debug</td>
<td></td>
<td>Turn on dynamic memory debugging.</td>
</tr>
<tr>
<td>-override_polynomial</td>
<td>poly</td>
<td>Set the override polynomial for finite fields to poly.</td>
</tr>
<tr>
<td>-orbiter_path</td>
<td>( p )</td>
<td>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 “/.”</td>
</tr>
<tr>
<td>-magma_path</td>
<td>( p )</td>
<td>Set the magma path to ( p ). This is useful in case the Orbiter session has to create a magma process.</td>
</tr>
<tr>
<td>-fork ( L M f t s )</td>
<td></td>
<td>Fork new Orbiter sessions in parallel. The new sessions will be indexed by the values ( i ) that result from a loop with start value ( f ) and increment ( s ) bounded from above by ( t ), equivalent to a C-loop of type “for (i=f; i &lt; t; i+= s).” Every occurrence 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 whose name is described through the mask ( M ). The actual file name results from using the printf command from the C-library for ( M ) with the integer value of the loop variable. 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 ((t - f)/s). The orbiter path from -orbiter_path is used when starting the forked sessions.</td>
</tr>
</tbody>
</table>

Table 2.1: Orbiter session commands
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.


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 means 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 19.1. 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 executed 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
The purpose of this command is to produce a file called

\[\text{GF} \_16.\text{tex}\]

which can then be processed through \text{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 \text{F} \_16. We can create a makefile like this:

\begin{verbatim}
F_16:
    $(ORBITER) -v 3 \
    -define F -finite_field -q 16 -end \
    -with F -do -finite_field_activity -cheat_sheet_GF -end
    pdflatex GF_16.tex
\end{verbatim}

With this file present, type the terminal command \texttt{make F}_\texttt{16} to execute the two line Orbiter command. Windows users can use \texttt{SuperShell}. The program \texttt{make} will look for the file \texttt{makefile} in the current directory. Once found, it will search for the label \texttt{F}_\texttt{16} in it and execute the commands beneath it. The given commands will invoke Orbiter and produce the \texttt{GF}\_\texttt{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 19.1. 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 \texttt{ORBITER\_PATH} below.

One difficulty in installing Orbiter is the path of installation. In the sample makefile, there is a makefile variable called \texttt{ORBITER\_PATH} which contains the path to the orbiter executable \texttt{orbiter.exe}. Depending on the local installation of orbiter, the makefile variable needs to be changed accordingly. The actual command to run the \texttt{F}\_\texttt{16} example is as follows:

\begin{verbatim}
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
\end{verbatim}
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

```plaintext
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

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

to

```plaintext
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?

Unfortunately, the answer is complicated. 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

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

creates a finite field object \( F \) for the field with two elements (see Section 3.2). The command

```
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 \( F_2 \) and uses it to construct \( PG(3,2) \). The -projective_space command requires additional options to set the dimension \( n \) and the field \( \mathbb{F}_q \) in \( PG(n,q) \). The -n command sets the dimension \( n \). The -field command can be used to specify a particular 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
```
<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-finite_field</td>
<td>A finite field $\mathbb{F}_q$. See Sections 3.2 and 3.3.</td>
</tr>
<tr>
<td>-polynomial_ring</td>
<td>A multivariate polynomial ring. See Section 8.2.</td>
</tr>
<tr>
<td>-linear_group</td>
<td>A linear group. See Section 5.2.</td>
</tr>
<tr>
<td>-permutation_group</td>
<td>A permutation group. See Section 5.1.</td>
</tr>
<tr>
<td>-projective_space</td>
<td>A projective space $\text{PG}(n, q)$. See Section 4.1.</td>
</tr>
<tr>
<td>-orthogonal_space</td>
<td>A non-degenerate orthogonal space $O^*(n, q)$. See Section 4.7.</td>
</tr>
<tr>
<td>-BLT_set_classify</td>
<td>An object to classify BLT-sets. See Section 12.4.</td>
</tr>
<tr>
<td>-spread_classify</td>
<td>An object to classify spreads. See Section 12.1.</td>
</tr>
<tr>
<td>-formula</td>
<td>An algebraic / symbolic expression. See Section 2.8.</td>
</tr>
<tr>
<td>-cubic_surface</td>
<td>A cubic surface. See Section 7.1.</td>
</tr>
<tr>
<td>-quartic_curve</td>
<td>A quartic curve. See Section 7.2.</td>
</tr>
<tr>
<td>-classification_of_cubic_surfaces_with_double_sixes</td>
<td>An object to classify cubic surfaces using double sixes. See Section 7.3.</td>
</tr>
<tr>
<td>-collection</td>
<td>A collection of objects.</td>
</tr>
<tr>
<td>-geometric_object</td>
<td>A geometric object. See Section 4.10.</td>
</tr>
<tr>
<td>-graph</td>
<td>A graph. See Section 13.1.</td>
</tr>
<tr>
<td>-code</td>
<td>A code. See Section 10.2.</td>
</tr>
<tr>
<td>-spread</td>
<td>A spread. See Section 12.1.</td>
</tr>
</tbody>
</table>

Table 2.2: Orbiter Objects (Part I)
<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-translation_plane</td>
<td>A translation plane. See Section 12.2.</td>
</tr>
<tr>
<td>-spread_table</td>
<td>A table of spreads. See Section 12.3.</td>
</tr>
<tr>
<td>-packing_with_symmetry_assumption</td>
<td>A generator for packings with assumed symmetry. See Section 12.3.</td>
</tr>
</tbody>
</table>
| -packing_choose_fixed_points                | A selection of fixed orbits for packings with assumed symmetry. See Sec-
|                                              |    tion 12.3.                                                           |
| -packing_long_orbits                        | A search for long orbits for packings with assumed symmetry. See Sec-
|                                              |    tion 12.3.                                                           |
| -graph_classification                       | An object which allows classifying graphs and tournaments. See Section 1
|                                              |    3.3.                                                                 |
| -diophant                                   | A diophantine system, i.e., a system of positive integer equations). See
|                                              |    Section 11.2.                                                        |
| -design                                      | A combinatorial design. See Section 11.5.                               |
| -design_table                               | A table of designs. It can be used to construct large sets of designs. A
|                                              |    large set is a set of designs satisfying certain properties. See Sec-
|                                              |    tion 11.5.                                                           |
| -large_set_with_symmetry_assumption         | An object to create a large set of designs. See Section 11.5.           |
| -set                                         | A set. See Section 2.6.                                                |
| -vector                                      | A vector over a finite field. See Section 2.7.                         |
| -geometry_builder                           | An object to classify incidence geometries. See Section 11.4.           |
| -vector_ge                                   | A vector of group elements. See Section 5.3.                           |

Table 2.3: Orbiter Objects (Part II)
command sequence is used. Here, \textit{LABEL} is the name under which the object is registered in the symbol table. \textit{DESCRIPTION} is the activity that should be applied. Some activities require more than one object, in which case the syntax

\begin{verbatim}
  -with \textit{LABEL1} -and \textit{LABEL2} -do \textit{DESCRIPTION} -end
\end{verbatim}

is used. Here, \textit{LABEL1} and \textit{LABEL2} are the objects for which the activity is invoked. For an example of an activity requiring two objects, see Sections 12.1 and 12.2.

Table 2.4 list the possible activities for Orbiter objects. More details will be given in the later sections of this guide.
<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-finite_field_activity</td>
<td>An activity for finite fields, see Sections 3.2 and 3.3.</td>
</tr>
<tr>
<td>-projective_space_activity</td>
<td>An activity for a projective space, see Section 4.1.</td>
</tr>
<tr>
<td>-orthogonal_space_activity</td>
<td>An activity for an orthogonal space, see Section 4.7.</td>
</tr>
<tr>
<td>-group_theoretic_activity</td>
<td>An activity for a group, see Section 5.6.</td>
</tr>
<tr>
<td>-cubic_surface_activity</td>
<td>An activity for a cubic surface, see Section 7.1.</td>
</tr>
<tr>
<td>-quartic_curve_activity</td>
<td>An activity for a quartic curve, see Section 7.2.</td>
</tr>
<tr>
<td>-combinatorial_object_activity</td>
<td>An activity for a combinatorial object, see Section 4.5.</td>
</tr>
<tr>
<td>-graph_theoretic_activity</td>
<td>An activity for a graph, see Section 13.1.</td>
</tr>
<tr>
<td>-classification_of_cubic_surfaces_with_double_sixes_activity</td>
<td>An activity for a cubic surface, see Section 7.1.</td>
</tr>
<tr>
<td>-spread_table_activity</td>
<td>An activity associated with a table of spreads, see Section 12.3.</td>
</tr>
<tr>
<td>-packing_with_symmetry_assumption_activity</td>
<td>An activity related to creating packings with assumed symmetry group, see Section 12.3.</td>
</tr>
<tr>
<td>-packing_fixed_points_activity</td>
<td>An activity related to creating packings with assumed symmetry group, see Section 12.3.</td>
</tr>
<tr>
<td>-graph_classification_activity</td>
<td>An activity for a classification of graphs problem, see Section 13.3.</td>
</tr>
<tr>
<td>-diophant_activity</td>
<td>An activity for a diophantine system, see Section 11.2.</td>
</tr>
<tr>
<td>-design_activity</td>
<td>An activity for a combinatorial design, see Section 11.5.</td>
</tr>
<tr>
<td>-large_set_with_symmetry_assumption_activity</td>
<td>An activity related to creating large sets of designs with assumed symmetry group, see Section 11.6.</td>
</tr>
</tbody>
</table>

Table 2.4: Orbiter Activities
<table>
<thead>
<tr>
<th>Object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLT-sets</td>
<td>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 $q \leq 73$.</td>
</tr>
<tr>
<td>Cubic Surfaces</td>
<td>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 $q \leq 128$.</td>
</tr>
<tr>
<td>Quartic curves</td>
<td>Orbiter knows the classification of smooth quartic curves with 28 bitangents in projective planes over all fields $\mathbb{F}_q$ for $q = 9, 13, 19, 25, 27, 29, 31$.</td>
</tr>
<tr>
<td>Spreads</td>
<td>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 $(q, k) \in {(2, 2), (3, 2), (2, 4), (4, 2), (5, 2), (3, 3)}$.</td>
</tr>
</tbody>
</table>
| Hyperovals   | A hyperoval in $\text{PG}(2, 2^e)$ 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 DH($k, n$). Orbiter knows the classification of dual hyperovals DH(4, 7) and DH(4, 8). |
| Packings     | Orbiter knows the classification of packings of $\text{PG}(3, 3)$.                                                                           |

Table 2.5: Mathematical Data Available 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 2.5. 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

```plaintext
create_BLT_5_1:
  ↳ $(\text{ORBITER}) -v 2 \$
  ↳  -define F -finite_field -q 5 -end \$
  ↳  -define O -orthogonal_space 0 5 F -end \$
```
recalls the BLT-set with Orbiter Catalogue Number 1 in $Q(4,5)$. A latex report
\texttt{catalogue\_q5\_iso1.tex} is written. For more details about BLT-sets, see Section 12.4.

The command

\begin{verbatim}
create_surface_4_0:
  $(ORBITER) -v 3 \\
  -with 0 -do -orthogonal_space_activity \\
  -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
\end{verbatim}

recalls the cubic surface with Orbiter Catalogue Number 0 in $PG(3,4)$. A latex report
\texttt{surface\_catalogue\_q4\_iso0\_report.tex} is written. For more details about cubic surfaces, see Section 7.1.
2.6 Set Builder

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

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

The next command creates the 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, the end value plus one, and the increment.

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

For C programmers, `-loop a b c` is equivalent to

```
for (i=a; i < b; i += c) {
}
```
2.7 Vector Builders

Orbiter allows to create objects of type vector. A vector is simply a data structure for a sequence of integers. It is similar to an array in a programming language. Orbiter does not force any kind of typing. The same vector can have many different meanings. For instance, indexing allows us to identify different types of objects with integers. For instance, a vector could be considered as a vector of elements of a finite field. This is because in Orbiter, finite field elements are ranked and represented as integers.

There are two different ways to define a vector, called dense and sparse format. 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 together with a field object, then Orbiter will force the vector entries to lie in the interval \([0, q - 1]\), where \(q\) is the order of the finite field. Otherwise, they can be any integer values. Note that there are limitations due to the word size of the machine and the processor. Most machines today have 64 bits, so any integer of absolute value less than \(2^{63} - 1\) can be represented (recall that the sign takes one bit away). On an older style 32 bit machine, only integers whose absolute value is less that \(2^{31} - 1\) can be stored. For more about limitations, see Section 17.2.

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.

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. In this case, the -dense option should not be used. 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 \(2 \times 3\) matrix over \(\mathbb{F}_5\):

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

28
For larger matrices, we can use makefile variables. For instance, the following command creates the generator matrix of the Hamming code:

```
HAMMING_CODE_GENERATOR="\n1,0,0,0,0,1,1, \n0,1,0,0,1,0,1, \n0,0,1,0,1,1,0, \n0,0,0,1,1,1"n
```

```
matrix_example1:
▷ $(ORBITER) -v 2 \n▷ -define F -finite_field -q 2 -end \n▷ -define v -vector -field F -format 4 \n▷ -dense $(HAMMING_CODE_GENERATOR) -end \n▷ -print_symbols
```

For large matrices over small fields, the `-compact` option can be given (instead of `-dense`). For instance, the following code creates a $22 \times 22$ matrix over the binary field:

```
CONWAY_GEN1="\n110111000100001010000\n111010111101000101101\n00000100000100010101\n11111001101000101110\n0101010000010011101\n00000100000100010101\n0010000000010010101\n0001000110000011111\n1110100110100011001\n0000000000000010101\n00000000001001000101\n0110111110101011101\n0000000000010100101\n0001000110000011111\n1110100110110010101\n0000000000000011101\n00000000000100100101\n0110111110101011111\n0000000000010100101\n0001000110000011111\n0000000000010100101\n00000000000100100101\n00000000000100010101\n0000000000000101001\n000000000000010010101\n0000000000000100010101\n```

29
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

Using the dense option, spaces in the input string are ignored. For large vectors, the sparse command can be used to enter non-zero coefficients as a list of pairs. For instance,

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

```
1 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 1
```

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.

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

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

$$(1,1,1,1,1,1,1,1,1,1,1)$$. 
2.8 Formula Builders

Orbiter can parse symbolic formulas from a minimalistic grammar. Here is an example. The formula is defined as a makefile variable:

\[ \text{TEST\_FORMULA} = "(-a+b*b)*x*x+a*b*x" \]

The command

\[
\text{formula\_example:}
\]
\[ \text{\$\{ORBITER\} -v 3 \}
\]
\[ \text{\> \> -define f -formula \}
\]
\[ \text{\> \> \> "test\_formula" "test\_formula" "" \}
\]
\[ \text{\> \> \> $(TEST\_FORMULA)}
\]
\[ \text{\> \> dot -Tpng test\_formula.gv >test\_formula.png}
\]
\[ \text{\> \> open test\_formula.png}
\]

parses the formula and produces an abstract syntax tree. The tree is exported in graphviz format, and can be processed using the dot command. The graphical representation of the abstract syntax tree is shown in Figure 2.4.

The next example evaluates the formula over the field $\mathbb{F}_5$, using the assignment $a = 2, b = 3, x = 4$:

\[
\text{formula\_evaluate:}
\]
\[ \text{\$\{ORBITER\} -v 3 \} \]
-define F -finite_field -q 5 -end \\n-define f -formula \\n  "test\_formula" "test\_\_formula" "" \\n  $(TEST\_FORMULA) \\n-with F -do -finite_field\_activity \\n-evaluate f "a=2,b=3,x=4" -end
Chapter 3

Basic Algebra

3.1 Basic Number Theory

Table 3.1 shows Orbiter commands for basic number theory, including integer factor rings and the Euclidean algorithm. To compute primitive roots, the \texttt{-primitive\_root} command can be used. The algorithm is randomized. For instance,

\texttt{PR29:}
\begin{verbatim}
 &>(ORBITER) -v 1 -smallest\_primitive\_root 29
\end{verbatim}

computes a primitive root modulo 29. The answer in this case is 2. For a large example, consider

\texttt{PR\_915839:}
\begin{verbatim}
 &>(ORBITER) -v 5 -primitive\_root 915839
\end{verbatim}

which computes a primitive root modulo 915839. The answer is 43085. The command

\texttt{PR\_915839\_check:}
\begin{verbatim}
 &>(ORBITER) -v 5 -power\_mod 43085 49842 915839
\end{verbatim}

computes

\[43085^{49842} \mod 915839\]

which is 487320.

The command \texttt{-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
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-power_mod</td>
<td>$a \ n \ p$</td>
<td>Computes $a^n \mod p$.</td>
</tr>
<tr>
<td>-discrete_log</td>
<td>$b \ a \ p$</td>
<td>Computes $n$ such that $a^n \equiv b \mod p$.</td>
</tr>
<tr>
<td>-extended_gcd</td>
<td>$a \ b$</td>
<td>Computes integers $g$, $u$, and $v$ such that $g = \gcd(a, b) = ua + vb$.</td>
</tr>
<tr>
<td>-square_root_mod</td>
<td>$a \ p$</td>
<td>Computes a square root of $a$ modulo $p$, i.e. an integer $b$ such that $b^2 \equiv a \mod p$.</td>
</tr>
<tr>
<td>-square_root</td>
<td>$a$</td>
<td>Computes $\lfloor \sqrt{a} \rfloor$ of an integer $a$.</td>
</tr>
<tr>
<td>-inverse_mod</td>
<td>$a \ p$</td>
<td>Computes the modular inverse of $a$ modulo $p$, i.e. an integer $b$ with $ab \equiv 1 \mod p$.</td>
</tr>
<tr>
<td>-draw_mod_n</td>
<td>descr</td>
<td>Draws the integers modulo $n$ on a circle.</td>
</tr>
<tr>
<td>-order_of_q_mod_n</td>
<td>$q \ n_{\min} \ n_{\max}$</td>
<td>Computes the order $\text{ord}(q,n)$ of $q$ modulo $n$ for all $n$ with $n_{\min} \leq n \leq n_{\max}$ for which $\gcd(n,q) = 1$. Also computes $\varphi(n)$ and $\varphi(n)/\text{ord}(q,n)$.</td>
</tr>
<tr>
<td>-Chinese_remainders</td>
<td>R M</td>
<td>Solves a system of congruences with remainders R and moduli M. R and M must be vectors whose labels are given.</td>
</tr>
</tbody>
</table>

Table 3.1: Basic Number Theory Commands
This command can be quite expensive.

Computing inverses modulo a prime $p$ is possible using the \texttt{-inverse\_mod} command. The command

\begin{verbatim}
IM:
\texttt{$(ORBITER) -v 5 \text{-inverse\_mod 1865025205 2147483647}$}
\end{verbatim}

computes the inverse of 1865025205 modulo 2147483647 which is 579785381.

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

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

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

\begin{verbatim}
IM.gcd:
\texttt{$(ORBITER) -v 5 \text{-extended\_gcd 1865025205 2147483647}$}
\end{verbatim}

The output is

$$1 = -503526232 \times 2147483647 + 579785381 \times 1865025205,$$

from which we see that $\text{gcd}(a, b) = 1$ and $u = -503526232$ and $v = 579785381$, which is the \texttt{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 \texttt{-power\_mod} command can be used. For instance,

\begin{verbatim}
PM3a:
\texttt{$(ORBITER) -v 5 \text{-power\_mod 16807 1073741823 2147483647}$}
\end{verbatim}

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

The modular square root of $a$ modulo $p$ is any $x$ in

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

The command \texttt{-square\_root\_mod} can be used to compute modular square roots using the algorithm of Tonelli and Shanks (cf. [19]). For instance,
Table 3.2: The order of 2 modulo $n$

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

$sqrt\mod$:

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

```
$($ORBITER$) -v 3 -order_of_q_mod_n 2 3 151

$($ORBITER$) -v 1 -csv_file_latex 1 \
$($ORBITER$) -v order_of_q_mod_n_q2_3_151.csv

pdflatex order_of_q_mod_n_q2_3_151.tex

open order_of_q_mod_n_q2_3_151.pdf
```

produces the output shown in Table 3.2.

The command
Table 3.3: The values of the Eulerfunction

<table>
<thead>
<tr>
<th>N</th>
<th>PHI</th>
<th>N</th>
<th>PHI</th>
<th>N</th>
<th>PHI</th>
<th>N</th>
<th>PHI</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>26</td>
<td>12</td>
<td>51</td>
<td>32</td>
<td>76</td>
<td>36</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>27</td>
<td>18</td>
<td>52</td>
<td>24</td>
<td>77</td>
<td>60</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>28</td>
<td>12</td>
<td>53</td>
<td>52</td>
<td>78</td>
<td>24</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>29</td>
<td>28</td>
<td>54</td>
<td>18</td>
<td>79</td>
<td>78</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>30</td>
<td>8</td>
<td>55</td>
<td>40</td>
<td>80</td>
<td>32</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>31</td>
<td>30</td>
<td>56</td>
<td>24</td>
<td>81</td>
<td>54</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>32</td>
<td>16</td>
<td>57</td>
<td>36</td>
<td>82</td>
<td>40</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>33</td>
<td>20</td>
<td>58</td>
<td>28</td>
<td>83</td>
<td>82</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
<td>34</td>
<td>16</td>
<td>59</td>
<td>58</td>
<td>84</td>
<td>24</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>35</td>
<td>24</td>
<td>60</td>
<td>16</td>
<td>85</td>
<td>64</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>36</td>
<td>12</td>
<td>61</td>
<td>60</td>
<td>86</td>
<td>42</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>37</td>
<td>36</td>
<td>62</td>
<td>30</td>
<td>87</td>
<td>56</td>
</tr>
<tr>
<td>13</td>
<td>12</td>
<td>38</td>
<td>18</td>
<td>63</td>
<td>36</td>
<td>88</td>
<td>40</td>
</tr>
<tr>
<td>14</td>
<td>6</td>
<td>39</td>
<td>24</td>
<td>64</td>
<td>32</td>
<td>89</td>
<td>88</td>
</tr>
<tr>
<td>15</td>
<td>8</td>
<td>40</td>
<td>16</td>
<td>65</td>
<td>48</td>
<td>90</td>
<td>24</td>
</tr>
<tr>
<td>16</td>
<td>8</td>
<td>41</td>
<td>40</td>
<td>66</td>
<td>20</td>
<td>91</td>
<td>72</td>
</tr>
<tr>
<td>17</td>
<td>16</td>
<td>42</td>
<td>12</td>
<td>67</td>
<td>66</td>
<td>92</td>
<td>44</td>
</tr>
<tr>
<td>18</td>
<td>6</td>
<td>43</td>
<td>42</td>
<td>68</td>
<td>32</td>
<td>93</td>
<td>60</td>
</tr>
<tr>
<td>19</td>
<td>18</td>
<td>44</td>
<td>20</td>
<td>69</td>
<td>44</td>
<td>94</td>
<td>46</td>
</tr>
<tr>
<td>20</td>
<td>8</td>
<td>45</td>
<td>24</td>
<td>70</td>
<td>24</td>
<td>95</td>
<td>72</td>
</tr>
<tr>
<td>21</td>
<td>12</td>
<td>46</td>
<td>22</td>
<td>71</td>
<td>70</td>
<td>96</td>
<td>32</td>
</tr>
<tr>
<td>22</td>
<td>10</td>
<td>47</td>
<td>46</td>
<td>72</td>
<td>24</td>
<td>97</td>
<td>96</td>
</tr>
<tr>
<td>23</td>
<td>22</td>
<td>48</td>
<td>16</td>
<td>73</td>
<td>72</td>
<td>98</td>
<td>42</td>
</tr>
<tr>
<td>24</td>
<td>8</td>
<td>49</td>
<td>42</td>
<td>74</td>
<td>36</td>
<td>99</td>
<td>60</td>
</tr>
<tr>
<td>25</td>
<td>20</td>
<td>50</td>
<td>20</td>
<td>75</td>
<td>40</td>
<td>100</td>
<td>40</td>
</tr>
</tbody>
</table>

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 in Table 3.3.

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 \( a \mapsto a^k \mod 11 \):

\texttt{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 in Table 3.3.
Table 3.4: The function $a \mapsto a^2 \mod 11$

<table>
<thead>
<tr>
<th>A</th>
<th>APOWK</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 3.1: Cycle of powers of 2 modulo 13

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

draw_mod_13:
    $\text{(ORBITER)} -v 2 \$
    $\text{-draw_options -embedded -end \$
    $\text{-draw_mod_n -n 13 -file mod_13 -power_cycle 2 -end}$
    $\text{pdflatex mod_13_draw.tex}$
    $\text{open mod_13_draw.pdf}$

uses a 13-gon to represent the elements modulo 13. It also computes the powers of 2 mod 13 and connects consecutive powers in the diagram (see Figure 3.1).

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

\[
x \equiv 2 \mod 5 \\
x \equiv 2 \mod 6 \\
x \equiv 5 \mod 7
\]
The following command creates one vector for the remainders and one for the moduli and then invokes the \texttt{-Chinese_remainders} command.

\begin{verbatim}
Chinese_remainders_A:
  $ (ORBITER) -v 2 \[
  \langle \text{-define } R \text{-vector -dense } "2,2,5" \text{-end } \rangle \[
  \langle \text{-define } M \text{-vector -dense } "5,6,7" \text{-end } \rangle \[
  \langle \text{-Chinese_remainders } R \text{ } M \rangle 
\end{verbatim}

The answer is \(x \equiv 152 \text{ modulo 210} \).

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

\[
x \equiv 2 \text{ mod } 2147483647 \\
x \equiv 3 \text{ mod } 5915587277
\]

using the command

\begin{verbatim}
Chinese_remainders_C2:
  $ (ORBITER) -v 2 \[
  \langle \text{-define } R \text{-vector -dense } "2,3" \text{-end } \rangle \[
  \langle \text{-define } M \text{-vector -dense } "2147483647,5915587277" \text{-end } \rangle \[
  \langle \text{-Chinese_remainders } R \text{ } M \rangle 
\end{verbatim}

The answer is

\[
x \equiv 5684294357108828365 \text{ mod } 12703626939758759219.
\]

A quick check with Maple shows that

\[
5684294357108828365 \text{ mod } 2147483647 \equiv 2 \\
5684294357108828365 \text{ mod } 5915587277 \equiv 3
\]

as required.
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-q</td>
<td>q</td>
<td>Specify the order of the field. Here, $q = p^k$ for some prime $p$ and some positive integer $k$.</td>
</tr>
<tr>
<td>-override_polynomial</td>
<td>n</td>
<td>Specify the polynomial used to create the finite field. The polynomial is given as integer, using the base $p$ representation. See Section 3.3.</td>
</tr>
<tr>
<td>-without_tables</td>
<td></td>
<td>Create the field without precomputing the tables.</td>
</tr>
</tbody>
</table>

Table 3.5: Options for Creating Finite Fields

### 3.2 Finite Fields

Let $F_q$ denote the finite field with $q$ elements. Up to isomorphism, there is only one field of order $q$. See Section 17.2 for a list of limitations of Orbiter. A finite field $F_q$ can be created using the `-finite_field` command. Table 3.5 lists Orbiter commands for creating a finite field. For instance,

\[
\text{F}_2: \\
\text{ORBITER} -v 3 -list_arguments \ \\
\text{define F -finite_field -q 2 -end} \ \\
\text{with F -do -finite_field_activity -cheat_sheet_GF -end} \\
pdflatex GF_2.tex \\
open GF_2.pdf
\]

creates the finite field $F_2$ and produces a report for it.

Table 3.6 lists basic Orbiter activities for finite fields. More activities will follow in Section 3.3. The command

\[
\text{F}_7: \\
\text{ORBITER} -v 3 \ \\
\text{define F -finite_field -q 7 -end} \ \\
\text{with F -do -finite_field_activity -cheat_sheet_GF -end} \\
pdflatex GF_7.tex \\
open GF_7.pdf
\]

creates a cheat sheet for $F_7$ as shown below. The element $\alpha$ is a primitive element.
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-cheat_sheet_GF</td>
<td></td>
<td>Produce a cheat sheet in latex which shows information about the field, including addition and multiplication tables.</td>
</tr>
<tr>
<td>-product_of v</td>
<td></td>
<td>Compute the product of all field elements in the vector $v$.</td>
</tr>
<tr>
<td>-sum_of v</td>
<td></td>
<td>Compute the sum of all field elements in the vector $v$.</td>
</tr>
<tr>
<td>-negate v</td>
<td></td>
<td>Negate each field element in the vector $v$.</td>
</tr>
<tr>
<td>-inverse v</td>
<td></td>
<td>Compute the multiplicative inverse of each field element in the vector $v$.</td>
</tr>
<tr>
<td>-power_map k v</td>
<td></td>
<td>Compute the $k$-th power of each field element in the vector $v$.</td>
</tr>
</tbody>
</table>

Table 3.6: Finite Field Activities

\[ Z_i = \log_\alpha (1 + \alpha^i) \]

<table>
<thead>
<tr>
<th>$i$</th>
<th>$\gamma_i$</th>
<th>$-\gamma_i$</th>
<th>$\gamma_i^{-1}$</th>
<th>$\log_\alpha(\gamma_i)$</th>
<th>$\alpha^i$</th>
<th>$Z_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 = 0</td>
<td>0</td>
<td>DNE</td>
<td>DNE</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1 = 1</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>2 = \alpha^2</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3 = \alpha</td>
<td>4</td>
<td>5</td>
<td>1</td>
<td>6</td>
<td>DNE</td>
</tr>
<tr>
<td>4</td>
<td>4 = \alpha^4</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>5 = \alpha^5</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>6 = \alpha^3</td>
<td>1</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

\[
\begin{array}{cccccccc}
+ & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\
0 & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\
1 & 1 & 2 & 3 & 4 & 5 & 6 & 0 \\
2 & 2 & 3 & 4 & 5 & 6 & 0 & 1 \\
3 & 3 & 4 & 5 & 6 & 0 & 1 & 2 \\
4 & 4 & 5 & 6 & 0 & 1 & 2 & 3 \\
5 & 5 & 6 & 0 & 1 & 2 & 3 & 4 \\
6 & 6 & 0 & 1 & 2 & 3 & 4 & 5 \\
\end{array}
\]
Suppose we want to check Wilson’s theorem that the product of all nonzero field elements of negative one. The following command so so, assuming that $p = 11$. 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 to compute the product of these elements. The answer is 10 which is congruent to $-1$ mod 11:

```plaintext
F_11.product_of_all_nonzero_elements:
> $(ORBITER) -v 3 \n
> -define F -finite_field -q 11 -end \n
> -define S -vector -field F -loop 1 11 1 -end \n
> -with F -do -finite_field_activity -product_of S -end
```

Suppose we want to create the Vandermonde matrix whose entries are $x_j^i$. 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.

```plaintext
F_7.vandermonde:
> $(ORBITER) -v 3 \n
> -define F -finite_field -q 7 -end \n
> -with F -do -finite_field_activity \n
> -Vandermonde_matrix \n
> -end
```

The output is shown below. The first matrix is $V = (x_j^i)$. The second matrix is $V^{-1}$.
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 $\alpha$ 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. In Figure 3.2, the addition and multiplication tables of $\mathbb{F}_7$ are shown with respect to the cyclic ordering of elements as $0, 3^0, 3^1, 3^2, \ldots, 3^6 = 0, 1, 3, 2, 6, 4, 5, 1$.

In the second ordering, the addition table of the prime field no longer exhibits cyclic structure.

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. Here is an example. We create the field $\mathbb{F}_{101}$ without precomputed tables:

```
F_101.wo:
▷ $(ORBITER) -v 3 \$
```

45
\begin{verbatim}
\define F -finite_field -q 101 -without_tables -end \n\define F -with F -do -finite_field_activity -cheat_sheet_GF -end
\pdflatex GF_101.tex
\open GF_101.pdf
\end{verbatim}
3.3 Extension Fields

Let $F$ be a field. An extension field of $F$ is any field $E$ which contains $F$. Because $E$ is a vector space over $F$, the dimension of $E/F$ is well-defined. It may be finite or infinite. An example of a field extension is a field of the form $E = F(\alpha)$, where $\alpha$ is any element over $F$. Here, $F(\alpha)$ is the smallest field which contains $F$ and $\alpha$. If $\gamma \in E$ satisfies a polynomial equation with coefficients in $F$, then $\gamma$ is called algebraic over $F$. The minimum polynomial of an element $\gamma$ in $E$ over $F$ is the monic, lowest degree polynomial in $F[X]$ which has $\gamma$ 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 $\alpha$ is. The degree of $E/F$ equals the degree of the minimum polynomial of $\alpha$ over $F$.

In this section, we will consider algebraic extension 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 $|E| = q^e$. 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

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

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

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

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 which has degree less than $e$ and leading coefficient one. By means of identification, we can take these polynomials to be the set of standard representatives of the residue classes. So, for instance, for $q = 4 = 2^2$, we can pick the irreducible polynomial $m(X) = X^2 + X + 1$ over $\mathbb{F}_2$ and have four standard representatives modulo $I(m)$, namely

$$0, \quad 1, \quad X, \quad X + 1.$$

Together, these make up a complete set of representatives of the residue classes modulo $I(m)$, 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

$$
\begin{array}{c|cccc}
 & 0 & 1 & X & X + 1 \\
0 & 0 & 1 & X & X + 1 \\
1 & 1 & 0 & X + 1 & X \\
X & X & X + 1 & 0 & 1 \\
X + 1 & X + 1 & X & 1 & 0
\end{array}
$$

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 $X + 1$:

$$
\begin{align*}
X \cdot X &= X^2 \equiv X + 1 \mod X^2 + X + 1, \\
X \cdot (X + 1) &= X^2 + X \equiv 1 \mod X^2 + X + 1, \\
(X + 1) \cdot X &= X^2 + X \equiv X \mod X^2 + X + 1, \\
(X + 1) \cdot (X + 1) &= X^2 + 1 \equiv X \mod X^2 + X + 1,
\end{align*}
$$

so the multiplication table of $\mathbb{F}_4$ turns out to be

$$
\begin{array}{c|cccc}
 & 0 & 1 & X & X + 1 \\
0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & X & X + 1 \\
X & 0 & X & X + 1 & 1 \\
X + 1 & 0 & X + 1 & 1 & X
\end{array}
$$

Figure 3.3 shows a graphical representation of the addition and multiplication tables of $\mathbb{F}_4$ using colors to represent the different elements: White is zero, black is one, red is $X$ and green is $X + 1$. In the multiplication table, the row and column associated with the zero elements are removed.

Table 3.7 lists Orbiter activities for finite fields. This extends Table 3.6 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
Table 3.7: More Finite Field Activities

<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-trace</td>
<td></td>
<td>Computes the partition of the field elements according to the value of their absolute trace.</td>
</tr>
<tr>
<td>-norm</td>
<td></td>
<td>Computes the partition of the field elements according to the value of their absolute norm.</td>
</tr>
<tr>
<td>-normal_basis d</td>
<td>d</td>
<td>Computes a normal basis for $\mathbb{F}_{q^d}$.</td>
</tr>
</tbody>
</table>

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 $\alpha$ 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 $\alpha$.

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 $\alpha$ 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 $\alpha^i$, for $0 \leq i < e$. Therefore, any element $\gamma$ of $\mathbb{F}_q$ has a unique expression of the form

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

The associated integer rank of $\gamma$ is obtained by replacing $\alpha$ by $p$ in this expression and evaluating the expression over the integers. So, the rank of $\gamma$ is

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

As $\gamma$ 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 $\alpha$, the primitive element, is $p$. The numerical ranks of the elements of the prime subfield are exactly the elements of $[0,p-1]$.

The primitive polynomials used by Orbiter to create small finite fields are listed in Table 3.8. 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
<table>
<thead>
<tr>
<th>$q$</th>
<th>Polynomial</th>
<th>Numerical</th>
<th>Relation</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>$X^2 + X + 1$</td>
<td>7</td>
<td>$\omega^2 = \omega + 1$</td>
</tr>
<tr>
<td>8</td>
<td>$X^3 + X^2 + 1$</td>
<td>13</td>
<td>$\gamma^3 = \gamma^2 + 1$</td>
</tr>
<tr>
<td>9</td>
<td>$X^2 + X + 2$</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>$X^4 + X^3 + 1$</td>
<td>25</td>
<td>$\delta^4 = \delta^3 + 1$</td>
</tr>
<tr>
<td>25</td>
<td>$X^2 + X + 2$</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>$X^3 + 2X + 1$</td>
<td>34</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>$X^5 + X^2 + 1$</td>
<td>37</td>
<td>$\eta^5 = \eta^2 + 1$</td>
</tr>
<tr>
<td>49</td>
<td>$X^2 + X + 3$</td>
<td>59</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>$X^6 + X^5 + 1$</td>
<td>97</td>
<td></td>
</tr>
<tr>
<td>81</td>
<td>$X^4 + X^3 + 2$</td>
<td>110</td>
<td></td>
</tr>
<tr>
<td>121</td>
<td>$X^2 + 4X + 2$</td>
<td>167</td>
<td></td>
</tr>
<tr>
<td>125</td>
<td>$X^3 + X^2 + X + 2$</td>
<td>86</td>
<td></td>
</tr>
<tr>
<td>128</td>
<td>$X^7 + X^6 + 1$</td>
<td>193</td>
<td>$\zeta^7 = \zeta^6 + 1$</td>
</tr>
<tr>
<td>169</td>
<td>$X^2 + X + 2$</td>
<td>184</td>
<td></td>
</tr>
<tr>
<td>243</td>
<td>$X^5 + 2X + 1$</td>
<td>250</td>
<td></td>
</tr>
<tr>
<td>256</td>
<td>$X^8 + X^4 + X^3 + X^2 + 1$</td>
<td>285</td>
<td></td>
</tr>
<tr>
<td>289</td>
<td>$X^2 + X + 3$</td>
<td>309</td>
<td></td>
</tr>
<tr>
<td>343</td>
<td>$X^3 + 3X + 2$</td>
<td>366</td>
<td></td>
</tr>
<tr>
<td>361</td>
<td>$X^2 + X + 2$</td>
<td>382</td>
<td></td>
</tr>
<tr>
<td>512</td>
<td>$X^9 + X^4 + 1$</td>
<td>529</td>
<td></td>
</tr>
<tr>
<td>529</td>
<td>$X^2 + 2X + 5$</td>
<td>580</td>
<td></td>
</tr>
<tr>
<td>625</td>
<td>$X^4 + X^3 + X + 2$</td>
<td>326</td>
<td></td>
</tr>
<tr>
<td>729</td>
<td>$X^6 + X^5 + 2$</td>
<td>974</td>
<td></td>
</tr>
<tr>
<td>841</td>
<td>$X^2 + 5X + 2$</td>
<td>988</td>
<td></td>
</tr>
<tr>
<td>961</td>
<td>$X^2 + 2X + 3$</td>
<td>1026</td>
<td></td>
</tr>
<tr>
<td>1024</td>
<td>$X^{10} + X^3 + 1$</td>
<td>1033</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.8: Orbiter primitive polynomials for fields $\mathbb{F}_q$ with $q \leq 1024$
Table 3.9: The field $\mathbb{F}_{16}$

$\mathbb{F}_{16}$:

$\diamond$ $(\text{ORBITER}) -v 3 \$

$\diamond$ $\diamond$ -define F -finite_field -q 4 -end \$

$\diamond$ $\diamond$ -with F -do -finite_field_activity -cheat_sheet_GF -end

$\diamond$ pdflatex GF_4.tex

$\diamond$ open GF_4.pdf

creates a report for the field $\mathbb{F}_{4}$. The command

$\mathbb{F}_{16}$:

$\diamond$ $(\text{ORBITER}) -v 3 \$

$\diamond$ $\diamond$ -define F -finite_field -q 16 -end \$

$\diamond$ $\diamond$ -with F -do -finite_field_activity -cheat_sheet_GF -end

$\diamond$ pdflatex GF_16.tex

creates a cheat sheet for $\mathbb{F}_{16}$. This command produces Table 3.9.

Unlike other computer algebra systems (GAP [29] and Magma [14]), Orbiter does not use Conway polynomials to create field extensions. Instead, it provides the option to override polynomial: $X^4 + X^3 + 1 = 25$

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

Subfields:

<table>
<thead>
<tr>
<th>Subfield</th>
<th>Polynomial</th>
<th>Numerical Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbb{F}_4$</td>
<td>$X^2 + X + 1$</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 3.9:

<table>
<thead>
<tr>
<th>$i$</th>
<th>$\gamma_i$</th>
<th>$-\gamma_i$</th>
<th>$\gamma_i^{-1}$</th>
<th>$\log_a (\gamma_i)$</th>
<th>$\alpha^i$</th>
<th>$Z_i$</th>
<th>$\phi(\gamma_i)$</th>
<th>$T(\gamma_i)$</th>
<th>$N(\gamma_i)$</th>
<th>$T_3(\gamma_i)$</th>
<th>$N_2(\gamma_i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>DNE</td>
<td>DNE</td>
<td>1</td>
<td>DNE</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>15</td>
<td>2</td>
<td>12</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>$\alpha = \delta$</td>
<td>2</td>
<td>12</td>
<td>1</td>
<td>4</td>
<td>9</td>
<td>4</td>
<td>1</td>
<td>11</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td>$\alpha + 1 = \delta^{12}$</td>
<td>3</td>
<td>8</td>
<td>12</td>
<td>8</td>
<td>4</td>
<td>5</td>
<td>1</td>
<td>11</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>$\alpha^2 = \delta^2$</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>9</td>
<td>3</td>
<td>9</td>
<td>1</td>
<td>10</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>$\alpha^2 + 1 = \delta^9$</td>
<td>5</td>
<td>15</td>
<td>9</td>
<td>11</td>
<td>10</td>
<td>8</td>
<td>1</td>
<td>10</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>$\alpha^2 + \alpha = \delta^{13}$</td>
<td>6</td>
<td>4</td>
<td>13</td>
<td>15</td>
<td>8</td>
<td>13</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>7</td>
<td>$\alpha^2 + \alpha + 1 = \delta^7$</td>
<td>7</td>
<td>14</td>
<td>7</td>
<td>7</td>
<td>13</td>
<td>12</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>8</td>
<td>$\alpha^3 = \delta^3$</td>
<td>8</td>
<td>3</td>
<td>3</td>
<td>14</td>
<td>6</td>
<td>15</td>
<td>1</td>
<td>11</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>9</td>
<td>$\alpha^3 + 1 = \delta^4$</td>
<td>9</td>
<td>13</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>14</td>
<td>1</td>
<td>11</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>10</td>
<td>$\alpha^3 + \alpha = \delta^{10}$</td>
<td>10</td>
<td>11</td>
<td>10</td>
<td>10</td>
<td>5</td>
<td>11</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>11</td>
</tr>
<tr>
<td>11</td>
<td>$\alpha^3 + \alpha + 1 = \delta^5$</td>
<td>11</td>
<td>10</td>
<td>5</td>
<td>13</td>
<td>14</td>
<td>10</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>12</td>
<td>$\alpha^3 + \alpha^2 = \delta^{14}$</td>
<td>12</td>
<td>2</td>
<td>14</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>13</td>
<td>$\alpha^3 + \alpha^2 + 1 = \delta^{11}$</td>
<td>13</td>
<td>9</td>
<td>11</td>
<td>6</td>
<td>7</td>
<td>7</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>14</td>
<td>$\alpha^3 + \alpha^2 + \alpha = \delta^8$</td>
<td>14</td>
<td>7</td>
<td>8</td>
<td>12</td>
<td>11</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>15</td>
<td>$\alpha^3 + \alpha^2 + \alpha + 1 = \delta^6$</td>
<td>15</td>
<td>5</td>
<td>6</td>
<td>1</td>
<td>DNE</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>10</td>
</tr>
</tbody>
</table>
Table 3.10: The subfields of $\mathbb{F}_{64}$

<table>
<thead>
<tr>
<th>Subfield</th>
<th>Polynomial</th>
<th>Numerical rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbb{F}_4$</td>
<td>$X^2 + X + 1$</td>
<td>7</td>
</tr>
<tr>
<td>$\mathbb{F}_8$</td>
<td>$X^3 + X + 1$</td>
<td>11</td>
</tr>
</tbody>
</table>

Figure 3.4: Addition and multiplication table of $\mathbb{F}_3$ and $\mathbb{F}_9$ using the lexicographic ordering of elements

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. For instance, Table 3.10 shows the subfields of $\mathbb{F}_{64}$ generated by the polynomial $X^6 + X^5 + 1$ whose numerical rank is 97.

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. For instance, in Figure 3.4, we find the tables for $\mathbb{F}_3$ in the upper left corners of the tables of $\mathbb{F}_9$, for instance. Recall that omit the zero element in the multiplication tables.

Orbiter uses primitive polynomials for creating extension fields. Because of this, the element $\alpha$ is always primitive. Since the numerical rank of $\alpha$ 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 Figure 3.5.
Figure 3.5: Addition and multiplication table of $\mathbb{F}_9$ using the cyclic ordering of elements
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-RREF</td>
<td>$m\ n\ L$</td>
<td>Compute the RREF of the $m \times n$ matrix $L$ over $\mathbb{F}_q$</td>
</tr>
<tr>
<td>-nullspace</td>
<td>$m\ n\ L$</td>
<td>Compute a basis for the right nullspace of the $m \times n$ matrix $L$</td>
</tr>
<tr>
<td>normalize_from_the_right</td>
<td></td>
<td>Normalizes the result of -RREF or nullspace from the right</td>
</tr>
<tr>
<td>normalize_from_the_left</td>
<td></td>
<td>Normalizes the result of -RREF or nullspace from the left</td>
</tr>
<tr>
<td>-eigenstuff</td>
<td>$d\ M$</td>
<td>Computes the eigenvalues and eigenvectors of the given $d \times d$ matrix $M$ over $\mathbb{F}_q$</td>
</tr>
<tr>
<td>all_rational_normal_forms</td>
<td>$d$</td>
<td>Produces a report of all rational normal forms of endomorphisms of $\mathbb{F}_q^d$</td>
</tr>
</tbody>
</table>

Table 3.11: Finite Field Activities for Linear Algebra

### 3.4 Linear Algebra Over Finite Fields

In Table 3.11, some finite field activities regarding linear algebra are shown. For instance, the command

```
RREF:
\[ \$(ORBITER) -v\ 2 \]
\[ \| -define\ F -finite_field -q\ 2 -end \]
\[ \| -define\ v -vector -field\ F -format\ 2 \]
\[ \| -dense\ "1,1,1,0,1,1,0,0,1" \]
\[ \| -end \]
\[ \| -with\ F -do -finite_field_activity \]
\[ \| -RREF\ v -normalize_from_the_right \]
\[ \| -end \]
```

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
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 $F_7$. 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:

```
V7_VANDERMONDE_EXTENDED="
1,0,0,0,0,0,1,0,0,0,0,0,0, \n1,1,1,1,1,1,0,0,0,0,0,0,0, \n1,2,4,1,2,4,1,0,0,1,0,0,0, \n1,3,2,6,4,5,1,0,0,1,0,0,0, \n1,4,2,1,4,2,1,0,0,0,1,0,0, \n1,5,4,6,2,3,1,0,0,0,0,1,0, \n1,6,1,6,1,6,1,0,0,0,0,0,1"
```

```
RREF_V7:
▷ $(ORBITER) -v 2 \\
▷ ▷ -define F -finite_field -q 7 -end \\
▷ ▷ -define V -vector -format 7 \\
▷ ▷ ▷ -dense $(V7_VANDERMONDE_EXTENDED) \\
▷ ▷ -end \\
▷ -with F -do -finite_field_activity \\
▷ ▷ ▷ -RREF V \\
▷ ▷ -end
```

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 $F_7$
```
Position \((i, j) = (0, 0)\), found pivot in column 0

After making pivot 1:

After elimination above pivot 0 in position \((0,0)\):
The inverse matrix agrees with the output obtained in Section 3.2.

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

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

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

```
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
produces a list of all conjugacy classes of 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 [41]. This command relies on the Orbiter catalogue of irreducible polynomials. For an introduction to the rational normal form of endomorphisms, see [47].
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
3.5 Advanced Topics in Finite Fields

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

First, in Tables 3.12-3.13, 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

```
normal_basis_2_3:
▷ $(ORBITER) -v 2 \\n▷ ▷ -define F -finite_field -q 2 -end \n▷ ▷ -with F -do -finite_field_activity \n▷ ▷ -normal_basis 3 -end
```

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 coefficient vectors with respect to the standard basis, the normal basis is

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

Let us apply the Frobenius mapping $\Phi$ to the elements of the normal bases:

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

Thus,

\[b_1 \mapsto b_2 \mapsto b_3 \mapsto b_1\]

under $\Phi$, 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.
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-write_code_for_division</code></td>
<td>fname $A \ B$</td>
<td>Write C++ source code for the polynomial division of $A$ by $B$. See Section 10.4.</td>
</tr>
<tr>
<td><code>-polynomial_division</code></td>
<td>$A \ B$</td>
<td>Divides polynomial $B$ by polynomial $A$.</td>
</tr>
<tr>
<td><code>-extended_gcd_for_polynomials</code></td>
<td>$A \ B$</td>
<td>Computes the extended gcd of polynomials $A$ and $B$.</td>
</tr>
<tr>
<td><code>-polynomial_mult_mod</code></td>
<td>$A \ B \ M$</td>
<td>Computes the product of polynomials $A$ and $B$ modulo the polynomial $M$.</td>
</tr>
<tr>
<td><code>-polynomial_power_mod</code></td>
<td>$A \ N \ M$</td>
<td>Computes the $n$-th power of the polynomial $A$ modulo the polynomial $M$.</td>
</tr>
<tr>
<td><code>-Berlekamp_matrix</code></td>
<td>$A$</td>
<td>Compute the Berlekamp matrix associated with the polynomial $A$.</td>
</tr>
<tr>
<td><code>-normal_basis</code></td>
<td>$d$</td>
<td>Computes a normal basis for $\mathbb{F}_{q^d}$ over $\mathbb{F}_q$.</td>
</tr>
<tr>
<td><code>-polynomial_find_roots</code></td>
<td>$A$</td>
<td>Computes the roots of the polynomial $A$.</td>
</tr>
<tr>
<td><code>-nullspace</code></td>
<td>$A$</td>
<td>Computes the right nullspace of the matrix $A$.</td>
</tr>
<tr>
<td><code>-RREF</code></td>
<td>$A$</td>
<td>Computes the RREF of the matrix $A$.</td>
</tr>
<tr>
<td><code>-weight Enumerator</code></td>
<td>$A$</td>
<td>Computes the weight enumerator of the code whose generator matrix is $A$.</td>
</tr>
<tr>
<td><code>-Walsh_Hadamard_transform</code></td>
<td>fname $n$</td>
<td>Computes the Walsh-Hadamard transform for the $n$-variable boolean function in the given file.</td>
</tr>
<tr>
<td><code>-algebraic_normal_form</code></td>
<td>fname $n$</td>
<td>Computes the algebraic normal form for the $n$-variable boolean function in the given file.</td>
</tr>
<tr>
<td><code>-apply_trace_function</code></td>
<td>fname</td>
<td>Applies the absolute trace function to the function in the given file.</td>
</tr>
<tr>
<td><code>-apply_power_function</code></td>
<td>fname $d$</td>
<td>Applies the raise-to-the-power-$d$ function to the function in the given file.</td>
</tr>
<tr>
<td><code>-identity_function</code></td>
<td>fname_csv</td>
<td>Creates the identity function and stores in the given csv file.</td>
</tr>
<tr>
<td><code>-Walsh_matrix</code></td>
<td>$n$</td>
<td>Creates the Walsh matrix of order $n$.</td>
</tr>
</tbody>
</table>

Table 3.12: Finite Field Activities (Part I)
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-Vandermonde_matrix</td>
<td>$n$</td>
<td>Creates the Vandermonde matrix of order $q \times q$. The entry $(i, j)$ is $x_j^i$ where $w_0, \ldots, x_{q-1}$ is the list of field elements in ordered according to the Orbiter ranks.</td>
</tr>
<tr>
<td>-transversal</td>
<td>$L_1 \ L_2 \ P$</td>
<td>Computes the unique transversal to the lines $L_1$ and $L_2$ through the point $P$ in $\text{PG}(3, q)$. The lines are given by a basis consisting of 8 field elements.</td>
</tr>
<tr>
<td>-intersection_of_two_lines</td>
<td>$L_1 \ L_2$</td>
<td>Computes the intersection of two lines in $\text{PG}(3, q)$. The lines are given by a basis consisting of 8 field elements.</td>
</tr>
<tr>
<td>-rank_point_in_PG</td>
<td>$P$</td>
<td>Computes the orbiter point rank of the point $P$ in $\text{PG}(n, q)$. $P$ is a label of a vector, which is the coefficient vector.</td>
</tr>
<tr>
<td>-unrank_point_in_PG</td>
<td>$r$</td>
<td>Computes the orbiter point in $\text{PG}(n, q)$ from the Orbiter rank value $r$.</td>
</tr>
<tr>
<td>-inverse_isomorphism_klein_quadric</td>
<td>$L_{36}$</td>
<td></td>
</tr>
<tr>
<td>-NTT</td>
<td>$k \ n$</td>
<td>Computes the Number-theoretic transform for $n = 2^k$, which must divide $q - 1$.</td>
</tr>
</tbody>
</table>

Table 3.13: Finite Field Activities (Part II)
Figure 3.6: The field reduction from $\mathbb{F}_{64}$ to $\mathbb{F}_8$

The output is shown in Figure 3.6. Note that the dimension of the vector space is 2, so the block matrices are $2 \times 2$. Observe that $\mathbb{F}_{64}$ has many subfields. Figure 3.7 shows the field reduction from $\mathbb{F}_{64}$ to $\mathbb{F}_4$ (left) and from $\mathbb{F}_{64}$ to $\mathbb{F}_2$ (right). Here, the block matrices have size $3 \times 3$ and $6 \times 6$, respectively.

The minimum polynomials associated with the $n$-th roots over $\mathbb{F}_q$ can be computed using the \texttt{-nth_roots} command, which is a finite field activity. The activity is applied 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 $\alpha$ be a primitive element of $\mathbb{F}_{q^m}$ and let $\beta$ be a generator of the subgroup of $n$-th roots.
Figure 3.7: The field reduction from $\mathbb{F}_{64}$ to $\mathbb{F}_4$ (left) and from $\mathbb{F}_{64}$ to $\mathbb{F}_2$ (right)

Also, let $\gamma$ 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 $\beta$. After that, the $q - 1$th roots are shown, generated by $\gamma$. 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 $\mathbb{F}_8$:

F_8_Nth_roots_21:
▷ $(\text{ORBITER})$ -v 3 \n▷ ▷ -define F -finite_field -q 8 -override_polynomial 11 -end \n▷ ▷ -with F -do -coding_theoretic_activity \n▷ ▷ ▷ -nth_roots 21 \n▷ ▷ -end \n▷ pdflatex Nth_roots_q8_n21.tex \n▷ open Nth_roots_q8_n21.pdf

The output is:

Let $\alpha$ be a primitive element of GF(64). Let $\beta$ be a primitive 21-th root in GF(64), so $\beta = \alpha^3$.

$\beta^0 = 100000 = 1$
$\beta^1 = 000100 = \alpha^3$
$\beta^2 = 100001 = \alpha^5 + 1$
$\beta^3 = 111101 = \alpha^5 + \alpha^3 + \alpha^2 + \alpha + 1$
\[ \beta^4 = 011111 = \alpha^5 + \alpha^4 + \alpha^3 + \alpha^2 + \alpha \]
\[ \beta^5 = 101010 = \alpha^4 + \alpha^2 + 1 \]
\[ \beta^6 = 110100 = \alpha^3 + \alpha + 1 \]
\[ \beta^7 = 100111 = \alpha^5 + \alpha^4 + \alpha^3 + 1 \]
\[ \beta^8 = 101101 = \alpha^5 + \alpha^3 + \alpha^2 + 1 \]
\[ \beta^9 = 011101 = \alpha^5 + \alpha^3 + \alpha^2 + \alpha \]
\[ \beta^{10} = 011011 = \alpha^5 + \alpha^3 + \alpha^2 + \alpha \]
\[ \beta^{11} = 001011 = \alpha^5 + \alpha^3 + \alpha^2 + 1 \]
\[ \beta^{12} = 010100 = \alpha^3 + \alpha \]
\[ \beta^{13} = 100011 = \alpha^5 + \alpha^3 + \alpha^2 \]

Let \( \gamma \) be a primitive 7-th root in GF(64), so \( \gamma = \alpha^9 \).
\[ \gamma^0 = 100000 = 1 \]
\[ \gamma^1 = 111101 = \alpha^5 + \alpha^3 + \alpha^2 + \alpha + 1 \]
\[ \gamma^2 = 110100 = \alpha^3 + \alpha + 1 \]
\[ \gamma^3 = 011101 = \alpha^5 + \alpha^3 + \alpha^2 + \alpha \]
\[ \gamma^4 = 001001 = \alpha^5 + \alpha^2 \]
\[ \gamma^5 = 100101 = \alpha^5 + \alpha^2 + 1 \]
\[ \gamma^6 = 010100 = \alpha^3 + \alpha \]

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:

<table>
<thead>
<tr>
<th>(i)</th>
<th>(r_i)</th>
<th>(\text{Cyc}(r_i))</th>
<th>(m_{\beta_i}(X))</th>
<th>(m_{\beta_i}(X))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>(0)</td>
<td>((100000)X^0 + (100000)X^1)</td>
<td>(X + 1)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>(1, 8)</td>
<td>((011101)X^0 + (101001)X^1 + (100000)X^2)</td>
<td>(X^2 + 7X + 3)</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>(2, 16)</td>
<td>((010100)X^0 + (011101)X^1 + (100000)X^2)</td>
<td>(X^2 + 3X + 5)</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>(3)</td>
<td>((111101)X^0 + (100000)X^1)</td>
<td>(X + 2)</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>(4, 11)</td>
<td>((101001)X^0 + (010100)X^1 + (100000)X^2)</td>
<td>(X^2 + 5X + 7)</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>(5, 19)</td>
<td>((111101)X^0 + (001001)X^1 + (100000)X^2)</td>
<td>(X^2 + 6X + 2)</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>(6)</td>
<td>((101000)X^0 + (100000)X^1)</td>
<td>(X + 4)</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>(7, 14)</td>
<td>((100000)X^0 + (100000)X^1 + (100000)X^2)</td>
<td>(X^2 + X + 1)</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>(9)</td>
<td>((011101)X^0 + (100000)X^1)</td>
<td>(X + 3)</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>(10, 17)</td>
<td>((110100)X^0 + (111101)X^1 + (100000)X^2)</td>
<td>(X^2 + 2X + 4)</td>
</tr>
<tr>
<td>10</td>
<td>12</td>
<td>(12)</td>
<td>((001001)X^0 + (100000)X^1)</td>
<td>(X + 6)</td>
</tr>
<tr>
<td>11</td>
<td>13</td>
<td>(13, 20)</td>
<td>((001001)X^0 + (110100)X^1 + (100000)X^2)</td>
<td>(X^2 + 4X + 6)</td>
</tr>
<tr>
<td>12</td>
<td>15</td>
<td>(15)</td>
<td>((101001)X^0 + (100000)X^1)</td>
<td>(X + 7)</td>
</tr>
<tr>
<td>13</td>
<td>18</td>
<td>(18)</td>
<td>((010100)X^0 + (100000)X^1)</td>
<td>(X + 5)</td>
</tr>
</tbody>
</table>

In Section 3.2, we considered the Vandermonde matrix over \(F_7\). Let us do the same for the field \(F_8\) instead. We use the following command:

\begin{verbatim}
F_8.vandermonde:
  $ (ORBITER) -v 3 \$
  $> -define F -finite_field -q 8 -end \$
\end{verbatim}
The output is shown below. Again, the first matrix is $V = (x_i^j)$. The second matrix is $V^{-1}$:

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

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 $\mathbb{F}_{1024}$:

```
F_1024.vandermonde:
▶ $(ORBITER) -v 3 \\n▶ ▶ -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:

```
SRC=$(MY_PATH)/src
MY_CPP = g++
MY_CC = gcc
CPPFLAGS = -Wall -I../../DEV.22/orbiter/src/lib -std=c++14
```

67
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 $F_{17}$. Here is the command to generate the Orbiter source code:

\texttt{NTT\_k4\_q17.cpp:}
\begin{verbatim}
  $(ORBITER) -v 3 \\
  -define F -finite_field -q 17 -end \\
  -with F -do -coding_theoretic_activity \\
  -NTT 4 17 \\
  -end
\end{verbatim}

This produces a C++ file \texttt{NTT\_k4\_q17.cpp}. This file should be compiled and linked against
the Orbiter library. The command

\texttt{F\_17\_NTT\_compile: NTT\_k4\_q17.cpp}
\begin{verbatim}
  $(MY\_CPP) NTT\_k4\_q17.cpp $(CPPFLAGS) \\
  $(LIB) $(LFLAGS) -o NTT\_k4\_q17.out \\
  ./NTT\_k4\_q17.out
\end{verbatim}

can be used to compile the code and run it. Note the dependency on the file \texttt{NTT\_k4\_q17.cpp}. This means that \texttt{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 homogenous 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:

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

For more on rings, see Chapter 8.
Chapter 4

Geometry

4.1 Finite Projective Spaces

Orbiter can create the projective space $\text{PG}(n, q)$. In order to do so, an object of type projective_space needs to be defined. Once the object exists, various commands are available. Let us look at a very simple example. Suppose we want to create $\text{PG}(3, 2)$. The following command sequence first creates the finite field $\mathbb{F}_2$. The symbol $F$ is used to store the field. After that, the projective space $\text{PG}(3, F)$ is created and stored in the symbol $P$. No other commands are given:

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

This means that Orbiter offers indexing for the subspaces of $\text{PG}(n, q)$ of a fixed dimension. For instance, there are enumerators for points and lines. Besides these, there are enumerators for subspaces of any 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 5.2. The indexing of points is not the lexicographic ordering. It emphasizes the role of frames in the geometry by assigning the smallest rank values to the members of the standard frame. After that, the other points are listed.

Orbiter can create cheat sheets, which summarize the properties of $\text{PG}(n, q)$ and list the various elements. The following command creates a cheat sheet for $\text{PG}(2, 4)$ using a finite field object:

```
PG_2_4:
\$ (ORBITER) -v 2 \n\$ \$ -define F -finite_field -q 4 -end \n\$ \$ -define P -projective_space -n 2 -field F -v 0 -end \n```
The cheat sheet contains a drawing of the plane as shown in Figure 4.1. 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:

\[ P_0 = (1, 0, 0) \]
\[ P_1 = (0, 1, 0) \]
\[ P_2 = (0, 0, 1) \]
\[ P_3 = (1, 1, 1) \]
\[ P_4 = (1, 1, 0) \]
\[ P_5 = (2, 1, 0) \]
\[ P_6 = (3, 1, 0) \]
\[ P_7 = (1, 0, 1) \]
\[ P_8 = (2, 0, 1) \]
\[ P_9 = (3, 0, 1) \]
\[ P_{10} = (0, 1, 1) \]

There are 7 elements in the Baer subgeometry.

Normalized from the left:

\[ P_0 = (1, 0, 0) \]
\[ P_1 = (0, 1, 0) \]
\[ P_2 = (0, 0, 1) \]
\[ P_3 = (1, 1, 1) \]
\[ P_4 = (1, 1, 0) \]
\[ P_5 = (2, 1, 0) \]
\[ P_6 = (3, 1, 0) \]
\[ P_7 = (1, 0, 1) \]
\[ P_8 = (2, 0, 1) \]
\[ P_9 = (3, 0, 1) \]
\[ P_{10} = (0, 1, 1) \]

The Lines of \( PG(2,4) \). \( PG(2,4) \) has 21 1-subspaces:
Here is a slightly larger example. The following command creates a cheat sheet for \( \text{PG}(3, 2) \).

\text{PG\_3\_2:}
\begin{itemize}
\item \$\text{ORBITER} -v 2 \\
\item \$ -define F -finite_field -q 2 -end \\
\item \$ -define P -projective_space -n 3 -field F -v 0 -end \\
\item \$ -with P -do -projective_space_activity \\
\item \$ -cheat_sheet \\
\item \$ -end
\end{itemize}

\text{pdflatex \text{PG\_3\_2.tex}}

\text{open \text{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.

\textbf{The projective space} \( \text{PG}(3, 2) \)

\begin{itemize}
\item \( q = 2 \)
\item \( p = 2 \)
\item \( e = 1 \)
\item \( n = 3 \)
\end{itemize}

Number of points = 15
The points of PG(3, 2)

PG(3, 2) has 15 points:

\[
\begin{align*}
P_0 &= (1,0,0,0) & P_4 &= (1,1,1,1) & P_8 &= (1,1,1,0) & P_{12} &= (0,0,1,1) \\
P_1 &= (0,1,0,0) & P_5 &= (1,1,0,0) & P_9 &= (1,0,0,1) & P_{13} &= (1,0,1,1) \\
P_2 &= (0,0,1,0) & P_6 &= (1,0,1,0) & P_{10} &= (0,1,0,1) & P_{14} &= (0,1,1,1) \\
P_3 &= (0,0,0,1) & P_7 &= (0,1,1,0) & P_{11} &= (1,1,0,1)
\end{align*}
\]

Normalized from the left:

\[
\begin{align*}
P_0 &= (1,0,0,0) & P_4 &= (1,1,1,1) & P_8 &= (1,1,1,0) & P_{12} &= (0,0,1,1) \\
P_1 &= (0,1,0,0) & P_5 &= (1,1,0,0) & P_9 &= (1,0,0,1) & P_{13} &= (1,0,1,1) \\
P_2 &= (0,0,1,0) & P_6 &= (1,0,1,0) & P_{10} &= (0,1,0,1) & P_{14} &= (0,1,1,1) \\
P_3 &= (0,0,0,1) & P_7 &= (0,1,1,0) & P_{11} &= (1,1,0,1)
\end{align*}
\]

The lines of PG(3, 2)

PG(3, 2) has 35 1-subspaces:

\[
\begin{align*}
L_0 &= \begin{bmatrix} 1000 \\ 0100 \end{bmatrix} = \text{Pl}(1,0,0,0,0,0) \\
L_1 &= \begin{bmatrix} 1000 \\ 0110 \end{bmatrix} = \text{Pl}(1,0,1,0,0,0) \\
L_2 &= \begin{bmatrix} 1000 \\ 0101 \end{bmatrix} = \text{Pl}(1,0,0,0,1,0) \\
L_3 &= \begin{bmatrix} 1000 \\ 0111 \end{bmatrix} = \text{Pl}(1,0,1,0,1,0) \\
L_4 &= \begin{bmatrix} 1000 \\ 0010 \end{bmatrix} = \text{Pl}(0,0,1,0,0,0) \\
L_5 &= \begin{bmatrix} 1000 \\ 0011 \end{bmatrix} = \text{Pl}(0,0,1,0,1,0) \\
\vdots
\end{align*}
\]
\[ L_{34} = \begin{bmatrix} 0010 \\ 0001 \end{bmatrix} = \mathbf{Pl}(0, 1, 0, 0, 0) \]

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}

PG(3, 2) has the following low weight Pluecker lines:

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

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

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

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

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

\[ L_{34} = \begin{bmatrix} 0010 \\ 0001 \end{bmatrix} = \mathbf{Pl}(0, 1, 0, 0, 0, 0) \]
The planes of PG(3, 2)

PG(3, 2) has 15 2-subspaces:

\[
L_0 = \begin{bmatrix}
1000 \\
0100 \\
0010
\end{bmatrix}
\]

\[
L_1 = \begin{bmatrix}
1000 \\
0100 \\
0011
\end{bmatrix}
\]

\[
L_i = \begin{bmatrix}
0100 \\
0010 \\
0001
\end{bmatrix}
\]

The polynomial rings associated with PG(3, 2)

<table>
<thead>
<tr>
<th>(h)</th>
<th>monomial</th>
<th>vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(X_0)</td>
<td>(1, 0, 0, 0)</td>
</tr>
<tr>
<td>1</td>
<td>(X_1)</td>
<td>(0, 1, 0, 0)</td>
</tr>
<tr>
<td>2</td>
<td>(X_2)</td>
<td>(0, 0, 1, 0)</td>
</tr>
<tr>
<td>3</td>
<td>(X_3)</td>
<td>(0, 0, 0, 1)</td>
</tr>
</tbody>
</table>
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 normalize 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, \textsc{Rank} and \textsc{Unrank}. The function \textsc{Rank} takes a vector \(x \in \mathbb{F}_q^n\), not zero, and maps it to the element in \(\mathbb{Z}_N\) representing the projective point \(P(x)\). A frame in \(\text{PG}(n, q)\) 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 \(e_i\) be the \(i\)-th unit vector. A frame for \(\text{PG}(n, q)\) is

\[
e_0, \ldots, e_{n-1}, e_0 + \cdots + e_{n-1}.
\]

This is the \textit{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). Thus, for \(\text{PG}(2, 2)\) the ordering is

\[(1, 0, 0), (0, 1, 0), (0, 0, 1), (1, 1, 1), (1, 1, 0), (1, 0, 1), (0, 1, 1)\).

Let us describe the two functions rank and unrank to perform the actual mappings between \(\text{PG}(n, q)\) and \(\mathbb{Z}_N\), where \(N = \theta_n(q)\). For this, assume that ranking and unranking functions have already been defined for the elements of the finite field \(\mathbb{F}_q\). Thus, we assume that for \(x \in \mathbb{F}_q\), \(\text{RANK}(\mathbb{F}_q, x)\) is a number \(b\) in \(\mathbb{Z}_q\). Also, for \(b \in \mathbb{Z}_q\), we assume that \(\text{UNRANK}(\mathbb{F}_q, b)\) is the corresponding \(x \in \mathbb{F}_q\). So, we assume that \textsc{Rank} and \textsc{Unrank} are mutually inverse functions. Consider the group \(\text{PGL}(3, 2)\) acting on \(\text{PG}(2, 2)\), for instance. The points of \(\text{PG}(2, 2)\) are listed in 4.1.

Let us look at an example. The following command computes the rank of \(P(3, 3, 1) = P(\omega + 1, \omega + 1, 1)\) in \(\text{PG}(2, 4)\):

```
PG_2_4_rank_point:
  ▶ $\langle$ORBITER$\rangle$ -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
```
Algorithm 1 Rank

1: procedure Rank(vector : x, field : $\mathbb{F}_q$, int : n)
2:   assert x is a nonzero vector in $\mathbb{F}_q^n$.
3:   if x = e, then
4:     return i
5:   if x = 1 then
6:     return n
7:   i ← max$\{j \in \mathbb{Z}_n \mid x_j \neq 0\}$
8:   x ← $\frac{1}{x_i}$x
9:   a := 0
10:  for j = i − 1, . . ., 1, 0 do
11:     a ← a + $\text{Rank}(\mathbb{F}_q, x_j)$
12:     if j > 0 then
13:        a ← a · q
14:     if i = n − 1 and a ≥ $\sum_{j=0}^{i-1} q^j$ then
15:        a ← a − 1
16:     a ← a + n − i + $\sum_{j=0}^{i-1} q^j$
17: return a

\[ a = \text{Rank}(x) \quad \text{and} \quad x = \text{Unrank}(a) \]

<table>
<thead>
<tr>
<th>a = Rank(x)</th>
<th>x = Unrank(a)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(1, 0, 0)</td>
</tr>
<tr>
<td>1</td>
<td>(0, 1, 0)</td>
</tr>
<tr>
<td>2</td>
<td>(0, 0, 1)</td>
</tr>
<tr>
<td>3</td>
<td>(1, 1, 1)</td>
</tr>
<tr>
<td>4</td>
<td>(1, 1, 0)</td>
</tr>
<tr>
<td>5</td>
<td>(1, 0, 1)</td>
</tr>
<tr>
<td>6</td>
<td>(0, 1, 1)</td>
</tr>
</tbody>
</table>

Table 4.1: Representatives of the points of PG(2, 2)
Algorithm 2 Unrank

1: procedure Unrank(int : a, field : $\mathbb{F}_q$, int : n)
2:  assert $a \in \mathbb{Z}_N$ where $N = \theta_{n-1}(q)$.
3:  if $a < n$ then
4:      return $e_a$
5:  $a \leftarrow a - n$
6:  if $a = 0$ then
7:      return 1
8:  $a \leftarrow a - 1$
9:  $x \leftarrow 0$
10:  for $i = 1, \ldots, n - 1$ do
11:      if $a \geq \sum_{j=1}^{i-1} q^j$ then
12:          $a \leftarrow a - \sum_{j=1}^{i-1} q^j$
13:      else
14:          $x_i \leftarrow 1$
15:      break
16:  for $k = i + 1, \ldots, n - 1$ do
17:      $x_k \leftarrow 0$
18:  $a \leftarrow a + 1$
19:  if $i = n - 1$ and $a \geq \sum_{j=0}^{i-1} q^j$ then
20:      $a \leftarrow a + 1$
21:  $j \leftarrow 0$
22:  while $a > 0$ do
23:      $r \leftarrow a \mod q$
24:      $x_j \leftarrow \text{Unrank}(\mathbb{F}_q, r)$
25:      $j \leftarrow j + 1$
26:      $a \leftarrow (a - r)/q$
27:  for $h = j, \ldots, i - 1$ do
28:      $x_h \leftarrow 0$
29:  return $x$
The rank turns out to be 20. Conversely, running

```bash
PG_2.4_unrank_point:
  ▶ $ (ORBITER) -v 2 \
  ▶ ▶ -define v -vector -dense "19,20" -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 \( P(3, 3, 1) \).

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

```bash
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 in Figure 4.2. The rows and columns correspond to points and lines, respectively. The Orbiter indexing of points and lines determines the ordering of rows and columns.
Figure 4.2: Incidence matrix of PG(2, 8) in Orbiter ordering
4.3 Finite Desarguesian Projective Planes

The projective spaces $\text{PG}(2, q)$ deserve special attention. They are examples of a more general structure called projective planes. The $\text{PG}(2, F)$, $F$ 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. For other projective planes, see Section 12.2.

The points in the desarguesian projective plane $\text{PG}(2, q)$ have the coordinates $P(x, y, z)$, with $x, y, z \in \mathbb{F}_q$. We can distinguish one line, for instance $z = 0$, and call it the line at infinity. The points not on that line form an affine plane $\text{AG}(2, q)$.

The command

```
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 $\text{PG}(2, 16)$ shown in Figure 4.3. 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 16.2 that the default values are 10000), which has the effect that the text appears smaller relative to the grid.

Projected spaces have a special property. They 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 $\text{PG}(2, 4)$

```
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 \ 
  -cheat_sheet_for_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:

\[
\begin{array}{c|c}
\rightarrow & 21 \\
21 & 5 \\
\end{array}
\]

Column scheme:
Figure 4.4: Cyclic incidence matrix of PG(2, 4)

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

```
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 shown in Figure 4.4.
If the number of points is not a prime, the group acts imprimitively. By considering various subgroups, 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:

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

> open PG_2_4_singer_incma_subgroup_index_3_draw.bmp
```

The tactical decomposition of the incidence matrix is shown in Figure 4.5.

Figure 4.5: Tactical decomposition of the incidence matrix of PG(2, 4)
4.4 The Grassmannian

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

$$\left[ \begin{array}{c} n \\ k \end{array} \right]_q = \prod_{i=0}^{k-1} q^{n-i} - 1,$$

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 $N = \left[ \begin{array}{c} n \\ k \end{array} \right]_q$. 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 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

GR_3_2_2:

```
$ (ORBITER) -v 2 \n$ -define F -finite_field -q 2 -end \n$ -with F -do -finite_field_activity \n$ -cheat_sheet Gr 3 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 PG(2,2):

\[
\begin{align*}
L_0 &= \begin{bmatrix} 100 \\ 010 \end{bmatrix} & L_3 &= \begin{bmatrix} 101 \\ 010 \end{bmatrix} & L_6 &= \begin{bmatrix} 010 \\ 001 \end{bmatrix} \\
L_1 &= \begin{bmatrix} 100 \\ 011 \end{bmatrix} & L_4 &= \begin{bmatrix} 101 \\ 011 \end{bmatrix} & L_5 &= \begin{bmatrix} 110 \\ 001 \end{bmatrix} \\
L_2 &= \begin{bmatrix} 100 \\ 001 \end{bmatrix}
\end{align*}
\]

The following command illustrates how to rank lines. In the example, we consider lines in 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.
PG_3.3_rank_lines:
\begin{verbatim}
$ (ORBITER) -v 2
  define v -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 v1 -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
\end{verbatim}

In the next example, we unrank six lines in PG(3,5).

PG_3.5_unrank_lines:
\begin{verbatim}
$ (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
\end{verbatim}

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

planes_in_pencil:
\begin{verbatim}
$ (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{verbatim}
4.5 Algebraic Sets

A set of points \( V \) in \( \text{PG}(n,q) \) 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 \( \text{PG}(n,q) \), 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 \( \text{PG}(n,q) \), every set is algebraic of degree at most \((n+1)(q-1)\) [30]. 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 8.2.

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 8.4, we record the coefficient vector of the equation as sequence

\[(1, 0, 3, 0, 0, 0, 10, 1, 0, 0).\]

The Orbiter command

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

EC_11.txt:

```plaintext
$ (ORBITER) -v 2 \n  ▶ -define F -finite_field -q 11 -end \n  ▶ -define R -polynomial_ring -field F \n  ▶   ▶ -number_of_variables 3 \n  ▶   ▶ -homogeneous_of_degree 3 \n  ▶   ▶ -end \n  ▶ -define P -projective_space -n 2 -field F -v 0 -end \n  ▶ -define EC -geometric_object P \n  ▶   ▶ -projective_variety R \n  ▶   ▶   ▶ "EC_11" "EC\_11" \n  ▶   ▶ -EC_11_EQUATION \n  ▶   ▶ -end \n  ▶ -with EC -do -combinatorial_object_activity -save \n  ▶ -end
```
Figure 4.6: Elliptic curve \( y^2 \equiv x^3 + x + 3 \mod 11 \)

creates the algebraic set associated to the cubic curve \( y^2 = x^3 + x + 3 \) in \( \text{PG}(2,11) \). It turns out that there are exactly 18 points over \( \mathbb{F}_{11} \) (cf. Figure 4.6). 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 8.5, the equation is coded as coefficient vector

\[
(0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0).
\]

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

Hirschfeld_surface_Equation="0,0,0,0,0,1,0,1,0,0,1,0,0,0,1,0,0,0,0,0,0"

Hirschfeld_surface_q4.txt:

```
> $(ORBITER) -v 2 \n>   -define F -finite_field -q 4 -end \n>   -define R -polynomial_ring -field F \n>     -number_of_variables 4 \n>     -homogeneous_of_degree 3 \n>   -end \n>   -define P -projective_space -n 3 -field F -v 0 -end \n>   -define H4 -geometric_object P \n>     -projective_variety R \n>     "Hirschfeld\_surface\_q4" \n>     "Hirschfeld\_surface\_q4"
```
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

```
gr 4 2 2:
  $(ORBITER) -v 2 \$
  $\ldots$ -define F -finite_field -q 2 -end \$
  $\ldots$ -with F -do -finite_field_activity \$
  $\ldots$ -cheat_sheet_Gr 4 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):

<table>
<thead>
<tr>
<th>Line</th>
<th>Plücker Coordinates</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_0$</td>
<td>$\begin{bmatrix}1000 \ 0100\end{bmatrix}$ = $\text{Pl}(1,0,0,0,0,0)$</td>
</tr>
<tr>
<td>$L_1$</td>
<td>$\begin{bmatrix}1000 \ 0110\end{bmatrix}$ = $\text{Pl}(1,0,1,0,0,0)$</td>
</tr>
<tr>
<td>$L_{34}$</td>
<td>$\begin{bmatrix}0010 \ 0001\end{bmatrix}$ = $\text{Pl}(0,1,0,0,0,0)$</td>
</tr>
</tbody>
</table>

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 $\text{PG}(3, q)$ in a second, different way. In the example of $\text{PG}(3, 2)$, this yields the following list (output shortened):
\begin{align*}
0 &= \text{Pl}(1, 0, 0, 0, 0) = L_0 = \begin{bmatrix} 1000 \\ 0100 \end{bmatrix} \\
1 &= \text{Pl}(0, 1, 0, 0, 0) = L_{34} = \begin{bmatrix} 0010 \\ 0001 \end{bmatrix} \\
2 &= \text{Pl}(0, 0, 1, 0, 0) = L_4 = \begin{bmatrix} 1000 \\ 0010 \end{bmatrix} \\
3 &= \text{Pl}(0, 1, 1, 1, 1) = L_{26} = \begin{bmatrix} 1101 \\ 0011 \end{bmatrix}
\end{align*}
Table 4.2: Nondegenerate Quadrics in PG(n, q) and the canonical form adopted in Orbiter

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

Table 4.3: Command options to create an orthogonal space

<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-label_txt</td>
<td>L</td>
<td>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.</td>
</tr>
<tr>
<td>-label_tex</td>
<td>L</td>
<td>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.</td>
</tr>
<tr>
<td>-without_group</td>
<td></td>
<td>Do not create the orthogonal group.</td>
</tr>
</tbody>
</table>

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 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 4.2. 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 $\epsilon$ d q -end
```

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

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

In Table 4.3, Orbiter command options for creating orthogonal spaces are shown.

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

```
0.5.2_incidence_matrix.csv:
$ (ORBITER) -v 2 \
  -define F -finite_field -q 2 -end \
  -define O -orthogonal_space 0 5 F -without_group -end \
  -with 0 -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 0.5.2_incidence_matrix.csv \
  -box_width 20 -bit_depth 8 \
  -partition 2 \
  -all_one_r all_one_c \
  -end 
open 0.5.2_incidence_matrix_draw.bmp
```

The command also creates a bitmap drawing of the incidence matrix between points and lines of $Q(4,2)$. The incidence matrix is shown in Figure 4.7. The Orbiter indexing for points and lines of quadrics is used to order the rows and columns.

By default, the orthogonal space is created together with the orthogonal group $PGO(n+1,q)$. When $q$ is prime, the group $PGO(n+1,q)$ is created instead (the groups are isomorphic in this case, and $PGO(n+1,q)$ 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 4.4, Orbiter activities for orthogonal spaces are shown.
Figure 4.7: Incidence matrix of $Q(4, 2)$

<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-create_BLT_set</td>
<td>descr</td>
<td>Creates a BLT-set of $Q(4, q)$. See Section 12.4.</td>
</tr>
<tr>
<td>-cheat_sheet_orthogonal</td>
<td></td>
<td>Create a cheat sheet.</td>
</tr>
<tr>
<td>-print_points</td>
<td>$v$</td>
<td>Print the points whose ranks are given in the vector $v$.</td>
</tr>
<tr>
<td>-print_lines</td>
<td>$v$</td>
<td>Print the lines whose ranks are given in the vector $v$.</td>
</tr>
<tr>
<td>-unrank_line_through_two_points</td>
<td>$p1$ $p2$</td>
<td>Determine the rank of the line through the points whose ranks are $p1$ and $p2$.</td>
</tr>
<tr>
<td>-lines_on_point</td>
<td>$p$</td>
<td>Create the ranks of all lines through the point whose rank is $p$.</td>
</tr>
<tr>
<td>-perp</td>
<td>$L$</td>
<td>Determine the common perp of a set of points. The point ranks are given in the list $L$.</td>
</tr>
<tr>
<td>-export_point_line_incidence_matrix</td>
<td></td>
<td>Create a csv file with the point line incidence matrix of the space.</td>
</tr>
<tr>
<td>-intersect_with_subspace</td>
<td>$M$</td>
<td>Find the points in the intersection of the quadric with the subspace whose generating matrix has label $M$.</td>
</tr>
</tbody>
</table>

Table 4.4: Activities related to orthogonal spaces
The command

\texttt{Op\_6\_2:}
\begin{verbatim}
  $\$(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
\end{verbatim}

\texttt{pdflatex 0\_1\_6\_2\_report.tex}
\texttt{open 0\_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):

\begin{center}
\begin{tabular}{r|cccccccc}
 & 9 & 36 & 18 & 18 & 6 & 9 & 9 \\
\hline
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 \\
\hline
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 \\
\end{tabular}
\end{center}

The number of points is 35 points:

- $P_0 = (1,0,0,0,0,0)$
- $P_1 = (0,1,0,0,0,0)$
- $P_2 = (0,0,1,0,0,0)$
- $P_3 = (1,0,1,0,0,0)$
- $P_4 = (0,1,1,0,0,0)$
- $P_5 = (0,0,0,1,0,0)$
- $P_6 = (1,0,0,1,0,0)$
- $P_7 = (0,1,0,1,0,0)$
The number of lines is 105

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 \texttt{-without\_group} option.

The command

\begin{verbatim}
Op_6_64_line_rank:
▷ $(ORBITER) -v 4 \\
▷ $\texttt{-define F -finite_field -q 64 -end}$
\end{verbatim}
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 $Q(5, 64)$:

```
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 0 -do -orthogonal_space_activity \\
  > -unrank_line_through_two_points 15447347 15225451 \\
  > -end
```

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:

$$X_0X_1 + X_2X_3 + X_4X_5 = 0$$

<table>
<thead>
<tr>
<th></th>
<th>16769025</th>
<th>1090252800</th>
<th>532350</th>
<th>532350</th>
<th>130</th>
<th>4225</th>
<th>4225</th>
</tr>
</thead>
<tbody>
<tr>
<td>16511040</td>
<td>65</td>
<td>4160</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>266175</td>
<td>0</td>
<td>4096</td>
<td>128</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>266175</td>
<td>0</td>
<td>4096</td>
<td>0</td>
<td>128</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4225</td>
<td>3969</td>
<td>0</td>
<td>126</td>
<td>126</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4225</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4225</td>
<td></td>
</tr>
</tbody>
</table>
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 12.4.

According to Table 4.2, 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

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

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 $Q^+(5, 5)$.

```
 elliptic_quadric_subspace:
  $ (ORBITER) -v 3 \n  \> -define F -finite_field -q 5 -end \n  \> -define v -vector -format 4 \n  \> \> -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" \n  \> \> -end \n  \> -define O -orthogonal_space 1 6 F -end \n  \> -with O -do -orthogonal_space_activity \n  \> \> -intersect_with_subspace v \n  \> \> -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 4.2, 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 \( Q(4,7) \):

```
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:

```
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 \( \mathbb{F}_7 \):

<table>
<thead>
<tr>
<th>Point</th>
<th>Coordinates</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 :</td>
<td>( P_0 = (0,1,0,0,0) )</td>
</tr>
<tr>
<td>1 :</td>
<td>( P_1 = (0,0,1,0,0) )</td>
</tr>
<tr>
<td>2 :</td>
<td>( P_{40} = (0,1,2,6,2) )</td>
</tr>
<tr>
<td>3 :</td>
<td>( P_{41} = (0,1,4,3,1) )</td>
</tr>
<tr>
<td>4 :</td>
<td>( P_{225} = (1,6,6,5,1) )</td>
</tr>
<tr>
<td>5 :</td>
<td>( P_{270} = (1,5,5,4,4) )</td>
</tr>
<tr>
<td>6 :</td>
<td>( P_{241} = (1,2,2,2,1) )</td>
</tr>
</tbody>
</table>
7 : \( P_{340} = (1,1,1,4,3) \)

More on BLT-sets can be found in Section 12.4.

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

**Doily_W_2:**

```bash
$ (ORBITER) -v 2 \
  -define F -finite_field -q 2 -end \
  -define 0 -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 0 -do \
  -orthogonal_space_activity \
  -print_points W2_points \
  -end \
  -with 0 -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 $H(k, Q)$. 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

```
H_2_4:
  $(ORBITER) -v 2 \n  -define F -finite_field -q 4 -end \n  -with F -do -finite_field_activity \n  -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 $H(2, 4)$ contains 9 points:

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

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

The command

```
H_3_4:
  $(ORBITER) -v 2 \n  -define F -finite_field -q 4 -end \n  -with F -do -finite_field_activity \n  -cheat_sheet hermitian 3 -end
  pdflatex H_3_4.tex
  open H_3_4.pdf
```

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

104
The Hermitian variety $H(3, 4)$ contains 45 points:

<table>
<thead>
<tr>
<th>Point Index</th>
<th>Coordinates</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_0$</td>
<td>$(1, 1, 0, 0)$</td>
<td>5</td>
</tr>
<tr>
<td>$P_1$</td>
<td>$(2, 1, 0, 0)$</td>
<td>6</td>
</tr>
<tr>
<td>$P_2$</td>
<td>$(3, 1, 0, 0)$</td>
<td>7</td>
</tr>
<tr>
<td>$P_3$</td>
<td>$(1, 0, 1, 0)$</td>
<td>8</td>
</tr>
<tr>
<td>$P_4$</td>
<td>$(2, 0, 1, 0)$</td>
<td>9</td>
</tr>
<tr>
<td>$P_5$</td>
<td>$(3, 0, 1, 0)$</td>
<td>10</td>
</tr>
<tr>
<td>$P_6$</td>
<td>$(0, 1, 1, 0)$</td>
<td>11</td>
</tr>
<tr>
<td>$P_7$</td>
<td>$(0, 2, 1, 0)$</td>
<td>15</td>
</tr>
<tr>
<td>$P_8$</td>
<td>$(0, 3, 1, 0)$</td>
<td>19</td>
</tr>
<tr>
<td>$P_9$</td>
<td>$(1, 0, 0, 1)$</td>
<td>23</td>
</tr>
<tr>
<td>$P_{10}$</td>
<td>$(2, 0, 0, 1)$</td>
<td>24</td>
</tr>
<tr>
<td>$P_{11}$</td>
<td>$(3, 0, 0, 1)$</td>
<td>25</td>
</tr>
<tr>
<td>$P_{12}$</td>
<td>$(0, 1, 0, 1)$</td>
<td>26</td>
</tr>
<tr>
<td>$P_{13}$</td>
<td>$(0, 2, 0, 1)$</td>
<td>30</td>
</tr>
<tr>
<td>$P_{14}$</td>
<td>$(0, 3, 0, 1)$</td>
<td>34</td>
</tr>
<tr>
<td>$P_{15}$</td>
<td>$(1, 1, 1, 1)$</td>
<td>4</td>
</tr>
<tr>
<td>$P_{16}$</td>
<td>$(2, 1, 1, 1)$</td>
<td>43</td>
</tr>
<tr>
<td>$P_{17}$</td>
<td>$(3, 1, 1, 1)$</td>
<td>44</td>
</tr>
<tr>
<td>$P_{18}$</td>
<td>$(1, 2, 1, 1)$</td>
<td>46</td>
</tr>
<tr>
<td>$P_{19}$</td>
<td>$(2, 2, 1, 1)$</td>
<td>47</td>
</tr>
<tr>
<td>$P_{20}$</td>
<td>$(3, 2, 1, 1)$</td>
<td>48</td>
</tr>
<tr>
<td>$P_{21}$</td>
<td>$(1, 3, 1, 1)$</td>
<td>50</td>
</tr>
<tr>
<td>$P_{22}$</td>
<td>$(2, 3, 1, 1)$</td>
<td>51</td>
</tr>
<tr>
<td>$P_{23}$</td>
<td>$(3, 3, 1, 1)$</td>
<td>52</td>
</tr>
<tr>
<td>$P_{24}$</td>
<td>$(0, 0, 1, 1)$</td>
<td>38</td>
</tr>
<tr>
<td>$P_{25}$</td>
<td>$(1, 1, 2, 1)$</td>
<td>58</td>
</tr>
<tr>
<td>$P_{26}$</td>
<td>$(2, 1, 2, 1)$</td>
<td>59</td>
</tr>
<tr>
<td>$P_{27}$</td>
<td>$(3, 1, 2, 1)$</td>
<td>60</td>
</tr>
<tr>
<td>$P_{28}$</td>
<td>$(1, 2, 2, 1)$</td>
<td>62</td>
</tr>
<tr>
<td>$P_{29}$</td>
<td>$(2, 2, 2, 1)$</td>
<td>63</td>
</tr>
<tr>
<td>$P_{30}$</td>
<td>$(3, 2, 2, 1)$</td>
<td>64</td>
</tr>
<tr>
<td>$P_{31}$</td>
<td>$(1, 3, 2, 1)$</td>
<td>66</td>
</tr>
<tr>
<td>$P_{32}$</td>
<td>$(2, 3, 2, 1)$</td>
<td>67</td>
</tr>
<tr>
<td>$P_{33}$</td>
<td>$(3, 3, 2, 1)$</td>
<td>68</td>
</tr>
<tr>
<td>$P_{34}$</td>
<td>$(0, 0, 2, 1)$</td>
<td>53</td>
</tr>
<tr>
<td>$P_{35}$</td>
<td>$(1, 1, 3, 1)$</td>
<td>74</td>
</tr>
<tr>
<td>$P_{36}$</td>
<td>$(2, 1, 3, 1)$</td>
<td>75</td>
</tr>
<tr>
<td>$P_{37}$</td>
<td>$(3, 1, 3, 1)$</td>
<td>76</td>
</tr>
<tr>
<td>$P_{38}$</td>
<td>$(1, 2, 3, 1)$</td>
<td>78</td>
</tr>
<tr>
<td>$P_{39}$</td>
<td>$(2, 2, 3, 1)$</td>
<td>79</td>
</tr>
<tr>
<td>$P_{40}$</td>
<td>$(3, 2, 3, 1)$</td>
<td>80</td>
</tr>
<tr>
<td>$P_{41}$</td>
<td>$(1, 3, 3, 1)$</td>
<td>82</td>
</tr>
<tr>
<td>$P_{42}$</td>
<td>$(2, 3, 3, 1)$</td>
<td>83</td>
</tr>
<tr>
<td>$P_{43}$</td>
<td>$(3, 3, 3, 1)$</td>
<td>84</td>
</tr>
<tr>
<td>$P_{44}$</td>
<td>$(0, 0, 3, 1)$</td>
<td>69</td>
</tr>
<tr>
<td>$P_{45}$</td>
<td>$(5, 6, 7, 8, 9, 10, 11, 15, 19, 23, 24, 25, 26, 30, 34, 4, 43, 44, 46, 47, 48, 50, 51, 52, 38, 58, 59, 60, 62, 63, 64, 66, 67, 68, 74, 75, 76, 78, 79, 80, 82, 83, 84, 69)$</td>
<td></td>
</tr>
</tbody>
</table>

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 4.5-4.8.

Table 4.9 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 element

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

which is a Baer collineation. It fixes a subgeometry $\text{PG}(3,2)$. The command

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

can be used.

Suppose we are looking for a projectivity of $\text{PG}(3,16)$ 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
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-export_point_line_incidence_matrix</td>
<td></td>
<td>Create a csv file of the point line incidence matrix.</td>
</tr>
<tr>
<td>-table_of_cubic_surfaces_compute_properties</td>
<td>fname q₀ col-offset</td>
<td>See Section 7.5.</td>
</tr>
<tr>
<td>-cubic_surface_properties_analyze</td>
<td>fname q₀</td>
<td>See Section 7.5.</td>
</tr>
<tr>
<td>-canonical_form_of_code</td>
<td>label m n matrix</td>
<td>Compute the automorphism group of a linear code using Nauty. See Section 10.</td>
</tr>
<tr>
<td>-map</td>
<td>label parameters</td>
<td>evaluate a formula using the given parameters</td>
</tr>
<tr>
<td>-analyze_del_Pezzo_surface</td>
<td>label parameters</td>
<td></td>
</tr>
<tr>
<td>-cheat_sheet_for_decomposition_by_element_PG</td>
<td>power elt fname</td>
<td>Analyzes the orbit structure of the cyclic group generated by the given element in the action on PG(n, q).</td>
</tr>
<tr>
<td>-cheat_sheet_for_decomposition_by_subgroup</td>
<td>label descr</td>
<td>Analyzes the orbit structure of the subgroup $H$ in the action on PG(n, q). The subgroup must be a linear group, and the description of $H$ must come from the commands from Section 5.2.</td>
</tr>
<tr>
<td>-table_of_quartic_curves</td>
<td></td>
<td>Export the classification of quartic curves to a csv file.</td>
</tr>
<tr>
<td>-table_of_cubic_surfaces</td>
<td></td>
<td>Export the classification of cubic surfaces to a csv file.</td>
</tr>
<tr>
<td>-classify_surfaces_with_double_sixes</td>
<td>label control</td>
<td>Classify cubic surfaces using the approach of double sixes. See Section 7.3.</td>
</tr>
</tbody>
</table>

Table 4.5: Projective Space Activities (Part 1)
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-classify_surfaces_through_arcs_and_two_lines</code></td>
<td></td>
<td>Classify cubic surfaces using the approach of six-arcs and two skew lines. See Section 7.3.</td>
</tr>
<tr>
<td><code>-classify_surfaces_through_arcs_and_trihedral_pairs</code></td>
<td></td>
<td>See Section 7.3.</td>
</tr>
<tr>
<td><code>-test_nb_Eckardt_points</code></td>
<td>$e$</td>
<td>Restrict to $e$ Eckardt points. See Section 7.3. To be used in conjunction with <code>-classify_surfaces_through_arcs_and_trihedral_pairs</code>.</td>
</tr>
<tr>
<td><code>-sweep</code></td>
<td>fname</td>
<td></td>
</tr>
<tr>
<td><code>-sweep_4</code></td>
<td>fname surface-descr</td>
<td></td>
</tr>
<tr>
<td><code>-sweep_4_27</code></td>
<td>fname surface-descr</td>
<td></td>
</tr>
<tr>
<td><code>-six_arcs_not_on_conic</code></td>
<td></td>
<td>Classify six-arcs not on a conic in a plane.</td>
</tr>
<tr>
<td><code>-filter_by_nb_Eckardt_points</code></td>
<td>$e$</td>
<td>Filter for the number of Eckardt points to be equal to $e$. Used in conjunction with <code>-six_arcs_not_on_conic</code>.</td>
</tr>
<tr>
<td><code>-trihedral1_control</code></td>
<td>poset-control</td>
<td>For <code>-classify_surfaces_through_arcs_and_trihedral_pairs</code></td>
</tr>
<tr>
<td><code>-trihedra2_control</code></td>
<td>poset-control</td>
<td>For <code>-classify_surfaces_through_arcs_and_trihedral_pairs</code></td>
</tr>
<tr>
<td><code>-control_six_arcs</code></td>
<td>poset-control</td>
<td>For <code>-classify_surfaces_through_arcs_and_trihedral_pairs</code></td>
</tr>
</tbody>
</table>

Table 4.6: Projective Space Activities (Part 2)
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-classify_semiﬁelds</td>
<td>descr</td>
<td></td>
</tr>
<tr>
<td>-cheat_sheet</td>
<td></td>
<td>Produce a cheat sheet for PG(n, q)</td>
</tr>
<tr>
<td>-classify_quartic_curves_nauty</td>
<td>fname-mask N</td>
<td>Classify quartic curves using Nauty.</td>
</tr>
<tr>
<td>-classify_quartic_curves_with_</td>
<td>fname-mask N k d</td>
<td>Classify quartic curves using substructure algorithm.</td>
</tr>
<tr>
<td>substructure</td>
<td>fname</td>
<td></td>
</tr>
<tr>
<td>-set_stabilizer</td>
<td>k frame-mask N col-label frame-out</td>
<td>Compute canonical form of sets using the substructure algorithm.</td>
</tr>
<tr>
<td>-conic_type</td>
<td>t set</td>
<td>Compute the conic type of the given set (given by its label). Record intersections of size $\geq t$ only.</td>
</tr>
<tr>
<td>-arc_with_given_set_as_s_lines_</td>
<td>sz d d_{\text{min}} s</td>
<td>Finds arcs with the given set as s-lines.</td>
</tr>
<tr>
<td>after_dualizing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-arc_with_two_given_sets_of_</td>
<td>sz d d_{\text{min}} s t T</td>
<td>Finds arcs with the two given sets as s-lines and t-lines, respectively.</td>
</tr>
<tr>
<td>lines_after_dualizing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-arc_with_three_given_sets_of_</td>
<td>sz d d_{\text{min}} s t T u</td>
<td>Finds arcs with the three given sets as s-lines and t-lines and u-lines, respectively.</td>
</tr>
<tr>
<td>lines_after_dualizing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-dualize_hyperplanes_to_points</td>
<td></td>
<td>Turns ranks of hyperplanes into ranks of points.</td>
</tr>
<tr>
<td>-dualize_points_to_hyperplanes</td>
<td></td>
<td>Turns ranks of points into ranks of hyperplanes.</td>
</tr>
<tr>
<td>-dualize_rank_k_subspaces</td>
<td>k</td>
<td>Turns ranks of $k$-subspaces into ranks of $n - k$ subspaces.</td>
</tr>
<tr>
<td>-classify_arcs</td>
<td>descr</td>
<td>Classify arcs.</td>
</tr>
<tr>
<td>-classify_cubic_curves</td>
<td></td>
<td>Classify cubic curves.</td>
</tr>
</tbody>
</table>

Table 4.7: Projective Space Activities (Part 3)
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-latex_homogeneous_equation</td>
<td>$d$</td>
<td>Produce a latex rendering of the equation of degree $d$</td>
</tr>
<tr>
<td></td>
<td>symb-txt</td>
<td></td>
</tr>
<tr>
<td></td>
<td>symb-tex</td>
<td></td>
</tr>
<tr>
<td></td>
<td>equation</td>
<td></td>
</tr>
<tr>
<td>-lines_on_point_but_within_a_plane</td>
<td>pt-rk</td>
<td>Compute the lines through a given point contained in a given plane.</td>
</tr>
<tr>
<td></td>
<td>plane-rk</td>
<td></td>
</tr>
<tr>
<td>-rank_lines_in_PG</td>
<td>$M$</td>
<td>Rank the lines given in rows of the matrix $M$.</td>
</tr>
<tr>
<td>-unrank_lines_in_PG</td>
<td>$v$</td>
<td>Unrank the lines whose ranks are given in the vector $v$.</td>
</tr>
<tr>
<td>-move_two_lines_in_hyperplane_stabilizer_text</td>
<td>l1 l2 m1 m2</td>
<td>Find the unique transvection fixing the hyperplane at infinity moving l1 and l2 to m1 and m2.</td>
</tr>
<tr>
<td>-planes_through_line</td>
<td>$l$</td>
<td>Find all planes through the line $l$.</td>
</tr>
</tbody>
</table>

Table 4.8: Projective Space Activities (Part 4)

```plaintext
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, $\delta$ is the primitive element in the built-in field $\mathbb{F}_{16}$, satisfying $\delta^4 = \delta^2 + 1$.

It is possible to define algebraic varieties directly from an algebraic equation. We distinguish between managed variables and arbitrary variables. We require that the polynomial is ho-
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-create_points_on_quartic</td>
<td>$\epsilon$</td>
<td>Creates a table of points on a specific quartic curve. Consecutive points are no more than $\epsilon$ apart.</td>
</tr>
<tr>
<td>-create_points_on_parabola</td>
<td>$\epsilon$ $a$ $b$ $c$</td>
<td>Creates a table of points on the parabola $y = ax^2 + bx + c$. Consecutive points are no more than $\epsilon$ apart.</td>
</tr>
<tr>
<td>-smooth_curve</td>
<td>$\epsilon$ $N$ $b$ $t_{\text{min}}$ $t_{\text{max}}$ function</td>
<td>Creates at least $N$ points on a continuous curve given by “function”. Consecutive points are no more than $\epsilon$ apart. The function must be in terms of a parameter $t$. The values of $t$ are taken from the interval $[t_{\text{min}}, t_{\text{max}}]$.</td>
</tr>
<tr>
<td>-make_table_of_surfaces</td>
<td></td>
<td>Produces a latex table summarizing the surfaces in the Orbiter catalogue.</td>
</tr>
<tr>
<td>-create_surface_reports</td>
<td>field-orders</td>
<td>Produce reports for all surfaces in the Orbiter catalogue over the give field orders.</td>
</tr>
<tr>
<td>-create_surface_atlas</td>
<td>$q_{\text{max}}$</td>
<td>Produce reports for all surfaces in the Orbiter catalogue for field orders $q \leq q_{\text{max}}$.</td>
</tr>
<tr>
<td>-create_dickson_atlas</td>
<td></td>
<td>Produce reports of Dickson surfaces.</td>
</tr>
</tbody>
</table>

Table 4.9: Orbiter commands related to projective geometries
mogeneous in the managed variables. The other variables can be used to represent scalar parameters, for instance. Here is an example. Suppose we want to study the del Pezzo surfaces

\[ f_3 : w^2 = x^4 + y^4 + z^4 + 8x^2y^2 + 8x^2z^2 + 8y^2z^2, \quad f_4 : w^2 = x^4 + y^4 + z^4 - x^2y^2. \]

Orbiter assumes that the equation has \( w^2 \) on the left hand side. Therefore, only the right hand side of the equation needs to be given. We translate the equation into simplified notation as follows:

\[ x*x*x*x+y*y*y*y+z*z*z*z+8*x*x*y*y+8*x*x*z*z+8*y*y*z*z \]

for \( f_3 \) and

\[ x*x*x*x+y*y*y*y+z*z*z*z-x*x*y*y \]

for \( f_4 \). The following command can be used to produce a report on the two surfaces over the field \( \mathbb{F}_{13} \).

```
del_Pezzo_F13ab_report:
  \$\texttt{ORBITER} -v 3 \\
  \texttt{-define F -finite_field -q 13 -end} \\
  \texttt{-define P -projective_space -n 3 -field F -v 0 -end} \\
  \texttt{-define f3 -formula} \texttt{"del_Pezzo_F13a" "x,y,z"} \\
  \texttt{-define f4 -formula} \texttt{"del_Pezzo_F13b" "x,y,z"} \\
  \texttt{-define del_Pezzo13 -collection "f3,f4"} \\
  \texttt{-with P -do} \\
  \texttt{-projective_space_activity} \\
  \texttt{-analyze_del_Pezzo_surface del_Pezzo13 ""} \\
  \texttt{-end} \\
  \texttt{pdflatex del_Pezzo_F13b_report.tex} \\
  \texttt{open del_Pezzo_F13b_report.pdf}
```

The third argument after the `-formula` command specifies the managed variables, which are \( x,y,z \). The command `-collection` is used to group objects together. In this case, both surfaces are grouped together under the new name. That way, we can issue the `-analyze_del_Pezzo_surface` once, and it applies to both surfaces.
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 4.10 and 4.11 can be issued. Modifier options as shown in Table 4.12 apply.

The following command creates an elliptic quadric ovoid on PG(3, 8):

```
elliptic_quadric_ovoid_q8:
  $(ORBITER) -v 2 \n  ▷ -define F -finite_field -q 8 -end \n  ▷ -define P -projective_space -n 3 -field F -v 0 -end \n  ▷ -define O -geometric_object P \n  ▷ -elliptic_quadric_ovoid \n  ▷ -end \n  ▷ -with O -do -combinatorial_object_activity -save \n  ▷ -end
```

The next command creates the Suzuki-Tits ovoid in PG(3, 8):

```
ovoid_ST_q8:
  $(ORBITER) -v 2 \n  ▷ -define F -finite_field -q 8 -end \n  ▷ -define P -projective_space -n 3 -field F -v 0 -end \n  ▷ -define O -geometric_object P \n  ▷ -ovoid_ST \n  ▷ -end \n  ▷ -with O -do -combinatorial_object_activity -save \n  ▷ -end
```

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 PG(2, 17) and saves it to file. The equation is encoded using the ordering of quartic monomials from Table 8.4.

```
EDGE_CURVE_Q17_EQUATION="1,16,16,0,0,0,0,0,0,0,0,0,1,12,0,0"
EDGE_CURVE_Q17_AS_POINTS="4, 7, 16, 19, 20, 23, 32, 35, 89, 100, 244, 251"
```

113
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-hyperoval</td>
<td></td>
<td>To create a hyperoval</td>
</tr>
<tr>
<td>-subiaco_oval</td>
<td>f_short</td>
<td>Create the Subiaco oval</td>
</tr>
<tr>
<td>-subiaco_hyperoval</td>
<td></td>
<td>Create the Subiaco hyperoval</td>
</tr>
<tr>
<td>-adelaide_hyperoval</td>
<td></td>
<td>Create the Adelaide hyperoval</td>
</tr>
<tr>
<td>-translation</td>
<td>i</td>
<td>Create the translation hyperoval with exponent i</td>
</tr>
<tr>
<td>-Segre</td>
<td></td>
<td>Create the Segre hyperoval</td>
</tr>
<tr>
<td>-Payne</td>
<td></td>
<td>Create the Payne hyperoval</td>
</tr>
<tr>
<td>-Cherowitzo</td>
<td></td>
<td>Create the Cherowitzo hyperoval</td>
</tr>
<tr>
<td>-OKeefe_Penttila</td>
<td></td>
<td>Create the O'Keefe, Penttila hyperoval</td>
</tr>
<tr>
<td>-BLT_database</td>
<td>k</td>
<td>Create the kth BLT-set of order $q$ from the database ($k = 0, 1, \ldots$)</td>
</tr>
<tr>
<td>-elliptic_quadric_ovoid</td>
<td></td>
<td>Create an elliptic quadric ovoid in $\text{PG}(3,q)$.</td>
</tr>
<tr>
<td>-ovoid_ST</td>
<td></td>
<td>Create the Suzuki Tits ovoid in $\text{PG}(3,q)$. Here, $q = 2^{2r+1}$.</td>
</tr>
<tr>
<td>-Baer</td>
<td></td>
<td>Create the (standard) Baer subgeometry</td>
</tr>
<tr>
<td>-orthogonal</td>
<td>$\epsilon$</td>
<td>Create the $Q^\epsilon(n,q)$ quadric</td>
</tr>
<tr>
<td>-hermitian</td>
<td></td>
<td>Create the Hermitian variety given by $\sum_{i=0}^{n} X_i^{\sqrt{q}+1} = 0$</td>
</tr>
<tr>
<td>-cuspidal_cubic</td>
<td></td>
<td>Create the cuspidal cubic $(s^3,ts^2,t^3)$ in $\text{PG}(2,q)$</td>
</tr>
<tr>
<td>-twisted_cubic</td>
<td></td>
<td>Create a twisted cubic $(s^3,s^2t,st^2,t^3)$ in $\text{PG}(3,q)$</td>
</tr>
<tr>
<td>-elliptic_curve</td>
<td>a b</td>
<td>Create the elliptic curve $y^2 = x^3 + ax + b$</td>
</tr>
<tr>
<td>-ttp_construction_A</td>
<td></td>
<td>Create the twisted tensor product code of type A [7]</td>
</tr>
<tr>
<td>-ttp_construction_A_hyperoval</td>
<td></td>
<td>Create the twisted tensor product code of type A [7]</td>
</tr>
<tr>
<td>-ttp_construction_B</td>
<td></td>
<td>Create the twisted tensor product code of type B [7]</td>
</tr>
</tbody>
</table>

Table 4.10: Orbiter Objects (Part 1)
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-unital_XXq_YZq_ZYq</td>
<td></td>
<td>Create the unital with equation $XX^q+YZ^q+ZY^q = 0$</td>
</tr>
<tr>
<td>-desarguesian_line_spread_in_PG_3_q</td>
<td></td>
<td>Create the desarguesian line spread in $\text{PG}(3,q)$ as a set of 2-subspaces</td>
</tr>
<tr>
<td>-Buekenhout_Metz</td>
<td></td>
<td>Create the Buekenhout Metz unital</td>
</tr>
<tr>
<td>-Uab</td>
<td>$a \ b$</td>
<td>Create the Buekenhout Metz unital in the form of Barwick and Ebert [5]</td>
</tr>
<tr>
<td>-whole_space</td>
<td></td>
<td>Create the whole space</td>
</tr>
<tr>
<td>-hyperplane</td>
<td>$\text{pt}$</td>
<td>Create the hyperplane given by dual coordinates associated with the given point</td>
</tr>
<tr>
<td>-segre_variety</td>
<td>$a \ b$</td>
<td>Create the Segre variety</td>
</tr>
<tr>
<td>-Maruta_Hamada_arc</td>
<td></td>
<td>Create the Maruta Hamada arc</td>
</tr>
<tr>
<td>-projective_variety</td>
<td>lab_ascii lab_tex $d$ coeffs</td>
<td>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.</td>
</tr>
<tr>
<td>-intersection_of_zariski_open_sets</td>
<td>$l \ d \ n \ C_1 \ ... \ C_n$</td>
<td>Create the intersection of the Zariski open sets given by equations $C_1, \ldots C_n$ of degree $d$ with label $l$, see Section 4.5.</td>
</tr>
<tr>
<td>-projective_curve</td>
<td>$l \ r \ d \ C$</td>
<td>Create the projective curve of degree $d$ with label $l$, with coefficient vector $C$ in $r$ variables</td>
</tr>
</tbody>
</table>

Table 4.11: Orbiter Objects (Part 2)

<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-embedded_in_PG_4_q</td>
<td></td>
<td>Create the BLT-set with ranks in $\text{PG}(n,q)$ instead of orthogonal point ranks</td>
</tr>
<tr>
<td>-BLT_in_PG</td>
<td></td>
<td>Create the BLT-set with ranks in $\text{PG}(n,q)$ instead of orthogonal point ranks</td>
</tr>
<tr>
<td>-monomial_type_LEX</td>
<td></td>
<td>Select lexicographic ordering of coefficients in an algebraic equation.</td>
</tr>
<tr>
<td>-monomial_type_PART</td>
<td></td>
<td>Select partition ordering of coefficients in an algebraic equation (default).</td>
</tr>
</tbody>
</table>

Table 4.12: Orbiter Objects: Modifiers

115
The following command computes the line type of the Edge curve:

```bash
> 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 -save
> -end
```

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

Group Theory

5.1 Permutation Groups

Permutation groups can be represented on a computer using the technique of stabilizer chains, or Sims chains (cf. [36, 63]). The stabilizer chain is defined 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.

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.

In Section 5.2, we will discuss matrix groups over finite fields. 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 \([0, q^n - 1]\).

Group elements can be defined using a compact representation as integer vectors. For instance, for linear groups, the coding of elements consists of the entries of the associated matrix (for projective matrix groups, the coding is not unique as scalar multiples of the matrix describe the same group element). For semilinear matrix groups, an extra integer is used to
describe the associated field automorphism as a power of the generator of the group of field automorphisms (the Frobenius endomorphism as transformation). For affine groups, the coding consists of a matrix, a vector and possible a integer describing a field automorphism. Generating sets of groups can be specified by listing generators in coded form.

Let us start with a cyclic group. The following command creates a cyclic group of order 6:

```
Cyclic_6:
> $(ORBITER) -v 3 \
>   -define G -permutation_group -cyclic_group 6 -end \
>   -with G -do \
>   -group_theoretic_activity \n>   -report \n>   -end
> pdflatex C_6_report.tex
> #open C_6_report.pdf
```

The following command produces a graphical representation of the group table of the cyclic group $C_6$, shown in Figure 5.1.

```
Cyclic_6_group_table:
> $(ORBITER) -v 3 \
>   -define G -permutation_group -cyclic_group 6 -end \
>   -with G -do \
>   -group_theoretic_activity \n>   -export_group_table \n>   -end
> $(ORBITER) -v 2 \
>   -define all_one_r -vector -repeat 1 6 -end \
```

Figure 5.1: The group table of $C_6$
Next, let us consider the symmetric group Sym(n). The following command creates Sym(3):

```bash
Symmetric_3:
- $(ORBITER) -v 3 
- -define G -permutation_group -symmetric_group 3 -end 
- -with G -do 
- -group_theoretic_activity 
- -report 
- -end
```

The report is shown below:

**Stabilizer chain**

<table>
<thead>
<tr>
<th>Level</th>
<th>Base pt</th>
<th>Orbit length</th>
<th>Subgroup order</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

**Basic Orbit 0**

```
```

121
Figure 5.2: The group table of Sym(3)

- Basic orbit 0 has size 3
  - 0, 1, 2

Basic Orbit 1

- Basic orbit 1 has size 2
  - 1, 2

The following command produces a graphical representation of the group table of the symmetric group Sym(3), shown in Figure 5.2.

```bash
Symmetric_3_group_table:
  $(ORBITER) -v 3 \
  -define G -permutation_group -symmetric_group 3 -end \
  -with G -do \
  -group_theoretic_activity \
```

122
The next command produces a graphical representation of the elements of the symmetric group Sym(3), shown in Figure 5.3.

```
$ (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
```

The next command produces a graphical representation of the elements of the symmetric group Sym(3), shown in Figure 5.3.

```
$ (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 \\
```

123
\begin{verbatim}
  \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
\end{verbatim}
5.2 Linear Groups

Orbiter provides support for matrix groups and their various permutation representationes. For background information about the classical groups of matrices over finite fields, see cf. [68]. Any group in Orbiter is associated with a permutation action. There can be multiple actions for the same group though. Using homomorphisms of permutation groups, new actions can be formed from old actions. Basic group actions are projective, affine, and general linear, as well as orthogonal, unitary and tensor product. Product actions can be defined also. In order to establish a permutation representation, the elements (aka points) of the permutation domain need to be made available. One way would be to make a table of all elements in the permutation domain. However, this would be time and memory intensive. For this reason, a different technique is used that creates points only when needed. The way this works is that the permutation domain is encoded implicitly, using a fixed bijection to a suitable integer interval (zero based), called the domain. Whenever we want the ith point in the domain, we can call a function that produces it. Conversely, whenever we have a point, we can call a function that tells us what the associated index in the domain. This is facilitated by two mutually inverse functions. The rank function turns a point into an index. The unrank function turns an index in the domain into a point. Rank and unrank functions are helpful because they eliminate the need for tables of all objects. The ranks lead to rather compact storage of objects in files. The objects can be reconstructed from the ranks.

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 $\text{PG}(n-1,q)$. Let $\pi$ be a projective space. A collineation of a projective space $\pi$ is a bijective mapping from the points of $\pi$ to themselves which preserves collinearity. That is, a collineation $\varphi$ 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 $\Psi_M$ is the induced map on projective space. By considering the homomorphism $M \mapsto \Psi_M$, the group $\text{GL}(n+1,q)$ of invertible endomorphisms becomes a subgroup of the group of collineations of $\text{PG}(n,q)$. This is the projectivity group $\text{PGL}(n+1,q)$. It is isomorphic to $\text{GL}(n+1,q)/\mathbb{F}_q^\times$. Another source of collineations is this: Let $\Phi \in \text{Aut}(\mathbb{F}_q)$ be a field automorphism. Then $\Phi$ acts on projective space by sending $P(x)$ to $P(x\Phi)$. This map is another type of collineation, called automorphic collineation. This way, $\text{Aut}(\mathbb{F}_q)$ gives rise to a group of collineations. If $q = p^h$ for some prime $p$ and some integer $h$ then

$$\Phi_0 : \mathbb{F}_q \rightarrow \mathbb{F}_q, \ x \mapsto x^p$$

is a generator for the cyclic group $C_h \simeq \text{Aut}(\mathbb{F}_q)$. The collineation group of $\text{PG}(n,q)$ ($n \geq 2$) is isomorphic to the semidirect product of the projectivity group and the automorphism group of the field. The collineation group is $\text{PGL}(n+1,q) = \text{PGL}(n+1,q) \rtimes \text{Aut}(\mathbb{F}_q)$. We use the following notation for elements of $\text{PGL}(n+1,q)$. Let $\Phi_0$ be a generator for $\text{Aut}(\mathbb{F}_q)$ and let $M \in \text{GL}(n+1,q)$. The map

$$(\Psi_M, \Phi_0^k) : \text{PG}(n,q) \rightarrow \text{PG}(n,q), \ P(x) \mapsto P(y), \ y = (x \cdot M)^{\Phi_0^k}$$
is denoted as
\[ M_k. \] (5.1)

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
\[ M_k \cdot N_l = (M \cdot N^{\Phi^{-k}})_{k+l}, \] (5.2)
\[ (M_k)^{-1} = \left( (M^{-1})^{\Phi^k} \right)^{-k}. \] (5.3)

The affine group \( \text{AGL}(n, q) \) is the semidirect product of \( \text{GL}(n, q) \) with \( \mathbb{F}_q^n \). The affine semi-linear group \( \text{AΓL}(n, q) \) is the semidirect product of \( \text{AGL}(n, q) \) with \( \text{Aut}(\mathbb{F}_q) \). The elements of \( \text{AΓL}(n, q) \) are triples
\[ M_{a,k} := (M, a, k) \in \text{GL}(n, q) \times \mathbb{F}_q^n \times \text{Aut}(\mathbb{F}_q), \]
which act on \( \mathbb{F}_q^n \):
\[ (x, (M, a, k)) \mapsto (x \cdot M + a)^{\Phi_k}. \]

The multiplication in \( \text{AΓL}(n, q) \) is
\[ M_{a,k} \cdot N_{b,l} = (MN)_{aN^{\Phi^{-k}} + b^{\Phi^{-k}}, k+l}. \]

The inverse of an element is
\[ \left( M_{a,k} \right)^{-1} = \left( M^{-1} \right)_{a^{\Phi_k}M^{-1}, -k}. \]

A correlation is a one-to-one mapping between the set of points and the set of hyperplanes which reverses incidence. So, if \( \rho \) is a correlation and \( P \) is a point and \( \ell \) is a hyperplane then \( P^{\rho} \) is a hyperplane and \( \ell^{\rho} \) 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
\[ \rho : \mathcal{P} \leftrightarrow \mathcal{L}, \quad P(x) \leftrightarrow [x]. \]

A group \( G \) can act on \( V \) in one of the types listed in Table 5.1. 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. The basic types of matrix groups in Orbiter are listed in Table 5.2.

For instance,
<table>
<thead>
<tr>
<th>Type</th>
<th>Perm. Domain</th>
<th>Degree</th>
</tr>
</thead>
<tbody>
<tr>
<td>General linear GL((n, q))</td>
<td>all vectors of (V)</td>
<td>(q^n)</td>
</tr>
<tr>
<td>Affine AGL((n, q))</td>
<td>all vectors of (V)</td>
<td>(q^n)</td>
</tr>
<tr>
<td>Projective PGL((n, q))</td>
<td>(\mathfrak{S}r_1(V))</td>
<td>(\frac{q^n - 1}{q - 1})</td>
</tr>
<tr>
<td>Wreath product GL((d, q) \wr \text{Sym}(n))</td>
<td>(\mathfrak{S}r_1(\mathbb{F}_d^q)^{\otimes n})) extended</td>
<td>(n + nq^d + \frac{q^n - 1}{q - 1})</td>
</tr>
<tr>
<td>Orthogonal PGO((n, q))</td>
<td>(Q(V))</td>
<td>(\frac{q^{n-1} - 1}{q - 1})</td>
</tr>
<tr>
<td>Orthogonal PGO(^+(n, q))</td>
<td>(Q^+(V))</td>
<td>(\frac{(q^{n/2} - 1)(q^{(n-2)/2} + 1)}{q - 1})</td>
</tr>
<tr>
<td>Orthogonal PGO(^-(n, q))</td>
<td>(Q^-(V))</td>
<td>(\frac{(q^{n/2} + 1)(q^{(n-2)/2} - 1)}{q - 1})</td>
</tr>
</tbody>
</table>

Table 5.1: Basic actions

PGL\(_{4,2}\):

▶ $(\text{ORBITER})$ -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
▶ pdflatex PGL\(_{4,2}\) report.tex
▶ open PGL\(_{4,2}\) report.pdf

creates the group PGL\((4, 2)\) acting on the 15 elements of \(\mathfrak{S}r_1(\mathbb{F}_2^4)\). At first, the field \(\mathbb{F}_2\) is created. Secondly, the group \(G = \text{PGL}(3, 2)\) 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 PGL\((4, 2)\)

The order of the group PGL\((4, 2)\) is 20160
The group acts on a set of size 15
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>-GL</td>
<td>n q</td>
<td>GL(n, q)</td>
</tr>
<tr>
<td>-GGL</td>
<td>n q</td>
<td>ΓL(n, q)</td>
</tr>
<tr>
<td>-SL</td>
<td>n q</td>
<td>SL(n, q)</td>
</tr>
<tr>
<td>-SSL</td>
<td>n q</td>
<td>ΣL(n, q)</td>
</tr>
<tr>
<td>-PGL</td>
<td>n q</td>
<td>PGL(n, q)</td>
</tr>
<tr>
<td>-PGGL</td>
<td>n q</td>
<td>PΓL(n, q)</td>
</tr>
<tr>
<td>-PSL</td>
<td>n q</td>
<td>PSL(n, q)</td>
</tr>
<tr>
<td>-PSSL</td>
<td>n q</td>
<td>PΣL(n, q)</td>
</tr>
<tr>
<td>-AGL</td>
<td>n q</td>
<td>AGL(n, q)</td>
</tr>
<tr>
<td>-AGGL</td>
<td>n q</td>
<td>AΓL(n, q)</td>
</tr>
<tr>
<td>-ASL</td>
<td>n q</td>
<td>ASL(n, q)</td>
</tr>
<tr>
<td>-ASSL</td>
<td>n q</td>
<td>AΣL(n, q)</td>
</tr>
<tr>
<td>-PGO</td>
<td>n q</td>
<td>PGO(n, q)</td>
</tr>
<tr>
<td>-PGOp</td>
<td>n q</td>
<td>PGO^+(n, q)</td>
</tr>
<tr>
<td>-PGO⁻</td>
<td>n q</td>
<td>PGO^−(n, q)</td>
</tr>
<tr>
<td>-PGGO</td>
<td>n q</td>
<td>PΓO(n, q)</td>
</tr>
<tr>
<td>-PGGO⁻</td>
<td>n q</td>
<td>PΓO^−(n, q)</td>
</tr>
<tr>
<td>-GL_d_q_wr_Sym_n</td>
<td>d q n</td>
<td>GL(d, q) ⋊ Sym(n)</td>
</tr>
</tbody>
</table>

Table 5.2: Basic types of Orbiter matrix groups
Strong generators for a group of order 20160:

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1
\end{bmatrix},
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 1
\end{bmatrix},
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 1 & 1
\end{bmatrix},
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0
\end{bmatrix},
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix},
\begin{bmatrix}
0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix},
\begin{bmatrix}
0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

The Action

Group action PGL(4, 2) of degree 15
We act on the following set:

0 = (1, 0, 0, 0)  8 = (1, 1, 1, 0)
1 = (0, 1, 0, 0)  9 = (1, 0, 0, 1)
2 = (0, 0, 1, 0)  10 = (0, 1, 0, 1)
3 = (0, 0, 0, 1)  11 = (1, 1, 0, 1)
4 = (1, 1, 1, 1)  12 = (0, 0, 1, 1)
5 = (1, 1, 0, 0)  13 = (1, 0, 1, 1)
6 = (1, 0, 1, 0)  14 = (0, 1, 1, 1)
7 = (0, 1, 1, 0)

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
The finite field $\mathbb{F}_2$

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

<table>
<thead>
<tr>
<th>$i$</th>
<th>$\gamma_i$</th>
<th>$-\gamma_i$</th>
<th>$\gamma_i^{-1}$</th>
<th>$\log_\alpha (\gamma_i)$</th>
<th>$\alpha^i$</th>
<th>$Z_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>DNE</td>
<td>DNE</td>
<td>1</td>
<td>DNE</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

$\alpha$ and $\beta$:

\[
\begin{array}{c}
+ \\
0 & 0 & 1 \\
0 & 0 & 1 \\
0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{c}
\cdot \\
1 & 1 & 0 \\
1 & 1 & 0 \\
1 & 1 & 0 \\
\end{array}
\]

$1^0 \equiv 1$

$1^1 \equiv 1$

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:

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 \\
\end{bmatrix},
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix},
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix},
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\]

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,
Stabilizer chain

<table>
<thead>
<tr>
<th>Level</th>
<th>Base pt</th>
<th>Orbit length</th>
<th>Subgroup order</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>15</td>
<td>20160</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>14</td>
<td>1344</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>12</td>
<td>96</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

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 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 4 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

133
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.

```
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:

```
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 5.2 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
134
```
The command

\texttt{L\_5\_3:
\begin{verbatim}
  $(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
\end{verbatim}
pdflatex PSL\_5\_3_report.tex
open PSL\_5\_3_report.pdf
}

creates PSL(5, 3) of order 237783237120.

The command

\texttt{PSP\_4\_4:
\begin{verbatim}
  $(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
\end{verbatim}
pdflatex PGL\_4\_4_Sp\_4\_4_report.tex
open PGL\_4\_4_Sp\_4\_4_report.pdf
}

creates the symplectic group PSp(4, 4) of order 979200.

The command

\texttt{PGO\_5\_2:
\begin{verbatim}
  $(ORBITER) -v 2 \ 
\end{verbatim}
}
creates the group PGO(5, 2) acting on the 15 points of the $Q(4, 2)$ quadric. The following latex report is produced:

**The Group PGO(5, 2)**

The order of the group PGO(5, 2) is 720

The group acts on a set of size 15

Strong generators for a group of order 720:

- $\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 \end{bmatrix}$,
- $\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$,
- $\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \end{bmatrix}$,
- $\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$,
- $\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix}$,
- $\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$,
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 \( \mathbb{F}_2 \)

\[ Z_i = \log_\alpha (1 + \alpha^j) \]

<table>
<thead>
<tr>
<th>( i )</th>
<th>( \gamma_i )</th>
<th>( -\gamma_i )</th>
<th>( \gamma_i^{-1} )</th>
<th>( \log_\alpha (\gamma_i) )</th>
<th>( \alpha^j )</th>
<th>( Z_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 = 0</td>
<td>0</td>
<td>DNE</td>
<td>DNE</td>
<td>1</td>
<td>DNE</td>
</tr>
<tr>
<td>1</td>
<td>1 = 1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>DNE</td>
</tr>
</tbody>
</table>

\[ + \]

\begin{array}{ccc}
0 & 0 & 1 \\
0 & 1 & 1 \\
1 & 1 & 0 \\
\end{array}

\[ \cdot \]

\begin{array}{c}
1 \\
1 \\
1 \\
\end{array}

137
$1^0 \equiv 1$

$1^1 \equiv 1$

## Base and Stabilizer Chain

Group order 720

$t_l=15, 8, 3, 1, 1, 2$

Base: $(0, 1, 2, 3, 4, 5)$

Strong generators for a group of order 720:

\[
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 1
\end{bmatrix},
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 1
\end{bmatrix},
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 \\
1 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 1
\end{bmatrix},
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 & 1
\end{bmatrix},
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 1
\end{bmatrix},
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

<table>
<thead>
<tr>
<th>Level</th>
<th>Base pt</th>
<th>Orbit length</th>
<th>Subgroup order</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>15</td>
<td>720</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>8</td>
<td>48</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
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 symplectic group PSp(6, 2) can be created using the following command:

```
PSP_6_2:
> $(ORBITER) -v 2 \
>   -define F -finite_field -q 2 -end \n>   -define G -linear_group -PGL 6 F \n>   -symplectic_group \n>   -end \n>   -with G -do \n>   -group_theoretic_activity \n```
The group PGO(7, 2), isomorphic to PSp(6, 2), can be created using the following command:

PGO_7_2:
  $\text{(ORBITER)} -v 2 \backslash$
  $\text{-define F -finite_field -q 2 -end} \backslash$
  $\text{-define G -linear_group -PGO 7 F -end} \backslash$
  $\text{-with G -do} \backslash$
  $\text{-group_theoretic_activity} \backslash$
  $\text{report} \backslash$
  $\text{-end} \backslash$
  pdflatex PGO_7_2_report.tex
  open PGO_7_2_report.pdf
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-Janko1</td>
<td></td>
<td>first Janko group, needs PGL(7,11)</td>
</tr>
<tr>
<td>-monomial</td>
<td></td>
<td>subgroup of monomial matrices</td>
</tr>
<tr>
<td>-diagonal</td>
<td></td>
<td>subgroup of diagonal matrices</td>
</tr>
<tr>
<td>-null_polarity_group</td>
<td></td>
<td>null polarity group</td>
</tr>
<tr>
<td>-symplectic_group</td>
<td></td>
<td>symplectic group</td>
</tr>
<tr>
<td>-singer</td>
<td>$k$</td>
<td>subgroup of index $k$ in the Singer cycle</td>
</tr>
<tr>
<td>-singer_and_frobenius</td>
<td>$k$</td>
<td>subgroup of index $k$ in the Singer cycle, extended by the Frobenius automorphism of $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$</td>
</tr>
<tr>
<td>-borel_upper</td>
<td></td>
<td>Borel subgroup of upper triangular matrices</td>
</tr>
<tr>
<td>-borel_lower</td>
<td></td>
<td>Borel subgroup of lower triangular matrices</td>
</tr>
<tr>
<td>-identity_group</td>
<td></td>
<td>identity subgroup</td>
</tr>
<tr>
<td>-subgroup_from_file</td>
<td>$f l$</td>
<td>read subgroup from file $f$ and give it the label $l$</td>
</tr>
<tr>
<td>-orthogonal</td>
<td>$\epsilon$</td>
<td>orthogonal group $O^\epsilon(n,q)$, with $\epsilon \in {\pm 1}$ when $n$ is even</td>
</tr>
<tr>
<td>-subgroup_by_generators</td>
<td>$o n s_1 \ldots s_n$</td>
<td>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 $s_1, \ldots, s_n$ are the generators for the subgroup in vector form.</td>
</tr>
</tbody>
</table>

Table 5.3: Commands for creating subgroups

5.3 Subgroups

There are many ways to create subgroups of a group. Table 5.3 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.

```
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)
```

C13:
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 generator is given in list notation, which is the second row in the array
\[
\begin{bmatrix}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 0
\end{bmatrix}.
\]

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 5.6.

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:

```
C13_generated:
```

```
$\$(ORBITER) -v 2 \\
```
The activity `-report` produces a report for the cyclic group, shown below:

### Stabilizer chain

<table>
<thead>
<tr>
<th>Level</th>
<th>Base pt</th>
<th>Orbit length</th>
<th>Subgroup order</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>13</td>
<td>13</td>
</tr>
</tbody>
</table>

#### 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"
```
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 $\text{Sym}(13)$.

**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
```

The `subgroup_by_generators` command will be discussed in more detail in Section 5.3.

For instance, the command

**J1:**

```
▷ $(ORBITER) -v 2 \\
▷ ▷ -define G -linear_group -PGL 7 11 -Janko1 -end \\
▷ ▷ -with G -do \\
▷ ▷ -group_theoretic_activity \\
▷ ▷ ▷ -report \\
```

The `subgroup_by_generators` command will be discussed in more detail in Section 5.3.
creates the first Janko group as a subgroup of $\text{PGL}(7,11)$.

The command

\begin{verbatim}
PGL_3.11_singer:
\$ (ORBITER) -v 2 \ 
\texttt{-define G -linear_group -PGL 3 11 -singer 19 -end} \ 
\texttt{-with G -do} \ 
\texttt{-group_theoretic_activity} \ 
\texttt{-report} \ 
\texttt{-end} \ 
pdfflatex PGL_3.11_Singer_3.11_19_report.tex \ 
\texttt{open PGL_3.11_Singer_3.11_19_report.pdf}
\end{verbatim}

creates a subgroup of the Singer cycle of order 7. The Singer cycle in $\text{GL}(d,q)$ is a generator for a subgroup of order $q^d - 1$. It induces an element of order $q^d - 1$ on the associated projective geometry $\text{PG}(d-1,q)$. The additional integer parameter $k$ after the \texttt{-singer} command is used to create the subgroup of index $k$ of the Singer cycle.

The command

\begin{verbatim}
PGL_3.11_singer_and_frobenius:
\$ (ORBITER) -v 2 \ 
\texttt{-define G -linear_group -PGL 3 11 -singer_and_frobenius 19 -end} \ 
\texttt{-with G -do} \ 
\texttt{-group_theoretic_activity} \ 
\texttt{-report} \ 
\texttt{-end} \ 
pdfflatex PGL_3.11_Singer_and_Frob3.11_19_report.tex \ 
\texttt{open PGL_3.11_Singer_and_Frob3.11_19_report.pdf}
\end{verbatim}

creates a subgroup of index 19 of the Singer cycle of $\text{PG}(2,11)$, 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 $\text{SL}(2,3)$. The Orbiter command
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 GL_2_3_Subgroup_quaternion_8_elements.tex
▷ open GL_2_3_Subgroup_quaternion_8_elements.pdf
▷ pdflatex GL_2_3_Subgroup_quaternion_8_report.tex
▷ open GL_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
```

149
The group of the cube can be created over the field $\mathbb{F}_3$:

cube_group:

```bash
$ (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, \
  2,0,0,1,0,0,0,1" \
  -end \
  -define G -linear_group -GL 3 3 \
  -subgroup_by_generators "cube" "48" 3 \
  -gns \
  -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
```

```
pdflatex GL_3_3_Subgroup_cube_48_elements.tex
open GL_3_3_Subgroup_cube_48_elements.pdf
```

The tetrahedral subgroup can be created as well:

tetra_group:

```bash
$ (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
```

```
pdflatex GL_3_3_Subgroup_tetra_12_elements.tex
open GL_3_3_Subgroup_tetra_12_elements.pdf
```

150
The Hesse group of order 216 extended by the automorphism group of the field can be created in PG(3, 4)

```
GENERATORS_HESSE_GROUP="\n3000300030 \n2000201230 \n1000100111 \n1000220200 \n1002312010 \n0331003211 \n2200011331"
```

Hesse group:
▷ $(ORBITER) -v 3 \n▷ ▷ -define gens -vector -compact \n▷ ▷ ▷ $(GENERATORS_HESSE_GROUP) \n▷ ▷ -end \n▷ ▷ -define G -linear_group -PGGL 3 4 \n▷ ▷ -subgroup_by_generators "Hesse" "432" 7 gens \n▷ ▷ -end \n▷ ▷ -with G -do \n▷ ▷ -group_theoretic_activity \n▷ ▷ ▷ -print_elements.tex \n▷ ▷ ▷ -report \n▷ ▷ -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 GL(8, 3) using the following command:

```
GENERATORS_WEYL_GROUP_E8="\n-1,-1,-1,-1,0,0,0,0, \n0,0,0,1,0,0,0,0, \n1,0,0,0,0,0,0,0, \n0,0,1,0,0,0,0,0, \n0,1,0,1,1,0,0,0, \n0,0,0,0,0,1,0,0, \n0,0,0,0,0,0,1,0, \n0,0,0,0,0,0,1,0, \n-1,0,-1,-1,-1,-1,-1, \n0,1,0,1,1,1,1,1, \n```

151
Weyl_E8:
▷ $(ORBITER) -v 3 \n▷ ▷ -define gens -vector -dense \n▷ ▷ ▷ $(GENERATORS_WEYL_GROUP_E8) \n▷ ▷ -end \n▷ ▷ -define G -linear_group -GL 8 3 \n▷ ▷ -subgroup_by_generators \n▷ ▷ ▷ "WeylE8" "696729600" 2 gens \n▷ ▷ -end \n▷ ▷ -with G -do \n▷ ▷ -group_theoretic_activity \n▷ ▷ ▷ -report \n▷ ▷ ▷ -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:


We can test if a group is a subgroup of another. In the following example, we test whether PGO\(^+(6,2)\) is a subgroup of 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.

test_subgroup:
▷ $(ORBITER) -v 2 \n▷ ▷ -define F -finite_field -q 2 -end \n▷ ▷ -define G1 -linear_group -PGOp 6 F -end \n▷ ▷ -define G2 -linear_group -PGL 6 F \n▷ ▷ ▷ -symplectic_group \n▷ ▷ -end \n▷ ▷ -with G1 -and G2 -do \n▷ ▷ -group_theoretic_activity \n▷ ▷ ▷ -is_subgroup_of \n▷ ▷ ▷ -end
Since the subgroup index is small (36), we create a set of coset representatives using the following command:

```bash
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:

<table>
<thead>
<tr>
<th>coset 0 / 36:</th>
<th>coset 35 / 36:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 0 0 0 0</td>
<td>1 0 1 1 1 0</td>
</tr>
<tr>
<td>0 1 0 0 0 0</td>
<td>1 0 1 0 0 0</td>
</tr>
<tr>
<td>0 0 1 0 0 0</td>
<td>0 1 1 1 0 1</td>
</tr>
<tr>
<td>0 0 0 1 0 0</td>
<td>0 1 1 1 1 1</td>
</tr>
<tr>
<td>0 0 0 0 1 0</td>
<td>1 0 0 0 1 0</td>
</tr>
<tr>
<td>0 0 0 0 0 1</td>
<td>1 1 0 1 0 0</td>
</tr>
</tbody>
</table>

\[
\begin{bmatrix}
100000 \\
010000 \\
001000 \\
000100 \\
000010 \\
000001
\end{bmatrix}
\begin{bmatrix}
101110 \\
101000 \\
011101 \\
011111 \\
100010 \\
110100
\end{bmatrix}
\]

The following command reads the vector of coset representatives from the file just created.

```bash
coset_reps_read:
```

153
$(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 \
-define CR -vector_ge -action G2 \
  -read_csv \
  PGOp_6_2_coset_reps.csv Element \
-define -end
5.4 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 [71].

The electronic Atlas of finite simple groups [71] lists generators for $U_3(3)$ as $3 \times 3$ matrices over the field $\mathbb{F}_9$ using the following short Magma [14] program:

```magma
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 [66]):

```magma
F<w>:=GF(9);
print DefiningPolynomial(F);
```

It results in

```magma
$.1^2 + 2*$.1 + 2
```

which is the Magma way of printing the polynomial $X^2 + 2X + 2$. If $\alpha$ 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

```magma
G<x,y>:=MatrixGroup<3,F|x,y>;
```
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

dflatex \ PGL_3.9\_Subgroup\_U_3.3_6048\_report.tex

Group theoretic activities will be discussed in Section 5.6.

As an example of a large group, consider the Conway group Co3. Following [65], the group can be generated using two matrices of dimension 22 over $F_2$. 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.

CONWAY_GEN1="\n110111000100001010000\n111101011110100001011\n00000100000010001010\n11111011101000100110\n01010100000000101110\n00000100000100010101\n00100000000100010101\n00100001100000111111\n11101001011010001011\n00000000000110010101\n00000000010001010101\n01101111101001110111\n00000000000110010101\n00000000000100010101\n00000000000000000101\n0000000000100100010101\n01101111101001110111\n00000000000110010101\n00000000000100010101\n00000000000000000101\n0000000000100100010101\n00000000000000000101\n00000000000100010101\n00000000000000000101\n00000000000100010101\n00000000000000000101\n00000000000100010101\n00000000000000000101\n00000000000100010101\n00000000000000000101\n00000000000100010101\n00000000000000000101\n00000000000100010101\n00000000000000000101"
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 -concatenate 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 $\mathbb{F}_{2^7}$. Again, we use makefile variables to specify the two generators as $7 \times 7$ matrices over $\mathbb{F}_{2^7}$ and concatenate them, before passing them to the `-subgroup_by_generators` command.

```
Ree_gen1="21,5,1,6,17,1,1, 3,13,5,21,6,6,18, 21,3,21,21,22,6,14, \n14,18,1,5,13,6,7, 3,3,2,1,24,16,3, 17,3,22,10,16,24,26, \n21,21,6,18,20,2,5"

Ree_gen2="16,3,11,5,16,22,20, 24,6,18,24,7,1,26, 9,23,17,18,23,20,13, \n9,7,2,15,17,5,11, 3,3,6,21,4,24,16, 25,8,6,24,21,12,7, \n24,15,2,13,11,14,24"

Ree_27:
  ▶ $(ORBITER) -v 2 \n  ▶ ▶ -define F -finite_field -q 27 -override_polynomial "34" -end \n  ▶ ▶ -define g1 -vector -field F -format 7 -dense $(Ree_gen1) -end \n  ▶ ▶ -define g2 -vector -field F -format 7 -dense $(Ree_gen2) -end \n  ▶ ▶ -define gens -vector -concatenate g1 -concatenate g2 -end \n  ▶ ▶ -define G -linear_group -PGL 7 F \n  ▶ ▶ ▶ -subgroup_by_generators "Ree_27" "10073444472" 2 gens \n  ▶ ▶ ▶ -end \n  ▶ ▶ -with G -do \n  ▶ ▶ -group_theoretic_activity \n  ▶ ▶ ▶ -report \n  ▶ ▶ -end
```
### Table 5.4: Commands for creating modified groups

<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-restricted_action</code></td>
<td>$S$</td>
<td>restricted action on the set $S$</td>
</tr>
<tr>
<td><code>-on_k_subspaces</code></td>
<td>$k$</td>
<td>induced action on $k$ dimensional subspaces</td>
</tr>
<tr>
<td><code>-on_k_subsets</code></td>
<td>$k$</td>
<td>induced action on $k$ subsets.</td>
</tr>
<tr>
<td><code>-on_wedge_product</code></td>
<td></td>
<td>action on the exterior square</td>
</tr>
<tr>
<td><code>-create_special_subgroup</code></td>
<td></td>
<td>Create the subgroup of special matrices.</td>
</tr>
<tr>
<td><code>-point_stabilizer</code></td>
<td>$pt$</td>
<td>Create the subgroup which stabilizes $pt$.</td>
</tr>
<tr>
<td><code>-from</code></td>
<td><code>label</code></td>
<td>Name of the original group (this option can be repeated).</td>
</tr>
</tbody>
</table>

#### 5.5 Induced Actions

It is possible to create new group actions from old. This is done using the command `-modified_group`. Table 5.4 lists Orbiter commands to do so.

Let us create the action of $\text{Sym}(4)$ on pairs:

```latex
\text{Symmetric}_4\text{on_pairs:}
\begin{verbatim}
▷ $(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_on_2_subsets_report.tex
▷ open Sym_4_on_2_subsets_report.pdf
\end{verbatim}
```

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 invariant subset. The example we show is related to cubic surfaces. At first, we create the Eckardt surface in $\text{PG}(3, 13)$ 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:
SURFACE_Q13_STAB_GENS="1,0,0,0,9,12,0,0,10,0,12,0,9,0,0,12, \
1,0,0,0,12,12,6,6,0,0,7,1,0,2,0, \
0,1,1,7,3,9,9,11,2,10,10,3,9,9,1,11"

surface_q13.Eckardt_on_tritangent_planes:
  $(ORBITER) -v 2 \n  > -orbiter_path $(ORBITER_PATH) \n  > -draw_options -embedded -end \n  > -define F -finite_field -q 13 -end \n  > -define gens -vector -field F -dense $(SURFACE_Q13_STAB_GENS) -end \n  > -define TriP -set -file \n  >  > arc_lifting_trihedral_q13 Arc0_1_2_3_43_113_tritangent_planef.csv \n  >  > -end \n  > -define G -linear_group -PGL 4 F \n  >  > -subgroup_by_generators "stab" \n  >  >  > 24 3 gens \n  >  > -end \n  > -define G_on_planes -modified_group -from G \n  >  > -on_k_subspaces 3 \n  >  > -end \n  > -define Gr -modified_group -from G_on_planes \n  >  > -restricted_action TriP \n  >  > -end \n  > -with Gr -do \n  > -group_theoretic_activity \n  >  > -report \n  >  > -end \n  > -define Orb -orbits -group Gr \n  >  > -on_points \n  >  > -end \n  > -with Orb -do -orbits_activity \n  >  > -report \n  >  > -end \n  > -with Orb -do -orbits_activity \n  >  > -draw_tree 0 \n  >  > -end \n  > -with Orb -do -orbits_activity \n  >  > -draw_tree 1 \n  >  > -end \n  > -with Orb -do -orbits_activity \n  >  > -stabilizer 36 \n  >  > -end \n  > pdflatex PGL_4_13_Gr_4_3_res45_orbits_report.tex
  > open PGL_4_13_Gr_4_3_res45_orbits_report.pdf
Table 5.5: Commands for specific actions

<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-wedge_detached</td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>-PGL2OnConic</td>
<td></td>
<td>induced action of PGL(2, q) on the conic in the plane PG(2, q)</td>
</tr>
<tr>
<td>-subfield_</td>
<td>s</td>
<td>action by field reduction to the subfield of index s</td>
</tr>
<tr>
<td>structure_action</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-on_tensors</td>
<td></td>
<td>induced action of GL(d, q)≀Sym(n) on the tensor space</td>
</tr>
<tr>
<td>-on_rank_one_</td>
<td></td>
<td>induced action of GL(d, q)≀Sym(n) on the tensor space</td>
</tr>
<tr>
<td>tensors</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5.5 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

T3_on_tensors:

```
$ (ORBITER) -v 2 \
  -define G \n  -linear_group -GL_d_q_wr_Sym_n 2 2 3 \n  -on_tensors -end \n  -with G -do \n  -group_theoretic_activity \n  -report \n  -end
```

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 (computed in Section 4.4).
creates the group $GL(2, 2) \wr Sym(3)$ acting on the 255 elements of $PG(7, 2)$ 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** $GL(2, 2) \wr Sym(3)$

The order of the group $GL(2, 2) \wr Sym(3)$ is 1296

The group acts on a set of size 255

**The Action**

Group action $GL(2, 2) \wr Sym(3)_{res255}$ 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.

\[
\begin{align*}
&\left( \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} ; \id \right),
&\left( \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} ; \id \right),
&\left( \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} ; \id \right),
&\left( \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} ; \id \right),
&\left( \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} ; (1, 2) \right),
&\left( \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} ; (0, 1) \right)
\end{align*}
\]
### Stabilizer chain

<table>
<thead>
<tr>
<th>Level</th>
<th>Base pt</th>
<th>Orbit length</th>
<th>Subgroup order</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>3</td>
<td>1296</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>432</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>216</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>3</td>
<td>216</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>2</td>
<td>72</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>3</td>
<td>36</td>
</tr>
<tr>
<td>6</td>
<td>9</td>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td>7</td>
<td>12</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>13</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

It is also possible to restrict the action on all rank-one tensors, as the following example shows:

T3r1:

- $(\text{ORBITER}) -v 4 \$
- $\text{-define G} \$
- $\text{-linear_group -GL_d_q_wr_Sym_n 2 2 3} \$
- $\text{-on_rank_one_tensors -end} \$
- $\text{-with G -do} \$
- $\text{-group_theoretic_activity} \$
- $\text{-report} \$
- $\text{-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 $\text{GL}(2,2) \wr \text{Sym}(3)$

The order of the group $\text{GL}(2,2) \wr \text{Sym}(3)$ is 1296

The group acts on a set of size 27
The Action

Group action \( \text{GL}(2, 2) \wr \text{Sym}(3) \) of degree 27
We act on the following set:

\[
\begin{align*}
0 &= (1, 0, 0, 0, 0, 0, 0, 0) \\
1 &= (0, 1, 0, 0, 0, 0, 0, 0) \\
2 &= (1, 1, 0, 0, 0, 0, 0, 0) \\
3 &= (0, 0, 1, 0, 0, 0, 0, 0) \\
4 &= (0, 0, 0, 1, 0, 0, 0, 0) \\
5 &= (0, 0, 1, 1, 0, 0, 0, 0) \\
6 &= (1, 0, 1, 0, 0, 0, 0, 0) \\
7 &= (0, 1, 1, 0, 0, 0, 0, 0) \\
8 &= (1, 1, 1, 0, 0, 0, 0, 0) \\
9 &= (0, 0, 0, 0, 1, 0, 0, 0) \\
10 &= (0, 0, 0, 0, 0, 1, 0, 0) \\
11 &= (0, 0, 0, 0, 0, 1, 1, 0) \\
12 &= (0, 0, 0, 0, 0, 0, 1, 0) \\
13 &= (0, 0, 0, 0, 0, 0, 0, 1) \\
14 &= (0, 0, 0, 0, 0, 0, 1, 1) \\
15 &= (0, 0, 0, 0, 1, 0, 1, 0) \\
16 &= (0, 0, 0, 0, 0, 1, 0, 1) \\
17 &= (0, 0, 0, 0, 1, 1, 1, 1) \\
18 &= (1, 0, 0, 0, 1, 0, 0, 0) \\
19 &= (0, 1, 0, 0, 0, 1, 0, 0) \\
20 &= (1, 1, 0, 0, 1, 1, 0, 0) \\
21 &= (0, 0, 1, 0, 0, 0, 1, 0) \\
22 &= (0, 0, 0, 1, 0, 0, 0, 1) \\
23 &= (0, 0, 1, 1, 0, 0, 1, 1) \\
24 &= (1, 0, 1, 0, 1, 0, 1, 0) \\
25 &= (0, 1, 0, 1, 0, 1, 0, 1) \\
26 &= (1, 1, 1, 1, 1, 1, 1, 1)
\end{align*}
\]

Base and Stabilizer Chain

Group order 1296
\( t_1=3, 2, 1, 3, 2, 3, 2, 3, 2, \)
Strong generators for a group of order 1296:

\[
\begin{align*}
\left( \begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix} \right); & \quad \left( \begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix} \right), \\
\left( \begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix} \right); & \quad \left( \begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix} \right), \\
\left( \begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix} \right); & \quad \left( \begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix} \right), \\
\left( \begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix} \right); & \quad \left( \begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix} \right)
\end{align*}
\]
The group of the projective line acts on a conic. The command PGL2OnConic can be used to create this action. The group elements are still represented as $2 \times 2$ matrices. Here is an example. We create the collineation group PGL(2, 8) of PG(1, 8) and act on PG(2, 8):

PGGL_2_8_on_conic:

```
$($ORBITER) -v 4 \n▷ -define G -linear_group \n▷ ▷ -PGGL 2 8 -PGL2OnConic \n▷ ▷ -end \n▷ ▷ -with G -do \n▷ ▷ -group_theoretic_activity \n▷ ▷ ▷ -report \n▷ ▷ -end
```

#pdflatex PGGL_2_8_onConic_2_8_report.tex

#open PGGL_2_8_onConic_2_8_report.pdf

This produces the following report. The generators are elements of PGL(2, 8) acting on PG(2, 8). The first basic orbit is the conic itself.

**The Group PGL(2, 8)OnConic(2, 8)**

The order of the group PGL(2, 8)OnConic(2, 8) is 1512
The group acts on a set of size 73

Strong generators for a group of order 1512:

\[
\begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix}, \begin{bmatrix}
\gamma & 0 \\
0 & 1
\end{bmatrix}, \begin{bmatrix}
1 & 0 \\
1 & 1
\end{bmatrix}, \\
\begin{bmatrix}
1 & 0 \\
\gamma & 1
\end{bmatrix}, \begin{bmatrix}
1 & 0 \\
\gamma^2 & 1
\end{bmatrix}, \begin{bmatrix}
0 & 1 \\
1 & 0
\end{bmatrix},
\]

1,0,0,1,1,
1,0,0,6,0,
1,0,1,1,0,
1,0,2,1,0,
1,0,4,1,0,
0,1,1,0,0,
The Action

Group action PΓL(2, 8)OnConic of degree 73
We act on the following set:

0 = (1, 0, 0) 5 = (2, 1, 0)
1 = (0, 1, 0) 72 = (7, 7, 1)
2 = (0, 0, 1)
3 = (1, 1, 1)
4 = (1, 1, 0)

The group is a matrix group.
The base action is on projective space PG(1, 8)
q = 8
p = 2
e = 3
n = 1
Number of points = 9
Number of lines = 1
Number of lines on a point = 1
Number of points on a line = 9

The finite field \( \mathbb{F}_8 \)

polynomial: \( X^3 + X^2 + 1 = 13 \)
\( Z_i = \log_\alpha (1 + \alpha^i) \)

\[
\begin{array}{|c|c|c|c|c|c|c|c|}
\hline
i & \gamma_i & -\gamma_i & \gamma_i^{-1} & \log_\alpha(\gamma_i) & \alpha^i & Z_i & \phi(\gamma_i) \\
\hline
0 & 0 = 0 & 0 & DNE & DNE & 1 & DNE & 0 \\
1 & 1 = 1 & 1 & 1 & 7 & 2 & 5 & 1 \\
2 & \alpha = \gamma & 2 & 6 & 1 & 4 & 3 & 4 \\
3 & \alpha + 1 = \gamma^5 & 3 & 4 & 5 & 5 & 2 & 5 \\
4 & \alpha^2 = \gamma^2 & 4 & 3 & 2 & 7 & 6 & 7 \\
5 & \alpha^2 + 1 = \gamma^3 & 5 & 7 & 3 & 3 & 1 & 6 \\
6 & \alpha^2 + \alpha = \gamma^6 & 6 & 2 & 6 & 6 & 4 & 3 \\
7 & \alpha^2 + \alpha + 1 = \gamma^4 & 7 & 5 & 4 & 1 & DNE & 2 \\
\hline
\end{array}
\]
<table>
<thead>
<tr>
<th>+</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>6</td>
<td>7</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>4</td>
<td>7</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>7</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>-</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>5</td>
<td>7</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>6</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>5</td>
<td>1</td>
<td>7</td>
<td>3</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>7</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>1</td>
<td>7</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>1</td>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>

167

$2^0 = 1 
2^1 = 2 
2^2 = 4 
2^3 = 5 
2^4 = 7 
2^5 = 3 
2^6 = 6 
2^7 = 1$
Basic Orbit 0

Basic orbit 0 has size 9
0, 1, 2, 3, 4, 5, 6, 7, 8

Basic Orbit 1

Basic orbit 1 has size 8
1, 2, 3, 4, 5, 6, 7, 8
Basic Orbit 2

Basic orbit 2 has size 7
2, 3, 4, 5, 6, 7, 8

Basic Orbit 3

Basic orbit 3 has size 3
4, 6, 7
5.6 Group Theoretic Activities

Once a group has been created as in Section 5.2, a group theoretic activity can be performed. For this purpose, Orbiter provides the \texttt{-group_theoretic_activity} option. Tables 5.6 and 5.7 list the possible commands that can come after it.

The command

\begin{verbatim}
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
\end{verbatim}

creates all elements of PGL(3, 2) and writes them into the file \texttt{PGL_3_2_elements.csv}.

The command

\begin{verbatim}
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
\end{verbatim}

creates a tree of the elements of Sym(3) (see Fig 5.4). The leaves are ordered lexicographically.

It is possible to compute all powers of a fixed element, as in the following command:
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-apply</td>
<td>a s</td>
<td>Applies the group element given by the coded vector s to the element a.</td>
</tr>
<tr>
<td>-multiply</td>
<td>s₁ s₂</td>
<td>Multiplies group elements s₁ and s₂, assuming the elements are given in coded form. Produces a latex report.</td>
</tr>
<tr>
<td>-inverse</td>
<td>s</td>
<td>Computes the inverse of s, which is given in coded form. Produces a latex report.</td>
</tr>
<tr>
<td>-consecutive_powers</td>
<td>s n</td>
<td>Computes all powers sⁱ for i = 1, . . . , n. s is given in coded form. Produces a latex report.</td>
</tr>
<tr>
<td>-raise_to_the_power</td>
<td>s n</td>
<td>Computes the n-th power of of s, which is given in coded form. Produces a latex report.</td>
</tr>
<tr>
<td>-export_orbiter</td>
<td></td>
<td>Exports the group to Orbiter.</td>
</tr>
<tr>
<td>-export_gap</td>
<td></td>
<td>Exports the group to GAP [29].</td>
</tr>
<tr>
<td>-export_magma</td>
<td></td>
<td>Exports the group to Magma [14].</td>
</tr>
<tr>
<td>-canonical_image</td>
<td>S</td>
<td>Compute the canonical image of the set S using GAP.</td>
</tr>
<tr>
<td>-search_element_of_order</td>
<td>i</td>
<td>Finds all elements of order i in the group (i ∈ N).</td>
</tr>
<tr>
<td>-find_standard_generators</td>
<td>a b c</td>
<td>Finds all pairs of elements x, y in G such that</td>
</tr>
<tr>
<td>-element_rank</td>
<td>s</td>
<td>Determines the rank of the group element s in the given group. s is given in coded form.</td>
</tr>
<tr>
<td>-element_unrank</td>
<td>r</td>
<td>Produces the group element whose rank is r.</td>
</tr>
<tr>
<td>-find_singer_cycle</td>
<td></td>
<td>Finds all Singer cycles whose matrix is a companion matrix.</td>
</tr>
<tr>
<td>-classes_based_on_normal_form</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-normalizer</td>
<td></td>
<td>See Section 5.7.</td>
</tr>
<tr>
<td>-centralizer_of_element</td>
<td></td>
<td>See Section 5.7.</td>
</tr>
</tbody>
</table>

Table 5.6: Group theoretic activities (Part 1)
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-permutation_representation_of_element</code></td>
<td>$g$</td>
<td>Compute the permutation representation of $g$</td>
</tr>
<tr>
<td><code>-conjugacy_class_of_element</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>-orbits_on_group_elements_under_conjugation</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>-normalizer_of_cyclic_subgroup</code></td>
<td></td>
<td>See Section 5.7.</td>
</tr>
<tr>
<td><code>-classes</code></td>
<td></td>
<td>See Section 5.7.</td>
</tr>
<tr>
<td><code>-report</code></td>
<td></td>
<td>Produce a latex report about the group.</td>
</tr>
<tr>
<td><code>-report_sylow</code></td>
<td></td>
<td>Include Sylow subgroups in the report (requires <code>-report</code>).</td>
</tr>
<tr>
<td><code>-report_group_table</code></td>
<td></td>
<td>Include the group table in the report (requires <code>-report</code>).</td>
</tr>
<tr>
<td><code>-report_classes</code></td>
<td></td>
<td>Include the conjugacy classes in the report (requires <code>-report</code>).</td>
</tr>
<tr>
<td><code>-export_group_table</code></td>
<td></td>
<td>Exports the group table as csv-file.</td>
</tr>
<tr>
<td><code>-test_if_geometric</code></td>
<td></td>
<td>Test is the action is geometric.</td>
</tr>
<tr>
<td><code>-conjugacy_class_of</code></td>
<td>$g$</td>
<td>Compute conjugacy class of $g$.</td>
</tr>
<tr>
<td><code>-isomorphism_Klein_quadric</code></td>
<td>fname</td>
<td></td>
</tr>
<tr>
<td><code>-print_elements</code></td>
<td></td>
<td>Produces a printout of all group elements.</td>
</tr>
<tr>
<td><code>-print_elements_tex</code></td>
<td></td>
<td>Produces a latex report of all group elements.</td>
</tr>
<tr>
<td><code>-save_elements_csv</code></td>
<td>fname</td>
<td>Writes the elements of the group to the given csv file.</td>
</tr>
<tr>
<td><code>-export_inversion_graphs</code></td>
<td>fname</td>
<td>Exports the inversion graphs associated to the group elements to the given file.</td>
</tr>
</tbody>
</table>

Table 5.7: Group theoretic activities (Part 2)
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-multiply_elements_csv_column_major_ordering</td>
<td>f1 f2 f3</td>
<td></td>
</tr>
<tr>
<td>-multiply_elements_csv_row_major_ordering</td>
<td>f1 f2 f3</td>
<td></td>
</tr>
<tr>
<td>-apply_elements_csv_to_set</td>
<td>f1 f2 S</td>
<td></td>
</tr>
<tr>
<td>-order_of_products</td>
<td>$g_1 \ldots g_n$</td>
<td>Creates a table of the orders of all products $g_i g_j$, $1 \leq i, j \leq n$.</td>
</tr>
<tr>
<td>-reverse_isomorphism_exterior_square</td>
<td></td>
<td>Given a set of generators of a subgroup of $\text{PGO}^+(6, q)$ as $6 \times 6$ matrixes, compute the inverse image of the generators in $\text{PGL}(4, q)$ (if possible).</td>
</tr>
<tr>
<td>-is_subgroup_of</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-coset_reps</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-orbit_of</td>
<td></td>
<td>pt</td>
</tr>
<tr>
<td>-orbits_on_set_system_from_file</td>
<td>f c n</td>
<td></td>
</tr>
<tr>
<td>-orbit_of_set_from_file</td>
<td></td>
<td>fname</td>
</tr>
<tr>
<td>-linear_codes</td>
<td>control $d$ $n_{\text{max}}$</td>
<td>Classify linear codes with prescribed minimum distance $d$. Assumes that the group is $\text{PGL}(r, q)$ or $\text{PΓL}(r, q)$. For each $n \leq n_{\text{max}}$, the $[n, k, \geq d]$ codes are classified with $n - k = r$. See Section 10.</td>
</tr>
<tr>
<td>-tensor_permutations</td>
<td></td>
<td>Computes the permutation representation of generators of the wreath product.</td>
</tr>
<tr>
<td>-classify_ovoids</td>
<td>descr</td>
<td></td>
</tr>
<tr>
<td>-classify_cubic_curves</td>
<td>descr</td>
<td>Classifies cubic curves. Expects an arc description options as in Table 6.4.</td>
</tr>
<tr>
<td>-representation_on_polynomials</td>
<td>$d$</td>
<td>Compute the representation in the action on polynomials of degree $d$.</td>
</tr>
</tbody>
</table>

Table 5.8: Group theoretic activities (Part 3)
We create the 12 powers of the cycle

\[(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11).\]
The command

```
PGL_3_4_singer:
▷ $(ORBITER) -v 5 \n▷ ▷ -define G -linear_group -PGL 3 4 -end \n▷ ▷ -with G -do \n▷ ▷ -group_theoretic_activity \n▷ ▷ ▷ -find_singer_cycle \n▷ ▷ -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 \( \omega \) 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:

```
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:

```
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
We can compute the inverse of a group element:

\[
\begin{bmatrix}
0 & 1 \\
2 & 3
\end{bmatrix}
\begin{bmatrix}
4 & 5 \\
6 & 0
\end{bmatrix} =
\begin{bmatrix}
6 & 0 \\
5 & 3
\end{bmatrix}
\]

0,1,2,3,
4,5,6,0,
6,0,5,3,

We can raise a group element to a power:

\[
\begin{bmatrix}
0 & 1 \\
2 & 3
\end{bmatrix}^{-1} =
\begin{bmatrix}
2 & 4 \\
1 & 0
\end{bmatrix}
\]

0,1,2,3,
2,4,1,0,
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.

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
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-classes</td>
<td></td>
<td>Compute a report of the conjugacy classes of elements.</td>
</tr>
<tr>
<td>-centralizer_of_element</td>
<td>label coding</td>
<td>Compute the centralizer of the coded group element, using label to create file names.</td>
</tr>
<tr>
<td>-normalizer_of_cyclic_subgroup</td>
<td>label s</td>
<td>Compute the normalizer of the cyclic subgroup generated by the element s.</td>
</tr>
<tr>
<td>-normalizer</td>
<td></td>
<td>Compute the normalizer of a subgroup in the larger group.</td>
</tr>
</tbody>
</table>

Table 5.9: Group theoretic activities based on Magma

5.7 Group Theoretic Activities Based on Magma

Through its interface to Magma [14], Orbiter can perform group theoretic computations. Table 5.9 list the group theoretic commands that rely on Magma. The communication to and from magma happens through files. This is a three step process: An Orbiter session receives a command to compute the conjugacy classes of a group. The Orbiter session writes a magma file. This file is read and executed by Magma. Magma writes a second file containing the conjugacy classes in coded form. Another Orbiter session reads the magma output file, decodes the information and produces the desired list of conjugacy classes. A latex report is written containing the classes, as well as related information regarding centralizers and normalizers.

For instance, the three-step command sequence

```
PGGL_2_4_classes:
  ▶ $(ORBITER) -v 3 \n  ▶ ▶ -define G \n  ▶ ▶ -linear_group -PGGL 2 4 \n  ▶ ▶ -end \n  ▶ ▶ -with G -do \n  ▶ ▶ -group_theoretic_activity \n  ▶ ▶ ▶ -classes \n  ▶ ▶ -end
  ▶ $(MAGMA_PATH)magma PGGL_2_4_classes.magma
  ▶ $(ORBITER) -v 3 \n  ▶ ▶ -define G \n  ▶ ▶ -linear_group -PGGL 2 4 \n  ▶ ▶ -end \n  ▶ ▶ -with G -do \n  ▶ ▶ -group_theoretic_activity \n  ▶ ▶ ▶ -classes \n  ▶ ▶ -end
```

180
computes the classes of elements in $\text{PGL}(2,4)$ using Orbiter-Magma-Orbiter. The first Orbiter command produces the file \texttt{PGGL\_2\_4\_classes.magma}. The magma command reads this file and produces the file \texttt{PGGL\_2\_4\_classes\_out.txt}. The second Orbiter command reads the file \texttt{PGGL\_2\_4\_classes\_out.txt} and produces the latex report \texttt{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 
\[ c_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \] 
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: 
\[ \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} \omega^2 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \] 
1,0,0,1,1,  
1,0,0,2,1,  
0,1,1,0,1, 

The command sequence

\texttt{PGGL\_2\_4\_cent\_2A:}  
\texttt{pdflatex PGGL\_2\_4\_classes\_out.tex}  
\texttt{open PGGL\_2\_4\_classes\_out.pdf}  
\texttt{open PGGL\_2\_4\_classes\_out.csv}
computes the centralizer of the Baer involution

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

The centralizer is a group of order 40320, isomorphic to \( \text{PGL}(4,2).Z_2 \). 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}_0
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
1 & 1 & 0 & 1
\end{bmatrix}_1
\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 & 0 & 1 \\
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 & 0 \\
1 & 1 & 1 & 1 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 1
\end{bmatrix}_0
\begin{bmatrix}
0 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 \\
1 & 0 & 1 & 0 \\
0 & 1 & 1 & 1
\end{bmatrix}_0
\begin{bmatrix}
0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}_0
\begin{bmatrix}
0 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 0 \\
0 & 0 & 1 & 1
\end{bmatrix}_0
\begin{bmatrix}
0 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 \\
0 & 1 & 1 & 1
\end{bmatrix}_0
\begin{bmatrix}
0 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 \\
0 & 1 & 1 & 1
\end{bmatrix}_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.

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 create 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:

```
GENERATORS_H5="1,2,3,4,0,6,7,8,9,5,10,11,12"
```

The command

```
Normalizer_of_H5:
  $(ORBITER) -v 2 \
  -define G -permutation_group -symmetric_group 13 \
  -subgroup_by_generators H5 5 1 \
  -with G -do \n  -group_theoretic_activity \n  -normalize \n  -end
  pdflatex Perm13_Subgroup_H5_5_normalizer.tex
  open Perm13_Subgroup_H5_5_normalizer.pdf
```

computes the normalizer of $H$ in $\text{Sym}(13)$. 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 Perm13SubgroupH5order5 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

\[H = \langle \begin{bmatrix} \alpha^4 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \rangle \cong C_2 \times C_2\]

is a subgroup of \(G = \text{PGL}(2, 9)\). To compute the normalizer of \(H\) in \(G\), the following command sequence can be used:

\[
\text{Normalizer_of}_Z22\text{in}_PGL\_2\_9:
\]
\[
\text{$(ORBITER) -v 2 \ \$
\]
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) Subgroup Z22 order 4 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,
Chapter 6

Orbit Algorithms

6.1 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 [17, 36, 63]. Posets based algorithms will be discussed in Section 6.2. Orderly generation using canonical forms is discussed in Section 15.2.

Table 6.1 lists the Orbiter commands that can be used to compute orbits. Table 6.2 lists activities for an object of type orbits.

Consider the group $\text{PGL}(4,2)$ in the natural action on the set of points of $\text{PG}(3,2)$. The degree of the action is 15. The action is transitive. The following example computes the Schreier tree for the action:

```
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
```
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-group</td>
<td>G</td>
<td>Specify the group that acts.</td>
</tr>
<tr>
<td>-on_points</td>
<td></td>
<td>Consider the (default) action that comes with $G$. This option will invoke the Schreier tree algorithm.</td>
</tr>
<tr>
<td>-on_subsets</td>
<td>$k$</td>
<td>Consider the induced action on subsets of size $k$. This option will invoke the poset algorithm.</td>
</tr>
<tr>
<td>-on_subspaces</td>
<td>$k$</td>
<td>Consider the induced action on subspaces of size $k$. We assume that the group action is linear. This option will invoke the poset algorithm.</td>
</tr>
<tr>
<td>-on_tensors</td>
<td>$d$</td>
<td>Consider the orbits on the $d$-fold tensor product.</td>
</tr>
<tr>
<td>-on_partition</td>
<td>$k$</td>
<td>Consider the action on partitions of type $k + k + k$. In this case, $3k$ must equal the degree of the defining action of $G$.</td>
</tr>
<tr>
<td>on_polynomials</td>
<td>$d$</td>
<td>Consider the action of $G$ on polynomials of degree $d$ in $n$ variables. Here, the action must be linear on an $n$-dimensional space.</td>
</tr>
<tr>
<td>-group</td>
<td>G</td>
<td>Specify the group that acts.</td>
</tr>
</tbody>
</table>

Table 6.1: Commands to compute orbits
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-report</code></td>
<td></td>
<td>Create a report about the orbits.</td>
</tr>
<tr>
<td><code>-export_something</code></td>
<td>type data</td>
<td>Export orbit data of the given type.</td>
</tr>
<tr>
<td><code>-export_trees</code></td>
<td></td>
<td>Create drawings of the Schreier trees.</td>
</tr>
<tr>
<td><code>-draw_tree</code></td>
<td>i</td>
<td>Create a drawing of the $i$-th orbit as a tree.</td>
</tr>
<tr>
<td><code>-stabilizer</code></td>
<td>i</td>
<td>Compute the stabilizer of the given point $i$.</td>
</tr>
<tr>
<td><code>-stabilizer_of_orbit_rep</code></td>
<td>i</td>
<td>Compute the stabilizer of the orbit representative of orbit $i$.</td>
</tr>
<tr>
<td><code>-Kramer_Mesner_matrix</code></td>
<td>t k</td>
<td>Compute the Kramer Mesner matrix of orbits on $t$-subsets and $k$-subsets. This requires a poset type problem.</td>
</tr>
<tr>
<td><code>-recognize</code></td>
<td>S</td>
<td>Recognize $S$ in the orbit data structure.</td>
</tr>
<tr>
<td><code>-report_options</code></td>
<td>descr</td>
<td>Set additional parameters for the report.</td>
</tr>
</tbody>
</table>

Table 6.2: Activities related to an object of type orbits

To draw the tree, the following command is used:

```
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 in Figure 6.1.

Consider the wreath product acting on rank-one tensors from Section 5.5. The following command sequence computes the orbits, exports the Schreier tree, and produces the drawing shown in Figure 6.2.

```
T3r1.orbits:
  ▶ $(ORBITER) -v 4 \n  ▶ ▶ -draw_options -embedded -end \n  ▶ ▶ -define G \n  ▶ ▶ -linear_group -GL_d_q_wr_Sym_n 2 2 3 \n  ▶ ▶ ▶ -on_rank_one_tensors -end \n  ▶ ▶ -define Orb -orbits -group G \n  ▶ ▶ ▶ -on_points \n  ▶ ▶ -end \n  ▶ ▶ -with Orb -do -orbits_activity \n  ▶ ▶ ▶ -report \n  ▶ ▶ -end \n  ▶ ▶ -with Orb -do -orbits_activity \n  ▶ ▶ ▶ -draw_tree 0 \n  ▶ ▶ -end
```

In the next example, we compute the orbits of the linear group PGL(4, 2) on homogeneous
Figure 6.2: The Schreier tree for the action on rank-one tensors

polynomials of degree 3 in 4 variables:

\begin{verbatim}
orbits_cubic_curves_q2:
  $(ORBITER) -v 4 \ 
  -define G -linear_group -PGL 3 2 -end \ 
  -define Orb -orbits -group G \ 
  -on_polynomials 3 \ 
  -end
  #pdflatex poly_orbits_d3_n3_q2.tex
  #open poly_orbits_d3_n3_q2.pdf
\end{verbatim}

This command computes the orbits of on all cubic forms in 4 variables, confirming the work of Dickson [24] and an enumerative result of Cooley [20].

The next example computes orbits in an induced action. Induced actions have been described in Section 5.5. One group can have many actions. In particular, Orbiter can work with induced actions without changing the representation of the group elements. This has the advantage that the stabilizers are expressed in terms of the original action. To consider an example, suppose we want to consider the action of the stabilizer of a conic on the points of the plane (this continues an example from Section 5.5). The following command can be used:

\begin{verbatim}
PGGL_2_8_on_conic_orbits:
  $(ORBITER) -v 4 \ 
  -define G \ 
  -linear_group -PGGL 2 8 -PGL2OnConic -end \ 
\end{verbatim}

191
The output shown below. First, the orbits are listed. Then for each orbit, the stabilizer is shown, together with the generators in the action on the plane. For the sake of space, some of the output has been shortened. The three orbits correspond to the conic, the nucleus and the remaining points of the plane.

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:

\[
\begin{pmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
\delta^4 & \delta^9 & 1
\end{pmatrix}
\begin{pmatrix}
1 & \delta^7 & \delta^{13} \\
\delta^8 & \delta^9 & \delta^{10} \\
\delta & \delta^6 & 1
\end{pmatrix}
\begin{pmatrix}
\delta^5 & \delta^5 & \delta^5 \\
\delta^5 & \delta^2 & \delta^{11} \\
\delta^5 & \delta^{14} & 1
\end{pmatrix}
\]

1,0,0,0,1,0,9,5,1,1,
1,7,6,14,5,10,2,15,1,3,
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\}{\text{16320}}

\begin{align*}
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)
\end{align*}

There are 0 extensions

Strong generators for a group of order 16320:

\begin{align*}
\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^{12} & 1 & \delta^2 \\
\delta^4 & \delta^3 & \delta^7 \\
\delta^6 & \delta^3 & 1
\end{bmatrix}_2, \\
\begin{bmatrix}
\delta^{15} & \delta^3 & \delta^6 \\
\delta^{11} & \delta^6 & \delta^{10} \\
\delta^{10} & \delta^6 & 1
\end{bmatrix}_3
\end{align*}

1,0,0,0,3,0,0,0,5,2, \\
1,0,0,0,6,0,0,0,15,1, \\
1,0,0,5,0,4,11,6,3, \\
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
6.2 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. Transitivity is implied. For instance, Figure 6.3 shows the power set lattice of a four-element subset.

Posets often come with group actions. We say that a group $G$ acts on a poset $\mathcal{P}$ if for all $x, y \in \mathcal{P}$ and all $g \in G$,

$$x \leq y \Rightarrow xg \leq yg.$$  

For background on poset actions, see Plesken [58]. The orbits of $G$ on $\mathcal{P}$ form another poset, the poset of orbits. The problem of classification of combinatorial objects can often be attacked by using group invariant relations. A layered poset can be decomposed into a series of relations. The layers allow to reduce the classification problem into small steps, namely from on layer to the next. This uses the incidence relation between adjacent layers. By iterating this method, one can form a poset of substructures, and the classification problem reduces to the problem of determining the orbits of the poset, which we will henceforth call the poset classification problem. Many classification problem in Combinatorics reduce to poset classification problems.

Orbiter uses the algorithm of Schmalz [62] to perform poset classification. Two versions are available: one for subset-type posets and one for subspace-type posets. Figure 6.4 shows the subspace lattice of $V(3, 2) = \mathbb{F}_2^3$. The basis elements are listed, using the enumerator for elements on the projective geometry $\text{PG}(2, 2)$ explained in Section 4.1.
The commands shown in Table 6.3 can be used to control the poset classification algorithm. By default, Orbiter will choose the lexicographically 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.

Suppose that orbiter chooses element $a$ at node $n$. Suppose we are interested in choosing element $b$ instead. The command

$$-\text{preferred\_choice}\ n\ a\ b$$

can be used to force Orbiter to choose $b$ instead of $a$ at node $n$. 

Figure 6.4: Subspace lattice of $V(3, 2)$
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-problem_label</td>
<td>str</td>
<td>Use str as a prefix for files that are created.</td>
</tr>
<tr>
<td>-path</td>
<td>p</td>
<td>Use path p for files that are created.</td>
</tr>
<tr>
<td>-depth</td>
<td>d</td>
<td>Set search depth to d.</td>
</tr>
<tr>
<td>-v</td>
<td>v</td>
<td>Set verbosity to v. Larger numbers mean more output.</td>
</tr>
<tr>
<td>-gv</td>
<td>v</td>
<td>Set verbosity for group theoretic operations to v. Larger numbers mean more output.</td>
</tr>
<tr>
<td>-recover</td>
<td>fname</td>
<td>Recover from the given file.</td>
</tr>
<tr>
<td>-lex</td>
<td></td>
<td>Use the lexicographic ordering to speed up the search.</td>
</tr>
<tr>
<td>-w</td>
<td></td>
<td>Save orbits at level d only.</td>
</tr>
<tr>
<td>-W</td>
<td></td>
<td>Save orbits at all levels.</td>
</tr>
<tr>
<td>-write_data_files</td>
<td></td>
<td>Save data to files.</td>
</tr>
<tr>
<td>-t</td>
<td></td>
<td>Write a file containing the search tree at level d.</td>
</tr>
<tr>
<td>-T</td>
<td></td>
<td>Write a file containing the search tree at all levels.</td>
</tr>
<tr>
<td>-draw_options</td>
<td>options</td>
<td>Drawing options according to Table 16.2.</td>
</tr>
<tr>
<td>-preferred_choice</td>
<td>n a b</td>
<td>At node n, choose b instead of a as orbit representative. This option can be repeated.</td>
</tr>
<tr>
<td>-clique_test</td>
<td>graph</td>
<td>Classify cliques in the given graph.</td>
</tr>
</tbody>
</table>

Table 6.3: Options to control the poset classification algorithm
6.3 Orbits on Subsets

The lattice of subsets of a set $X$ is $\mathcal{P}(X)$, 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 $\mathcal{P}(X)$ but rather in a subposet of it. We require that the subposet is closed under the group action and that the following property holds:

$$x, y \in \mathcal{P}(X) \text{ and } x \leq y \Rightarrow \left(y \in \mathcal{P} \rightarrow x \in \mathcal{P}\right).$$

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 $\text{PG}(3,2)$. The following command computes the orbits of the group $G$ generated by a Singer cycle in $\text{PG}(3,2)$:

```
PGL_3_2_singer:
  > $(ORBITER) -v 3 \n  > -orbiter_path $(ORBITER_PATH) \n  > -define Control -poset_classification_control \n  > -define -problem_label PGL_3_2_singer.1 \n  > -W -depth 7 \n  > -draw_options \n  > -radius 200 -embedded \n  > -end \n  > -end \n  > -define G -linear_group -PGL 3 2 -singer 1 -end \n  > -define Orb -orbits -group G \n  > -on_subsets 7 Control \n  > -end \n  > -with Orb -do -orbits_activity \n  > -report \n  > -report_options -draw_poset -end \n  > -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 $\text{PGL}(4,2)$ acting on all subsets of $\text{PG}(3,2)$:

```
PG_3_2_subsets:
  > $(ORBITER) -v 3 \n  > -orbiter_path $(ORBITER_PATH) \n  > -define Control -poset_classification_control \n```
A drawing of the poset of orbits as in Figure 6.5 is produced.

Orbiter can compute orbits of groups acting in various different actions. The following example computes the orbits of $\text{PGL}(3,2)$ on the subsets of lines of $\text{PG}(2,2)$.

**PGL\_3\_2\_on\_lines:**

```bash
$(ORBITER) -v 3 \
  -oriters_path $(ORBITER\_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 -end \
  -end
```

`pdflatex PGL_3_2_lines_poset.tex`

`open PGL_3_2_lines_poset.pdf`
Figure 6.5: The orbits of $\text{PGL}(4, 2)$ on subsets
The following example computes the orbits of PGO(5, 2) on the power set lattice of points of $Q(4, 2)$:

```
P5_2_on_subsets:
  \$(ORBITER) -v 3 \n  \$-orbiter_path \$(ORBITER_PATH) \n  \$-define Control -poset_classification_control \n  \$-define -problem_label PGO_5_2 \n  \$-define -depth 15 \n  \$-define -w \n  \$-end \n  \$-define F -finite_field -q 2 -end \n  \$-define G -linear_group -PGO 5 F -end \n  \$-define Orb -orbits -group G \n  \$-on_subsets 15 Control \n  \$-end \n  \$-with Orb -do -orbits_activity \n  \$-report \n  \$-report_options -draw_poset -end \n  \$-end
```

\pifx{PGO_5_2_poset.tex}
\open{PGO_5_2_poset.pdf}

The poset of orbits is shown in Figure 6.6.
Figure 6.6: Orbits of PGO(5, 2) on the poset of subsets of $Q(4, 2)$
6.4 Orbits on Subspaces

Orbiter can compute the orbits of a group on the lattice of subspaces of a finite vector space.

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 finite version of the quadric given by the equation

$$x_0x_1 + x_2x_3 = 0,$$

and depicted over the real numbers in Figure 6.7. PG(3, 2) has 15 points:

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

The $Q^+(3, 2)$ quadric given by the equation above consists of the nine points

$$P_0, P_1, P_2, P_3, P_4, P_5, P_6, P_9, P_{10}.$$

The quadric is stabilized by the group $\text{PGO}^+(4, 2)$ of order 72. The command

```
subspaces.Op.4.2:
▷ $\text{ORBITER\ -v\ 5}$ \ \$
▷ ▷ $\text{-orbiter_path ($\text{ORBITER\_PATH}$)}$
▷ ▷ $\text{-define Control -poset\_classification\_control}$
▷ ▷ ▷ $\text{-draw\_options -radius\ 200}$ -end$
```
Figure 6.8: Hasse-diagram for the orbits of the orthogonal group \( \text{PGO}^+(4, 2) \) on subspaces of \( \text{PG}(3, 2) \)

\[
\begin{align*}
\text{\texttt{\( \triangleright \text{-problem\_label Op.4.2 -W -depth 4 \)}} \\
\text{\texttt{\( \triangleright \text{-end \)}}} \\
\text{\texttt{\( \triangleright \text{-define G -linear\_group -PGL 4 2 -orthogonal 1 -end \)}}} \\
\text{\texttt{\( \triangleright \text{-define Orb -orbits -group G \)}}} \\
\text{\texttt{\( \triangleright \text{-on\_subspaces 4 Control \)}}} \\
\text{\texttt{\( \triangleright \text{-end \)}}} \\
\text{\texttt{\( \triangleright \text{-with Orb -do -orbits\_activity \)}}} \\
\text{\texttt{\( \triangleright \text{-report \)}}} \\
\text{\texttt{\( \triangleright \text{-report\_options -draw\_poset -end \)}}} \\
\text{\texttt{\( \triangleright \text{-end \)}}} \\
\text{\texttt{\( \texttt{pdflatex PGL.4.2_Orthogonal\_plus.4.2\_poset.tex} \)}} \\
\text{\texttt{\( \texttt{open PGL.4.2_Orthogonal\_plus.4.2\_poset.pdf} \)}}
\end{align*}
\]

produces a classification of all subspaces of \( \text{PG}(3, 2) \) under \( \text{PGO}^+(4, 2) \). The option \texttt{-draw\_poset} creates a Hasse diagram of the classification as shown in Figure 6.8. The nodes show the ranks of points in \( \text{PG}(3, 2) \) as described in Section 4.1.
6.5 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

```c
C6_on_partition:
  ▷ $(ORBITER) -v 5 \n  ▷ ▷ -orbiter_path $(ORBITER_PATH) \n  ▷ ▷ -define Control -poset_classification_control \n  ▷ ▷ ▷ -problem_label C6 \n  ▷ ▷ ▷ -depth 2 \n  ▷ ▷ ▷ -W \n  ▷ ▷ ▷ -draw_options \n  ▷ ▷ ▷ ▷ -radius 200 -embedded \n  ▷ ▷ ▷ ▷ -end \n  ▷ ▷ ▷ -end \n  ▷ ▷ ▷ -define G -permutation_group -cyclic_group 6 -end \n  ▷ ▷ ▷ -define Orb -orbits -group G \n  ▷ ▷ ▷ ▷ -on_partition 2 Control \n  ▷ ▷ ▷ ▷ -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

$$6 \left( \frac{1}{3} + \frac{1}{1} + \frac{1}{2} + \frac{1}{2} + \frac{1}{6} \right) = 15.$$ 

The command

```c
PGL_2_17_on_partition:
  ▷ $(ORBITER) -v 5 \n  ▷ ▷ -define Control -poset_classification_control \n  ▷ ▷ ▷ -problem_label PGL_2_17 \n  ▷ ▷ ▷ -depth 6 \n  ▷ ▷ ▷ -W \n  ▷ ▷ ▷ -end \n  ▷ ▷ ▷ -define G -linear_group -PGL 2 17 -end \n  ▷ ▷ ▷ -define Orb -orbits -group G \n  ▷ ▷ ▷ ▷ -on_partition 6 Control \n  ▷ ▷ ▷ ▷ -end
```

205
computes the orbits of the group $\text{PGL}(2, 17)$ 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
\[
4896 \left( \frac{480}{1} + \frac{184}{2} + \frac{11}{3} + \frac{20}{4} + \frac{15}{6} + \frac{1}{8} + \frac{6}{12} + \frac{1}{18} + \frac{1}{24} + \frac{1}{36} + \right) = 2858856.
\]
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-control</td>
<td>label</td>
<td>Set the label of the poset classification control object.</td>
</tr>
<tr>
<td>-d</td>
<td>d</td>
<td>Require that no more than d points lie on a line.</td>
</tr>
<tr>
<td>-target_size</td>
<td>t</td>
<td>Specify the size of the arc to be t.</td>
</tr>
<tr>
<td>-conic_test</td>
<td></td>
<td>Require that no 6 points of the arc lie on a conic.</td>
</tr>
<tr>
<td>-test_nb_Eckardt_points</td>
<td>n</td>
<td>Require exactly n Eckardt points.</td>
</tr>
<tr>
<td>-affine</td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>-no_arc_testing</td>
<td></td>
<td>Do not test the at most d points per line condition.</td>
</tr>
<tr>
<td>-has_forbidden_point_set</td>
<td>set</td>
<td>The arc must not contain any of the given points.</td>
</tr>
<tr>
<td>-override_group</td>
<td>label</td>
<td>Override the group under which we classify.</td>
</tr>
</tbody>
</table>

Table 6.4: Commands for Classifying Arcs

### 6.6 Arcs and Caps in Projective Spaces

In 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 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 PG(2, 16) (cf. [48]) and the Pellegrino cap in AG(4, 3). The uniqueness of this cap was proven by Hill [31].

A $(k,d)$-arc in a projective plane $\pi$ is a set $S$ of $k$ points such that every 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 $\Phi$ 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 6.4 list the commands available to classify arcs. Here is an example. A hyperoval in a plane 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 PGL(3, 2$^e$). The command

```
hyperoval_16_classify:
```
\texttt{$(ORBITER) \ -v \ 4 \ \$
\texttt{\indent -oriter_path $(ORBITER\_PATH) \$
\texttt{\indent -define F \ -finite\_field \ -q \ 16 \ -end \$
\texttt{\indent -define P \ -projective\_space \ -n \ 2 \ -field F \ -v \ 0 \ -end \$
\texttt{\indent -define Control \ -poset\_classification\_control \$
\texttt{\indent \indent -problem\_label \ hyperoval\_q16 \$
\texttt{\indent \indent -W \ -depth \ 18 \$
\texttt{\indent \indent -draw\_options \$
\texttt{\indent \indent \indent -radius \ 200 \$
\texttt{\indent \indent -end \$
\texttt{\indent -end \$
\texttt{\indent -with P \ -do \$
\texttt{\indent \indent -projective\_space\_activity \$
\texttt{\indent \indent \indent -classify\_arcs \$
\texttt{\indent \indent \indent \indent -control Control \$
\texttt{\indent \indent \indent \indent \indent -target\_size \ 18 \$
\texttt{\indent \indent \indent \indent \indent \indent -d \ 2 \$
\texttt{\indent \indent \indent \indent -end \$
\texttt{\indent -end \$

performs the classification of hyperovals in \(\text{PG}(2,16)\). 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):

\begin{center}
\textbf{Orbits at Level 18}
\end{center}
There are 2 orbits at level 18.

\begin{center}
\textbf{Orbit 0 / 2 at Level 18}
\end{center}
Node number: 4212

\{0, 1, 2, 3, 52, 67, 89, 106, 126, 141, 159, 176, 184, 199, 220, 235, 245, 262\}_{144}
Strong generators for a group of order 144:

\[
\begin{pmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
\delta^4 & \delta^9 & 1
\end{pmatrix},
\begin{pmatrix}
1 & \delta^7 & \delta^{13} \\
\delta^8 & \delta^9 & \delta^{10} \\
\delta^6 & 1 & 1
\end{pmatrix},
\begin{pmatrix}
\delta^5 & \delta^5 & \delta^5 \\
\delta^5 & \delta^2 & \delta^{11} \\
\delta^5 & \delta^{14} & 1
\end{pmatrix}
\]

1, 0, 0, 1, 0, 9, 5, 1, 1, 1, 7, 9, 14, 5, 10, 2, 15, 1, 3, 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}
Strong generators for a group of order 16320:

\[
\begin{bmatrix}
  \delta^6 & 0 & 0 \\
  0 & \delta^3 & 0 \\
  0 & 0 & 1
\end{bmatrix}_2,
\begin{bmatrix}
  \delta^9 & 0 & 0 \\
  0 & \delta^7 & 0 \\
  0 & 0 & 1
\end{bmatrix}_1,
\begin{bmatrix}
  \delta^{11} & 0 & 0 \\
  0 & \delta^4 & \delta^7 \\
  \delta^4 & \delta^7 & 1
\end{bmatrix}_3,
\begin{bmatrix}
  \delta^{10} & 0 & 0 \\
  0 & \delta^3 & 0 \\
  \delta^3 & \delta^{11} & 1
\end{bmatrix}_3,
\begin{bmatrix}
  \delta^{12} & \delta^6 \\
  \delta^4 & \delta^3 & \delta^7 \\
  \delta^6 & \delta^3 & 1 \\
\end{bmatrix}_2,
\begin{bmatrix}
  \delta^{14} & \delta^{10} & 1 \\
  \delta^5 & \delta^3 & \delta^6 \\
  \delta^{11} & \delta^6 & \delta^{10}
\end{bmatrix}_1
\]

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:

```
nc_arcs_16:
  > $(ORBITER) -v 4 \\n  >   -define F -finite_field -q 16 -end \n  >   -define P -projective_space -n 2 -field F -v 0 -end \n  >   -define Control -poset_classification_control \n  >   > -problem_label nc_arcs_q16_d2 \n  >   > -W -depth 6 \n  >   > -end \n  >   > -with P -do \n  >   > -projective_space_activity \n  >   >   -classify_arcs \n  >   >   > -control Control \n  >   >   > -target_size 6 \n```
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. The following command shows how to create all arcs associated with cubic surfaces with 13 Eckardt points over the field $\mathbb{F}_{32}$:

```bash
carcs 32 E13:
  $(ORBITER) -v 4 \n  -orbiter_path $(ORBITER_PATH) \n  -define F -finite_field -q 32 -end \n  -define P -projective_space -n 2 -field F -v 0 -end \n  -define Control -poset_classification_control \n  -problem_label nc_arcs_q32_d2 \n  -W -depth 6 \n  -draw_options -end \n  -end \n  -with P -do \n  -projective_space_activity \n  -classify_arcs \n  -control Control \n  -target_size 6 \n  -test_nb_Eckardt_points 13 \n  -d 2 \n  -conic_test \n  -end \n  -end
```

#pdflatex nc_arcs_q32_d2_poset.tex
#open nc_arcs_q32_d2_poset.pdf
6.7 Cubic Curves

Orbiter can classify cubic curves in $\text{PG}(2,q)$. To this end, the $(9,3)$-arcs in $\text{PG}(2,q)$ 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

cubic_curves_PG_2_8:
  ▶ $(\text{ORBITER}) \ -v \ 3 \ \$
  ▶ ▶ -define F -finite_field -q 8 -end \\
  ▶ ▶ -define P -projective_space -n 2 -field F -v 0 -end \\
  ▶ ▶ -define Control -poset_classification_control \\
  ▶ ▶ ▶ -problem_label cc_8 -W -depth 9 \\
  ▶ ▶ ▶ -draw_options -radius 200 -embedded -end \\
  ▶ ▶ -end \\
  ▶ ▶ -with P -do \\
  ▶ ▶ -projective_space_activity \\
  ▶ ▶ -classify_cubic_curves \\
  ▶ ▶ ▶ -control Control \\
  ▶ ▶ ▶ -target_size 9 -d 3 \\
  ▶ ▶ -end \\
  ▶ $(\text{ORBITER}) \ -v \ 2 \ -draw_matrix \\
  ▶ ▶ -input_csv_file cc_8_KM_6_9.csv \\
  ▶ ▶ -box_width 50 -bit_depth 8 -end

classifies the cubic curves in $\text{PG}(2,8)$.
Chapter 7

Cubic Surfaces

7.1 Creation

Orbiter can create, classify and investigate cubic surfaces over small finite fields. In this section, we describe ways in which surfaces can be created. The following sections will be about classification and investigation.

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 7.1-7.3 summarize the Orbiter commands that can be used to create cubic surfaces. The commands are applied to a projective space object, which must be created first. Not all of the surfaces created may have 27 lines, and some of the constructions may yield degenerate surfaces.

Table 7.4 lists activities for a cubic surface object.

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.

```
surface_4_0:
  ▶ $(ORBITER) -v 3 \n  ▶ ▶ -define F -finite_field -q 4 -end \n  ▶ ▶ -define P -projective_space -n 3 -field F -v 0 -end \n  ▶ ▶ -define S -cubic_surface -space P -catalogue 0 -end \n  ▶ ▶ -with S -do \n  ▶ ▶ -cubic_surface_activity \n  ▶ ▶ ▶ -report \n  ▶ ▶ ▶ -end \n  ▶ ▶ -with S -do \n  ▶ ▶ -cubic_surface_activity \n  ▶ ▶ ▶ -exportSomething "points" \n```
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-space</td>
<td>P</td>
<td>Specify the 3-dimensional projective space ( P = PG(3, q) ).</td>
</tr>
<tr>
<td>-label_txt</td>
<td>label</td>
<td>Override the ascii label of the curve.</td>
</tr>
<tr>
<td>-label_tex</td>
<td>label</td>
<td>Override the latex label of the curve.</td>
</tr>
<tr>
<td>-label_for_summary</td>
<td>label</td>
<td>Override the ascii label of the curve, to be used in summary commands.</td>
</tr>
<tr>
<td>-catalogue</td>
<td>( i )</td>
<td>Create the ( i )-th surface in the Orbiter catalogue. Here, ( i ) is an index variable used to index all surfaces in ( PG(3, q) ). The index ( i ) is zero-based. The automorphism group is created as well.</td>
</tr>
<tr>
<td>-by_coefficients</td>
<td>list-of-coeff-pairs</td>
<td>Create a surface from a list of coefficient-monomial pairs. The automorphism group is not created.</td>
</tr>
<tr>
<td>-by_rank</td>
<td>rank ( q_0 )</td>
<td>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 ( PG(19, q_0) ) and use the Orbiter point rank.</td>
</tr>
<tr>
<td>-family_Eckardt</td>
<td>( a ) ( b )</td>
<td>Create the Eckardt surface with parameters ( (a, b) ) as in see [12] (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^2}{a}(a^2 + 1)X_0X_1X_2 = 0 ). The automorphism group is created as well.</td>
</tr>
</tbody>
</table>

Table 7.1: Commands to create a cubic surface (Part 1)
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-family_G13</td>
<td>$a$</td>
<td>Create a member of the $G_{13}$ family with parameter $a$. The surface has 13 Eckardt points.</td>
</tr>
<tr>
<td>-family_F13</td>
<td>$a$</td>
<td>Create a member of the $F_{13}$ family with parameter $a$. The surface has 13 Eckardt points.</td>
</tr>
<tr>
<td>-family_bes</td>
<td>$a$ $c$</td>
<td>Create a member of the “bes”-family with parameter $a$. The surface has 5 Eckardt points.</td>
</tr>
<tr>
<td>-family_general_abcd</td>
<td>$a$ $b$ $c$ $d$</td>
<td>Create a member of the general family with parameters $a, b, c, d$.</td>
</tr>
<tr>
<td>-arc_lifting</td>
<td>$A$</td>
<td>Create the surface associated with the arc $A = a_1, \ldots, a_6$ in $\text{PG}(2,q)$ by means of the Clebsch map. Each of the $a_i$ is the rank of a point in $\text{PG}(2,q)$. Use the trihedral pair algorithm. Here, $A$ is a comma-separated string containing the numerical ranks of the $P_i$ in $\text{PG}(2,q)$.</td>
</tr>
<tr>
<td>-arc_lifting_with_two_lines</td>
<td>$A$ $L$</td>
<td>Create the surface associated with the arc $A = a_1, \ldots, a_6$ in $\text{PG}(2,q)$ by means of the Clebsch map, defined by the lines $\ell_1$ and $\ell_2$ 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.</td>
</tr>
</tbody>
</table>

Table 7.2: Commands to create a cubic surface (Part 2)
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-Cayley_form</td>
<td>(k \ l \ m \ n)</td>
<td>Create the surface from Cayley’s normal form, using the parameters (k, l, m, n).</td>
</tr>
<tr>
<td>-by_equation</td>
<td></td>
<td>Create the surface from an equation.</td>
</tr>
<tr>
<td>-by_double_six</td>
<td>(D)</td>
<td>Create the surface from a double six (D).</td>
</tr>
<tr>
<td>-by_skew_hexagon</td>
<td>label label</td>
<td>Create the surface from a skew hexagon.</td>
</tr>
<tr>
<td>-select_double_six</td>
<td>(L)</td>
<td>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.</td>
</tr>
<tr>
<td>-override_group</td>
<td>descr</td>
<td>Override the automorphism group of the surface by the given group.</td>
</tr>
<tr>
<td>-transform</td>
<td>elt</td>
<td>Apply the transformation given by the group element.</td>
</tr>
<tr>
<td>-transform_inverse</td>
<td>elt</td>
<td>Apply the inverse transformation given by the group element.</td>
</tr>
</tbody>
</table>

Table 7.3: Commands to create a cubic surface (Part 3)
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-report</td>
<td></td>
<td>Produce a latex report about the cubic surface.</td>
</tr>
<tr>
<td>-export_something</td>
<td>type</td>
<td>Exports data about the surface to a file. The type of data to be exported must be specified according to Table 7.5.</td>
</tr>
<tr>
<td>-all_quartic_curves</td>
<td></td>
<td>Creates all quartic curves with 28 bitangents from the surface by projecting along the tangent cone of a point not on any line.</td>
</tr>
<tr>
<td>-export_all_quartic_curves</td>
<td></td>
<td>Creates all quartic curves with 28 bitangents from the surface by projecting along the tangent cone of a point not on any line. Writes a csv file with the curves that have been created.</td>
</tr>
</tbody>
</table>

Table 7.4: Activities related to cubic surfaces

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>points</td>
<td>The points on the surface.</td>
</tr>
<tr>
<td>points_off</td>
<td>The points off the surface.</td>
</tr>
<tr>
<td>Eckardt_points</td>
<td>The Eckardt points of the surface.</td>
</tr>
<tr>
<td>double_points</td>
<td>The double points of the surface.</td>
</tr>
<tr>
<td>single_points</td>
<td>The single points of the surface.</td>
</tr>
<tr>
<td>zero_points</td>
<td>The zero points of the surface.  A point is a zero point if it does not lie on any line.</td>
</tr>
<tr>
<td>singular_points</td>
<td>The singular points of the surface.</td>
</tr>
<tr>
<td>lines</td>
<td>The set of lines of the surface.</td>
</tr>
<tr>
<td>tritangent_planes</td>
<td>The set of tritangent planes of the surface.</td>
</tr>
<tr>
<td>Hesse_planes</td>
<td>The set of Hesse planes of the surface.</td>
</tr>
</tbody>
</table>

Table 7.5: Options for exporting data
A report is created and several data elements of the surface are exported: the set of points on and off the surface, the set of lines, and the set of Hesse planes.

In the following command, the surface over $\mathbb{F}_7$ with catalogue number 0 is created, a report is written, and the set of lines is exported:

```
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 \
  -all_quartic_curves \
  -end \
  -with S -do \
  -cubic_surface_activity \
  -export Something "lines" \
  -end
```

Another way of creating surfaces is as members of known infinite families. For instance,

```
surface_eckardt_13_4_12:
  $(ORBITER) -v 6 \
  -define F -finite_field -q 13 -end \
```
creates the member of the Eckardt family described in [12] with parameters \((a, b) = (4, 12)\) over the field \(\mathbb{F}_{13}\).

It is often desired to produce a nice equation of a cubic surface. This can be done by applying coordinate changes. In the following example, a “nice” equation for the cubic surface over \(\mathbb{F}_{16}\) with catalogue number 0 is produced:

```
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 "15,11,4,0,0,1,12,0,0,12,0,0,0,0,0,1,3" \
  -end \
  -with S16_0 -do \
  -cubic_surface_activity \
  -report \
  -end
```

Let us try the 4-parameter normal form of cubic surfaces with parameters \(a, b, c, d\). The formula can be encoded in a makefile variable:

```
F_abcd_eqn_no_exponents="-(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*(b - d)*X0*X0*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*a*c - a*a*d - a*c*c + b*c*c + 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*X2 \ 
- (a*a*c*d - a*b*c*c - a*a*d + a*b*d + b*c*c - 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*X1*X2 \ 
- (a - c)*(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*X1*X1*X3 \ 
+ (a*d - b*c)*(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*X1*X2*X2 \ 
+ ((1+1)*a*a*b*c*d - a*a*b*d*d - (1+1)*a*a*c*d*d \
- (1+1)*a*b*b*c*c + a*b*b*c*d + (1+1)*a*b*c*c*d + a*b*c*d*d \
- b*b*c*c*d - a*a*b*c + a*a*c*d + a*a*d*d + a*b*b*c + a*b*c*c"
```
\[- (1+1+1+1)abc \cdot d - a\cdot c\cdot c\cdot d + a\cdot c\cdot d\cdot d + b\cdot b\cdot c\cdot c)\cdot x_1\cdot x_2\cdot x_3 \\]
\[+ \ c\cdot a\cdot (a\cdot d - b\cdot c - a + b + c - d)\cdot (b - d)\cdot x_1\cdot x_3\cdot x_3"

The following command parses the formula and creates the surface with \((a, b, c, d) = (4, 2, 2, 4)\) over \(\mathbb{F}_7\):

```
surface_F_abcd_q7:
  ▶ $(ORBITER) -v 3 \\
  ▶ ▶ -define F -finite_field -q 7 -end \\
  ▶ ▶ -with F -do \\
  ▶ ▶ -finite_field_activity \\
  ▶ ▶ -parse_and_evaluate "Fabcd" "X0,X1,X2,X3" \\
  ▶ ▶ ▶ $(F_abcd_eqn_no_exponents) "a=4,b=2,c=2,d=4" \\
  ▶ ▶ -end
```

It is possible to recreate the surface with the generators for the automorphism group. The following command creates two reports about the surface. One with and one without information about the group action.

```
surface_F_alpha_beta_gamma_delta_q7_overwrite_group:
  ▶ $(ORBITER) -v 3 \\
  ▶ ▶ -define F -finite_field -q 7 -end \\
  ▶ ▶ -define P -projective_space -n 3 -field F -v 0 -end \\
  ▶ ▶ -define F_2345 -cubic_surface -space P \\
  ▶ ▶ ▶ -by_equation "F_alpha_beta_gamma_delta" \\
  ▶ ▶ ▶ "\DF{\alpha,\beta,\gamma,\delta})x_0,x_1,x_2,x_3" \\
  ▶ ▶ ▶ $(F_ALPHA_BETA_GAMMA_DELTA) \\
  ▶ ▶ ▶ "alpha=2,beta=3,\gamma=4,\delta=5" \\
  ▶ ▶ ▶ "\D\alpha=2,\beta=3,\gamma=4,\delta=5\D" \\
  ▶ ▶ ▶ -overwrite_group 6 2 \\
  ▶ ▶ ▶ "1,5,0,0,3,6,0,0,1,1,3,0,5,5,0,3, \\
  ▶ ▶ ▶ 1,0,2,5,0,1,6,1,0,0,3,5,0,0,4,4" \\
  ▶ ▶ ▶ -end \\
  ▶ ▶ -with F_2345 -do \\
  ▶ ▶ -cubic_surface_activity \\
  ▶ ▶ -report \\
  ▶ ▶ -end
```

```
pdflatex surface_equation_F_alpha_beta_gamma_delta_q7_report.tex
open surface_equation_F_alpha_beta_gamma_delta_q7_report.pdf
```
### Table 7.6: Options to create a quartic curve

<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-space</td>
<td>P</td>
<td>Specify the 2-dimensional projective space $P = PG(2, q)$</td>
</tr>
<tr>
<td>-label_txt</td>
<td>label</td>
<td>Override the ascii label of the curve.</td>
</tr>
<tr>
<td>-label_tex</td>
<td>label</td>
<td>Override the latex label of the curve.</td>
</tr>
<tr>
<td>-label_for_summary</td>
<td>label</td>
<td>Override the ascii label of the curve, to be used in summary commands.</td>
</tr>
<tr>
<td>-catalogue</td>
<td>OCN</td>
<td>Create the quartic curve from the Orbiter catalogue with the given Orbiter catalogue number.</td>
</tr>
<tr>
<td>-by_coefficients</td>
<td>coeffs</td>
<td>Create a quartic curve given the coefficients of the equation.</td>
</tr>
<tr>
<td>-by_equation</td>
<td>eqn</td>
<td>Create a quartic curve from an equation.</td>
</tr>
<tr>
<td>-from_cubic_surface</td>
<td>$S \ i$</td>
<td>Create the quartic curve from $i$th orbit of the automorphism group of the surface $S$ on points not on lines ($i$ is zero based).</td>
</tr>
<tr>
<td>-override_group</td>
<td>descr</td>
<td>Override the automorphism group of the curve by the given group.</td>
</tr>
<tr>
<td>-transform</td>
<td>elt</td>
<td>Apply the transformation given by the group element.</td>
</tr>
<tr>
<td>-transform_inverse</td>
<td>elt</td>
<td>Apply the inverse transformation given by the group element.</td>
</tr>
</tbody>
</table>

#### 7.2 Quartic Curves

Cubic surfaces with 27 lines are associated with quartic curves with 28 bitangents (see [32]), 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 7.6 lists options to create a quartic curve object.

Table 7.7 lists activities for a quartic curve object.

Suppose we want to study the (unique) quartic curve for $q = 9$. The following command creates the curve whose catalogue number is 0, and produces a report:

```bash
quartic_curve_9_0_report:
  $(ORBITER) -v 3 \
  ▶ $define F -finite_field -q 9 -end \
```
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-report</td>
<td></td>
<td>Produce a latex report about the curve.</td>
</tr>
<tr>
<td>-export_something</td>
<td>type</td>
<td>Exports data of the quartic curve to a file. The type of data to be exported must be specified according to Table 7.8.</td>
</tr>
<tr>
<td>-create_surface</td>
<td></td>
<td>Create a cubic surface from the curve.</td>
</tr>
<tr>
<td>-extract_orbit_on_bitangents_by_length</td>
<td>l</td>
<td>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.</td>
</tr>
<tr>
<td>-extract_specific_orbit_on_bitangents_by_length</td>
<td>l i</td>
<td>Extract the i-th orbit on bitangents of length l.</td>
</tr>
<tr>
<td>-extract_specific_orbit_on_kovalevski_points_by_length</td>
<td>l i</td>
<td>Extract the i-th orbit on Kovalevski points of length l.</td>
</tr>
</tbody>
</table>

Table 7.7: Activities related to quartic curves

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>points</td>
<td>The points on the quartic curve.</td>
</tr>
<tr>
<td>equation</td>
<td>The equation of the quartic curves as a coefficient vector of length 15.</td>
</tr>
<tr>
<td>bitangents</td>
<td>The 28 bitangents.</td>
</tr>
<tr>
<td>Kovalevski_points</td>
<td>The Kovalevski points (points off the curve where 4 bitangents meet).</td>
</tr>
<tr>
<td>singular_points</td>
<td>The singular points of the curve.</td>
</tr>
</tbody>
</table>

Table 7.8: Options for exporting data

222
The report contains the following information:

**The equation**

The equation of the quartic curve is:

\[
\alpha^3 X_0^3 X_1 + \alpha^4 X_0^3 X_2 + \alpha^7 X_0^3 X_2^3 + \alpha^6 X_1^3 X_2 + \alpha^2 X_0 X_2^3 + X_1 X_2^3
\]

\((0, 0, 0, 8, 2, 4, 5, 7, 1, 0, 0, 0, 0, 0, 0)\)

**The gradient**

The gradient of the quartic curve is:

\[
\alpha^7 X_1^3 + \alpha^2 X_2^3
\]

\((0, 4, 7, 0, 0, 0, 0, 0, 0, 0, 0)\)

\[
\alpha^3 X_0^3 + X_2^3
\]

\((8, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0)\)

\[
\alpha^4 X_0^3 + \alpha^6 X_1^3
\]

\((2, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0)\)
General information

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of bitangents</td>
<td>28</td>
</tr>
<tr>
<td>Number of points</td>
<td>28</td>
</tr>
<tr>
<td>Fullness</td>
<td>is full</td>
</tr>
<tr>
<td>Number of Kovalevski points</td>
<td>63</td>
</tr>
<tr>
<td>Bitangent line type ((a_0, a_1, a_2))</td>
<td>((0, 28, 0))</td>
</tr>
<tr>
<td>Number of singular points</td>
<td>0</td>
</tr>
</tbody>
</table>

All points on the curve

The surface has 28 points:
The points on the quartic curve are:

\[
\begin{align*}
0: P_0 & = (1, 0, 0) \\
1: P_1 & = (0, 1, 0) \\
2: P_2 & = (0, 0, 1) \\
3: P_3 & = (1, 1, 1) \\
4: P_4 & = (1, 1, 0) \\
5: P_5 & = (2, 1, 0) \\
6: P_{14} & = (3, 0, 1) \\
7: P_{17} & = (6, 0, 1) \\
8: P_{24} & = (5, 1, 1) \\
9: P_{25} & = (6, 1, 1) \\
10: P_{30} & = (2, 2, 1) \\
11: P_{32} & = (4, 2, 1) \\
12: P_{34} & = (6, 2, 1) \\
13: P_{38} & = (1, 3, 1) \\
14: P_{41} & = (4, 3, 1) \\
15: P_{44} & = (7, 3, 1) \\
16: P_{46} & = (0, 4, 1) \\
17: P_{51} & = (5, 4, 1) \\
18: P_{53} & = (7, 4, 1) \\
19: P_{57} & = (2, 5, 1) \\
20: P_{58} & = (3, 5, 1) \\
21: P_{62} & = (7, 5, 1) \\
22: P_{76} & = (3, 7, 1) \\
23: P_{77} & = (4, 7, 1) \\
24: P_{78} & = (5, 7, 1) \\
25: P_{82} & = (0, 8, 1) \\
26: P_{83} & = (1, 8, 1) \\
27: P_{84} & = (2, 8, 1) \\
28: P_{85} & = (8, 0, 1) \\
\end{align*}
\]

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:

\[
\begin{align*}
0: P_7 & = (4, 1, 0) = c_{13} \cap c_{14} \cap c_{36} \cap c_{46} \\
1: P_8 & = (5, 1, 0) = a_2 \cap a_4 \cap c_{25} \cap c_{45} \\
2: P_9 & = (6, 1, 0) = b_1 \cap b_6 \cap c_{12} \cap c_{26} \\
3: P_{10} & = (7, 1, 0) = a_3 \cap b_5 \cap c_{35} \cap d \\
4: P_{11} & = (8, 1, 0) = b_2 \cap b_3 \cap c_{24} \cap c_{34} \\
5: P_{12} & = (1, 0, 1) = a_3 \cap a_4 \cap c_{23} \cap c_{24} \\
6: P_{13} & = (2, 0, 1) = c_{34} \cap c_{36} \cap c_{45} \cap c_{56} \\
7: P_{15} & = (4, 0, 1) = b_3 \cap b_6 \cap c_{13} \cap c_{16} \\
8: P_{16} & = (5, 0, 1) = a_5 \cap a_6 \cap c_{25} \cap c_{26} \\
9: P_{18} & = (7, 0, 1) = a_2 \cap b_1 \cap c_{35} \cap c_{46} \\
10: P_{19} & = (8, 0, 1) = b_4 \cap b_5 \cap c_{14} \cap c_{15} \\
\end{align*}
\]
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: \( P_{29} = (1, 2, 1) = b_1 \cap b_5 \cap c_{16} \cap c_{36} \)
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: \( P_{65} = (1, 6, 1) = a_4 \cap b_4 \cap c_{25} \cap c_{36} \)
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: \( P_{74} = (1, 7, 1) = a_5 \cap b_3 \cap c_{12} \cap c_{16} \)
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: \( P_{80} = (7, 7, 1) = a_6 \cap b_5 \cap c_{13} \cap c_{24} \)
The lines and their points of contact are:

55 : $P_{51} = (8, 7, 1) = a_2 \cap b_6 \cap c_{15} \cap c_{34}$
56 : $P_{55} = (3, 8, 1) = c_{14} \cap c_{16} \cap c_{24} \cap c_{26}$
57 : $P_{56} = (4, 8, 1) = a_4 \cap a_5 \cap c_{34} \cap c_{35}$
58 : $P_{57} = (5, 8, 1) = b_5 \cap b_6 \cap c_{45} \cap c_{46}$
59 : $P_{58} = (6, 8, 1) = a_6 \cap b_3 \cap c_{36} \cap d$
60 : $P_{59} = (7, 8, 1) = a_3 \cap b_4 \cap c_{12} \cap c_{56}$
61 : $P_{60} = (8, 8, 1) = b_1 \cap b_2 \cap c_{15} \cap c_{25}$
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, 90, 90)

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 : $P_{30} = (8, 0, 1)$
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 : $P_{33} = (5, 2, 1)$
\[
\begin{align*}
a_1 &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & \alpha^3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 8 \end{bmatrix}, \quad P_0 = P(1, 0, 0) 4x \\
a_2 &= \begin{bmatrix} 1 & 0 & \alpha^6 \\ 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 1 \end{bmatrix}, \quad P_{83} = P(1, 8, 1) 4x \\
a_3 &= \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & \alpha^6 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 5 \end{bmatrix}, \quad P_{57} = P(2, 5, 1) 4x \\
a_4 &= \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & \alpha^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 7 \end{bmatrix}, \quad P_{53} = P(7, 4, 1) 4x \\
a_5 &= \begin{bmatrix} 1 & 0 & \alpha^2 \\ 0 & 1 & \alpha^7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 7 \\ 0 & 1 & 4 \end{bmatrix}, \quad P_{30} = P(2, 2, 1) 4x \\
a_6 &= \begin{bmatrix} 1 & 0 & \alpha^2 \\ 0 & 1 & \alpha^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 7 \\ 0 & 1 & 7 \end{bmatrix}, \quad P_5 = P(2, 1, 0) 4x \\
b_1 &= \begin{bmatrix} 1 & 0 & \alpha^6 \\ 0 & 1 & \alpha^7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 4 \end{bmatrix}, \quad P_{58} = P(3, 5, 1) 4x \\
b_2 &= \begin{bmatrix} 1 & 0 & \alpha^7 \\ 0 & 1 & \alpha^6 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 4 \\ 0 & 1 & 5 \end{bmatrix}, \quad P_{14} = P(3, 0, 1) 4x \\
b_3 &= \begin{bmatrix} 1 & 0 & \alpha \\ 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 1 \end{bmatrix}, \quad P_{62} = P(7, 5, 1) 4x \\
b_4 &= \begin{bmatrix} 1 & 0 & \alpha^5 \\ 0 & 1 & \alpha^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 6 \\ 0 & 1 & 7 \end{bmatrix}, \quad P_{77} = P(4, 7, 1) 4x \\
b_5 &= \begin{bmatrix} 1 & 0 & \alpha^5 \\ 0 & 1 & \alpha^3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 6 \\ 0 & 1 & 8 \end{bmatrix}, \quad P_{41} = P(4, 3, 1) 4x \\
b_6 &= \begin{bmatrix} 1 & 0 & \alpha \\ 0 & 1 & \alpha^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 7 \end{bmatrix}, \quad P_3 = P(1, 1, 1) 4x \\
c_{12} &= \begin{bmatrix} 1 & 0 & \alpha^3 \\ 0 & 1 & \alpha^4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 8 \\ 0 & 1 & 2 \end{bmatrix}, \quad P_{17} = P(6, 0, 1) 4x \\
c_{13} &= \begin{bmatrix} 1 & 0 & \alpha \\ 0 & 1 & \alpha^4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 2 \end{bmatrix}, \quad P_{84} = P(2, 8, 1) 4x \\
c_{14} &= \begin{bmatrix} 1 & 0 & \alpha^5 \\ 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 6 \\ 0 & 1 & 1 \end{bmatrix}, \quad P_{32} = P(4, 2, 1) 4x \\
c_{15} &= \begin{bmatrix} 1 & 0 & \alpha^5 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 6 \\ 0 & 1 & 0 \end{bmatrix}, \quad P_1 = P(0, 1, 0) 4x \\
c_{16} &= \begin{bmatrix} 1 & 0 & \alpha \\ 0 & 1 & \alpha^6 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 5 \end{bmatrix}, \quad P_{51} = P(5, 4, 1) 4x \\
\end{align*}
\]
\[
c_{23} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & \alpha^5 \end{bmatrix}_{16} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 6 \end{bmatrix}_{16} \quad P_{82} = \mathbf{P}(0,8,1) \ 4 \times
\]
\[
c_{24} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & \alpha^7 \end{bmatrix}_{14} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 4 \end{bmatrix}_{14} \quad P_{25} = \mathbf{P}(6,1,1) \ 4 \times
\]
\[
c_{25} = \begin{bmatrix} 1 & 0 & \alpha^2 \\ 0 & 1 & \alpha^4 \end{bmatrix}_{72} = \begin{bmatrix} 1 & 0 & 7 \\ 0 & 1 & 2 \end{bmatrix}_{72} \quad P_{76} = \mathbf{P}(3,7,1) \ 4 \times
\]
\[
c_{26} = \begin{bmatrix} 1 & 0 & \alpha^2 \\ 0 & 1 & \alpha^3 \end{bmatrix}_{78} = \begin{bmatrix} 1 & 0 & 7 \\ 0 & 1 & 8 \end{bmatrix}_{78} \quad P_{44} = \mathbf{P}(7,3,1) \ 4 \times
\]
\[
c_{34} = \begin{bmatrix} 1 & 0 & \alpha^4 \\ 0 & 1 & \alpha^3 \end{bmatrix}_{28} = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 8 \end{bmatrix}_{28} \quad P_{38} = \mathbf{P}(1,3,1) \ 4 \times
\]
\[
c_{35} = \begin{bmatrix} 1 & 0 & \alpha^6 \\ 0 & 1 & \alpha^4 \end{bmatrix}_{52} = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 2 \end{bmatrix}_{52} \quad P_{24} = \mathbf{P}(5,1,1) \ 4 \times
\]
\[
c_{36} = \begin{bmatrix} 1 & 0 & \alpha^4 \\ 0 & 1 & \alpha^7 \end{bmatrix}_{24} = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 4 \end{bmatrix}_{24} \quad P_{78} = \mathbf{P}(5,7,1) \ 4 \times
\]
\[
c_{45} = \begin{bmatrix} 1 & 0 & \alpha^4 \\ 0 & 1 & \alpha^6 \end{bmatrix}_{25} = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 5 \end{bmatrix}_{25} \quad P_{34} = \mathbf{P}(6,2,1) \ 4 \times
\]
\[
c_{46} = \begin{bmatrix} 1 & 0 & \alpha^6 \\ 0 & 1 & \alpha \end{bmatrix}_{53} = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 3 \end{bmatrix}_{53} \quad P_{46} = \mathbf{P}(0,4,1) \ 4 \times
\]
\[
c_{56} = \begin{bmatrix} 1 & 0 & \alpha^4 \\ 0 & 1 & 1 \end{bmatrix}_{21} = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 1 \end{bmatrix}_{21} \quad P_{4} = \mathbf{P}(1,1,0) \ 4 \times
\]
\[
d = \begin{bmatrix} 1 & \alpha^6 & 0 \\ 0 & 0 & 1 \end{bmatrix}_{59} = \begin{bmatrix} 1 & 5 & 0 \\ 0 & 0 & 1 \end{bmatrix}_{59} \quad P_{2} = \mathbf{P}(0,0,1) \ 4 \times
\]

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: $1^{28}$

\[
\begin{array}{c|c}
28 & 1 \\
\hline
1 & 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, \\
11 & 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, \\
21 & 21, 22, 23, 24, 25, 26, 27, 0 \\
\end{array}
\]

point types: $1^{28}$

\[
\begin{array}{c|c}
28 & 1 \\
\hline
1 & 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, \\
11 & 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, \\
21 & 21, 22, 23, 24, 25, 26, 27, 0 \\
\end{array}
\]

point types for points off the curve: $4^{63}$
<table>
<thead>
<tr>
<th></th>
<th>1, 2, 3, 4, 5, 6, 7, 8, 9, 10,</th>
<th>11, 12, 13, 14, 15, 16, 17, 18, 19, 20,</th>
<th>21, 22, 23, 24, 25, 26, 27, 28, 29, 30,</th>
<th>31, 32, 33, 34, 35, 36, 37, 38, 39, 40,</th>
<th>41, 42, 43, 44, 45, 46, 47, 48, 49, 50,</th>
<th>51, 52, 53, 54, 55, 56, 57, 58, 59, 60,</th>
<th>61, 62, 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>63</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Lines on points off the curve:

Off point 0 = \( P_6 = (3, 1, 0) \) lies on 4 bisectors : \{ 4, 9, 16, 17 \}

Off point 1 = \( P_7 = (4, 1, 0) \) lies on 4 bisectors : \{ 13, 14, 23, 25 \}

Off point 2 = \( P_8 = (5, 1, 0) \) lies on 4 bisectors : \{ 1, 3, 19, 24 \}

Off point 3 = \( P_9 = (6, 1, 0) \) lies on 4 bisectors : \{ 6, 11, 12, 20 \}

Off point 4 = \( P_{10} = (7, 1, 0) \) lies on 4 bisectors : \{ 2, 10, 22, 27 \}

Off point 5 = \( P_{11} = (8, 1, 0) \) lies on 4 bisectors : \{ 7, 8, 18, 21 \}

Off point 6 = \( P_{12} = (1, 0, 1) \) lies on 4 bisectors : \{ 2, 3, 17, 18 \}

Off point 7 = \( P_{13} = (2, 0, 1) \) lies on 4 bisectors : \{ 21, 23, 24, 26 \}

Off point 8 = \( P_{15} = (4, 0, 1) \) lies on 4 bisectors : \{ 8, 11, 13, 16 \}

Off point 9 = \( P_{16} = (5, 0, 1) \) lies on 4 bisectors : \{ 4, 5, 19, 20 \}

Off point 10 = \( P_{18} = (7, 0, 1) \) lies on 4 bisectors : \{ 1, 6, 22, 25 \}

Off point 11 = \( P_{19} = (8, 0, 1) \) lies on 4 bisectors : \{ 9, 10, 14, 15 \}

Off point 12 = \( P_{20} = (0, 1, 1) \) lies on 4 bisectors : \{ 1, 8, 14, 26 \}

Off point 13 = \( P_{21} = (2, 1, 1) \) lies on 4 bisectors : \{ 7, 9, 20, 25 \}

Off point 14 = \( P_{22} = (3, 1, 1) \) lies on 4 bisectors : \{ 3, 10, 12, 23 \}

Off point 15 = \( P_{23} = (4, 1, 1) \) lies on 4 bisectors : \{ 5, 6, 17, 24 \}

Off point 16 = \( P_{26} = (7, 1, 1) \) lies on 4 bisectors : \{ 16, 19, 21, 27 \}

Off point 17 = \( P_{27} = (8, 1, 1) \) lies on 4 bisectors : \{ 2, 4, 13, 15 \}

Off point 18 = \( P_{28} = (0, 2, 1) \) lies on 4 bisectors : \{ 12, 13, 19, 22 \}

Off point 19 = \( P_{29} = (1, 2, 1) \) lies on 4 bisectors : \{ 6, 10, 16, 26 \}

Off point 20 = \( P_{31} = (3, 2, 1) \) lies on 4 bisectors : \{ 2, 5, 21, 25 \}

Off point 21 = \( P_{33} = (5, 2, 1) \) lies on 4 bisectors : \{ 1, 9, 18, 27 \}

Off point 22 = \( P_{35} = (7, 2, 1) \) lies on 4 bisectors : \{ 7, 11, 17, 23 \}

Off point 23 = \( P_{36} = (8, 2, 1) \) lies on 4 bisectors : \{ 3, 8, 15, 20 \}

Off point 24 = \( P_{37} = (0, 3, 1) \) lies on 4 bisectors : \{ 4, 6, 18, 23 \}

Off point 25 = \( P_{39} = (2, 3, 1) \) lies on 4 bisectors : \{ 1, 5, 12, 16 \}

Off point 26 = \( P_{40} = (3, 3, 1) \) lies on 4 bisectors : \{ 8, 9, 22, 24 \}

Off point 27 = \( P_{42} = (5, 3, 1) \) lies on 4 bisectors : \{ 3, 7, 13, 26 \}

Off point 28 = \( P_{43} = (6, 3, 1) \) lies on 4 bisectors : \{ 2, 11, 14, 19 \}

Off point 29 = \( P_{45} = (8, 3, 1) \) lies on 4 bisectors : \{ 15, 17, 25, 27 \}

Off point 30 = \( P_{47} = (1, 4, 1) \) lies on 4 bisectors : \{ 5, 7, 14, 22 \}

Off point 31 = \( P_{48} = (2, 4, 1) \) lies on 4 bisectors : \{ 8, 10, 17, 19 \}

Off point 32 = \( P_{49} = (3, 4, 1) \) lies on 4 bisectors : \{ 4, 11, 26, 27 \}
Off point 33 = $P_{50} = (4, 4, 1)$ lies on 4 bisecants: \{1, 2, 20, 23\}
Off point 34 = $P_{32} = (6, 4, 1)$ lies on 4 bisecants: \{6, 9, 13, 21\}
Off point 35 = $P_{34} = (8, 4, 1)$ lies on 4 bisecants: \{12, 15, 18, 24\}
Off point 36 = $P_{55} = (5, 5, 1)$ 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 = $P_{39} = (4, 5, 1)$ lies on 4 bisecants: \{18, 19, 25, 26\}
Off point 39 = $P_{60} = (0, 6, 1)$ lies on 4 bisecants: \{0, 10, 20, 21\}
Off point 40 = $P_{61} = (1, 6, 1)$ lies on 4 bisecants: \{0, 9, 19, 23\}
Off point 41 = $P_{62} = (2, 6, 1)$ lies on 4 bisecants: \{0, 11, 18, 22\}
Off point 42 = $P_{63} = (3, 6, 1)$ lies on 4 bisecants: \{0, 7, 12, 27\}
Off point 43 = $P_{64} = (4, 6, 1)$ lies on 4 bisecants: \{0, 13, 17, 26\}
Off point 44 = $P_{65} = (5, 6, 1)$ lies on 4 bisecants: \{0, 2, 6, 8\}
Off point 45 = $P_{66} = (6, 6, 1)$ lies on 4 bisecants: \{0, 16, 25\}

The following command creates the curve over $\mathbb{F}_{13}$ with catalogue number 0 and produces a report. The command also exports the orbit of bitangents of length 4:

```plaintext
 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 \
```

230
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:

\[
\text{NB_QUARTIC_CURVES_{Q19}=14}
\]

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:

\[
\text{quartic\_curves\_19\_report:}
\]

\[
\text{$(ORBITER) -v 3 \ \backslash} \\
\text{\textcircled{1} -define F -finite\textunderscore field -q 19 -end \ \backslash} \\
\text{\textcircled{2} -define P -projective\textunderscore space -n 2 -field F -v 0 -end \ \backslash} \\
\text{\textcircled{3} -loop L 0 $(NB\_QUARTIC\_CURVES\_Q19) 1 \ \backslash} \\
\text{\textcircled{4} -define C -quartic\_curve -space P -catalogue %L -end \ \backslash} \\
\text{\textcircled{5} -with C -do \ \backslash} \\
\text{\textcircled{6} -quartic\_curve\_activity \ \backslash} \\
\text{\textcircled{7} -report \ \backslash} \\
\text{\textcircled{8} -end \ \backslash} \\
\text{\textcircled{9} -end\_loop}
\]

\[
\text{pdflatex \\textit{quartic\_curve\_catalogue\_q19\_iso0\_report.tex}} \\
\text{pdflatex \\textit{quartic\_curve\_catalogue\_q19\_iso1\_report.tex}} \\
\text{pdflatex \\textit{quartic\_curve\_catalogue\_q19\_iso2\_report.tex}} \\
\text{pdflatex \\textit{quartic\_curve\_catalogue\_q19\_iso3\_report.tex}} \\
\text{pdflatex \\textit{quartic\_curve\_catalogue\_q19\_iso4\_report.tex}} \\
\text{pdflatex \\textit{quartic\_curve\_catalogue\_q19\_iso5\_report.tex}} \\
\text{pdflatex \\textit{quartic\_curve\_catalogue\_q19\_iso6\_report.tex}} \\
\text{pdflatex \\textit{quartic\_curve\_catalogue\_q19\_iso7\_report.tex}} \\
\text{pdflatex \\textit{quartic\_curve\_catalogue\_q19\_iso8\_report.tex}} \\
\text{pdflatex \\textit{quartic\_curve\_catalogue\_q19\_iso9\_report.tex}} \\
\text{pdflatex \\textit{quartic\_curve\_catalogue\_q19\_iso10\_report.tex}} \\
\text{pdflatex \\textit{quartic\_curve\_catalogue\_q19\_iso11\_report.tex}} \\
\text{pdflatex \\textit{quartic\_curve\_catalogue\_q19\_iso12\_report.tex}} \\
\text{pdflatex \\textit{quartic\_curve\_catalogue\_q19\_iso13\_report.tex}}
\]
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:

```
quartic_curve_q9_1:
  $(ORBITER_PATH)orbiter.out -v 3 \n  -define F -finite_field -q 9 -end \n  -define P2 -projective_space -n 2 -field F -end \n  -define P3 -projective_space -n 3 -field F -end \n  -define S9_1 -cubic_surface -space P3 -catalogue 1 -end \n  -define C -quartic_curve -space P2 -from_cubic_surface S9_1 0 -end \n  -with C -do \n  -quartic_curve_activity \n  -report \n  -end
```

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 substructures 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

```
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 $q = 7$:

```
quartic_curves_q7:
  $(ORBITER_PATH)orbiter.out -v 3 \n```

232
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:

$\text{NB\_CUBIC\_SURFACES\_Q13=4}$

Just like before, we create all quartic curves arising from projections:

```
quartic_curves_q13:
  $(ORBITER\_PATH)orbiter.out -v 3 \n  -define F -finite_field -q 13 -end \n  -define P -projective_space -n 3 -field F -end \n  -loop L 0 $(NB\_CUBIC\_SURFACES\_Q13) 1 \n  -define S_%L -cubic_surface -space P -catalogue %L -end \n  -end_loop \n  -print_symbols \n  -loop L 0 $(NB\_CUBIC\_SURFACES\_Q13) 1 \n  -with S_%L -do \n  -cubic_surface_activity \n  -export_all_quartic_curves \n  -end \n  -end_loop \n  -print_symbols
```

Each cubic surface creates a number of quartic curves (one for each orbit of the automorphism group on points not on lines). The curves arising from one surface are written to file. There is one file for each surface. First, we combine the output files into one big file:
quartic_curves_q13_combine:
▷ $(ORBITER_PATH)oritzer.out -v 3 \\
▷ ▷ -csv_file_concatenate_from_mask $(NB_CUBIC_SURFACES_Q13) \\
▷ ▷ ▷ surface_catalogue_q13_iso%ld_quartics.csv \\
▷ ▷ ▷ quartics_q13.csv

The next command performs a classification up to isomorphism based on the combined file that we have just created. The result is the classification of quartic curves with 28 bitangents over the field $\mathbb{F}_{13}$:

quartic_curves_q13_classify:
▷ $(ORBITER) -v 3 \\
▷ ▷ -define F -finite_field -q 13 -end \\
▷ ▷ -define P -projective_space -n 2 -field F -v 0 -end \\
▷ ▷ -with P -do \\
▷ ▷ -projective_space_activity \\
▷ ▷ ▷ -classify_quartic_curves_with_substructure \\
▷ ▷ ▷ ▷ quartics_q13.csv \\
▷ ▷ ▷ ▷ 1 4 4 quartic_curves_q13 \\
▷ ▷ ▷ -end \\
▷ ▷ -print_symbols

We find exactly two isomorphism types.
7.3 Classification

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 \( \text{PGL}(4, q) \) of \( \text{PG}(3, q) \). The approach described in [12] relies on Schlaefli's notion of a double six as a substructure [61]. The approach described in [37] utilizes the relation to non-conical six-arcs in a plane. A third approach is described in [38]. All three approaches are available in Orbiter.

In \( \text{PG}(3, 4) \), there is only one type of cubic surfaces with 27 lines. It is a member of the Hirschfeld family, described in [33]. 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 [12] to do so:

```bash
surface_classify_q4:
> $(ORBITER) -v 5 \
>   -define F -finite_field -q 4 -end \
>   -define P -projective_space -n 3 -field F -v 0 -end \
>   -define Control -poset_classification_control -W -end \
>   -with P -do \
>     -projective_space_activity \
>     -classify_surfaces_with_double_sixes Surf27 Control \
>   -end \
>   -with Surf27 -do \
>     -classification_of_cubic_surfaces_with_double_sixes_activity \
>     -report -end \
>   -end \
>   -print_symbols
> pdflatex Surfaces_q4.tex
> open Surfaces_q4.pdf
```

The \(-\text{report}\) option creates a latex report. After some redactions, the report contains the following elements.

### The semilinear group

#### The Action

Group action \( \text{PGL}(4, 4) \) of degree 85

The group is a matrix group.

The base action is on projective space \( \text{PG}(3, 4) \)
\[ q = 4 \]
\[ p = 2 \]
\[ e = 2 \]
\[ n = 3 \]
Number of points = 85
Number of lines = 357
Number of lines on a point = 21
Number of points on a line = 5

\section*{The orthogonal group}

\subsection*{The Action}

Group action $\PGL(4, 4)_{\text{OnWedge}}$ of degree 1365
The group is a matrix group.
The base action is on projective space $\PG(3, 4)$
\[ q = 4 \]
\[ p = 2 \]
\[ e = 2 \]
\[ n = 3 \]
Number of points = 85
Number of lines = 357
Number of lines on a point = 21
Number of points on a line = 5

\section*{The group stabilizing the fixed line}

\subsection*{The Action}

Group action $\PGL(4, 4)_{\text{OnWedge} \ast 100}$ of degree 100

Strong generators for a group of order 5529600: 

\[ $\vdots$ \]
The classification of five-plus-ones

Poset classification up to depth 5

The Orbits

Number of Orbits By Level

<table>
<thead>
<tr>
<th>Depth</th>
<th>Nb of orbits</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

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 7.9: Orbit Representatives

<table>
<thead>
<tr>
<th>N</th>
<th>D</th>
<th>O</th>
<th>Rep</th>
<th>(S,O)</th>
<th>L</th>
<th>F</th>
<th>Gen</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>{ }</td>
<td>(5529600, 1)</td>
<td>100</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>{ 0 }</td>
<td>(55296, 100)</td>
<td>64</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
<td>{ 0, 3 }</td>
<td>(1728, 3200)</td>
<td>36</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0</td>
<td>{ 0, 3, 56 }</td>
<td>(144, 38400)</td>
<td>16</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0</td>
<td>{ 0, 3, 56, 76 }</td>
<td>(288, 19200)</td>
<td>4</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>1</td>
<td>{ 0, 3, 56, 77 }</td>
<td>(96, 57600)</td>
<td>4</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>2</td>
<td>{ 0, 3, 56, 80 }</td>
<td>(72, 76800)</td>
<td>4</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0</td>
<td>{0, 3, 56, 76, 96}</td>
<td>(1440, 3840)</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>-----</td>
<td>-----</td>
<td>-------------------</td>
<td>--------------</td>
<td>------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>1</td>
<td>{0, 3, 56, 76, 97}</td>
<td>(96, 57600)</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>5</td>
<td>2</td>
<td>{0, 3, 56, 80, 92}</td>
<td>(360, 15360)</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>3</td>
<td>{0, 3, 56, 80, 93}</td>
<td>(120, 46080)</td>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Poset of Orbits in Detail

Classification of $5 + 1$ Configurations in $\text{PG}(3, 4)$

The order of the group is 1974067200
The group has 4 orbits on five plus one configurations in $\text{PG}(3, 4)$.

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
$\text{PG}(3, 4)$ 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=$\langle 3,0 \rangle$ up=$\langle 0, -1 \rangle$ 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:

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & \omega^2 & 0 & 0 \\
0 & 0 & \omega^2 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & \omega & 0 & 0 \\
\omega & 0 & \omega & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
\omega^2 & \omega & 0 & 0 \\
\omega & 0 & \omega & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & \omega^2 & 0 & 0 \\
\omega & 0 & \omega & 0 \\
0 & 0 & 1 & 1
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
\omega & \omega & 0 & 0 \\
\omega & 0 & \omega & 0 \\
\omega & 0 & 0 & 1
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 1 & 0 & 0 \\
\omega^2 & \omega & 0 & 0 \\
0 & 0 & \omega^2 & \omega \\
0 & 0 & 1 & 0
\end{bmatrix}
\]
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:

\[
\begin{bmatrix}
1 & \omega & 0 & 0 \\
0 & 1 & 0 & 0 \\
\omega & \omega & 1 & \omega^2 \\
0 & \omega^2 & 0 & 1
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
\omega^2 & 0 & 0 & 0 \\
0 & 0 & \omega^2 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
\omega & 0 & \omega & 0 \\
1 & \omega & 0 & 0 \\
\omega & 1 & \omega & 1
\end{bmatrix}
\]

\[
\begin{bmatrix}
0 & 0 & \omega^2 & 0 \\
0 & 0 & 1 & \omega^2 \\
\omega^2 & 0 & \omega & 0 \\
\omega & \omega^2 & \omega & 1
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 1 & 0 & 0 \\
\omega & \omega & 0 & \omega \\
0 & 0 & \omega^2 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}
\]
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:

\[
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & \omega^2 & 0 & 0 \\
0 & 0 & \omega^2 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix},
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & \omega^2 & 0 & 0 \\
\omega & 0 & \omega & 0 \\
1 & \omega & 0 & 0
\end{pmatrix},
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & \omega^2 & 0 & 0 \\
\omega & 0 & \omega & 0 \\
1 & \omega & \omega & 1
\end{pmatrix},
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & \omega^2 & 0 \\
\omega & 0 & \omega & 0 \\
1 & 0 & 0 & 0
\end{pmatrix},
\begin{pmatrix}
1 & 1 & 0 & 0 \\
0 & 0 & \omega^2 & 0 \\
\omega & 0 & \omega & 0 \\
1 & \omega & \omega & 1
\end{pmatrix},
\begin{pmatrix}
1 & 1 & 0 & 0 \\
0 & 0 & \omega^2 & 0 \\
\omega & 0 & \omega & 0 \\
1 & \omega & \omega & 1
\end{pmatrix}.
\]

1,0,0,0,0,3,0,0,0,0,0,3,0,0,0,0,1,1, 1,0,0,0,3,0,0,0,3,0,0,0,1,0,1,0, 1,0,0,0,3,0,2,0,2,0,0,3,0,3,1,1,0, 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, 1,1,0,0,3,0,0,0,0,0,0,3,3,0,0,1,0,1, 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:

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix},
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & \omega & 0 & 0 \\
0 & 0 & \omega & 0 \\
0 & 0 & 0 & \omega^2
\end{bmatrix},
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & \omega & 1 & 0 \\
0 & 0 & 0 & 1 \\
1 & 0 & 0 & 1
\end{bmatrix},
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & \omega^2 & \omega & 0 \\
0 & 0 & 0 & 1 \\
1 & 0 & 0 & 1
\end{bmatrix},
\begin{bmatrix}
1 & 0 & 0 & 0 \\
1 & 0 & \omega^2 & 0 \\
\omega & 0 & 0 & 1 \\
0 & 0 & 1 & 0
\end{bmatrix},
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & \omega^2 \\
\omega & 1 & 0 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix},
\begin{bmatrix}
\omega^2 & \omega & 1 & 0 \\
\omega & 0 & 1 & 0 \\
\omega & 1 & 0 & 0 \\
1 & 0 & 0 & 1
\end{bmatrix}
\]

The overall number of objects is: 38080

The Group $\text{PGL}(4, 4)$

The order of the group is 1974067200

Cubic Surfaces with 27 Lines in $\text{PG}(3, 4)$

The order of the group is 1974067200
The group has 1 orbits:

The orbits are:
Strong generators for a group of order 51840:

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}, \quad
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & \omega & 0 & 0 \\
0 & 0 & \omega & 0 \\
0 & 0 & 0 & \omega^2
\end{bmatrix}, \quad
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & \omega & 0 & 0 \\
0 & 0 & \omega & 0 \\
0 & 0 & 0 & \omega^2
\end{bmatrix},
\]

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 \\
1 & 1 & 0 & 1
\end{bmatrix}, \quad
\begin{bmatrix}
1 & 0 & 0 & 0 \\
\omega^2 & \omega & \omega \\
0 & 0 & \omega \\
1 & 0 & 0
\end{bmatrix}, \quad
\begin{bmatrix}
1 & 0 & 0 & 0 \\
\omega & \omega & \omega \\
1 & 0 & 0 \\
0 & 1 & 0
\end{bmatrix}, \quad
\begin{bmatrix}
\omega^2 & \omega & \omega \\
\omega & 0 & 0 \\
1 & 0 & 0
\end{bmatrix}
\]

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, 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:

\[
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{pmatrix} _1,
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & \omega & 0 & 0 \\
0 & 0 & \omega & 0 \\
0 & 0 & 0 & \omega^2 \\
\end{pmatrix} _0,
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & \omega & 0 & 0 \\
0 & 0 & \omega & 0 \\
0 & 0 & 0 & \omega^2 \\
\end{pmatrix} _0,
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 \\
1 & 1 & 0 & 1 \\
\end{pmatrix} _0,
\begin{pmatrix}
1 & 0 & 0 & 0 \\
\omega & \omega & \omega & 0 \\
0 & 0 & \omega & 0 \\
0 & 0 & \omega & 0 \\
\end{pmatrix} _0,
\begin{pmatrix}
\omega^2 & \omega & 0 & 0 \\
\omega & 0 & \omega & 0 \\
\omega & 0 & \omega & 0 \\
1 & 0 & 0 & 0 \\
\end{pmatrix} _0
\]

1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1,1,0,0,0,0,0,0,2,0,0,0,0,0,1,0,1,0,0,0,0,0,0,3,0,1,0,0,1,0,1,0,1,0,1,0,0,0,0,3,0,1,0,1,0,1,0,1,0,1,0,0,0,0,3,0,1,0,2,0,0,0,0,2,0,1,0,3,1,0,1,0,0,0,0,1,0,0,3,2,2,0,0,0,2,0,1,0,3,1,0,1,0,0,0,2,0,1,0,3,1,0,0,2,2,1,1,0,1,0,0,1,0,2,0,2,2,0,0,2,2,2,1,1,0,1,3,1,2,1,0,2,0,3,2,0,0,2,0,0,0,0,1,1,3,3,0,3,0,1,1,2,0,1,0,3,0,0,0,0,1,1,3,3,0,3,0,1,1,2,0,1,0,3,0,0,0,0

General information

Points on lines: \(5^{27}\)

Lines on points: \(3^{45}\)

The 27 Lines

\(\ell_0 = a_1 = \begin{pmatrix}
1 & 0 & \omega^2 & 0 \\
0 & 1 & 1 & \omega \\
\end{pmatrix} _{72} = \begin{pmatrix}
1 & 0 & 3 & 0 \\
0 & 1 & 1 & 2 \\
\end{pmatrix} _{72} = \text{Pl}(3,2,3,0,3,1)_{308}\)

\(\ell_1 = a_2 = \begin{pmatrix}
1 & 0 & \omega & 0 \\
0 & 1 & 0 & \omega^2 \\
\end{pmatrix} _{54} = \begin{pmatrix}
1 & 0 & 2 & 0 \\
0 & 1 & 0 & 3 \\
\end{pmatrix} _{54} = \text{Pl}(2,3,0,0,2,1)_{238}\)
\[\ell_2 = a_3 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}_{25} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}_{25} = \Pi_l(1, 1, 0, 0, 1, 1)_{177}\]

\[\ell_3 = a_4 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_{356} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_{356} = \Pi_l(0, 1, 0, 0, 0, 0)_{1}\]

\[\ell_4 = a_5 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}_0 = \Pi_l(1, 0, 0, 0, 0, 0)_{0}\]

\[\ell_5 = a_6 = \begin{bmatrix} 1 & 0 & \omega^2 & 1 \\ 0 & 1 & 0 & \omega \end{bmatrix}_{155} = \begin{bmatrix} 1 & 0 & 3 & 1 \\ 0 & 1 & 0 & 2 \end{bmatrix}_{155} = \Pi_l(3, 2, 0, 2, 3, 1)_{314}\]

\[\ell_6 = b_1 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_{340} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_{340} = \Pi_l(0, 0, 0, 1, 0, 0)_{9}\]

\[\ell_7 = b_2 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}_{38} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}_{38} = \Pi_l(0, 0, 1, 1, 1, 1)_{198}\]

\[\ell_8 = b_3 = \begin{bmatrix} 1 & \omega & 0 & 0 \\ 0 & 0 & 1 & \omega^2 \end{bmatrix}_{61} = \begin{bmatrix} 1 & 2 & 0 & 0 \\ 0 & 0 & 1 & 3 \end{bmatrix}_{61} = \Pi_l(0, 0, 2, 3, 2, 1)_{265}\]

\[\ell_9 = b_4 = \begin{bmatrix} 1 & 0 & \omega^2 & 1 \\ 0 & 1 & 1 & \omega \end{bmatrix}_{156} = \begin{bmatrix} 1 & 0 & 3 & 1 \\ 0 & 1 & 1 & 2 \end{bmatrix}_{156} = \Pi_l(3, 0, 3, 2, 3, 1)_{335}\]

\[\ell_{10} = b_5 = \begin{bmatrix} 1 & \omega^2 & 0 & 1 \\ 0 & 0 & 1 & \omega \end{bmatrix}_{165} = \begin{bmatrix} 1 & 3 & 0 & 1 \\ 0 & 0 & 1 & 2 \end{bmatrix}_{165} = \Pi_l(0, 2, 3, 2, 3, 1)_{337}\]

\[\ell_{11} = b_6 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}_{16} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}_{16} = \Pi_l(0, 0, 1, 0, 0, 0)_{2}\]

\[\ell_{12} = c_{12} = \begin{bmatrix} 1 & 0 & \omega & 1 \\ 0 & 1 & 0 & \omega^2 \end{bmatrix}_{138} = \begin{bmatrix} 1 & 0 & 2 & 1 \\ 0 & 1 & 0 & 3 \end{bmatrix}_{138} = \Pi_l(2, 3, 0, 3, 2, 1)_{256}\]

\[\ell_{13} = c_{13} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}_{109} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}_{109} = \Pi_l(1, 1, 0, 1, 1, 1)_{189}\]

\[\ell_{14} = c_{14} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_{345} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_{345} = \Pi_l(0, 1, 0, 1, 0, 0)_{13}\]

\[\ell_{15} = c_{15} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}_{84} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}_{84} = \Pi_l(1, 0, 0, 1, 0, 0)_{10}\]
\[ \ell_{16} = c_{16} = \begin{bmatrix} 1 & 0 & \omega^2 & 0 \\ 0 & 1 & 0 & \omega \end{bmatrix}_{71} = \begin{bmatrix} 1 & 0 & 3 & 0 \\ 0 & 1 & 0 & 2 \end{bmatrix}_{71} = \text{Pl}(3, 2, 0, 0, 3, 1)_{299} \]

\[ \ell_{17} = c_{23} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}_{85} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}_{85} = \text{Pl}(1, 1, 1, 1, 0, 0)_{16} \]

\[ \ell_{18} = c_{24} = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}_{122} = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}_{122} = \text{Pl}(0, 1, 1, 1, 1, 1)_{202} \]

\[ \ell_{19} = c_{25} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}_{110} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}_{110} = \text{Pl}(1, 0, 1, 1, 1, 1)_{199} \]

\[ \ell_{20} = c_{26} = \begin{bmatrix} 1 & 0 & \omega & 0 \\ 0 & 1 & 1 & \omega^2 \end{bmatrix}_{55} = \begin{bmatrix} 1 & 0 & 2 & 0 \\ 0 & 1 & 1 & 3 \end{bmatrix}_{55} = \text{Pl}(2, 3, 2, 0, 2, 1)_{244} \]

\[ \ell_{21} = c_{34} = \begin{bmatrix} 1 & \omega & 0 & 1 \\ 0 & 0 & 1 & \omega^2 \end{bmatrix}_{145} = \begin{bmatrix} 1 & 2 & 0 & 1 \\ 0 & 0 & 1 & 3 \end{bmatrix}_{145} = \text{Pl}(0, 3, 2, 3, 2, 1)_{271} \]

\[ \ell_{22} = c_{35} = \begin{bmatrix} 1 & 0 & \omega & 1 \\ 0 & 1 & 1 & \omega^2 \end{bmatrix}_{139} = \begin{bmatrix} 1 & 0 & 2 & 1 \\ 0 & 1 & 1 & 3 \end{bmatrix}_{139} = \text{Pl}(2, 0, 2, 3, 2, 1)_{267} \]

\[ \ell_{23} = c_{36} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}_{26} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}_{26} = \text{Pl}(1, 1, 1, 0, 1, 1)_{180} \]

\[ \ell_{24} = c_{45} = \begin{bmatrix} 1 & \omega^2 & 0 & 0 \\ 0 & 0 & 1 & \omega \end{bmatrix}_{81} = \begin{bmatrix} 1 & 3 & 0 & 0 \\ 0 & 0 & 1 & 2 \end{bmatrix}_{81} = \text{Pl}(0, 0, 3, 2, 3, 1)_{332} \]

\[ \ell_{25} = c_{46} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}_{100} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}_{100} = \text{Pl}(0, 1, 1, 0, 0, 0)_{6} \]

\[ \ell_{26} = c_{56} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}_{1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}_{1} = \text{Pl}(1, 0, 1, 0, 0, 0)_{3} \]

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 : $E_{56} = a_5 \cap b_6 \cap c_{56} = P_0 = P_0 = P(1,0,0,0) = P(1,0,0,0), \ T = 0$
1 : $E_{51} = a_5 \cap b_1 \cap c_{15} = P_1 = P_1 = P(0,1,0,0) = P(0,1,0,0), \ T = 4$
2 : $E_{46} = a_4 \cap b_6 \cap c_{46} = P_2 = P_2 = P(0,0,1,0) = P(0,0,1,0), \ T = 20$
3 : $E_{41} = a_4 \cap b_1 \cap c_{14} = P_3 = P_3 = P(0,0,0,1) = P(0,0,0,1), \ T = 84$
4 : $E_{32} = a_3 \cap b_2 \cap c_{32} = P_4 = P_4 = P(1,1,1,1) = P(1,1,1,1), \ T = 27$
5 : $E_{35} = a_5 \cap b_2 \cap c_{25} = P_5 = P_5 = P(1,1,0,0) = P(1,1,0,0), \ T = 1$
6 : $E_{54} = a_5 \cap b_4 \cap c_{45} = P_6 = P_6 = P(\omega,1,0,0) = P(2,1,0,0), \ T = 2$
7 : $E_{53} = a_5 \cap b_3 \cap c_{35} = P_7 = P_7 = P(\omega^2,1,0,0) = P(3,1,0,0), \ T = 3$
8 : $E_{36} = a_3 \cap b_6 \cap c_{36} = P_8 = P_8 = P(1,0,1,0) = P(1,0,1,0), \ T = 5$
9 : $E_{16} = a_1 \cap b_6 \cap c_{16} = P_9 = P_9 = P(\omega,0,1,0) = P(2,0,1,0), \ T = 10$
10 : $E_{26} = a_2 \cap b_6 \cap c_{26} = P_{10} = P_{10} = P(\omega^2,0,1,0) = P(3,0,1,0), \ T = 15$
11 : $E_{14,23,56} = c_{14} \cap c_{23} \cap c_{56} = P_{11} = P_{11} = P(0,1,1,0) = P(0,1,1,0), \ T = 9$
12 : $E_{13,24,56} = c_{13} \cap c_{24} \cap c_{56} = P_{12} = P_{12} = P(1,1,1,0) = P(1,1,1,0), \ T = 6$
13 : $E_{55} = a_6 \cap b_5 \cap c_{56} = P_{13} = P_{13} = P(\omega,1,1,0) = P(2,1,1,0), \ T = 12$
14 : $E_{12,34,56} = c_{12} \cap c_{34} \cap c_{56} = P_{14} = P_{14} = P(\omega^2,1,1,0) = P(3,1,1,0), \ T = 18$
15 : $E_{15,23,46} = c_{15} \cap c_{23} \cap c_{46} = P_{15} = P_{23} = P(1,0,0,1) = P(1,0,0,1), \ T = 21$
16 : $E_{31} = a_3 \cap b_1 \cap c_{13} = P_{16} = P_{26} = P(0,1,0,1) = P(0,1,0,1), \ T = 25$
17 : $E_{15,24,36} = c_{15} \cap c_{24} \cap c_{36} = P_{17} = P_{27} = P(1,1,0,1) = P(1,1,0,1), \ T = 22$
18 : $E_{21} = a_2 \cap b_1 \cap c_{12} = P_{18} = P_{30} = P(0,\omega,0,1) = P(0,2,0,1), \ T = 46$
19 : $E_{15,26,34} = c_{15} \cap c_{26} \cap c_{34} = P_{19} = P_{31} = P(1,\omega,0,1) = P(1,2,0,1), \ T = 24$
20 : $E_{61} = a_6 \cap b_1 \cap c_{16} = P_{20} = P_{34} = P(0,\omega^2,0,1) = P(0,3,0,1), \ T = 67$
21 : $E_{15} = a_1 \cap b_5 \cap c_{15} = P_{21} = P_{35} = P(1,\omega^2,0,1) = P(1,3,0,1), \ T = 23$
22 : $E_{42} = a_4 \cap b_2 \cap c_{24} = P_{22} = P_{38} = P(0,0,1,1) = P(0,0,1,1), \ T = 41$
23 : $E_{13,25,46} = c_{13} \cap c_{25} \cap c_{46} = P_{23} = P_{39} = P(1,0,1,1) = P(1,0,1,1), \ T = 26$
24 : $E_{14,25,36} = c_{14} \cap c_{25} \cap c_{36} = P_{24} = P_{42} = P(0,1,1,1) = P(0,1,1,1), \ T = 30$
25 : $E_{62} = a_6 \cap b_2 \cap c_{26} = P_{25} = P_{47} = P(\omega,\omega,1,1) = P(2,2,1,1), \ T = 53$
26 : $E_{25} = a_2 \cap b_5 \cap c_{25} = P_{26} = P_{48} = P(\omega^2,\omega,1,1) = P(3,2,1,1), \ T = 80$
27 : $E_{16,25,34} = c_{16} \cap c_{25} \cap c_{34} = P_{27} = P_{51} = P(\omega,\omega^2,1,1) = P(2,3,1,1), \ T = 55$
28 : $E_{12} = a_1 \cap b_2 \cap c_{12} = P_{28} = P_{32} = P(\omega^2,\omega^2,1,1) = P(3,3,1,1), \ T = 79$
29 : $E_{43} = a_4 \cap b_3 \cap c_{34} = P_{29} = P_{53} = P(0,0,\omega,1) = P(0,0,2,1), \ T = 62$
30 : $E_{12,35,46} = c_{12} \cap c_{35} \cap c_{46} = P_{30} = P_{54} = P(1,0,\omega,1) = P(1,0,2,1), \ T = 36$
31 : $E_{35} = a_3 \cap b_5 \cap c_{35} = P_{31} = P_{59} = P(\omega,\omega,1,1) = P(2,1,2,1), \ T = 49$
32 : $E_{63} = a_6 \cap b_3 \cap c_{36} = P_{32} = P_{60} = P(\omega^2,1,\omega,1) = P(3,1,2,1), \ T = 76$
33 : $E_{14,26,35} = c_{14} \cap c_{26} \cap c_{35} = P_{33} = P_{61} = P(0,\omega,\omega,1) = P(0,2,2,1), \ T = 51$
34 : $E_{23} = a_2 \cap b_3 \cap c_{23} = P_{34} = P_{62} = P(1,\omega,\omega,1) = P(1,2,2,1), \ T = 39$
35 : $E_{13} = a_1 \cap b_3 \cap c_{13} = P_{35} = P_{67} = P(\omega,\omega,\omega,1) = P(2,3,2,1), \ T = 50$
36 : $E_{16,24,35} = c_{16} \cap c_{24} \cap c_{35} = P_{36} = P_{68} = P(\omega^2,\omega^2,\omega,1) = P(3,3,2,1), \ T = 74$
37 : $E_{45} = a_4 \cap b_5 \cap c_{45} = P_{37} = P_{69} = P(0,0,\omega^2,1) = P(0,0,3,1), \ T = 83$
38 : $E_{64} = a_6 \cap b_4 \cap c_{46} = P_{38} = P_{70} = P(1,0,\omega^2,1) = P(1,0,3,1), \ T = 31$
39 : $E_{12,36,45} = c_{12} \cap c_{36} \cap c_{45} = P_{39} = P_{75} = P(\omega,\omega,\omega^2,1) = P(2,1,3,1), \ T = 59$
40 : $E_{34} = a_3 \cap b_4 \cap c_{34} = P_{40} = P_{76} = P(\omega^2,\omega,\omega^2,1) = P(3,1,3,1), \ T = 71$
41 : $E_{24} = a_2 \cap b_4 \cap c_{24} = P_{41} = P_{79} = \mathbb{P}((\omega, \omega, \omega^2, 1)) = \mathbb{P}(2, 2, 3, 1), \; T = 58$

42 : $E_{13,26,45} = c_{13} \cap c_{26} \cap c_{45} = P_{42} = P_{80} = \mathbb{P}((\omega^2, \omega, \omega^2, 1)) = \mathbb{P}(3, 2, 3, 1), \; T = 70$

43 : $E_{14} = a_1 \cap b_4 \cap c_{14} = P_{43} = P_{81} = \mathbb{P}(0, \omega^2, \omega^2, 1) = \mathbb{P}(0, 3, 3, 1), \; T = 72$

44 : $E_{16,23,45} = c_{16} \cap c_{23} \cap c_{45} = P_{44} = P_{82} = \mathbb{P}(1, \omega^2, \omega^2, 1) = \mathbb{P}(1, 3, 3, 1). \; T = 33$

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}$

248
Tritangent planes

The 45 tritangent planes are:

\[
\pi_{12} = \pi_0 = 79 = \begin{bmatrix}
1 & 0 & 0 & \omega^2 \\
0 & 1 & 0 & \omega^2 \\
0 & 0 & 1 & 1 \\
\end{bmatrix}
= \begin{bmatrix}
1 & 0 & 0 & 3 \\
0 & 1 & 0 & 3 \\
0 & 0 & 1 & 1 \\
\end{bmatrix}
\]

\[
= V(\omega^2X_0 + \omega^2X_1 + X_2 + X_3) = V(3X_0 + 3X_1 + X_2 + X_3)
\]
dual pt rank = 52 = (3, 3, 1, 1).

\[
\pi_{16,25,34} = \pi_{44} = 55 = \begin{bmatrix}
1 & 0 & 0 & \omega \\
0 & 1 & 0 & \omega \\
0 & 0 & 1 & \omega^2 \\
\end{bmatrix}
= \begin{bmatrix}
1 & 0 & 0 & 2 \\
0 & 1 & 0 & 2 \\
0 & 0 & 1 & 3 \\
\end{bmatrix}
\]

\[
= V(\omega X_0 + \omega X_1 + \omega^2X_2 + X_3) = V(2X_0 + 2X_1 + 3X_2 + X_3)
\]
dual pt rank = 79 = (2, 2, 3, 1).

Karaoglu [37] describes a different algorithm, based on non-conical six-arcs and Steiner trihedral pairs. The command

```
surface_classify_q4_arc_lifting_two_lines:
  $\text{(ORBITER) \ -v 10 \}
  \ -define F -finite_field -q 4 -end \n  \ -define P -projective_space -n 3 -field F -v 0 -end \n  \ -with P -do \n  \ -projective_space_activity \n  \ -control_six_arcs -problem_label sixarcs_q4 -end \n  \ -classify_surfaces_through_arcs_and_two_lines \n  \ -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.
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-identify_Eckardt</td>
<td></td>
<td>Identifies the isomorphism type of the Eckardt surface with parameter ( a ). All values of ( a ) are considered.</td>
</tr>
<tr>
<td>-identify_F13</td>
<td></td>
<td>Identifies the isomorphism type of the ( F_{13} ) surface with parameter ( a ). All values of ( a ) are considered.</td>
</tr>
<tr>
<td>-identify_Bes</td>
<td></td>
<td>Identifies the isomorphism type of the Bes surface with parameters ( a ) and ( c ). All values of ( a, c ) are considered.</td>
</tr>
<tr>
<td>-identify_general_abcd</td>
<td></td>
<td>Identifies the isomorphism type of the general surface with parameters ( a, b, c, d ). All values of ( a, b, c, d ) are considered.</td>
</tr>
<tr>
<td>-isomorphism_testing</td>
<td>S1 S2</td>
<td>Computes an isomorphism from surface S1 to surface S2 or concludes that none exists.</td>
</tr>
<tr>
<td>-recognize</td>
<td>S</td>
<td>Identifies the isomorphism type of the given surface ( S ).</td>
</tr>
<tr>
<td>-create_source_code</td>
<td></td>
<td>Creates source code for the classification of cubic surfaces with 27 lines over the given field.</td>
</tr>
<tr>
<td>-sweep_Cayley</td>
<td></td>
<td>Identifies all surfaces given by the Cayley normal form over the given field.</td>
</tr>
</tbody>
</table>

Table 7.10: Activities related to the classification of cubic surfaces

### 7.4 Isomorphism Testing and Recognition

Besides classification, Orbiter provides recognition and isomorphism testing of cubic surfaces. Table 7.10 lists the relevant Orbiter commands. These commands are activities of type “classification of cubic surfaces with double sixes.”

The `-recognize` option can be used to identify a given surface in the list produced by the classification. The command computes an isomorphism between the given surface and the surface in the catalogue. For instance,

```
$ORBITER -v 3
```

```
define F -finite_field -q 7 -end \\
define P -projective_space -n 3 -field F -v 0 -end \\
define Control -poset_classification_control -W -end \\
with P -do \\
projective_space_activity \\
classify_surfaces_with_double_sixes Surf Control \\
```
identifies the surface (cf. Table 8.5)

$$X_0^2X_3 + X_1^2X_2 + X_1X_2^2 + X_0X_3^2 + X_1X_2X_3 = 0$$

(7.1)

in the classification of surfaces over the field $\mathbb{F}_7$. This means that an isomorphism from the given surface to the surface in the list is computed. Also, the generators of the automorphism group of the given surface are computed, using the known generators for the automorphism group of the surface in the classification. For instance, executing the command above produces the isomorphism

$$
\begin{bmatrix}
1 & 4 & 4 & 0 \\
6 & 0 & 0 & 0 \\
6 & 2 & 0 & 1 \\
7 & 0 & 4 & 0
\end{bmatrix}
$$

(7.2)

Orbiter can compute isomorphism between two given surfaces. Both surfaces must have 27 lines. Let us consider an example. Suppose we want to find an isomorphism between the surfaces

$$
X_0^2X_2 + X_1^2X_2 + X_1X_2^2 + X_0X_3^2 + X_1X_2X_3 + \delta^{13}X_1X_3^2 + \delta^{12}X_2X_3^2 + \\
\delta^7X_0X_2X_3 + \delta^7X_1X_2X_3 = 0
$$

and

$$
\delta^{11}X_0^2X_3 + \delta^{12}X_1^2X_2 + \delta^{12}X_1X_2^2 + \delta^{11}X_0X_3^2 + X_1X_2X_3 = 0,
$$

over the field $\mathbb{F}_{16}$. The command

```
surface_isomorph_16:
```

```bash
$\text{(ORBITER)} \ -v \ 3 \ \$
```

```bash
$\text{(ORBITER)} \ -v \ 3 \$
```

```bash
$\text{(ORBITER)} \ -d \$
```

```bash
$\text{(ORBITER)} \ -d \$
```

```bash
$\text{(ORBITER)} \ -d \$
```

```bash
$\text{(ORBITER)} \ -d \$
```

```bash
$\text{(ORBITER)} \ -d \$
```

```bash
$\text{(ORBITER)} \ -d \$
```

```bash
$\text{(ORBITER)} \ -d \$
```

```bash
$\text{(ORBITER)} \ -d \$
```
computes an isomorphism from the first surface to the second, given by the matrix:

\[
\begin{bmatrix}
12 & 13 & 0 & 0 \\
8 & 13 & 0 & 0 \\
0 & 0 & 13 & 0 \\
12 & 13 & 11 & 1
\end{bmatrix}.
\]

Orbiter can recognize the isomorphism type of a cubic surface with 27 lines inside the Orbiter catalogue. Given a surface, Orbiter will return the orbiter catalogue number of the surface isomorphic to it. Let us consider an example. Suppose we want to determine the isomorphism type of the surface

\[X_0^2X_3 + X_1^2X_2 + X_1X_2^2 + X_0X_3^2 + X_1X_2X_3 = 0.\]

The command

\texttt{surface\_recognize\_8:} \\
\texttt{\$\texttt{(ORBITER)} -v 3 \} \\
\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{-define F -finite\_field -q 8 -end \}}} \\
\texttt{\texttt{\texttt{\texttt{\texttt{-define P -projective\_space -n 3 -field F -v 0 -end \}}} \\
\texttt{\texttt{\texttt{\texttt{\texttt{-define Control -poset\_classification\_control -W -end \}}} \\
\texttt{\texttt{\texttt{\texttt{\texttt{-with P -do \}}} \\
\texttt{\texttt{\texttt{\texttt{\texttt{-projective\_space\_activity \}}} \\
\texttt{\texttt{\texttt{\texttt{\texttt{-classify\_surfaces\_with\_double\_sixes Surf27 Control \}}} \\
\texttt{\texttt{\texttt{\texttt{\texttt{-end \}}} \\
\texttt{\texttt{\texttt{\texttt{\texttt{-with Surf27 -do \}}} \\
\texttt{\texttt{\texttt{\texttt{\texttt{-classification\_of\_cubic\_surfaces\_with\_double\_sixes\_activity \}}} \\
\texttt{\texttt{\texttt{\texttt{\texttt{-recognize \}}} \\
\texttt{\texttt{\texttt{\texttt{\texttt{-space P \}}} \\
\texttt{\texttt{\texttt{\texttt{\texttt{-by\_coefficients "1,6,1,8,1,11,1,13,1,19" \}}} \\
\texttt{\texttt{\texttt{\texttt{\texttt{-end \}}} \\
\end{verbatim}
finds that the surface is isomorphic to the surface with OCN equal to 0. An isomorphism will be computed as well.

The command

```
surface_sweep_Cayley_13:
  $(ORBITER) -v 3 
  -define F -finite_field -q 13 -end 
  -define P -projective_space -n 3 -field F -v 0 -end 
  -define Control -poset_classification_control -W -end 
  -with P -do 
  -projective_space_activity 
  -classify_surfaces_with_double_sixes Surf27 Control 
  -end 
  -with Surf27 -do 
  -classification_of_cubic_surfaces_with_double_sixes_activity 
  -sweep_Cayley 
  -end 
  -print_symbols
```

creates all surfaces in Cayley’s 4-parameter normal form over the field $\mathbb{F}_{13}$ and determines their isomorphism types.
7.5 Dickson Surfaces

For very small values of \( q \), the cubic surfaces over \( \mathbb{F}_q \) can be classified using the basic Schreier algorithm from Section 6.1. Let us look at an example. Suppose we want to classify all cubic surfaces in PG(3, 2). The non-singular ones have been classified by Dickson [24]. Orbiter can be used to recreate this classification and to investigate these surfaces further.

In Section 6.1, cubic surfaces in PG(3, 2) were classified using this Orbiter command:

```plaintext
orbits_cubic_curves_q2:
  $(ORBITER) -v 4 \
  $\diamond$ define G -linear_group -PGL 3 2 -end \
  $\diamond$ define Orb -orbits -group G \
  $\diamond$ $\diamond$ -on_polynomials 3 \
  $\diamond$ $\diamond$ -end
  #pdflatex poly_orbits_d3_n3_q2.tex
  #open poly_orbits_d3_n3_q2.pdf
```

To investigate the properties of these surfaces, the following two commands can be used:

```plaintext
poly_orbits_d3_n3_q2_F2.csv: poly_orbits_d3_n3_q2.csv
  $(ORBITER) -v 4 \
  $\diamond$ $\diamond$ -define F -finite_field -q 2 -end \
  $\diamond$ $\diamond$ -define P -projective_space -n 3 -field F -v 0 -end \
  $\diamond$ $\diamond$ -with P -do \
  $\diamond$ $\diamond$ -projective_space_activity \
  $\diamond$ $\diamond$ -table_of_cubic_surfaces_compute_properties \
  $\diamond$ $\diamond$ $\diamond$ poly_orbits_d3_n3_q2.csv 2 0 \
  $\diamond$ $\diamond$ -end
```

and

```plaintext
Dickson_q2_analyze: poly_orbits_d3_n3_q2_F2.csv
  $(ORBITER) -v 4 \
  $\diamond$ $\diamond$ -define F -finite_field -q 2 -end \
  $\diamond$ $\diamond$ -define P -projective_space -n 3 -field F -v 0 -end \
  $\diamond$ $\diamond$ -with P -do \
  $\diamond$ $\diamond$ -projective_space_activity \
  $\diamond$ $\diamond$ -cubic_surface_properties_analyze \
  $\diamond$ $\diamond$ $\diamond$ poly_orbits_d3_n3_q2_F2.csv 2 \
  $\diamond$ $\diamond$ -end
  pdflatex poly_orbits_d3_n3_q2_F2_report.tex
  open poly_orbits_d3_n3_q2_F2_report.pdf
```
To investigate the properties of these surfaces over the extension field $\mathbb{F}_4$, the following two commands can be used:

```bash
poly_orbits_d3_n3_q2_F4.csv: poly_orbits_d3_n3_q2.csv
  $(ORBITER) -v 4 \
  -define F -finite_field -q 4 -end \
  -define P -projective_space -n 3 -field F -v 0 -end \
  -with P -do \
  -projective_space_activity \
  -table_of_cubic_surfaces_compute_properties \
  poly_orbits_d3_n3_q2.csv 2 0 \
  -end
```

and

```bash
Dickson_q4_analyze: poly_orbits_d3_n3_q2_F4.csv
  $(ORBITER) -v 4 \
  -define F -finite_field -q 4 -end \
  -define P -projective_space -n 3 -field F -v 0 -end \
  -with P -do \
  -projective_space_activity \
  -cubic_surface_properties_analyze \
  poly_orbits_d3_n3_q2_F4.csv 2 \
  -end
  pdflatex poly_orbits_d3_n3_q2_F4_report.tex
  open poly_orbits_d3_n3_q2_F4_report.pdf
```
The data in Orbiter can be exported to be used for automated processing. It is possible to create a csv file with the cubic surfaces with 27 lines for a given \( q \). The following example shows how to export the data about cubic surfaces with \( q = 17 \):

```
MAKE_TABLE_OF_CUBIC_SURFACES=-define \ 
  P -projective_space -n 3 -field F -v 0 -end \ 
  -with P -do \ 
  P -projective_space_activity \ 
  -table_of_cubic_surfaces \ 
  -end
```

cubic_surfaces_tables_17:
```
$(ORBITER) -v 3 \ 
  -define F -finite_field -q 17 -end \ 
  $(MAKE_TABLE_OF_CUBIC_SURFACES)
```

A file `table_of_cubic_surfaces_q17_info.csv` is created. The command

```
cubic_surfaces_table_latex_17:
```
```
$(ORBITER) -v 3 -csv_file_latex 1 \ 
  table_of_cubic_surfaces_q17_info.csv
```

produces a latex table from the csv file.
Chapter 8

Ring Theory

8.1 Polynomials Over Finite Fields

For \( p \) prime, the finite field \( \mathbb{F}_p \) of order \( p \) can be constructed as factorring of the integers modulo \( p \). 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 8.1 lists Orbiter activities for polynomials over finite fields. The activities are finite field activities. For instance, the command

\[
\text{poly\_division:}
\]

\[
\text{poly\_division2:}
\]

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.

\[
\text{poly\_division2:}
\]

\[
\text{poly\_division2:}
\]
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-polynomial_division</td>
<td>$A(X) B(X)$</td>
<td>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.</td>
</tr>
<tr>
<td>-extended_gcd_for_polynomials</td>
<td>$A(X) B(X)$</td>
<td>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.</td>
</tr>
<tr>
<td>-polynomial_mult_mod</td>
<td>$A(X) B(X) M(X)$</td>
<td>Multiply the polynomials $A(X)$ and $B(X)$ modulo $M(X)$ in $\mathbb{F}_q[X]$.</td>
</tr>
<tr>
<td>-Berlekamp_matrix</td>
<td>$A(X)$</td>
<td>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: $1, X, X^2, \ldots, X^{d-1}$.</td>
</tr>
<tr>
<td>-polynomial_find_roots</td>
<td>$A(X)$</td>
<td>Find the roots of $A(X)$ over $\mathbb{F}_q$.</td>
</tr>
<tr>
<td>-make_table_of_irreducible_polynomials</td>
<td>$d$</td>
<td>Produces a list of all irreducible polynomials of degree $d$ over $\mathbb{F}_q$.</td>
</tr>
</tbody>
</table>

Table 8.1: Finite Field Activities Related to Polynomials
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,

```
poly_gcd:
```

```
$($ORBITER$) -v 2 \n$define F -finite_field -q 2 -end \n$define A -vector -field F -sparse 11 "1,0,1,10" -end \n$define B -vector -field F -dense "1,0,1,1" -end \n-with F -do \n-finite_field_activity \n-polynomial_division A B -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

$$(3X^2 + 2X + 1) \cdot (5X^2 + 4X + 3) \mod (X^3 + 7) \mod 7.$$

```
poly_mult_mod1:
```

```
$($ORBITER$) -v 2 \n$define F -finite_field -q 7 -end \n-with F -do \n-finite_field_activity \n-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

$$(2X^2 + X + 3) \cdot (4X^2 + 3X + 5) \mod (X^3 + 7) \mod 7.$$
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:

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 $\mathbb{F}_2$:

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:
Here is a somewhat larger example for numerical arguments. We wish to multiply 999 by 997 modulo 1033. The first command performs multiplication:

```bash
mul_pols_999_997:
$ (ORBITER) -v 2 \
  -define F -finite_field -q 2 -end \
  -with F -do \n  -finite_field_activity \
  -mul_pols 999 997 \
  -end
```

The next command performs division with remainder:

```bash
polyed_m_div_349147_1033:
$ (ORBITER) -v 2 \
  -define F -finite_field -q 2 -end \
  -with F -do \n  -finite_field_activity \
  -polyed_m_div 349147 1033 \
  -end
```

The next command performs an independent check, using the finite field with 1024 elements. This check relies on the fact that the irreducible polynomial to create the field $\mathbb{F}_{1024}$ is exactly the polynomial by which we did mod out in the example before:

```bash
mul_pols_1024_999_997_check:
$ (ORBITER) -v 3 \
  -define F -finite_field -q 1024 -end \
  -with F -do \
```

261
In this last command, the formula $a \times b$ is used and evaluated over $\mathbb{F}_{1024}$, using $a = 999$ and $b = 997$.

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 $2^{32} - 2$:

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\"

TWO\_TO\_THE\_32\_MINUS\_2=4294967294

\begin{verbatim}
power_mod_inverse:
  ▶ $(ORBITER) -v 2 \n  ▶ ▶ -define F -finite_field -q 2 -end \n  ▶ ▶ -define M -vector -field F -sparse 33 $(CRC32\_SPARSE) -end \n  ▶ ▶ -define A -vector -field F -sparse 2 "1,1" -end \n  ▶ ▶ -with F -do \n  ▶ ▶ -finite_field_activity \n  ▶ ▶ -polynomial_power_mod A $(TWO\_TO\_THE\_32\_MINUS\_2) M \n  ▶ ▶ -end
\end{verbatim}

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:

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\"

\begin{verbatim}
mult_mod_to_get_one:
  ▶ $(ORBITER) -v 2 \n  ▶ ▶ -define F -finite_field -q 2 -end \n  ▶ ▶ -define M -vector -field F -sparse 33 $(CRC32\_SPARSE) -end \n\end{verbatim}
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

\[
\text{Berlekamp\_matrix\_2,3:}
\]

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 polynomials. For a given degree over a given field \(\mathbb{F}_q\). We distinguish two tasks: The first task is finding one irreducible polynomial of the given degree and with the given field of coefficients. The second task is finding all irreducible polynomials given that one has already been found.

For instance, the command

\[
\text{search\_primitive\_poly\_2:}
\]

searches for primitive polynomials over \(\mathbb{F}_2\) of degree 2 to 10. The unix command `grep` is used to filter the output for lines containing the given pattern “//”. This yields the list

263
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 $s = 2$).

The following command creates a list of all irreducible polynomials of degree 3 over $\mathbb{F}_4$:

```
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 $\mathbb{F}_4$:

<table>
<thead>
<tr>
<th>Degree</th>
<th>Polynomial</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>112 : 91</td>
</tr>
<tr>
<td>1</td>
<td>1031 : 77</td>
</tr>
<tr>
<td>2</td>
<td>1213 : 103</td>
</tr>
<tr>
<td>3</td>
<td>1323 : 123</td>
</tr>
<tr>
<td>4</td>
<td>1322 : 122</td>
</tr>
<tr>
<td>5</td>
<td>1222 : 106</td>
</tr>
<tr>
<td>6</td>
<td>1021 : 73</td>
</tr>
<tr>
<td>7</td>
<td>1101 : 81</td>
</tr>
<tr>
<td>8</td>
<td>1333 : 127</td>
</tr>
<tr>
<td>9</td>
<td>1232 : 110</td>
</tr>
<tr>
<td>10</td>
<td>1113 : 87</td>
</tr>
<tr>
<td>11</td>
<td>1233 : 111</td>
</tr>
<tr>
<td>12</td>
<td>1301 : 113</td>
</tr>
<tr>
<td>13</td>
<td>1003 : 67</td>
</tr>
<tr>
<td>14</td>
<td>1112 : 86</td>
</tr>
</tbody>
</table>
15 : 1002 : 66
16 : 1312 : 118
17 : 1011 : 69
18 : 1132 : 94
19 : 1201 : 97
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-field</td>
<td>label</td>
<td>Specify the field of coefficients.</td>
</tr>
<tr>
<td>-homogeneous_of_degree</td>
<td>$d$</td>
<td>Specify the degree $d$ of polynomials.</td>
</tr>
<tr>
<td>-number_of_variables</td>
<td>$n$</td>
<td>Specify the number $n$ of variables.</td>
</tr>
<tr>
<td>-monomial_ordering_partition</td>
<td></td>
<td>Set monomial ordering to partition ordering.</td>
</tr>
<tr>
<td>-monomial_ordering_lex</td>
<td></td>
<td>Set monomial ordering to lexicographic ordering.</td>
</tr>
<tr>
<td>-variables</td>
<td>label-tex label-tex</td>
<td>Specify variable labels in ascii and in latex.</td>
</tr>
</tbody>
</table>

Table 8.2: Commands to create a multivariate polynomial ring

8.2 Multivariate Polynomial Rings

Orbiter can work with multivariate polynomial rings. Table 8.2 lists the commands for creating a multivariate polynomial ring.

Table 8.3 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 8.4 shows the monomials in the partition ordering for degrees 1, 2, 3 and 4 in a plane.

Table 8.5 shows the partition ordering monomials of degree at most 3 in 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.

```plaintext
CREMONA_MAP_Y0="3*y0*y0*y0*y0*y0*y2+4*y0*y0*y0*y1*y1*y2+2*y0*y0*y0*y2*y2*y2+y0*y1*y1*y1*y1*y2+6*y0*y1*y1*y2*y2*y2+9*y0*y2*y2*y2*y2*y2"
CREMONA_MAP_Y1="y0*y0*y0*y0*y1+5*y0*y0*y1*y1*y1"
```

266
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-cheat_sheet</td>
<td></td>
<td>Create a report about the polynomial ring.</td>
</tr>
<tr>
<td>-ideal</td>
<td>label_txt label_tex label_set</td>
<td>Compute the ideal of the set of points.</td>
</tr>
<tr>
<td>-apply_transformation</td>
<td>eqn elt</td>
<td>Apply the transformation given by the group element to the given equation.</td>
</tr>
<tr>
<td>-set_variable_names</td>
<td>txt tex</td>
<td>Choose new variable names in text and tex format.</td>
</tr>
<tr>
<td>-print_equation</td>
<td>equation</td>
<td>Print the given equation.</td>
</tr>
</tbody>
</table>

Table 8.3: Activities for a multivariate polynomial ring

<table>
<thead>
<tr>
<th>$h$</th>
<th>mon</th>
<th>vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$X_0$</td>
<td>(1,0,0)</td>
</tr>
<tr>
<td>1</td>
<td>$X_1$</td>
<td>(0,1,0)</td>
</tr>
<tr>
<td>2</td>
<td>$X_2$</td>
<td>(0,0,1)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$h$</th>
<th>mon</th>
<th>vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$X_0^2$</td>
<td>(2,0,0)</td>
</tr>
<tr>
<td>1</td>
<td>$X_1^2$</td>
<td>(0,2,0)</td>
</tr>
<tr>
<td>2</td>
<td>$X_2^2$</td>
<td>(0,0,2)</td>
</tr>
<tr>
<td>3</td>
<td>$X_0X_1$</td>
<td>(1,1,0)</td>
</tr>
<tr>
<td>4</td>
<td>$X_0X_2$</td>
<td>(1,0,1)</td>
</tr>
<tr>
<td>5</td>
<td>$X_1X_2$</td>
<td>(0,1,1)</td>
</tr>
</tbody>
</table>

Table 8.4: The partition ordering of monomials of degree 1, 2, 3 and 4 in a plane

<table>
<thead>
<tr>
<th>$h$</th>
<th>mon</th>
<th>vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$X_0^4$</td>
<td>(4,0,0)</td>
</tr>
<tr>
<td>1</td>
<td>$X_1^4$</td>
<td>(0,4,0)</td>
</tr>
<tr>
<td>2</td>
<td>$X_2^4$</td>
<td>(0,0,4)</td>
</tr>
<tr>
<td>3</td>
<td>$X_0^3X_1$</td>
<td>(3,1,0)</td>
</tr>
<tr>
<td>4</td>
<td>$X_0^2X_2$</td>
<td>(3,0,1)</td>
</tr>
<tr>
<td>5</td>
<td>$X_0X_1^3$</td>
<td>(1,3,0)</td>
</tr>
<tr>
<td>6</td>
<td>$X_1^3X_2$</td>
<td>(0,3,1)</td>
</tr>
<tr>
<td>7</td>
<td>$X_0X_2^3$</td>
<td>(1,0,3)</td>
</tr>
<tr>
<td>8</td>
<td>$X_1X_3^3$</td>
<td>(0,1,3)</td>
</tr>
<tr>
<td>9</td>
<td>$X_0^2X_1^2$</td>
<td>(2,2,0)</td>
</tr>
<tr>
<td>10</td>
<td>$X_0X_1^2X_2^2$</td>
<td>(2,0,2)</td>
</tr>
<tr>
<td>11</td>
<td>$X_1^2X_2^3$</td>
<td>(0,2,2)</td>
</tr>
<tr>
<td>12</td>
<td>$X_0^2X_1X_2$</td>
<td>(2,1,1)</td>
</tr>
<tr>
<td>13</td>
<td>$X_0X_1^2X_2$</td>
<td>(1,2,1)</td>
</tr>
<tr>
<td>14</td>
<td>$X_0X_1^3$</td>
<td>(1,1,2)</td>
</tr>
</tbody>
</table>

267
Table 8.5: The partition ordering of monomials of degree 1, 2 and 3 in PG(3, q)
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 $\text{PG}(2,11)$:

\begin{verbatim}
  arcs_5_2_q11:
    $\text{ORBITER} -v 4$
    -define F -finite_field -q 11 -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 -formula "y0" "y_0" "y0,y1,y2"
  -define Y1 -formula "y1" "y_1" "y0,y1,y2"
  -define Y2 -formula "y2" "y_2" "y0,y1,y2"
  -define Cremona -collection "Y0,Y1,Y2"
    -with P -do
    -projective_space_activity
    -map R Cremona ""
  -end
\end{verbatim}

269
It finds exactly two isomorphism types of arcs. The representative sets are

\{0, 1, 2, 3, 37\}, \{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 \(\mathbb{F}_{11}\):

\begin{verbatim}
$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_objects \ 
  -file_of_points arcs_5_2_q11_lvl_5 \ 
  -end \ 
  -with C -do \ 
  -combinatorial_object_activity \ 
  -ideal R \ 
  -end
\end{verbatim}

The ideals are generated by

\[ 7x0*x1 + 5*x0*x2 + 10*x1*x2 \]
and
\[ 4x_0 x_1 + 8x_0 x_2 + 10x_1 x_2, \]
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.

\[
\text{PTS}_\text{OF}_\text{SURFACE}_\text{ORBIT211}_Q3_L9_E4 = "\\n\begin{align*}
0,1,2,5,7,8,10,14,9,12, \\
15,3,16,37,31,34,20,19,17,32,36,33"
\end{align*}
\]

Then, we create a ring and compute the ideal:

```plaintext
surface_9lines_4E_ideal:
\> $(\text{ORBITER}) \text{ -v 2 } \\
\> \> -define Pts -vector -dense \\
\> \> \> $(\text{PTS}_\text{OF}_\text{SURFACE}_\text{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:

\[
x_0 x_0 x_1 + 2x_0 x_1 x_1 + 2x_0 x_1 x_3 \quad \text{and} \quad 2x_2 x_2 x_3 + 2x_2 x_3 x_3.
\]

Let us take the sum of the two polynomials and create the cubic surface:

\[
\text{SURFACE}_F_9 = "x_0 x_0 x_1 - x_0 x_1 x_1 -x_0 x_1 x_3 -x_2 x_2 x_3 - x_2 x_3 x_3"
\]

\[
F_9_{q7}:
\> $(\text{ORBITER}) \text{ -v 3 } \\
\> \> -define F -finite_field -q 7 -end \\
\> \> -define P -projective_space -n 3 -field F -v 0 -end \\
\]
In the next example, we wish to explore the relationship between conics and (5, 2)-arcs. We consider the plane $PG(2, 11)$. 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:

```
rando 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:

```
line type in PG 2 11:
$ORBITER) -v 3
    -orbiter path $(ORBITER_PATH)
    -define F -finite_field -q 11 -end
    -define P -projective_space -n 2 -field F -v 0 -end
    -define C -combinatorial_objects
    -file_of_points random_k_subsets_n133_k5_nb20.csv
    -end
    -with C -do
    -combinatorial_object_activity
    -line_type P random sets
```

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:

```
random arc 5 2 q11 ideal:
$(ORBITER) -v 2
```

272
The ideal is generated by

$$10x_0x_0 + 3x_0x_1 + 8x_0x_2 + 2x_1x_1 + 10x_2x_2.$$

The conic contains the following 12 points:

$\{3, 15, 19, 33, 40, 46, 50, 83, 88, 102, 108\}.$

The next command creates the Endrass surface over $\mathbb{F}_7$. The surface is defined as a makefile variable in sparse form.

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"

Endrass_F7.txt:

```
$\text{(ORBITER) -v 2} \\
$\text{-define F -finite_field -q 7 -end} \\
$\text{-define R -polynomial_ring -field F} \\
$\text{-number_of_variables 4} \\
$\text{-homogeneous_of_degree 8} \\
$\text{-define eqn -vector -field F -sparse 165} \\
$\text{-define P -projective_space -n 3 -field F -v 0} \\
$\text{-define Endrass_F7 -geometric_object P} \\
$\text{-projective_variety R} \\
```
Suppose we want to create the monomials of degree 8 in 4 variables. We use a 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:

```
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 "8,8,1"
  -x_min_global 0 -x_max_global 8
  -end
  -with D -do
  -diophant_activity -solve_mckay
  -end
  sort -r octic_monomials.sol >octic_monomials_sorted.txt
```

There are 165 monomials. They are listed in the file `octic_monomials_sorted.txt`. 
Chapter 9
Applications

9.1 Number Theory

In Table 9.1, some number theoretic commands are shown. For instance,

\[ \text{inverse\_mod\_a:} \]
\[ \text{\$ (ORBITER) -v 2 -inverse\_mod 18059241 58014043} \]

computes the inverse of 18059241 modulo 58014043.

The Legendre symbol tells us 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

\[ \text{jacobi\_a:} \]
\[ \text{\$ (ORBITER) -v 5 -jacobi 2221 7817} \]
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-jacobi</td>
<td>(a), (p)</td>
<td>Computes the Jacobi symbol (\left(\frac{a}{p}\right))</td>
</tr>
<tr>
<td>-sift_smooth</td>
<td>(a), (n), primes</td>
<td>Computes all smooth numbers in the interval ([a, a + n - 1]). Smooth means that they factor completely over the list of primes given.</td>
</tr>
<tr>
<td>-random</td>
<td>(n), (fname)</td>
<td>Creates (n) random numbers and writes them to the csv file (fname)</td>
</tr>
<tr>
<td>-random_last</td>
<td>(n)</td>
<td>Creates (n) random numbers prints the last one</td>
</tr>
<tr>
<td>-affine_sequence</td>
<td>(a), (b), (p)</td>
<td>Splits the interval ([0, p - 1]) into affine sequences of the form (x_{n+1} = ax_n + b \mod p)</td>
</tr>
</tbody>
</table>

Table 9.1: Number Theoretic Commands

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. The steps correspond to the basic rules for computing the Jacobi symbol and can be found in many textbooks. After reformatting, the description looks like this:

\[
\left(\frac{2221}{7817}\right) = \left(\frac{7817}{2221}\right) \cdot \left(\frac{2221}{7817}\right)^{\frac{2221-1}{2}} \\
= \left(\frac{7817}{2221}\right) \cdot \left(\frac{2221}{7817}\right)^{\frac{2221-1}{2}} \\
= \left(\frac{7817}{2221}\right) \cdot \left(\frac{1154}{2221}\right) \\
= \left(\frac{2}{2221}\right) \cdot \left(\frac{577}{2221}\right) \\
= \left(\frac{2}{2221}\right) \cdot \left(\frac{577}{2221}\right) \\
= (-1)^{\frac{2221^2-1}{2}} \cdot \left(\frac{577}{2221}\right) \\
= (-1)^{\frac{2221^2-1}{2}} \cdot (-1)^{\frac{577-1}{2}} \cdot \left(\frac{2221}{577}\right) \\
= (-1)^{\frac{2221^2-1}{2}} \cdot \left(\frac{490}{577}\right) \\
= (-1)^{\frac{2221^2-1}{2}} \cdot \left(\frac{490}{577}\right)
\]
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 from Section 3.1 to compute a square root of 2221 modulo 7817 and verify this fact. The command

```
$ (ORBITER) -v 2 -square_root_mod 2221 7817
```

yields that 7634 is a square root. Indeed,

$$7634^2 \equiv 2221 \mod 7817.$$
-orbits_on_polynomials $d$

<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-orbits_on_polynomials</td>
<td>$d$</td>
<td>Computes the representation of the group $G$ on homogeneous polynomials of degree $d$. This is a group theoretic activity as described in Section 5.6. The group $G$ must be constructed first.</td>
</tr>
</tbody>
</table>

Table 9.2: Representation Theory Commands

9.2 Representation Theory

Orbiter has some commands for representations of finite groups. Table 9.2 lists the commands available to classify arcs. The command

```
representation_on_polynomials_of_degree_3:
▷ $(ORBITER) -v 4 \n ▷ ▷ -define G -linear_group -PGL 4 3 -end \n ▷ ▷ -with G -do \n ▷ ▷ -group_theoretic_activity \n ▷ ▷ ▷ -representation_on_polynomials 3 \n ▷ ▷ -end \n ▷ $(ORBITER) -v 2 \n ▷ ▷ -loop L 0 9 1 -draw_matrix \n ▷ ▷ ▷ -input_csv_file PGL_4_3_rep_3_%L.csv \n ▷ ▷ ▷ -box_width 40 -bit_depth 24 -partition 3 20 20 -end \n ▷ ▷ -end_loop
```

creates $G = \text{PGL}(4, 3)$ 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. Figure 9.1 shows the representing matrices for a generating set of size 9.
Figure 9.1: Representation of PGL(4, 3) on cubic polynomials
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-solovay_strassen</td>
<td>$a \ n$</td>
<td>Performs $n$ Solovay / Strassen tests on the number $a$</td>
</tr>
<tr>
<td>-miller_rabin</td>
<td>$a \ n$</td>
<td>Performs $n$ Miller / Rabin tests on the number $a$</td>
</tr>
<tr>
<td>-fermat</td>
<td>$a \ n$</td>
<td>Performs $n$ Fermat tests on the number $a$</td>
</tr>
<tr>
<td>-find_pseudoprime</td>
<td>$a \ n_1 \ n_2 \ n_3$</td>
<td>Computes a pseudoprime which survives $n_1$ Fermat tests, $n_2$ Miller Rabin tests, $n_3$ Solovay Strassen tests</td>
</tr>
<tr>
<td>-find_strong_pseudoprime</td>
<td>$a \ n_1 \ n_2$</td>
<td>Computes a pseudoprime which survives $n_1$ Fermat tests and $n_2$ Miller Rabin tests</td>
</tr>
<tr>
<td>-RSA_encrypt_text</td>
<td>$d \ n \ b \text{ text}$</td>
<td>Using blocks of $b$ letters at a time, encrypt “text” using RSA with exponent $d$ modulo $n$</td>
</tr>
<tr>
<td>-RSA</td>
<td>$d \ n \ list-of-integers$</td>
<td>encrypt the given sequence of integers using RSA with exponent $d$ modulo $n$</td>
</tr>
</tbody>
</table>

Table 9.3: Cryptographic Commands

### 9.3 Cryptography

In Table 9.3, some global cryptographic commands are shown. Some cryptographic commands require a finite field and appear as a finite field activity, see Table 9.4. For instance,

```bash
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

```bash
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
```

280
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-EC_add</td>
<td>a b i₁ i₂</td>
<td>On the elliptic curve $y^2 \equiv x^3 + ax + b$ in $\mathbb{F}_q$, add the points with indices $i₁$ and $i₂$, each given as a pair $x, y$</td>
</tr>
<tr>
<td>-EC_points</td>
<td>a b</td>
<td>Computes all points of the elliptic curve $y^2 \equiv x^3 + ax + b$ over $\mathbb{F}_q$</td>
</tr>
<tr>
<td>-EC_multiple_of</td>
<td>a b pt n</td>
<td>Computes the $n$ fold multiple of the given point pt on the elliptic curve $y^2 \equiv x^3 + ax + b$ over $\mathbb{F}_q$</td>
</tr>
<tr>
<td>-EC_cyclic_subgroup</td>
<td>a b pt</td>
<td>Computes the cyclic subgroup generated by the given point pt on the elliptic curve $y^2 \equiv x^3 + ax + b$ over $\mathbb{F}_q$</td>
</tr>
<tr>
<td>-EC_Koblitz_encoding</td>
<td>a b s pt plain</td>
<td>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</td>
</tr>
<tr>
<td>-EC_bsgs</td>
<td>a b pt n cipher</td>
<td>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</td>
</tr>
<tr>
<td>-EC_bsgs_decode</td>
<td>a b pt n cipher</td>
<td>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”</td>
</tr>
<tr>
<td>-EC_discrete_log</td>
<td>a b pt base-pt</td>
<td>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 $\mathbb{F}_q$</td>
</tr>
<tr>
<td>-NTRU_encrypt</td>
<td>N p H R M</td>
<td>NTRU encryption for the message $M(X)$ using the public key $H(X)$ and one-time-key $R(X)$.</td>
</tr>
<tr>
<td>-polynomial_center_lift</td>
<td>A(X)</td>
<td>Compute the center lift mod $q$ for the coefficients of $A$</td>
</tr>
<tr>
<td>-polynomial_reduce_mod_p</td>
<td>p A(X)</td>
<td>Reduce the coefficients of the polynomial $A$ modulo $p$</td>
</tr>
</tbody>
</table>

Table 9.4: Finite Field Activities related to Cryptography
computes the cyclic subgroup generated by the point \((1, 4)\) on the curve \(y^2 = x^3 + x + 3 \mod 11\). The command

\[
\text{EC\_points\_199:}
\]
\[
\text{\&\& \$\text{(ORBITER)} -v 2 \}
\]
\[
\text{\&\& \&\& -\text{define F -finite\_field -q 199 -end \}
\]
\[
\text{\&\& \&\& -with F -do \}
\]
\[
\text{\&\& \&\& -finite\_field\_activity \}
\]
\[
\text{\&\& \&\& -EC\_points "EC\_5\_7\_q199" 5 7 -end}
\]
\[
\text{\&\& \$\text{(ORBITER)} -v 2 \}
\]
\[
\text{\&\& \&\& -draw\_matrix -input\_csv\_file EC\_5\_7\_q199\_points\_xy.csv \}
\]
\[
\text{\&\& \&\& -box\_width 10 -bit\_depth 24 \}
\]
\[
\text{\&\& \&\& -partition 2 199 199 -end}
\]

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 shown in Figure 9.2. Both the \(x\)-axis and the \(y\)-axis are indexed by the field elements from 0 to 198.

The command

\[
\text{Figure 9.2: The elliptic curve } y^2 = x^3 + 5x + 7 \mod 199
\]
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 \( i \)th 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

**EC.bsGs**:  
\[
\text{EC.Koblitz_encoding:} \quad \text{EC.bsGs:} \\
\quad \text{encode the message “DEADBEEF” on the curve } y^2 = x^3 + 5x + 7 \text{ mod 199 using the base point (147, 164) and the secret key 67. The } i \text{th input character is encoded as two points } (R_i, T_i) \text{ on the curve using the Elgamal scheme. A random round key is generated for each plaintext symbol. As seen in this example, the -seed command can be used to seed the random number generator with an arbitrary integer (here 17).}
\]

**EC.bsGs**:  
\[
\text{The command:} \\
\quad \text{EC.Koblitz_encoding:} \\
\quad \text{-seed 17} \\
\quad \text{define F -finite_field -q 199 -end} \\
\quad \text{-with F -do} \\
\quad \text{-finite_field_activity} \\
\quad \text{-EC.Koblitz_encoding 5 7 67 "147,164" "DEADBEEF"} \\
\quad \text{-end}
\]

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

**EC.bsGs_decode**:  
\[
\text{EC.bsGs_decode:} \\
\quad \text{EC.bsGs_decode 5 7 "129,176" 212} \\
\quad \text{"127,188,51,141,85,29,106,90,41,105,179,71,171,2,16,197,183,72,27,129,37,10"} \\
\quad \text{"50,179,169,13,153,169,115,116,188,110,176"} \\
\quad \text{-end}
\]
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 [35]). 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 \(A(X) = a_0 + a_1X + \cdots + a_dX^d\) 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 \(-\frac{p}{2} \leq a_i \leq \frac{p}{2}\). So, in an extension to the conventions 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

\begin{verbatim}
NTRU_N=7
NTRU_P=3
NTRU_Q=41
NTRU_D=2
NTRU_XN1="-1,0,0,0,0,0,0,1,"
ALICE_PRIVATE_F="-1,0,1,1,-1,0,1"
ALICE_PRIVATE_G="0,-1,-1,0,1,0,1"
\end{verbatim}

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 9.1. The following two makefile commands do the job:

NTRU_Alice1:
- $(ORBTER) -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

ALICE_PRIVATE_FQ="37,2,40,21,31,26,8"

NTRU_Alice2:
- $(ORBTER) -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.

ALICE_PRIVATE_FP="1,1,1,1,0,2,1"

There is a chance that the polynomial $f(x)$ 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:

NTRU_Alice_public_key:
- $(ORBTER) -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
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).

BOB_MESSAGE="1,-1,1,1,0,-1"

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:

NTRU_encrypt:
    $(ORBITER) -v 2 \
    -define F -finite_field -q $(NTRU_Q) -end \
    -with F -do \n    -finite_field_activity \n    -NTRU_encrypt $(NTRU_N) $(NTRU_P) $(ALICE_PUBLIC_KEY) \n    $(BOB_ONE_TIME_KEY) $(BOB_MESSAGE) -end

BOB_ENCRYPT= "25,3,40,2,4,19,31"

Decryption is done in five steps.

NTRU_decrypt1:
    $(ORBITER) -v 2 \
    -define F -finite_field -q $(NTRU_Q) -end \
    -with F -do \n    -finite_field_activity \n    -polynomial_mult_mod $(ALICE_PRIVATE_F) \n    $(BOB_ENCRYPT) $(NTRUE_XN1) \n    -end

ALICE_C1="40,1,40,40,33,10,1"

NTRU_decrypt2:
    $(ORBITER) -v 2 \
    -define F -finite_field -q $(NTRU_Q) -end \
    -with F -do \n    -finite_field_activity \n    -polynomial_mult_mod $(ALICE_PRIVATE_F) \n    $(BOB_ENCRYPT) $(NTRUE_XN1) \n    -end

286
Decryption produces Bob’s message to Alice.

ToDo:

- RSA
- sqrt mod

Decryption produces Bob’s message to Alice.

ToDo:

- RSA
- sqrt mod
• quadratic sieve
• pseudoprimes
Chapter 10
Coding Theory

10.1 Introduction

Orbiter supports research in coding theory. Global Orbiter commands for coding theory are summarized in Table 10.1. Additional commands, associated with objects of type code will be discussed below and in later sections.

The command

```bash
Allen_Gates_noise_1_percent:
  > $(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 Figure 10.1. The original is on the left. The effect of noise can be seen on the right. The picture shows Paul Allen and Bill Gates in the early 1970s.

The command

```bash
Hamming_space_4_2_distance_matrix:
  > $(ORBITER) -Hamming_space_distance_matrix 4 2
```

creates the distance matrix of the Hamming graph $H(n,q)$. The data is written to the file `Hamming_n4_q2.csv`. The command

```bash
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
```
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-make_macwilliams_system</td>
<td>q n k</td>
<td>Create the MacWilliams equations for the weight enumerator of the dual code.</td>
</tr>
<tr>
<td>-table_of_bounds</td>
<td>n_max q</td>
<td>Make a table of bounds for q-ary linear code for all ( k \leq n \leq n_{\text{max}} )</td>
</tr>
<tr>
<td>-make_bounds_for_d_given_n_and_k_and_q</td>
<td>n k q</td>
<td>Make bounds for the minimum distance of a ([n,k]_q) code</td>
</tr>
<tr>
<td>-Hamming_space_distance_matrix</td>
<td>n q</td>
<td>Make the distance matrix of the Hamming graph ( H(n,q) ).</td>
</tr>
<tr>
<td>-random_noise_in_bitmap_file</td>
<td>f1 f2 n d</td>
<td>Apply random noise at the ( d/n ) level to the bitmap file ( f1 ) and write to ( f2 ).</td>
</tr>
<tr>
<td>-introduce_errors</td>
<td>CRC-options</td>
<td>Introduce errors to a file. See Table 10.6.</td>
</tr>
<tr>
<td>-check_errors</td>
<td>CRC-options</td>
<td>Find errors in a CRC coded file. See Table 10.6.</td>
</tr>
<tr>
<td>-extract_block</td>
<td>CRC-options</td>
<td>Extract a block from a CRC coded file. See Table 10.6.</td>
</tr>
</tbody>
</table>

Table 10.1: Global Coding Theoretic Commands

Figure 10.1: Random noise at the 1% level
Figure 10.2: The color-coded distance matrix of the Hamming graph $H(4,2)$

The command

```
$ (ORBITER) -v 2
$ -partition 4 16 16 \
$ -end
$ open Hamming_n4_q2_draw.bmp
```

produces the bitmap graphic `Hamming_n4_q2_draw.bmp` shown in Figure 10.2.

The command

```
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:
For examples concerning the bounds, see Section 10.8.

Tables 10.2 and 10.3 list coding theoretic activities in Orbiter. Depending on the activity, an object of type code or an object of type finite field is required.

The following command creates the $[5, 2]_2$ code whose codewords are $\{0, 7, 25, 30\}$:

```
CODE_5_2_3_CODEWORDS="0,7,25,30"
```

code_5_2_3_diagram:
```
$\text{ORBITER} -v 2 \$
```
```
\text{-define} F \text{-finite_field} -q 2 \text{-end} \$
```
```
\text{-with} F \text{-do -coding_theoretic_activity} \$
```
```
\text{-code_diagram "code_5_2_3"} \$
```
```
\text{-metric_balls 1} \$
```
```
\text{-end} \$
```
```
$\text{ORBITER} -v 2 \$
```
```
\text{-draw_matrix} \$
```
```
\text{-input_csv_file code_5_2_3_diagram.01_5_4.csv} \$
```
```
\text{-box_width 25 -bit_depth 24} \$
```
```
\text{-partition 4 8 4} \$
```
```
\text{-end} \$
```

The Hamming graph $H(5, 2)$ can be created with the following command:

```
Hamming_5_2_graph:
```
```
$\text{ORBITER} -v 2 \$
```
```
\text{-define} G \text{-graph -Hamming 5 2 \text{-end} \$
```
```
\text{-with} G \text{-do \$
```
```
\text{-graph_theoretic_activity -export_csv \text{-end} \$
```
```
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-general_code_binary</td>
<td>n text</td>
<td></td>
</tr>
<tr>
<td>-code_diagram</td>
<td>label codewords n</td>
<td></td>
</tr>
<tr>
<td>-code_diagram_from_file</td>
<td>fname</td>
<td></td>
</tr>
<tr>
<td>-enhance</td>
<td>radius</td>
<td></td>
</tr>
<tr>
<td>-metric_balls</td>
<td>radius</td>
<td></td>
</tr>
<tr>
<td>-long_code</td>
<td>n generators</td>
<td></td>
</tr>
<tr>
<td>-encode_text_5bits</td>
<td>input fname</td>
<td></td>
</tr>
<tr>
<td>-field_induction</td>
<td>fname-in fname-out nb-bits</td>
<td></td>
</tr>
<tr>
<td>-weight Enumerator</td>
<td>matrix</td>
<td>Compute the complete weight enumerator of the linear code generated by the $m \times n$ matrix $L$</td>
</tr>
<tr>
<td>-minimum_distance</td>
<td>code-object-label</td>
<td>Compute the minimum distance of the linear code object.</td>
</tr>
<tr>
<td>-generator_matrix_cyclic_code</td>
<td>n poly</td>
<td></td>
</tr>
<tr>
<td>-nth_roots</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>-NTT</td>
<td>n q</td>
<td>Create Number theoretic transform of dimension $n$ over $\mathbb{F}_q$.</td>
</tr>
<tr>
<td>-fixed_code</td>
<td>perm</td>
<td>Create the subcode fixed by the given permutation of the positions of the code.</td>
</tr>
<tr>
<td>-export_magma</td>
<td>fname</td>
<td></td>
</tr>
<tr>
<td>-export_codewords</td>
<td>fname</td>
<td></td>
</tr>
<tr>
<td>-export_codewords_by_weight</td>
<td>fname</td>
<td></td>
</tr>
<tr>
<td>-export_genma</td>
<td>fname</td>
<td></td>
</tr>
<tr>
<td>-export_checkma</td>
<td>fname</td>
<td></td>
</tr>
</tbody>
</table>

Table 10.2: Coding Theoretic Activities (Part I)
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-crc32</td>
<td>text</td>
<td></td>
</tr>
<tr>
<td>-crc32_hexdata</td>
<td>hexdata</td>
<td></td>
</tr>
<tr>
<td>-crc32_test</td>
<td>block-length</td>
<td></td>
</tr>
<tr>
<td>-crc256_test</td>
<td>message-length R k</td>
<td></td>
</tr>
<tr>
<td>-crc32_remainders</td>
<td>msg-length</td>
<td></td>
</tr>
<tr>
<td>-crc_encode_file_based</td>
<td>frame-in frame-out block-length crc_type block_length</td>
<td></td>
</tr>
<tr>
<td>-find_CRC_polynomials</td>
<td>nb-errors info-bits check-bits</td>
<td></td>
</tr>
<tr>
<td>-write_code_for_division</td>
<td>fname A B</td>
<td></td>
</tr>
<tr>
<td>-polynomialdivision_from_file</td>
<td>fname r1</td>
<td></td>
</tr>
<tr>
<td>-polynomialdivision_from_file_all_k_bit_error_patterns</td>
<td>fname r1 k</td>
<td></td>
</tr>
</tbody>
</table>

Table 10.3: Coding Theoretic Activities (Part II)
Figure 10.3: Drawing of the Hamming graph $H(5, 2)$

Using the unix dot program, this command sequence creates the drawing of $H(5, 2)$ shown in Figure 10.3.
### Table 10.4: Commands to create codes

<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-field</td>
<td>$F$</td>
<td>Specify the field of definition.</td>
</tr>
<tr>
<td>-linear_code_through_generator_matrix</td>
<td>$M$</td>
<td>Create a code defined by a generator matrix.</td>
</tr>
<tr>
<td>-linear_code_from_projective_set</td>
<td>$nmk \ S$</td>
<td>Create a code defined by a projective set in the dual.</td>
</tr>
<tr>
<td>-linear_code_by_columns_of_parity_check</td>
<td>$nmk \ M$</td>
<td>Create a code defined by an affine set in the dual.</td>
</tr>
<tr>
<td>-first_order_Reed_Muller</td>
<td>$m$</td>
<td>Create a first order Reed-Muller code of degree $m$.</td>
</tr>
<tr>
<td>-BCH</td>
<td>$n \ d$</td>
<td>BCH code of length $n$ with prescribed minimum distance $d$.</td>
</tr>
<tr>
<td>-Reed_Solomon</td>
<td>$n \ d$</td>
<td>Not yet implemented.</td>
</tr>
<tr>
<td>-Gilbert_Varshamov</td>
<td>$n \ k \ d$</td>
<td>Create a Gilbert-Varshamov code of length $n$ with dimension $k$ and minimum distance at least $d$.</td>
</tr>
</tbody>
</table>

### Table 10.5: Code modifications

<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-dual</td>
<td></td>
<td>Compute the dual code.</td>
</tr>
</tbody>
</table>

### 10.2 Linear Codes

In this section, we will see how linear codes can be created and studied in Orbiter. A code object is used to represent a specific code. Table 10.4 list the commands to create a code object. Table 10.5 lists code modifications. These are commands used to create a new code from an old one.

The following command creates the first order Reed-Muller code in three variables:

```
RM_3_1:
   ▶ $(ORBITER) -v 2 \
   ▶ ▶ -define F -finite_field -q 2 -end \
   ▶ ▶ -define C -code -field F \
   ▶ ▶ ▶ -first_order_Reed_Muller 3 \n   ▶ ▶ -end \
   ▶ ▶ -with C -and F -do -coding_theoretic_activity \n```
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:

```
SIMPLEX_CODE_GENERATOR="\n1,0,1,0,1,0,1, \n0,1,1,0,0,1,1, \n0,0,0,1,1,1,1"
```

The following command computes the nullspace of this matrix, which is the Hamming code:

```
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 
▷ ▷ ▷ -linear_code_through_generator_matrix v 
▷ ▷ -end
```

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:

```bash
$ (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 \
  -linear_code_through_generator_matrix v \
  -dual \
  -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:

```magma
K<w> := GF(2);
V := VectorSpace(K, 7);
C := LinearCode(sub<V | [1,1,0,0,0,0,0], [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.

```bash
$ (ORBITER) -v 2 \
  -define F -finite_field -q 2 -end \
  -define C -code -field F -first_order_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:

```
CODE_RM_3_1_GENMA="\n11111111\n01010101\n00110011\n00001111"
```

The following command creates the Hamming code from its generator matrix directly:

```
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 \\
▷ ▷ ▷ -linear_code_through_generator_matrix genma \\
▷ ▷ -end \\
#pdflatex code_n8_k4_q2.tex
#open code_n8_k4_q2.pdf
```

The following command creates the Hamming code and produces a list of codewords.

```
RM_3_1_and_codewords:
▷ $(ORBITER) -v 2 \\
▷ ▷ -define F -finite_field -q 2 -end \\
▷ ▷ -define C -code -field F -first_order_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
```

The Hamming code is cyclic. To see this, we need to consider the action of the Singer cycle on the set of points of PG(2, 2). The following command creates the Singer cycle:

```
```

299
Figure 10.4: The Hamming code

Hamming.singer:
\[
\begin{array}{c}
\text{\$} \text{ORBITER} \text{ -v 3 \ }\\
\text{\$} \text{define G -linear_group -PGL 3 2 -singer 1 -end \ }\\
\text{\$} \text{define Orb -orbits -group G \ }\\
\text{\$} \text{on_points \ }\\
\text{\$} \text{end}
\end{array}
\]
\[
\text{#pdflatex PGL_3_2.Singer_3_2_1_report.tex}
\]
\[
\text{#open PGL_3_2.Singer_3_2_1_report.pdf}
\]

This produces the following output:

Strong generators for a group of order 7:

\[
\begin{bmatrix}
0 & 1 & 0 \\
0 & 0 & 1 \\
1 & 0 & 1
\end{bmatrix}
\]
Basic Orbit 0

Basic orbit 0 has size 7
0, 1, 2, 3, 4, 5, 6

From this, we know how to rearrange the points of PG(2, 2) to exhibit the cyclic structure. We issue the following command to recreate the Hamming code:

```
SIMPLEX_CODE_GENMA_CYCLIC="\n1,0,0,1,1,1,0, \
0,1,0,0,1,1,1, \
0,0,1,1,1,0,1"

Hamming cyclic generator:
▷ $(ORBITER) -v 2 \\
▷ ▷ -define F -finite_field -q 2 -end \\
▷ ▷ -define v -vector -format 3 -field F \\
▷ ▷ ▷ -dense $(SIMPLEX_CODE_GENMA_CYCLIC) \\
▷ ▷ -end \\
▷ ▷ -with F -do -finite_field_activity \\
▷ ▷ -nullspace v \\
▷ -end
▷ pdflatex nullspace_3_7.tex
▷ open nullspace_3_7.pdf
```

This produces the following output:
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:

```
HAMMING_CODE_GENERATOR="\n1,0,0,0,0,1,1, \n0,1,0,0,1,0,1, \n0,0,1,0,1,1,0, \n0,0,0,1,1,1,1"
```

The next command computes the weight enumerator:

```
Hamming weight enumerator:
\$ (ORBITER) -v 2 \n  -define F -finite_field -q 2 -end \n  -define v -vector -field F -format 4 \n  -dense $(HAMMING_CODE_GENERATOR) \n  -end \n  -define C -code -field F \n  -linear_code_through_generator_matrix v \n  -end \n  -with C -do \n  -coding_theoretic_activity \n  -weight Enumerator \n  -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:

```
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:

```
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:

GOLAY_23COLUMN_RANKS_PROJECTIVELY="0, 1, 2, 3, 4, 5, 6, 7, \n8, 9, 10, 11, 132, 913, 1460, 1750, 1898, 2518, 2787, 2874, \n3320, 3357, 3662"

Suppose we want to list the code words. The following command can be used:

Golay23.code_words:
  $ (ORBITER) -v 2 \\
  -define v -vector -dense $(GOLAY_23COLUMN_RANKS_PROJECTIVELY) -end \\
  -define F -finite_field -q 2 -end \\
  -define C -code -field F \\
  -linear_code_from_from_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
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-input</td>
<td>fname</td>
<td>Input file name.</td>
</tr>
<tr>
<td>-output</td>
<td>fname</td>
<td>Output file name.</td>
</tr>
<tr>
<td>-block_length</td>
<td>L</td>
<td>Set block length to $L$ field elements.</td>
</tr>
<tr>
<td>-block_based_error_generator</td>
<td></td>
<td>Apply block-based error generator.</td>
</tr>
<tr>
<td>-file_based_error_generator</td>
<td>threshold</td>
<td>Apply file-based error generator.</td>
</tr>
<tr>
<td>-nb_repeats</td>
<td>N</td>
<td>Set the number of repeats to $N$.</td>
</tr>
<tr>
<td>-threshold</td>
<td>t</td>
<td>Set probability of error per experiment to $t/1000000$.</td>
</tr>
<tr>
<td>-error_log</td>
<td>fname</td>
<td>Set file name for error logging.</td>
</tr>
<tr>
<td>-selected_block</td>
<td>i</td>
<td>Set block number.</td>
</tr>
</tbody>
</table>

Table 10.6: CRC options

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 10.6 summarizes options associated with commands for CRC-codes.

Here is an example. We consider a short string of English text and encode it with 5 bits per character. This is done using the `-encode_text_5bits` command. The encoded text is stored in a csv file, which we decide to call `text.csv`.

```
encode_text_5bits:
  $(ORBITER) -encode_text_5bits \
  "Hithere" "text.csv" \
  $(ORBITER) -v 2 \
  -define F -finite_field -q 2 -end \
  -with F -do \
  -coding_theoretic_activity \
  -polynomial_division_from_file \
  text.csv 13 -end \
  pdflatex polynomial_division_file_13.tex \
  open polynomial_division_file_13.pdf
```
We decide to pick the binary polynomial $13 = X^3 + X^2 + 1$. We divide the information polynomial by the check polynomial:

```
encode_text_5bits_check:
  $ (ORBITER) -v 2 \
  \> -define F -finite_field -q 2 -end \n  \> -with F -do \n  \> -coding_theoretic_activity \n  \> \> -polynomial_division_from_file \n  \> \> text_with_1error.csv 13 \n  \> \> -end
  \> pdflatex polynomial_division_file_13.tex
  \> open polynomial_division_file_13.pdf
```

This creates the following output:

```
text.csv / 13 =
1010110100110101010111000010111100 / 1101 = 11011011100111111101000011011100010111100

1101 | 1010110100110101010111000010111100
  1101
  ===
  1111010101010111000010111100
  1101
  ===
  1010101101010111000010111100
  1101
  ===
  111100101010111000010111100
  1101
  ===
  10001101010111000010111100
  1101
  ===
  101110101010111000010111100
  1101
  ===
  110101010111000010111100
  1101
  ===
  101010111000010111100
  1101
  ===
  11110111000010111100
```

The remainder after division by the check polynomial is 5, or the polynomial $X^2 + 1$, or the bit-sequence 101.

The following command investigates all 1-bit errors, to see which of them can be detected using the given CRC-polynomial:

```
encode_text_5bits_1error:
  $(ORBITER) -encode_text_5bits \n  "Hithere" "text.csv"
  $(ORBITER) -v 2 \n  -define F -finite_field -q 2 -end \n  -with F -do \n```
\[ \text{-coding_theoretic_activity} \]
\[ \text{-polynomial_division_from_file_all_k_bit_error_patterns} \]
\[ \text{text.csv 13 1} \]
\[ \text{-end} \]

```
open polynomial_division_file_all_1_error_patterns_13.pdf
```

The following output is created:

```
: 0101011010011010101010111000010111100
0 : 01010110011010101010111000010111101 : 100 : 4 : X^{2}
1 : 01011010011010101011011000010111101 : 111 : 7 : X^{2} + X + 1
2 : 010101101010101010101011000010111100 : 001 : 1 : 1
3 : 01011010101011010101011000010111100 : 000 : 0 : 0
4 : 01010110101010101011011000010111100 : 010 : 2 : X
5 : 01010110101010101011011000010111100 : 110 : 6 : X^{2} + X
6 : 01010110101010101011011000010111100 : 011 : 3 : X + 1
7 : 01010110101010101011011000010111100 : 100 : 4 : X^{2}
8 : 01010110101010101011011000010111100 : 111 : 7 : X^{2} + X + 1
9 : 01010110101010101011011000010111100 : 001 : 1 : 1
10 : 01010110101010101011011000010111100 : 000 : 0 : 0
11 : 01010110101010101011011000010111100 : 010 : 2 : X
13 : 01010110101010101011011000010111100 : 011 : 3 : X + 1
14 : 01010110101010101011011000010111100 : 100 : 4 : X^{2}
15 : 01010110101010101011011000010111100 : 111 : 7 : X^{2} + X + 1
16 : 01010110101010101011011000010111100 : 001 : 1 : 1
17 : 01010110101010101011011000010111100 : 000 : 0 : 0
18 : 01010110101010101011011000010111100 : 010 : 2 : X
20 : 01010110101010101011011000010111100 : 011 : 3 : X + 1
21 : 01010110101010101011011000010111100 : 100 : 4 : X^{2}
22 : 01010110101010101011011000010111100 : 111 : 7 : X^{2} + X + 1
23 : 01010110101010101011011000010111100 : 001 : 1 : 1
24 : 01010110101010101011011000010111100 : 000 : 0 : 0
25 : 01010110101010101011011000010111100 : 010 : 2 : X
26 : 01010110101010101011011000010111100 : 110 : 6 : X^{2} + X
27 : 01010110101010101011011000010111100 : 011 : 3 : X + 1
28 : 01010110101010101011011000010111100 : 100 : 4 : X^{2}
29 : 01010110101010101011011000010111100 : 111 : 7 : X^{2} + X + 1
30 : 01010110101010101011011000010111100 : 001 : 1 : 1
31 : 01010110101010101011011000010111100 : 000 : 0 : 0
32 : 01010110101010101011011000010111100 : 010 : 2 : X
33 : 01010110101010101011011000010111100 : 110 : 6 : X^{2} + X
```

308
It shows that 5 single bit errors are undetected.

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 \( n = 128 \).

\[
\text{CRC}\_3\_128\_10:
\]
\[
\triangleright \ $(\text{ORBITER})$ -v 1 \\
\triangleright \triangleright -define F -finite_field -q 2 -end \\
\triangleright \triangleright -with F -do -coding_theoretic_activity \\
\triangleright \triangleright \triangleright -find_CRC_polynomials 3 128 10 \\
\triangleright \triangleright -end
\]

The program finds 244 polynomials in about 1 minute.

Here is a collection of CRC polynomials from various sources:

CRC4="1,4,1,2,1,1,1,0"

CRC7="1,7,1,3,1,0"

CRC8\_ATM="1,8,1,2,1,1,1,0"

CRC16\_CCITT="1,16,1,12,1,5,1,0"

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"

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"

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"

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:

```
crc32_Berlekamp_matrix:
  $(ORBITER) -v 2 \n  -define F -finite_field -q 2 -end \n  -define v -vector -field F -sparse 33 $(CRC32 ETHERNET) -end \n  -with F -do \n  -finite_field_activity \n  -Berlekamp_matrix v \n  -end
```

Now, we create some new CRC polynomials over the field $\mathbb{F}_{256}$. To begin with, we create the 771st roots over $\mathbb{F}_{256}$:

```
CRC_F256_roots_771:
  $(ORBITER) -v 3 \n  -define F -finite_field -q 256 -end \n  -with F -do -coding_theoretic_activity \n  -nth_roots 771 \n  -end
```

We create a BCH code of length 771 over $\mathbb{F}_{256}$ with designed distance 2:

```
CRC_F256_BCH_code_d2:
  $(ORBITER) -v 2 \n  -define F -finite_field -q 256 -end \n  -define C -code -field F \n  -BCH 771 2 \n  -end \n  -with C -and F -do -coding_theoretic_activity \n  -export_magma BCH_lq8_n771_d2.magma \n  -end
  pdflatex BCH_codes_q256_n771_d2.tex
  open BCH_codes_q256_n771_d2.pdf
```

The polynomial in dense coding

```
CRC_POLY_Q256_DEG2_DENSE="214,167,1"
```

We generate C++ source code for the use of this polynomial:

```
CRC_F256_BCH_write_code_for_division_d2:
  $(ORBITER) -v 2 \n```

310
We create a BCH code of length 771 over $\mathbb{F}_{256}$ with designed distance 16:

\begin{verbatim}
F256_BCH_code_d16:
$\$(ORBITER) -v 3 \$
$\$(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
\end{verbatim}

The polynomial in sparse coding is:

\begin{verbatim}
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"
\end{verbatim}

The polynomial in dense coding is:

\begin{verbatim}
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"
\end{verbatim}

We generate C++ source code for the use of this polynomial:

\begin{verbatim}
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 \$
\end{verbatim}
We confirm that the polynomial divides $X^{771} - 1$ as it should:

\[
F_{256} \text{BCH code division:
}\]
\[
$(ORBITER) -v 2 \\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
\>
10.5 Reed-Muller Codes

The following command creates the Reed Muller code $\text{RM}_{3,1}$.

\[ \text{RM\_3\_1\_Hamming\_space\_diagram:} \]
\[ \text{\textit{\$(ORBITER) -v 2 \}} \]
\[ \text{\textit{\quad -define F -finite\_field -q 2 -end \}} \]
\[ \text{\textit{\quad -with F -do \}} \]
\[ \text{\textit{\quad -coding\_theoretic\_activity \}} \]
\[ \text{\textit{\quad -code\_diagram "RM\_3\_1" \}} \]
\[ \text{\textit{\quad $(REED\_MULLER\_3\_1\_CODEWORDS) 8 \}} \]
\[ \text{\textit{\quad -metric\_balls 1 \}} \]
\[ \text{\textit{\quad -end \}} \]

The following command produces a diagram of the characteristic function of the code in the Hamming space $H(8, 2)$, shown in Figure 10.5. The different codewords are given different colors.

\[ \text{RM\_3\_1\_draw:} \]
\[ \text{\textit{\$(ORBITER) -v 2 \}} \]
\[ \text{\textit{\quad -draw\_matrix \}} \]
\[ \text{\textit{\quad -input\_csv\_file RM\_3\_1\_holes\_8\_16.csv \}} \]
\[ \text{\textit{\quad -box\_width 25 -bit\_depth 8 \}} \]
\[ \text{\textit{\quad -partition 4 16 16 \}} \]
\[ \text{\textit{\quad -end \}} \]
\[ \text{\textit{\$(ORBITER) -v 2 \}} \]
\[ \text{\textit{\quad -draw\_matrix \}} \]
\[ \text{\textit{\quad -input\_csv\_file RM\_3\_1\_diagram\_01\_8\_16.csv \}} \]
\[ \text{\textit{\quad -box\_width 25 -bit\_depth 8 \}} \]
\[ \text{\textit{\quad -partition 4 16 16 \}} \]
\[ \text{\textit{\quad -end \}} \]
\[ \text{\textit{\$(ORBITER) -v 2 \}} \]
\[ \text{\textit{\quad -draw\_matrix \}} \]
\[ \text{\textit{\quad -input\_csv\_file RM\_3\_1\_diagram\_8\_16.csv \}} \]
\[ \text{\textit{\quad -box\_width 25 -bit\_depth 8 \}} \]
\[ \text{\textit{\quad -partition 4 16 16 \}} \]
\[ \text{\textit{\quad -end \}} \]
\[ \text{\textit{\texttt{open RM\_3\_1\_diagram\_8\_16\_draw.bmp} \}} \]

313
Figure 10.5: Boolean function representation of $\text{RM}_{3,1}$ in $H(8, 2)$
10.6 BCH Codes

Let $\beta$ be an $n$-th root of unity over $\mathbb{F}_q$. The minimum polynomial of $\beta$ 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

$$\text{lcm}(m_{\beta^1, \mathbb{F}_q}, m_{\beta^2, \mathbb{F}_q}, \ldots, m_{\beta^{d-1}, \mathbb{F}_q}).$$

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

```
$($\text{ORBITER}$)$ -v 2 \\
$>$ -define \$F -finite_field -q 8 -override_polynomial 11 -end \\
$>$ -define \$C -code -field \$F \\
$>$ -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 in Figure 10.6. We will try BCH-codes with minimum distances 3, 5 and 7. Here is distance 3:

```
$($\text{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 is described in a latex output file:

```
BCH-code:
n = 21, \ k = 17, \ d_0 = 3, \ q = 8, 
g(x) = m_1m_2 = X^4 + 4X^3 + 4X^2 + 3X + 4 
Chosen cyclotomic sets:
```

315
Figure 10.6: The 8-cyclotomic sets modulo 21

\{ 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:

\[
\begin{bmatrix}
4 & 3 & 4 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 4 & 3 & 4 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 4 & 3 & 4 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 4 & 3 & 4 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 4 & 3 & 4 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 4 & 3 & 4 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 4 & 3 & 4 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 3 & 4 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 3 & 4 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 3 & 4 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 3 & 4 & 4 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 3 & 4 & 4 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 3 & 4 & 4 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 3 & 4 & 4 & 1 & 0 & 0
\end{bmatrix}
\]

And now for \( d = 5 \):

\begin{verbatim}
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
\end{verbatim}

The output file is:

\begin{verbatim}
BCH-code:
\( n = 21, \, k = 14, \, d_0 = 5, \, q = 8, \)  
\( g(x) = m_1m_2m_3m_4 = X^7 + 3X^6 + 3X^5 + 2X^4 + X^3 + 2X^2 + X + 2 \)  
Chosen cyclotomic sets:
\{ 1, 8 \}
\{ 2, 16 \}
\{ 3 \}
\end{verbatim}

317
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:

\[
\begin{bmatrix}
  2 & 1 & 2 & 1 & 2 & 3 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
  0 & 2 & 1 & 2 & 1 & 2 & 3 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
  0 & 0 & 2 & 1 & 2 & 1 & 2 & 3 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
  0 & 0 & 0 & 2 & 1 & 2 & 1 & 2 & 3 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
  0 & 0 & 0 & 0 & 2 & 1 & 2 & 1 & 2 & 3 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
  0 & 0 & 0 & 0 & 0 & 2 & 1 & 2 & 1 & 2 & 3 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
  0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 & 2 & 1 & 2 & 3 & 3 & 1 & 0 & 0 & 0 & 0 & 0 \\
  0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 & 2 & 1 & 2 & 3 & 3 & 1 & 0 & 0 & 0 & 0 \\
  0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 & 2 & 1 & 2 & 3 & 3 & 1 & 0 & 0 & 0 \\
  0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 & 2 & 1 & 2 & 3 & 3 & 1 & 0 & 0 \\
  0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 & 2 & 1 & 2 & 3 & 3 & 1 & 0 \\
  0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 & 2 & 1 & 2 & 3 & 3 & 1 \\
\end{bmatrix}
\]

We compute the minimum distance:

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_OVERR}\e 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 $d = 7$:

F_8_BCH_code_d7:
▷ $(\text{ORBITER})$ -v 3 \n▷ ▷ -define F -finite_field -q 8 -override_polynomial 11 -end \n▷ ▷ -define C -code -field F \n▷ ▷ ▷ -BCH 21 7 \n▷ ▷ -end

The output file is:

BCH-code:
$n = 21, k = 11, d_0 = 7, q = 8,$
$g(x) = m_1m_2m_3m_4m_5m_6 = X^{10} + X^9 + 2X^8 + 5X^7 + 2X^6 + 4X^4 + 6X^3 + 5X^2 + 6X + 6$
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:

\[
\begin{bmatrix}
6 & 6 & 5 & 6 & 4 & 0 & 2 & 5 & 2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 6 & 6 & 5 & 6 & 4 & 0 & 2 & 5 & 2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 6 & 6 & 5 & 6 & 4 & 0 & 2 & 5 & 2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 6 & 6 & 5 & 6 & 4 & 0 & 2 & 5 & 2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 6 & 6 & 5 & 6 & 4 & 0 & 2 & 5 & 2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 6 & 6 & 5 & 6 & 4 & 0 & 2 & 5 & 2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 6 & 6 & 5 & 6 & 4 & 0 & 2 & 5 & 2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & 6 & 5 & 6 & 4 & 0 & 2 & 5 & 2 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & 6 & 5 & 6 & 4 & 0 & 2 & 5 & 2 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & 6 & 5 & 6 & 4 & 0 & 2 & 5 & 2 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & 6 & 5 & 6 & 4 & 0 & 2 & 5 & 2 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & 6 & 5 & 6 & 4 & 0 & 2 & 5 & 2 & 1 \\
\end{bmatrix}
\]

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.

\begin{verbatim}
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
\end{verbatim}

The drawing is shown in Figure 10.7.

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:

\begin{verbatim}
BCH_F256_roots_771:
  $(ORBITER) -v 3 \
  -define F -finite_field -q 256 -end \
  -with F -do -coding_theoretic_activity \
\end{verbatim}

320
The next command creates a BCH-code of length 771 over $\mathbb{F}_{256}$ with minimum distance at least 16:

```
$($ORBITER$) -v 3 \n$define F -finite_field -q 256 -end \n$define C -code -field F \n$ BCH 771 16 \n$end
```

```
pdflatex BCH\_codes\_q256\_n771\_d16.tex
open BCH\_codes\_q256\_n771\_d16.pdf
```
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 $\mathbb{F}_7$, we use the primitive element $\alpha = 3$. The Reed-Solomon code of designed distance 3 over $\mathbb{F}_7$ is the cyclic code generated by

$$(X - \alpha)(X - \alpha^2) = (X - 3)(X - 2) = X^2 + 2X + 6.$$ 

The generator matrix of the code in cyclic form is

$$\begin{bmatrix}
6 & 2 & 1 & 0 & 0 & 0 \\
0 & 6 & 2 & 1 & 0 & 0 \\
0 & 0 & 6 & 2 & 1 & 0 \\
0 & 0 & 0 & 6 & 2 & 1
\end{bmatrix}.$$ 

Let us investigate this code. We start with the weight enumerator. The command

```bash
CODE_RS_6_4_7="\\
621000 \
062100 \\
006210 \\
000621"
```

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: The Reed Solomon code of designed distance 3 over \( \mathbb{F}_8 \) is the cyclic code generated by

\[(X - \alpha)(X - \alpha^2) = X^2 + 6X + 5.\]

The associated cyclic generator matrix is

\[
\begin{bmatrix}
  5 & 6 & 1 & 0 & 0 & 0 & 0 \\
  0 & 5 & 6 & 1 & 0 & 0 & 0 \\
  0 & 0 & 5 & 6 & 1 & 0 & 0 \\
  0 & 0 & 0 & 5 & 6 & 1 & 0 \\
  0 & 0 & 0 & 0 & 5 & 6 & 1
\end{bmatrix}
\]

We use the makefile variable CODE_RS_8 to hold this generator matrix. The following command computes the weight enumerator

\[
\text{CODE\_RS\_8}="\text{\begin{array}{cccccccc}
5 & 6 & 1 & 0 & 0 & 0 & 0 \\
0 & 5 & 6 & 1 & 0 & 0 & 0 \\
0 & 0 & 5 & 6 & 1 & 0 & 0 \\
0 & 0 & 0 & 5 & 6 & 1 & 0 \\
0 & 0 & 0 & 0 & 5 & 6 & 1
\end{array}}"
\]

which turns out to be

\[
y^7 + 245x^3y^4 + 1225x^4y^3 + 5586x^5y^2 + 12838x^6y + 12873x^7.
\]

Computing the automorphism group of the code is computationally infeasible. The next command performs field reduction on the code. This produces a \([21, 15]_2\) code.
The reduced matrix is shown in Figure 10.8. Let us compute the weight enumerator of the reduced code. The command

```
RS_8_reduced="\n01000110000000000000000\n00111001000000000000000\n11001100100000000000000\n00001000110000000000000\n00000011100100000000000\n00000111100100000000000\n00000011100100000000000\n00000011100100000000000\n00000011100100000000000\n"
RREF_RS_8_reduced_weight_enumerator:
  ▶ $(ORBITER) -v 2 \n  ▶ ▶ -define F -finite_field -q 2 -end \n  ▶ ▶ -define v -vector -format 15 -field F \n  ▶ ▶ ▶ -compact $(RS_8\_reduced) \n  ▶ ▶ -end \n  ▶ ▶ -define C -code -field F \n  ▶ ▶ ▶ -linear_code_through_generator_matrix v \n  ▶ ▶ -end \n  ▶ ▶ -with C -do \n  ▶ ▶ -coding_theoretic_activity \n  ▶ ▶ ▶ -weight Enumerator \n  ▶ ▶ -end

computes the weight enumerator of the binary code. It is

\[ 1y^{21} + 28x^3y^{18} + 84x^4y^{17} + 273x^5y^{16} + 924x^6y^{15} + 1956x^7y^{14} + \\
2982x^8y^{13} + 4340x^9y^{12} + 5796x^{10}y^{11} + 5796x^{11}y^{10} + 4340x^{12}y^{9} + \\
2982x^{13}y^8 + 1956x^{14}y^7 + 924x^{15}y^6 + 273x^{16}y^5 + 84x^{17}y^4 + 28x^{18}y^3 + \\
x^{21} \]

In particular, the field reduced Reed-Solomon code is not optimal. It has minimum distance
three, but there are codes of minimum distance 4. Here is one. We store the code to a file
and then draw the generator matrix as bitmap.

CODE_21_15_4="\n111000100000000000000000 \n110100010000000000000000 \n101100001000000000000000 \n011100000100000000000000 \n110010000010000000000000 \n101010000001000000000000 \n011010000000100000000000 \n
We compute the weight enumerator

\[
\text{1y}^{21} + 221x^4y^{17} + 1600x^6y^{15} + 6498x^8y^{13} + 10912x^{10}y^{11} + 9250x^{12}y^9 + 3584x^{14}y^7 + 669x^{16}y^5 + 32x^{18}y^3 + 1x^{20}y.
\]

This shows that this code is a \([21, 15, 4]_2\). It is optimal.
10.8 Bounds

In coding theory, one main question is to determine the best value of \(d_{\text{max}}\) 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 \(d \geq d_{\text{max}}\) exists. A lower bound tells us that a code with \(d \geq d_{\text{max}}\) exists. The command

\[
\text{bounds\_for\_d\_given\_n15\_k6\_q2:}
\]

\[
\text{
\(\text{\$ (ORBITER) -v 2 \ ( \text{-make\_bounds\_for\_d\_given\_n\_and\_k\_and\_q\ 15\ 6\ 2} \)} \)
}
\]

gives upper and lower bounds on the optimal minimum distance \(d_{\text{max}}\) 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:

\[
d_{\text{GV}} = 5
\]
\[
d_{\text{singleton}} = 10
\]
\[
d_{\text{hamming}} = 6
\]
\[
d_{\text{plotkin}} = 7
\]
\[
d_{\text{griesmer}} = 6
\]

This shows that \(5 \leq d_{\text{max}} \leq 6\). The command

\[
\text{coding\_theory\_bounds\_q2:}
\]

\[
\text{
\(\text{\$ (ORBITER) -v 2 \ -table\_of\_bounds\ 20\ 2} \)
}
\]

produces a table of bounds for binary codes with \(n, k \leq 20\). A file

\[
\text{table\_of\_bounds\_n20\_q2.csv}
\]

is computed. The command

\[
\text{GV\_n15\_k6\_d5:}
\]

\[
\text{
\(\text{\$ (ORBITER) -v 2 \ ( \text{-define\ F \ -finite\ field \ -q 2 \ -end} \ ( \text{-define\ C \ -code \ -field\ F} \ ( \text{-Gilbert\_Varshamov\ 15\ 6\ 5} \ ( \text{-end} \)
}
\]

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:

\[
\begin{array}{cccccccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\
\end{array}
\]
To compute the minimum distance of the code, we do:

```bash
CODE_GV_N15_K6="\n111111111100000\n11110000010000\n11100110000100\n11010101000100\n10101011000010\n10110100100001" 

GV_n15_k6_d5_weight Enumerator:
▷ $(ORBITER) -v 2 \n▷ -define F -finite_field -q 2 -end \n▷ -define v -vector -format 6 -field F \n▷ -compact $(CODE_GV_N15_K6) \n▷ -end \n▷ -define C -code -field F \n▷ -linear_code_through_generator_matrix v \n▷ -end \n▷ -with C -do \n▷ -coding_theoretic_activity \n▷ -weight Enumerator \n▷ -end
```

The weight enumerator is

\[ 1y^{15} + 27x^6y^9 + 24x^8y^7 + 9x^{10}y^5 + 3x^{12}y^3. \]

From this, we see that the code has minimum distance 6, which is better than predicted.
10.9 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 $\text{PG}(r-1,q)$. 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 $\text{PG}(r-1,q)$ for which any $d-1$-subset (if any) is independent. Under the correspondence, the action of $\text{PGL}(r,q)$ 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 $\text{PGL}(r,q)$ 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

```
codes_8_4_4:
  > $(ORBITER) -v 6 \n  > -orbiter_path $(ORBITER_PATH) \n  > -define G \n  > -linear_group -PGL 4 2 -end \n  > -with G -do \n  > -group_theoretic_activity \n  > -poset_classification_control \n  > -problem_label codes_8_4_4 \n  > -draw_options -embedded -radius 250 \n  > -line_width 1.0 -spanning_tree -end \n  > -end \n  > -linear_codes 3 8 \n  > -end
```

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 $\text{PGL}(4,2)$ on the poset $\Lambda_{3,3}(2)$. Using poset classification, Orbiter then produces the poset of orbits shown in Figure 10.9. In this diagram, the numbers stand for Orbiter ranks of points in $\text{PG}(3,2)$. 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 $\text{PG}(3,2)$. We write the
Figure 10.9: Orbits of PGL(4, 2) on the poset $\Lambda_{3,3}(2)$
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 $C$. This means that the words of the code are the vectors $c$ such that $c \cdot H^\top = 0$. 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 $\text{AGL}(3, 2)$, 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

```
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. A pie chart of the class size distribution is shown in Fig. 11.1.

The next command computes the character table of the symmetric group Sym(4):

```
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

\[
\begin{bmatrix}
1 & -1 & 1 & 1 & -1 \\
3 & 1 & 0 & -1 & -1 \\
2 & 0 & -1 & 2 & 0 \\
3 & -1 & 0 & -1 & 1 \\
1 & 1 & 1 & 1 & 1
\end{bmatrix}
\]
```

The following command creates the character table of Sym(4).
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-random_permutation</td>
<td>n fname</td>
<td>Creates a random permutation in $\text{Sym}(n)$ and stores it in the given file.</td>
</tr>
<tr>
<td>-create_random_k_subsets</td>
<td>n k N</td>
<td>Creates $N$ random $k$-subsets of an $n$-set.</td>
</tr>
<tr>
<td>-read_poset_file</td>
<td>fname</td>
<td>Reads a poset from the given file.</td>
</tr>
<tr>
<td>-read_poset_file_with_grouping</td>
<td>fname x-stretch</td>
<td>Reads a poset from the given file and sets stretch factor for orbit grouping.</td>
</tr>
<tr>
<td>-list_parameters_of_SRG</td>
<td>$v_{\text{max}}$</td>
<td>Performs a sift for putative parameter sets of SRGs.</td>
</tr>
<tr>
<td>-conjugacy_classes_Sym_n</td>
<td>n</td>
<td>Compute a list of conjugacy classes of $\text{Sym}(n)$.</td>
</tr>
<tr>
<td>-tree_of_all_k_subsets</td>
<td>n k</td>
<td>Creates a tree-file for all $k$-subsets of an $n$-set.</td>
</tr>
<tr>
<td>-Delandtsheer_Doyen</td>
<td></td>
<td>See Section 11.7.</td>
</tr>
<tr>
<td>-tdo_refinement</td>
<td></td>
<td>See Section 11.8.</td>
</tr>
<tr>
<td>-tdo_print</td>
<td></td>
<td>See Section 11.8.</td>
</tr>
<tr>
<td>-convert_stack_to_tdo</td>
<td></td>
<td>See Section 11.8.</td>
</tr>
<tr>
<td>-maximal_arc_parameters</td>
<td></td>
<td>See Section 11.8.</td>
</tr>
<tr>
<td>-arc_parameters</td>
<td></td>
<td>See Section 11.8.</td>
</tr>
<tr>
<td>-pentomino_puzzle</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 11.1: Commands related to Combinatorics (Part 1)
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-draw_layered_graph</td>
<td>options</td>
<td>Draws a graph.</td>
</tr>
<tr>
<td>-make_elementary_symmetric_functions</td>
<td>$n$ $k_{\text{max}}$</td>
<td>Computes the elementary symmetric functions in $n$ variables of degree $1,\ldots,k_{\text{max}}$.</td>
</tr>
<tr>
<td>-Dedekind_numbers</td>
<td>$n_{\text{min}}$ $n_{\text{max}}$ $q_{\text{min}}$ $q_{\text{max}}$</td>
<td>Computes the Dedekind numbers $D_{n,q}$ for $n_{\text{min}} \leq n \leq n_{\text{max}}$ and $q_{\text{min}} \leq q \leq q_{\text{max}}$.</td>
</tr>
<tr>
<td>-rank_k_subset</td>
<td>$n$ $k$ $L$</td>
<td>Computes the ranks of $k$-subsets chosen from an $n$-set. $L$ is a list of $k$-sets taken from an $n$-set.</td>
</tr>
<tr>
<td>-geometry_builder</td>
<td></td>
<td>See Section 11.4.</td>
</tr>
<tr>
<td>-character_table_symmetric_group</td>
<td>$n$</td>
<td>Computes the character table of $\text{Sym}(n)$ using the algorithm of Burnside.</td>
</tr>
<tr>
<td>-domino_portrait</td>
<td>$D$ $s$ $\text{fname}$</td>
<td>Computes a domino portrait for a graphics file in $r/g/b$ format using double $D$ domino sets.</td>
</tr>
</tbody>
</table>

Table 11.2: Commands related to Combinatorics (Part 2)
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:

```
random_k_subsets:
  $(ORBITER) -v 4 \n  -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:

```
rank_k_subsets_test:
  $(ORBITER) -v 2 \n  -rank_k_subset 10 3 0,1,2,0,3,4,1,3,5,2,4,5,3,6,7,1,6,8,0,6,9
```

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:
Walsh_matrix_4:

\[
\begin{array}{cccc}
\text{\textbackslash $(ORBITER)$} & \text{-v 3} & \text{-define F -finite_field -q 2 -end} \\
\text{-with F -do -finite_field_activity} & \text{-Walsh_matrix 4 -end} \\
\text{$\textbackslash $(ORBITER)$ -v 2 -draw_matrix} & \text{-input_csv_file Walsh_01_4.csv} \\
\text{-box_width 10 -bit_depth 24 -partition 3 16 16 -end} \\
\text{#pdflatex GF_2.tex} \\
\text{#open GF_2.pdf}
\end{array}
\]

The following command creates the matrix of Dedekind numbers of order at most 10:

Dedekind_10_10:

\[
\begin{array}{cccc}
\text{\textbackslash $(ORBITER)$} & \text{-v 3} & \text{-Dedekind_numbers 2 10 2 10} \\
\end{array}
\]

The following command creates the elementary symmetric functions in 4 variables.

elementary_symmetric_functions_4:

\[
\begin{array}{cccc}
\text{\textbackslash $(ORBITER)$} & \text{-make_elementary_symmetric_functions 4 4} \\
\end{array}
\]

The output is:

\begin{align*}
\text{k=1} : & \quad x_0 + x_1 + x_2 + x_3 \\
\text{k=2} : & \quad x_0 x_1 + x_0 x_2 + x_0 x_3 + x_1 x_2 + x_1 x_3 + x_2 x_3 \\
\text{k=3} : & \quad x_0 x_1 x_2 + x_0 x_1 x_3 + x_0 x_2 x_3 + x_1 x_2 x_3 \\
\text{k=4} : & \quad x_0 x_1 x_2 x_3
\end{align*}

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.

domino_portrait:

\[
\begin{array}{cccc}
\text{\textbackslash $(ORBITER)$} & \text{-v 3} & \text{-domino_portrait 7 4 anton_28x32 -end} \\
\end{array}
\]

The portrait is shown in Figure 11.2. It is possible to compare the domino portrait with a grayscale version of the input image. The following command creates a grayscale image of the input file that was written during the previous command.
Figure 11.2: Domino Portrait
Figure 11.3: Domino portrait input file in grayscale

```plaintext
domino_portrait_input:
  ▶ $(ORBITER) -v 2 \n  ▶ ▶ -define all_one_r -vector -repeat 1 28 -end \n  ▶ ▶ -define all_one_c -vector -repeat 1 32 -end \n  ▶ ▶ -draw_matrix \n  ▶ ▶ ▶ -grayscale \n  ▶ ▶ ▶ -invert_colors \n  ▶ ▶ ▶ -input_csv_file anton_28x32.m.csv \n  ▶ ▶ ▶ -box_width 20 -bit_depth 8 \n  ▶ ▶ ▶ -partition 3 \n  ▶ ▶ ▶ ▶ all_one_c all_one_r \n  ▶ ▶ -end
  ▶ open anton_28x32_m_draw.bmp
```

The grayscale version of the input file is shown in Figure 11.3.
11.2 Diophantine Systems

Diophantine systems of equations and inequalities arise frequently in Combinatorics. In Table 11.3, Orbiter commands for creating and solving diophantine systems are shown. In Table 11.4, 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 \( x \) with entries in 0, 1 such that

\[ A x = 1 \]

where 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 \textsc{TEST\_SYSTEM} for the coefficient matrix and \textsc{TEST\_RHS} for the right hand side.

\textsc{TEST\_SYSTEM}="\n0,1,0,1,0,0, \n0,0,1,0,1,0, \n1,0,1,0,0,0, \n0,1,0,1,0,1, \n1,0,0,0,1, \n1,0,1,0,0, \n0,1,0,0,1,1
"

\textsc{TEST\_RHS}="1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1"

\begin{verbatim}
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 $(TEST\_RHS) \\
  -x_min_global 0 -x_max_global 1 \\
  -end
\end{verbatim}
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-label</td>
<td>label</td>
<td>Use the given name as file name.</td>
</tr>
<tr>
<td>-coefficient_matrix</td>
<td>A</td>
<td>Set the coefficient matrix to the previously created vector with format information.</td>
</tr>
<tr>
<td>-coefficient_matrix_csv</td>
<td>fname</td>
<td>Read the coefficient matrix from the given csv-file.</td>
</tr>
<tr>
<td>-RHS</td>
<td>list-of-integers</td>
<td>3n values: (RHS-low, RHS-high, RHS-type) for each row of the system.</td>
</tr>
<tr>
<td>-RHS_csv</td>
<td>fname</td>
<td>Read the RHS information from the given csv file.</td>
</tr>
<tr>
<td>-RHS_constant</td>
<td>low,high,type</td>
<td>Set the RHS according to low,high,type.</td>
</tr>
<tr>
<td>-x_max_global</td>
<td>a</td>
<td>Set the upper bound for all variables to a</td>
</tr>
<tr>
<td>-x_min_global</td>
<td>a</td>
<td>Set the lower bound for all variables to a</td>
</tr>
<tr>
<td>-x_bounds</td>
<td>list-of-values</td>
<td>Set the lower and upper bounds for all variables.</td>
</tr>
<tr>
<td>-x_bounds_csv</td>
<td>fname</td>
<td>Read the lower and upper bounds for all variables from the given file.</td>
</tr>
<tr>
<td>-has_sum</td>
<td>s</td>
<td>For the sum of the variables to be s.</td>
</tr>
<tr>
<td>-maximal_arc</td>
<td>s d secants subset</td>
<td>Create system for a maximal arc of size s and degree d in PG(2,q). Use the given set of two pencil lines. The subset picks the lines from the given pencils which are external.</td>
</tr>
<tr>
<td>-q</td>
<td>q</td>
<td>Use PG(2,q) for maximal arcs.</td>
</tr>
<tr>
<td>-override_polynomial</td>
<td>a</td>
<td>Use polynomial numerically coded as a for creating $F_q$.</td>
</tr>
</tbody>
</table>

Table 11.3: Orbiter Commands to create Diophantine systems
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-print</td>
<td></td>
<td>Print the system.</td>
</tr>
<tr>
<td>-solve_mckay</td>
<td></td>
<td>Solve the system using McKay’s pos-solve.</td>
</tr>
<tr>
<td>-solve_DLX</td>
<td></td>
<td>Solve the system using Knuth’s dancing links.</td>
</tr>
<tr>
<td>-solve_standard</td>
<td></td>
<td>Solve the system using the standard solver.</td>
</tr>
<tr>
<td>-draw</td>
<td></td>
<td>Produce a drawing of the coefficient matrix of the system.</td>
</tr>
<tr>
<td>-draw_as_bitmap</td>
<td>w b</td>
<td>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.</td>
</tr>
<tr>
<td>-perform_column_reductions</td>
<td></td>
<td>Eliminate variables which must be zero and write a reduced system.</td>
</tr>
<tr>
<td>-test_single_equation</td>
<td></td>
<td>For each row of the system, compute the number of solutions of the system restricted to the nonzero coefficients.</td>
</tr>
<tr>
<td>-project_to_single_equation_and_solve</td>
<td>i j</td>
<td>Solve the system assuming the $j$th solution to the restricted system consisting of the $i$th row.</td>
</tr>
<tr>
<td>-project_to_two_equations_and_solve</td>
<td>i j r m</td>
<td>Solve the system assuming any solution to the restricted system consisting of the $i$th and the $j$-th row whose number is congruent to $r \mod m$.</td>
</tr>
</tbody>
</table>

Table 11.4: Orbiter activities for Diophantine systems
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:

McKay_test:
```
$ (ORBITER) -v 4 \
  -define A -vector -format 7 -dense $(TEST_SYSTEM) -end \
  -define D -diophant \
  -label test_system \
  -coefficient_matrix A \
  -RHS $(TEST_RHS) \
  -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. And now the dancing links solver:

DLX_test:
```
$ (ORBITER) -v 4 \
  -define A -vector -format 7 -dense $(TEST_SYSTEM) -end \
  -define D \
  -diophant -label test_system \
  -coefficient_matrix A \
  -RHS $(TEST_RHS) \
  -x_min_global 0 -x_max_global 1 \
  -end \
  -with D -do \
  -diophant_activity -solve_DLX \
  -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 moreover 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

\[
\binom{n}{2} = \sum_{j=2}^{n} a_j \binom{j}{2}
\]  

(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

\[
\begin{bmatrix}
15 & 10 & 6 & 3 & 1 \\
\end{bmatrix}.
\]

The Orbiter command

\[
\text{linsp6:}
\]

\[
\text{\# Found 15 solutions with 22 backtrack steps}
\]

solves the system using McKay’s program possolve [50]. The program finds 15 solutions, written to the file \text{linsp6.sol}.

Let us consider a problem from [11]. Suppose we are interested in linear spaces on 30 points 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

\[
p_j = \#j\text{-lines through } P.
\]
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

\[6p_7 + 4p_5 + 3p_4 = 29, \quad p_7 \leq 1, \quad p_5 \leq 27, \quad p_4 \leq 24.\]

Using the Orbiter commands

```
linsp30_pt_types:
  $(ORBITER) -v 4 \n  -define A -vector -format 1 -dense "6,4,3" -end \n  -define D -diophant \n  -label linsp30_pt_types \n  -coefficient_matrix A \n  -RHS "29,29,1" -x_bounds "0,1,0,27,0,24" \n  -end \n  -with D -do \n  -diophant_activity -solve_mckay \n  -end
```

we determine the possibilities

\[
(p_7, p_5, p_4) = \begin{pmatrix}
1 & 5 & 1 \\
1 & 2 & 5 \\
0 & 5 & 3 \\
0 & 2 & 7
\end{pmatrix}
\]

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:

\[
b_0 + b_1 + b_2 + b_3 = 30
\]
\[
b_0 + b_1 = 7
\]
\[
5b_0 + 2b_1 + 5b_2 + 2b_3 = 135 = 27 \cdot 5
\]
\[
b_0 + 5b_1 + 3b_2 + 7b_3 = 96 = 24 \cdot 4
\]
\[
10b_0 + b_1 + 10b_2 + b_3 \leq 351 = \binom{27}{2}
\]
\[
10b_1 + 3b_2 + 21b_3 \leq 276 = \binom{24}{2}
\]

Using the Orbiter commands
linsp30_pt_distribution:
▶ $(ORBITER) -v 4 \\
▶ ▶ -define A -vector -format 6 -dense \\
▶ ▶ ▶ "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 "30,30,1,7,7,1,135,135,1,96,96,1,0,351,2,0,276,2" \\
▶ ▶ ▶ -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 $L = (V, B)$, where $B$ is a set of distinct subsets of $V$, called lines. The members of $V \cup B$ are called elements. For two elements $x, y$, we say that $x$ is incident with $y$, written $x I y$, if either $x \in y$ or $y \in x$. We require that any line has at least two points and 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 $V \cup B$ such that each $C_i$ either is a subset of $V$ or a subset of $B$. A decomposition is called tactical if for all $i$, the incidence number

$$\iota(C_i, C_j) = |\{y \in C_j, x I y\}|$$

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 $\text{Aut}(L)$ be the automorphism group of the linear space, which is the subgroup of $\text{Sym}(V)$ which preserves incidence. For $\alpha \in \text{Aut}(L)$ we say that the decomposition $\Pi$ preserves $\alpha$ if $\alpha$ fixes every class of $\Pi$. For $A \leq \text{Aut}(L)$, we say that $\Pi$ preserves $A$ is $\Pi$ preserves every element $\alpha \in A$. Mostly, we are interested in those decompositions $\Pi$ which preserve $\text{Aut}(L)$. 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 $V \cup B$ 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 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 decomposition of any given linear space $P$. 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 $\text{Aut}(L)$.

Let us consider the opposite question. Given a tactical decomposition, does there exist a linear space whose coarsest tactical decomposition is the given one? If so, how many nonisomorphic partial linear spaces are there for a given tactical decomposition? in other words, we would like to classify the linear spaces which admit a given tactical decomposition. The \texttt{-geometry\_builder} option can answer these kinds of questions. Table 11.5 shows the options for the geometry builder.

The command

\begin{verbatim}
geo10_3:
▷ $(ORBITER) -v 2 \n▷ ▷ -define Test_lines -set -loop 4 11 1 -end \n▷ ▷ -define Geo -geometry\_builder \n▷ ▷ ▷ -V 10 -B 10 -TDO 3 -fuse 1 \n▷ ▷ ▷ -fname GEO 10_3 \n\end{verbatim}
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-V</td>
<td>part</td>
<td>The initial partition of points (rows).</td>
</tr>
<tr>
<td>-B</td>
<td>part</td>
<td>The initial partition of blocks (columns).</td>
</tr>
<tr>
<td>-TDO</td>
<td>tdo</td>
<td>The initial row-tactical decomposition scheme.</td>
</tr>
<tr>
<td>-fuse</td>
<td>fuse</td>
<td>The partition of row classes.</td>
</tr>
<tr>
<td>-girth_test</td>
<td>g</td>
<td>Require the girth of the collinearity graph to be at least g.</td>
</tr>
<tr>
<td>-lambda</td>
<td>λ</td>
<td>Set λ for two-design test. Every pair of points lies in λ blocks.</td>
</tr>
<tr>
<td>-find_square</td>
<td></td>
<td>Construct linear spaces.</td>
</tr>
<tr>
<td>-simple</td>
<td></td>
<td>Construct simple designs (needs -lambda)</td>
</tr>
<tr>
<td>-search_tree</td>
<td></td>
<td>Write a file containing the search tree (at the level of rows of the partial geometry).</td>
</tr>
<tr>
<td>-search_tree_flags</td>
<td></td>
<td>Write a file containing the search tree (at the level of flags of the partial geometry). A flag is an incident point-block pair.</td>
</tr>
<tr>
<td>-orderly</td>
<td></td>
<td>User orderly generation.</td>
</tr>
<tr>
<td>-special_test_orderly</td>
<td></td>
<td>Use a special test. This option only applies to orderly generation.</td>
</tr>
<tr>
<td>-split</td>
<td>l r m</td>
<td>Split the search tree. After l lines, continue only cases congruent to r modulo m.</td>
</tr>
<tr>
<td>-fname_GEO</td>
<td>fname</td>
<td>Set the output file name base (no extension).</td>
</tr>
<tr>
<td>-output_to_inc_file</td>
<td></td>
<td>Set output to inc file.</td>
</tr>
<tr>
<td>-output_to_sage_file</td>
<td></td>
<td>Set output to sage file.</td>
</tr>
<tr>
<td>-output_to_blocks_file</td>
<td></td>
<td>Set output to a file containing the blocks in coded form.</td>
</tr>
<tr>
<td>-output_to_blocks_latex_file</td>
<td></td>
<td>Set output to a file containing the blocks in latex.</td>
</tr>
</tbody>
</table>

Table 11.5: Orbiter commands to build geometries
classifies the configurations $10_3$. 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.

1. The option `-output_to_inc_file` writes $10_3$\_inc\_file. The file contains the incidences in increasing order. The position in the incidence matrix is given. One linear space is given per row, except for the first row and the last. The first row contains the number of points, the number of lines, and the number of incidences. The incidences are given in numeric form. The last row start with $-1$. 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. In this case, with $b = 10$, zero represents the incidence between point 0 and block zero. The number $99$ represents the incidence between point 9 and block 9. Here is the file $10_3$\_inc:

```
10 10 30
0 1 2 10 13 14 20 25 26 31 33 35 41 44 47 52 53 58 62 68 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 54 58 62 67 69 73 76 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 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 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 87 89 96 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 97 99
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 97 99
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 97 99
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 97 99
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 97 99
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 97 99
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 97 99
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 97 99
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 97 99
0 1 2 10 13 14 20 25 26 31 33 35 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 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 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 76 79 84 86 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 [64].

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
```

349
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 command below does so for the example that we just did. Note the additional option `-search_tree`. This 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`. In a second invocation of Orbiter, the `-tree_draw` command is used to draw a tree from the file `10_3_tree.txt` that was 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. Here, the 10 green nodes at the very bottom of the diagram represent the 10 isomorphism types of configurations $10_3$.

```
geo_10_3_tree:
  $(ORBITER) -v 20 \
  -define Test_lines -set -loop 0 11 1 -end \n  -define GEO -geometry_builder \n  -V 10 -E 10 -TDO 3 -fuse 1 \n  -fname_GEO 10_3 \n  -output_to_inc_file \n  -search_tree \n  -test Test_lines \n  -end \
  $(ORBITER) -v 20 \
  -draw_options -embedded -radius 40 \
  -paperheight 220 \n  -paperwidth 330 \n  -xin 10000 -yin 10000 \n  -xout 1000000 -yout 500000 \n  -scale 2 -line_width 0.3 \n  -nodes_empty \n```
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 in Figure 11.4.

Any incidence structure defines a graph on its underlying set of points. The vertices are the points of the incidence structure. Two vertices are adjacent if and only if the incidence structure contains a block which contains the associated points. In a geometric context, the graph is known as the collinearity graph of the geometry. The distance between two points is the distance of the associated vertices in the collinearity graph. The girth is the length of the shortest cycle. It is often desired to classify incidence structures with a given girth. This means that we are given an integer $g$ (the girth), and that we are looking for incidence structures whose collinearity graph has no cycles of length less than $g$. For instance, the following example classifies all cubic graphs on 10 vertices with girth at least 5:
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.

We can classify configurations with a given girth. 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. It is associated to a cubic surface. The following command classifies all configurations 15\_3:

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 
  -end

This command takes about 8 minutes of time to complete.

The next command classifies the configurations 15\_3 with girth 4. Only one configuration arises, the Cremona Richmond, with automorphism group Sym(6) of order 720.

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 
  -test Test_lines 
  -end

352
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.

40_4_g4.inc:
```bash
$ (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 \
$ (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 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_orderly` to speed up the search.

geo_63_3_g6:
```bash
$ (ORBITER) -v 2 \
```

The search tree is shown in Figure 11.5.
Figure 11.5: The search tree for the configurations 40_4 with girth 4
The search tree is much larger than for the previous problem.
11.5 Design Theory

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 contained 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 \( j \)th block class depends only on the class of the point. If 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 \( i \)th 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 \( \text{PG}(2,q) \). 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.

The command

```
$ (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
```

creates the design \( \text{PG}(2,3) \).

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:

\[
\begin{bmatrix}
2 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix},
\begin{bmatrix}
1 & 0 & 0 \\
0 & 2 & 0 \\
0 & 0 & 1
\end{bmatrix},
\begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 1 & 1
\end{bmatrix},
\begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 1 & 1
\end{bmatrix},
\begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 1 & 0
\end{bmatrix},
\begin{bmatrix}
0 & 1 & 0 \\
0 & 0 & 1 \\
0 & 0 & 1
\end{bmatrix},
\begin{bmatrix}
1,0,0,0,2,0,0,0,2, \\
1,0,0,0,2,0,0,0,1, \\
1,0,0,0,1,0,1,0,1, \\
1,0,0,0,1,0,0,1,1, \\
1,0,0,0,0,1,0,1,0, \\
0,1,0,1,0,0,0,0,1,
\end{bmatrix}
\]

The blocks of the design are encoded in the lexicographic ordering of \(k\)-subsets (here \(k = 4\)).

The program also displays the tactical decomposition schemes of the design, which are

\[
\begin{array}{c|c}
\rightarrow & 13_1 \\
13_0 & 4
\end{array} \qquad \begin{array}{c|c}
\downarrow & 13_1 \\
13_0 & 4
\end{array}
\]

In Section 15.4, we will show how to compute further properties of the design.

The command

\begin{verbatim}
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
\end{verbatim}

creates a design on 8 points invariant under the wreath product \(\text{Sym}(4) \wr \text{Sym}(2)\). The design has 12 blocks of size 4. The command

\begin{verbatim}
wreath_product_designs_n8_k6_inc.txt:
  > $(ORBITER) -v 8 \ 
\end{verbatim}

357
creates a design on 16 points invariant under the wreath product \( \text{Sym}(8) \wr \text{Sym}(2) \). The design has 3920 blocks of size 6. We will compute the automorphism groups of these two designs in Section 15.3.

One way to construct designs is by assuming a suitable group of symmetries. Let us consider an example. It is possible to construct \( t-(v, k, \lambda) \) 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. \([40]\)). 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 \( G \leq \text{Sym}(V) \) defined in this way. The \( t-(v, k, \lambda) \) designs invariant under \( G \) are in one-to-one correspondence to the solutions of

\[
M_{t,k}(G) \cdot x = \lambda 1,
\]

where \( x \) is a column vector of zeros and ones and \( 1 \) is the column vector of all ones. The length of \( x \) is the number of \( k \)-orbits of \( G \) on \( V \), while the length of \( 1 \) is the number of \( t \)-orbits of \( G \) on \( V \). Any vector \( x \) 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 \( x \) is one. We assume the group \( \text{PGL}(2,32) \) in the action on points of the projective line \( \text{PG}(1,32) \) over the field \( F_{32} \). The parameters of the design are \( 7-(33,8,10) \), that is, each 7-subset of \( \text{PG}(1,32) \) is covered exactly 10 times by the chosen 8-subsets comprising the design. The first orbiter command creates the group \( \text{PGL}(2,32) \) and computes the Kramer-Mesner matrix

\[
M_{7,8}(\text{PGL}(2,32)).
\]

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

\[
\text{KM}_{-\text{PGGL}}_{\_2\_32\_\text{-KM}}_{\_7\_8}.\text{csv}.
\]

The second command produces the graphical representation of the matrix shown in Figure 11.6 (different colors represent different values of entries in the matrix). The third Orbiter command creates the diophantine system associated with the Kramer-Mesner matrix.
Figure 11.6: Kramer-Mesner matrix $M_{7,8}(\text{PGL}(2,32))$

**KM_PGGL_2_32:**

```bash
$($(ORBITER) -v 3 \n  -orbiter_path $(ORBITER_PATH) \n  -define Control -poset_classification_control \n  -problem_label KM_PGGL_2_32 \n  -W -depth 8 \n  -draw_options -embedded -sideways -radius 50 \n  -scale 0.5 -line_width 0.3 -end \n  -end \n  -define G -linear_group -PGGL 2 32 -end \n  -define Orb -orbits -group G \n  -on_subsets 8 Control \n  -end \n  -with Orb -do -orbits_activity \n  -Kramer_Mesner_matrix 7 8 \n  -end \n  -with Orb -do -orbits_activity \n  -report \n  -report_options -draw_poset -end \n  -end
```

```bash
$(ORBITER) -v 2 -draw_matrix \n -input_csv_file KM_PGGL_2_32_KM_7.8.csv \n -box_width 20 -bit_depth 24 \n -partition 3 32 97 -end
```

```bash
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
```

```bash
$($(ORBITER) -v 4 \n  -define A -vector -file KM_PGGL_2_32_KM_7.8.csv -end \n  -define D -diophant \n```

359
The last command performs a complete enumeration of all solutions by solving the system and producing the solution vectors $x$ which correspond to the designs.
11.6 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.

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 if it is unique, we use the following command:

```
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 \
>  -end
```

The command produces the file AG_2_3.inc, which contains the following lines:

```
9 12 36
0 1 2 3 12 16 18 24 31 32 33 37 43 46 49 53 56 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 commands, we will treat blocks of the design as sets of ranks of $k$-subsets. We can now create a table of all designs AG(2, 3), as orbit under the group Sym(9). The following command does that:

```
AG_2_3_BLOCKS="0,13,22,27,35,41,47,53,55,59,71,76"
```

```
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 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.
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:

The final step to classify the large sets up to isomorphism will be discussed in Section 15.4.
11.7 Design Theory – Delandtsheer-Doyen

Delandtsheer and Doyen in [23] 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 $\text{AGL}(d_1, q_1) \times \text{AGL}(d_2, q_2)$ acting on a grid of size $q_1^{d_1} \times q_2^{d_2}$ in product action.

Finite projective planes often arise in this context. However, not all examples are projective planes. Orter 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 $3 \times 7$. To this end, we consider the group $\text{AGL}(1, 3) \times \text{AGL}(1, 7)$. 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. The question remains to construct all blocks and to classify the resulting designs. The Desarguesian plane $\text{PG}(2, 4)$ provides a solution. The question is to decide whether there are any other, nonisomorphic designs. The following Orbiter commands can be used:

```
PP4= -d1 1 -q1 3 -d2 1 -q2 7 -K 5 -search_control -W -end -problem_label PP4

PP4_GROUP1=-subgroup "1,1,1,1, " "21" -group_label "cyclic21"

PP4_MASK1=\n ▷ -nb_orbits_on_blocks 1 \n ▷ -depth 5 \n ▷ -mask_label "no_mask"
```

The command `DD_PP4` sets up the orbits of the group on pairs and writes the file `PP4_pair_covering.csv`.

```
DD_PP4:
 ▷ $(\text{ORBITER}) -v 6 \n ▷ ▷ -Delandtsheer_Doyen $(PP4) $(PP4_GROUP1) $(PP4_MASK1) \n ▷ ▷ ▷ -end \n```

The command `DD_PP4_system` creates a diophantine system of Steiner type and solves it.
DD_PP4_system:
▷ $(ORBITER) -v 4  
▷ ▷ -define D -diophant -label PP4  
▷ ▷ -problem_of_Steiner_type 10 PP4.pair_covering.csv  
▷ ▷ -has_sum 1  
▷ ▷ -end  
▷ ▷ -with D -do  
▷ ▷ ▷ -diophant_activity -solve_mckay  
▷ ▷ -end

It finds exactly one solution. This must be the PG(2,4) design. Since there are no more designs, isomorphism testing is not needed.
11.8 Tactical Decompositions

Table 11.6 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

```
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 $i$th 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

```
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} |
```

365
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-lambda3</td>
<td>$\lambda_3$ $s$</td>
<td>Refine as 3-design with $\lambda_3$ and with block-size $s$</td>
</tr>
<tr>
<td>-solution</td>
<td>$i$ $fname$</td>
<td>Use solutions to system $i$ from file $fname$.</td>
</tr>
<tr>
<td>-range</td>
<td>$f$ $l$</td>
<td>Refine cases $i$ with $f \leq i &lt; f + l$ only.</td>
</tr>
<tr>
<td>-select</td>
<td>label</td>
<td>Select the case for refinement by label.</td>
</tr>
<tr>
<td>-o1</td>
<td>$s$</td>
<td>Omit $s$ variables from the first refinement system.</td>
</tr>
<tr>
<td>-o2</td>
<td>$s$</td>
<td>Omit $s$ variables from the second refinement system.</td>
</tr>
<tr>
<td>-D1_upper_bound_x0</td>
<td>$b$</td>
<td>Add the bound $x_0 \leq b$ in the first refinement.</td>
</tr>
<tr>
<td>-reverse</td>
<td></td>
<td>Sort the distributions in reverse order.</td>
</tr>
<tr>
<td>-reverse_inverse</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-nopacking</td>
<td></td>
<td>Do not use packing inequalities.</td>
</tr>
<tr>
<td>-dual_is_linear_space</td>
<td></td>
<td>Assume that the dual incidence structure is a linear space also. This is valid for projective planes, for instance.</td>
</tr>
<tr>
<td>-geometric_test</td>
<td></td>
<td>Subject the distributions to the geometric test.</td>
</tr>
<tr>
<td>-once</td>
<td></td>
<td>Find at most one refinement in each case. This can be used to test which cases can be refined.</td>
</tr>
<tr>
<td>-mckay</td>
<td></td>
<td>Use McKay’s solver instead (by default, a lexicographic solver is used).</td>
</tr>
<tr>
<td>-input_file</td>
<td>$fname$</td>
<td>Specify the input TDO-file for refinement.</td>
</tr>
</tbody>
</table>

Table 11.6: TDO refinement options
Next, we can compute all coarsest column-tactical refinements of the decomposition. To this end, the command

```
max_arc_16_4_refine:
  $(ORBITER) -v 4 -tdo_refinement \
  -input_file max_arc_q16_r4.tdo -dual_is_linear_space -end
```

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:

```
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} 
  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
```

```
367
```
| 52_{0} | 4 0 |
| 221_{3} | 17 17 |

extra_col_scheme at level 3:
is 1 x 2

| 273_{0} | 17 17 |
Chapter 12

Finite Geometry

12.1 Spreads

A $t$-spread of $\text{PG}(n,q)$ is a set of disjoint $\text{PG}(t,q)$ that cover all of $\text{PG}(n,q)$ pointwise. $t$-spreads in $\text{PG}(n,q)$ exist if and only if $t+1$ divides $n+1$. In order to create a spread, Orbiter offers several commands, as summarized in Table 12.1. The following two commands create the two spreads of order 9, relying on the Orbiter knowledge base.

create spread 9a:
- $(\text{ORBITER}) -v 3 \newline
- \text{-define F -finite_field -q 3 -end} \newline
- \text{-define G -linear_group -PGL 4 F -end} \newline
- \text{-define S -spread -kernel_field F} \newline
- \text{-group G -k 2 -catalogue 0} \newline
- \text{-end}$

create spread 9b:
- $(\text{ORBITER}) -v 3 \newline
- \text{-define F -finite_field -q 3 -end} \newline
- \text{-define G -linear_group -PGL 4 F -end} \newline
- \text{-define S -spread -kernel_field F} \newline
- \text{-group G -k 2 -catalogue 1} \newline
- \text{-end}$

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 [54] can be defined using the following makefile variable.
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-kernel_field</td>
<td>$F$</td>
<td>Define the kernel of the spread. $F$ must be an object of type finite field.</td>
</tr>
<tr>
<td>-group</td>
<td>$G$</td>
<td>Define the group acting on the spread. Should be $\text{PGL}(2k; F)$.</td>
</tr>
<tr>
<td>-k</td>
<td>$k$</td>
<td>Set the dimension of the spread.</td>
</tr>
<tr>
<td>-catalogue</td>
<td>$i$</td>
<td>Pull spread number $i$ from the catalogue of spreads associated with the given field and the given dimension.</td>
</tr>
<tr>
<td>-family</td>
<td>$L$</td>
<td>Define a spread from a named family $L$. So far, no family has been provided.</td>
</tr>
<tr>
<td>-spread_set</td>
<td>$S$</td>
<td>Define a spread from the named spreadset $S$. The spreadset $S$ must be a vector object. It must contain $q^3k^2$ entries over $F$.</td>
</tr>
</tbody>
</table>

Table 12.1: Orbiter commands to define a spread

```plaintext
SPREAD_SET_27_RAO_RAO="
0,0,0,0,0,0,0,0,0,0,
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,
0,2,2,2,2,2,0,2,\n2,1,2,0,2,0,2,0,1,\n0,1,2,2,0,1,0,1,1,\n1,0,0,0,1,0,0,0,1,\n2,1,0,1,2,1,0,2,0,\n0,2,0,2,2,1,1,2,\n0,0,1,0,1,2,2,2,1,\n2,0,1,2,2,1,1,0,1,\n0,1,1,1,0,1,2,2,\n2,2,0,2,0,0,0,2,2,\n2,1,1,1,1,2,2,1,2,\n2,2,1,2,1,0,2,0,0,\n1,2,0,2,0,2,1,0,0,\n```

370
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:

```
create_spread_Rao_Rao_27:
  ▷ $(ORBITER) -v 3 \n  ▷ ▷ -define F -finite_field -q 3 -end \n  ▷ ▷ -define SS -vector -dense $(SPREAD_SET_27_RAO_RAO) -end \n  ▷ ▷ -define G -linear_group -PGL 6 F -end \n  ▷ ▷ -define S -spread -kernel_field F \n  ▷ ▷ ▷ -group G -k 3 -spread_set SS \n  ▷ ▷ -end
```

The following command creates the Desarguesian line-spread in PG(3, 2):

```
desarguesian_spread_in_PG_3_2:
  ▷ $(ORBITER) -v 3 \n  ▷ ▷ -define FQ -finite_field -q 4 -end \n  ▷ ▷ -define Fq -finite_field -q 2 -end \n  ▷ ▷ -with FQ -and Fq -do -finite_field_activity \n  ▷ ▷ ▷ -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) =
  1 0 0 0 0
  0 1 0 0
  0 0 1 0
  0 0 0 1

  Spread element 1 is (0, 1) =
  1 0 1 0
  0 1 0 1
  0 0 1 0
  0 0 0 1

  Spread element 2 is (1, 1) =
  1 1 1 0
  0 1 0 1
  0 0 1 0
  1 0 0 1

  Spread element 3 is (2, 1) =
  1 0 0 1
  1 0 1 0
  0 1 0 1
  1 1 1 0

```

371
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):

```
$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 $(1, 0) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}_{0}$

Spread element 1 is $(0, 1) = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix}_{1394}$

Spread element 2 is $(1, 1) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}_{189}$

Spread element 3 is $(2, 1) = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}_{671}$

Spread element 4 is $(3, 1) = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix}_{562}$

Spread element 5 is $(4, 1) = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix}_{1040}$
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. The problem is computationally difficult. 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 \( \text{PG}(3, 4) \) under the action of \( \text{PGL}(4, 4) \). Under the André, Bruck-Bose construction \([3, 16]\), these spreads correspond to translation planes of order 16 with kernel \( \mathbb{F}_4 \). In order to classify the spreads of \( \text{PG}(3, 4) \), we use the command

```bash
classify_spreads_16_4:
```

Because

```bash
$ (ORBITER) -v 4 \
  > -define F -finite_field -q 4 -end \n  > -define P -projective_space -n 3 -field F -v 0 -end \n  > -define C -spread_classifier \n  >  > -projective_space P \n  >  > -k 2 \n  >  > -starter_size 17 \n  >  > -poset_classification_control \n  >  >  > -draw_options \n  >  >  >  > -radius 20 \n  >  >  >  > -nodes_empty \n  >  >  >  > -line_width 0.2 \n  >  >  >  > -embedded \n  >  >  >  > -end \n  >  >  > -problem_label_spreads_16_4 \n  >  >  > -end \n```
The command uses poset classification to classify the spreads. To this end, it computes the poset of orbits for the group $G = \mathrm{PGL}(4,4)$ acting on the poset of partial spreads in $\mathrm{PG}(3,4)$, shown in Figure 12.1. Up to isomorphism, there are exactly three line-spreads in $\mathrm{PG}(3,4)$ (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:

$$
\begin{bmatrix}
\omega^2 & 0 & 0 & 0 \\
0 & \omega^2 & 0 & 0 \\
1 & \omega & 0 & 1 \\
\omega^2 & \omega & 1 & \omega
\end{bmatrix}
\begin{bmatrix}
\omega^2 & 0 & 0 & 0 \\
\omega & \omega^2 & 0 & 0 \\
\omega & \omega & 1 & \omega^2 \\
0 & 1 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
\omega & 1 & \omega & \omega \\
\omega^2 & \omega & \omega & 0 \\
\omega & 0 & 0 & 1 \\
0 & \omega & \omega & 1
\end{bmatrix}
$$

1,0,0,0,0,1,0,0,2,3,0,2,1,1,3,2,0,
1,0,0,3,1,0,0,3,3,2,1,0,2,2,0,1,
1,3,1,1,2,2,0,1,0,0,3,0,1,1,3,0,
There are 0 extensions
Number of generators 3
Figure 12.1: The poset of orbits of partial spreads in $PG(3,4)$
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:

\[
\begin{bmatrix}
\omega & 0 & 0 & 0 \\
0 & \omega & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix},
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & \omega^2 & 0 \\
0 & 0 & 1 & 0 \\
\end{bmatrix},
\begin{bmatrix}
\omega^2 & 0 & 0 & 0 \\
0 & \omega^2 & 0 & 0 \\
1 & \omega & \omega^2 & 0 \\
1 & \omega & 1 & 1 \\
\end{bmatrix}.
\]

\[
\begin{bmatrix}
\omega^2 & 0 & 0 & 0 \\
\omega & \omega^2 & 0 & 0 \\
0 & 0 & \omega^2 & 0 \\
0 & 0 & \omega & 1 \\
\end{bmatrix},
\begin{bmatrix}
\omega^2 & 0 & 0 & \omega \\
\omega & \omega^2 & 1 & \omega^2 \\
0 & 0 & 1 & 0 \\
0 & 0 & \omega^2 & 1 \\
\end{bmatrix},
\begin{bmatrix}
0 & 0 & 0 & 1 \\
0 & 0 & \omega & 1 \\
0 & 1 & \omega^2 & 0 \\
\omega & 1 & 1 & 1 \\
\end{bmatrix}.
\]

There are 0 extensions
Number of generators 7

Orbit 2 / 3 at Level 17
Node number: 1128

\{0, 25, 50, 75, 90, 108, 124, 140, 158, 183, 199, 217, 233, 250, 268, 312, 345\}_576
| OCN | |Aut| | Name |
|-----|-----|-----|-----|
| 0   | 1200 |     | Hall spread |
| 1   | 81600|     | Desarguesian spread |
| 2   | 576  |     | Semifield spread |

Table 12.2: Spreads in PG(3,4) in the Orbiter Catalogue

Strong generators for a group of order 576:

\[
\begin{bmatrix}
\omega & 0 & 0 & 0 \\
0 & \omega^2 & 0 & 0 \\
0 & 0 & \omega^2 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix},
\begin{bmatrix}
\omega^2 & 0 & 0 & 0 \\
0 & \omega & 0 & 0 \\
0 & \omega & 0 & 1 \\
\omega & 0 & 0 & 1
\end{bmatrix},
\begin{bmatrix}
\omega & \omega & 0 & 0 \\
\omega & 0 & \omega & 0 \\
\omega & 0 & 0 & \omega \\
\omega & 0 & 0 & 1
\end{bmatrix},
\begin{bmatrix}
\omega & \omega & \omega & \omega \\
\omega & 0 & \omega & 1 \\
\omega & \omega & 0 & 0 \\
\omega & 0 & \omega & 1
\end{bmatrix},
\begin{bmatrix}
1 & 0 & \omega & 0 \\
1 & 0 & \omega & 1 \\
0 & 0 & \omega & 1 \\
0 & 0 & 0 & 1
\end{bmatrix},
\begin{bmatrix}
0 & \omega & \omega & 0 \\
0 & \omega & 0 & \omega \\
0 & 0 & \omega & 0 \\
0 & 0 & 0 & 1
\end{bmatrix},
\begin{bmatrix}
\omega & \omega & \omega & 0 \\
\omega & \omega & 0 & 1 \\
\omega & 0 & \omega & 0 \\
\omega & 1 & \omega & 1
\end{bmatrix},
\begin{bmatrix}
1,0,0,0,0,2,0,0,0,0,2,0,0,0,0,3,1, \\
1,0,0,0,1,0,0,3,0,3,2,1,0,0,2,0, \\
1,0,0,0,3,1,0,0,3,0,2,2,1,0,1,2,0, \\
1,1,1,1,2,0,2,0,2,0,2,1,0,2,2,3,0, \\
1,0,3,1,1,3,1,0,1,0,2,2,0,0,0,1,1, \\
0,1,1,0,0,0,0,1,2,0,2,1,3,2,3,2,0,
\end{bmatrix}
\]

There are 0 extensions
Number of generators 6

The three spreads in PG(3,4) can be distinguished by their stabilizer orders. Table 12.2 lists the line spreads in PG(3,4) according to their orbiter catalogue number (OCN).

Let us now look at a more difficult problem. We wish to classify the spreads in PG(3,5). To this end, we will use the method of classification by substructure. We pick a size \(s\) of a partial spread, and classify all partial spreads of size \(s\). These are the substructures. Next, we perform the lifting, which means we construct all spreads of PG\(3,5\) containing one of the orbit representatives of the substructures. In a final step, we 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. Because of this, we use specialized algorithms from graph theory, which enhance the performance of the lifting. Specifically, we perform a search for rainbow cliques. We will go over some examples to illustrate the technique. To begin with, we choose the parameter \(s = 5\).
The command

classify_spreads_25_starter_lift_case_0:
  ▶ $(ORBITER) -v 3 \\
  ▶ ▶ -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 \\
  ▶ ▶ ▶ ▶ -draw_options \\
  ▶ ▶ ▶ ▶ ▶ -radius 20 \\
  ▶ ▶ ▶ ▶ ▶ -nodes_empty \\
  ▶ ▶ ▶ ▶ ▶ -line_width 0.2 \\
  ▶ ▶ ▶ ▶ ▶ -embedded \\
  ▶ ▶ ▶ ▶ -end \\
  ▶ ▶ ▶ -W \\
  ▶ ▶ ▶ -problem_label spreads_25 \\
  ▶ ▶ ▶ -end \\
  ▶ ▶ -output_prefix "" \\
  ▶ -end \\
  ▶ -with C -do -spread_classify_activity \\
  ▶ ▶ -compute_starter \\
  ▶ ▶ ▶ -problem_label spreads_25 \\
  ▶ ▶ ▶ -W -depth 5 \\
  ▶ ▶ ▶ -report -end \\
  ▶ ▶ -end \\
  ▶ -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 of 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 clique in the graph, the command

spreads_25_starter_0_cliques:
  ▶ $(ORBITER) -v 2 \\
  ▶ ▶ -define G -graph -load spreads_25_graph_0.bin -end \\
  378
can be used.

The command

```plaintext
classify_spreads_25_starter_lift_all_cases:
  $(ORBITER) -v 3 \n  -define F -finite_field -q 5 -end \n  -define P -projective_space -n 3 -field F -v 0 -end \n  -define C -spread_classifier \n  -projective_space P \n  -k 2 \n  -starter_size 5 \n  -recoordinatize \n  -poset_classification_control \n  -draw_options \n  -radius 20 \n  -nodes_empty \n  -line_width 0.2 \n  -end \n  -W \n  -problem_label_spreads_25 \n  -end \n  -output_prefix "" \n  -end \n  -with C -do -spread_classify_activity \n  -compute_starter \n  -problem_label_spreads_25 \n  -W -depth 5 \n  -report -end \n  -end \n  -with C -do -spread_classify_activity \n  -prepare_lifting_all_cases \n  -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:
spreads\_25\_starter\_cliques:
\[
\text{-} \text{(ORBITER) -v 2} \text{ \textbackslash} \\
\text{-} \text{-loop L 0 29 1} \text{ \textbackslash} \\
\text{-} \text{-define G -graph -load spreads\_25\_graph\_%L.bin -end \textbackslash} \\
\text{-} \text{-with G -do \textbackslash} \\
\text{-} \text{-graph\_theoretic\_activity \textbackslash} \\
\text{-} \text{-define \_prefix} \text{ -isomorph \textbackslash} \\
\text{-} \text{\_projective\_space P \textbackslash} \\
\text{-} \text{-k 2 \textbackslash} \\
\text{-} \text{-recoordinatize \textbackslash} \\
\text{-} \text{-poset\_classification\_control \textbackslash} \\
\text{-} \text{-draw\_options \textbackslash} \\
\text{-} \text{-radius 20 \textbackslash} \\
\text{-} \text{-nodes\_empty \textbackslash} \\
\text{-} \text{-line\_width 0.2 \textbackslash} \\
\text{-} \text{-embedded \textbackslash} \\
\text{-} \text{-end \textbackslash} \\
\text{-} \text{-W \textbackslash} \\
\text{-} \text{-problem\_label spreads\_25 \textbackslash} \\
\text{-} \text{-end \textbackslash} \\
\text{-} \text{-output\_prefix "" \textbackslash} \\
\text{-} \text{-end \textbackslash} \\
\text{-} \text{-with C -do -spread\_classify\_activity \textbackslash} \\
\text{-} \text{-compute\_starter \textbackslash} \\
\text{-} \text{-problem\_label spreads\_25 \textbackslash} \\
\text{-} \text{-W -depth 5 \textbackslash} \\
\text{-} \text{-report -end \textbackslash} \\
\text{-} \text{-end \textbackslash} \\
\text{-} \text{-with C -do -spread\_classify\_activity \textbackslash} \\
\text{-} \text{-isomorph \textbackslash} \\
\text{-} \text{-prefix_iso "/spreads\_25" \textbackslash} \\
\text{-} \text{-use\_database\_for\_starter \textbackslash} \\
\text{-} \text{-build\_db \textbackslash}
\]

The resulting cliques are again stored in files. The command

classify\_spreads\_25\_isomorph:
\[
\text{-} \text{(ORBITER) -v 3} \text{ \textbackslash} \\
\text{-} \text{-define F -finite\_field -q 5 -end \textbackslash} \\
\text{-} \text{-define P -projective\_space -n 3 -field F -v 0 -end \textbackslash} \\
\text{-} \text{-define C -spread\_classifier \textbackslash} \\
\text{-} \text{-k 2 \textbackslash} \\
\text{-} \text{-starter\_size 5 \textbackslash} \\
\text{-} \text{-recoordinatize \textbackslash} \\
\text{-} \text{-poset\_classification\_control \textbackslash} \\
\text{-} \text{-draw\_options \textbackslash} \\
\text{-} \text{-radius 20 \textbackslash} \\
\text{-} \text{-nodes\_empty \textbackslash} \\
\text{-} \text{-line\_width 0.2 \textbackslash} \\
\text{-} \text{-embedded \textbackslash} \\
\text{-} \text{-end \textbackslash} \\
\text{-} \text{-W \textbackslash} \\
\text{-} \text{-problem\_label spreads\_25 \textbackslash} \\
\text{-} \text{-end \textbackslash} \\
\text{-} \text{-output\_prefix "" \textbackslash} \\
\text{-} \text{-end \textbackslash} \\
\text{-} \text{-with C -do -spread\_classify\_activity \textbackslash} \\
\text{-} \text{-compute\_starter \textbackslash} \\
\text{-} \text{-problem\_label spreads\_25 \textbackslash} \\
\text{-} \text{-W -depth 5 \textbackslash} \\
\text{-} \text{-report -end \textbackslash} \\
\text{-} \text{-end \textbackslash} \\
\text{-} \text{-with C -do -spread\_classify\_activity \textbackslash} \\
\text{-} \text{-isomorph \textbackslash} \\
\text{-} \text{-prefix_iso "/spreads\_25" \textbackslash} \\
\text{-} \text{-use\_database\_for\_starter \textbackslash} \\
\text{-} \text{-build\_db \textbackslash}
\]
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. Of course, this agrees with the results in the literature, see [22].

Table 12.3 lists the solid spreads in PG(7, 2) according to their Orbiter catalogue number (OCN).

| OCN | |Aut|   | Name |
|-----|-----------------|-----|
| 0   | 1008            |     |
| 1   | 1008            |     |
| 2   | 1728            |     |
| 3   | 216             |     |
| 4   | 360             |     |
| 5   | 288             |     |
| 6   | 3600            |     |
| 7   | 244800          |     |

Table 12.3: Spreads in PG(7, 2) in the Orbiter Catalogue
12.2 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, 16]). In order to perform the construction, we need a field $F = F_q$ which is the kernel of the plane, a spread of $k$-subspaces, and the groups $\text{PGL}(2k, F)$ and $\text{PGL}(2k + 1, F)$. For instance, the command

```
create_translation_plane_9b:
  $(ORBITER) -v 3 \n  -define F -finite_field -q 3 -end \n  -define G -linear_group -PGL 4 F -end \n  -define G1 -linear_group -PGL 5 F -end \n  -define S -spread -kernel_field F \n  -group G -k 2 -catalogue 1 \n  -end \n  -define T -translation_plane S G G1 -end \n  -with T -do -translation_plane_activity \n  -export_incma \n  -end \n  -with T -do -translation_plane_activity \n  -report \n  -end \n  -define A -linear_group -import_group_of_plane T -end \n  -define Orb -orbits -group A \n  -on_points \n  -end \n  -with Orb -do -orbits_activity \n  -report \n  -end \n  -with Orb -do -orbits_activity \n  -stabilizer 92 \n  -end \n  -with Orb -do -orbits_activity \n  -export_trees \n  -end
```

```
$(ORBITER) -v 2 \n -draw_matrix \n -input_csv_file plane_catalogue_q3_k2_1_incma.csv \n -box_width 6 -bit_depth 8 \n -partition 2 91 91 \n -end
```

```
$(ORBITER) -v 3 \n -draw_layered_graph \n -orbit_PGL_5.3_on_andre_3.layered_graph \n -radius 250 -spanning_tree -embedded -nodes_empty \n```
creates the (projective) Hall plane of order 9 from the Hall spread. In this example, we use the fact that $\mathrm{PGL}(n,q) = \mathrm{PGL}(n,q)$ if $q$ is prime. The example also creates a bitmap drawing of the incidence matrix of the plane, shown in Figure 12.2.

In the next example, we create a translation plane of order 16 with kernel of order 4:

```
create_translation_plane_16_4_0:
> $\$(ORBITER) -v 3 \n> > -define F -finite_field -q 4 -end \n> > -define G -linear_group -PGGL 4 F -end \n```
This plane is the Hall plane, and the spread is the Hall spread. The spread has a stabilizer of order 1200.

In the next example, we create a translation plane of order 16 with kernel of order 2:

create_translation_plane_16_2.0:

```
$ORBITER) -v 3 \n  -define F -finite_field -q 2 -end \n  -define G -linear_group -PGL 8 F -end \n  -define G1 -linear_group -PGL 9 F -end \n  -define S -spread -kernel_field F \n  -group G -k 4 -catalogue 0 \n  -end \n  -define T -translation_plane S G G1 -end \n  -with T -do -translation_plane_activity \n  -export_incma \n  -end \n$ORBITER) -v 2 \n  -draw_matrix \n  -input_csv_file plane_catalogue_q2_k4_0_incma.csv \n  -box_width 6 -bit_depth 8 \n  -partition 4 273 273 \n  -end \n  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 [52], there are two planes whose associated spreads have this automorphism group 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:

\[
\text{RREF}_{\text{plane}_16.2.0\_rank\_of\_incma}: \\
\text{\texttt{\$}}(\text{ORBITER}) -v 2 \backslash \\
\quad \text{\texttt{-define F -finite\_field -q 2 -end \}} \\
\quad \text{\texttt{-define v -vector -field F \}} \\
\quad \text{\texttt{-file plane\_catalogue\_q2\_k4\_0\_incma.csv \}} \\
\quad \text{\texttt{-end \}} \\
\quad \text{\texttt{-with F -do -finite\_field\_activity \}} \\
\quad \text{\texttt{-RREF v -normalize\_from\_the\_right \}} \\
\quad \text{\texttt{-end}}
\]

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 12.1, which we know is isomorphic to the spread in the Orbiter catalogue with number 0 and projective stabilizer of order 84. The command

\[
\text{create\_translation\_plane\_27\_Rao\_Rao:} \\
\text{\texttt{\$}}(\text{ORBITER}) -v 3 \backslash \\
\quad \text{\texttt{-define F -finite\_field -q 3 -end \}} \\
\quad \text{\texttt{-define SS -vector -dense $(\text{SPREAD\_SET\_27\_RAO\_RAO}) \text{-end \}}}} \\
\quad \text{\texttt{-define G -linear\_group -PGL 6 F -end \}} \\
\quad \text{\texttt{-define G1 -linear\_group -PGL 7 F -end \}} \\
\quad \text{\texttt{-define S -spread -kernel\_field F \}} \\
\quad \text{\texttt{-group G -k 3 -spread\_set SS \}} \\
\quad \text{\texttt{-end \}} \\
\quad \text{\texttt{-define T -translation\_plane S G G1 -end \}} \\
\quad \text{\texttt{-with T -do -translation\_plane\_activity \}} \\
\quad \text{\texttt{-export\_incma \}} \\
\quad \text{\texttt{-end}}
\]

creates the translation plane from the spread (there is an error message which we can ignore; this is because we did not create the stabilizer of the spread). To compute the 3-rank of the incidence matrix, we issue the following command:

\[
\text{RREF}_{\text{Rao\_Rao\_plane\_incma\_rank}:} \\
\text{\texttt{\$}}(\text{ORBITER}) -v 2 \backslash \\
\quad \text{\texttt{-define F -finite\_field -q 3 -end \}} \\
\quad \text{\texttt{-define v -vector -field F \}} \\
\quad \text{\texttt{-file plane\_incma.csv \}} \\
\quad \text{\texttt{-end \}} \\
\quad \text{\texttt{-with F -do -finite\_field\_activity \}}
The 3-rank turns out to be 271. According to the Moorhouse tables [53], the plane is Moorhouse IV.
12.3 Packings

A packing of $\text{PG}(3, q)$ is a set of pairwise line-disjoint spreads of $\text{PG}(3, q)$ 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 12.4 list Orbiter commands related to the construction of packings with assumed symmetry.

Table 12.5 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 $\text{PG}(3, 4)$. There are three isomorphism types of spreads in $\text{PG}(3, 4)$. The command computes the orbits of each. In total, this gives 5096448 labeled spreads.

```
spreads_PG_3_4:
▷ - mkdir SPREAD_TABLES_4
▷ $(ORBITER) -v 6 \
▷ ▷ -define F -finite_field -q 4 -end \n▷ ▷ -define P -projective_space -n 3 -field F -v 0 -end \n▷ ▷ -define T -spread_table P 2 "0,1,2" "SPREAD_TABLES_4/"
```

There are 21 isomorphism types of spreads in $\text{PG}(3, 5)$. The regular spread has Orbiter catalogue number equal to 12. The following command creates a table of all labeled regular spreads:

```
spreads_PG_3_4:
▷ - mkdir SPREAD_TABLES_4
▷ $(ORBITER) -v 6 \
▷ ▷ -define F -finite_field -q 4 -end \n▷ ▷ -define P -projective_space -n 3 -field F -v 0 -end \n▷ ▷ -define T -spread_table P 2 "0,1,2" "SPREAD_TABLES_4/"
```

There are 155000 regular spreads.
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-H</td>
<td>description</td>
<td>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.</td>
</tr>
<tr>
<td>-N</td>
<td>description</td>
<td>Specify the normalizer of $H$.</td>
</tr>
<tr>
<td>-cliques_on_fixpoint_graph</td>
<td>$s$</td>
<td>Using poset classification, classify the orbits of $N$ on cliques of size $\leq s$ in the graph on fixed points.</td>
</tr>
<tr>
<td>-cliques_on_fixpoint_graph_control</td>
<td>descr</td>
<td>Specify poset classification options related to the classification of cliques on the fixed point graph as in Table 6.3.</td>
</tr>
<tr>
<td>-fixp_clique_types_save_individually</td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>-process_long_orbits</td>
<td>descr</td>
<td>Proceed on to long orbits using Table 12.5.</td>
</tr>
<tr>
<td>-spread_tables_prefix</td>
<td>$P$</td>
<td>Use prefix $P$ to access spread tables.</td>
</tr>
<tr>
<td>-report</td>
<td></td>
<td>Create a report of the classification process.</td>
</tr>
<tr>
<td>-regular_packing</td>
<td></td>
<td>Initialize Klein correspondence and identify (regular) spreads with external lines to the Klein quadric using the polarity of the Klein quadric.</td>
</tr>
</tbody>
</table>

Table 12.4: Orbiter commands related to the construction of packings with assumed symmetry
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-list_of_cases_from_file</td>
<td>fname</td>
<td>Define a subset of cases of fixed point cliques to be worked on. Only the cases listed the given file are considered.</td>
</tr>
<tr>
<td>-split</td>
<td>r m</td>
<td>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.</td>
</tr>
<tr>
<td>-orbit_length</td>
<td>l</td>
<td>Use orbits of length l.</td>
</tr>
<tr>
<td>-clique_size</td>
<td>s</td>
<td>Use exactly s orbits of length l.</td>
</tr>
<tr>
<td>-solution_path</td>
<td>P</td>
<td>Use P as a prefix for all solution files.</td>
</tr>
<tr>
<td>-create_graphs</td>
<td></td>
<td>For each case, create the graph that describes whether two orbits of length l are compatible.</td>
</tr>
<tr>
<td>-solve</td>
<td></td>
<td>Perform clique finding and write solutions to file.</td>
</tr>
<tr>
<td>-read_solutions</td>
<td></td>
<td>Read solutions from file.</td>
</tr>
</tbody>
</table>

Table 12.5: Orbiter commands related to the construction of packings with assumed symmetry related to picking long orbits
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-catalogue</td>
<td>$i$</td>
<td>Create BLT-set number $i$ from the Orbiter catalogue ($i$ is zero-based).</td>
</tr>
<tr>
<td>-family</td>
<td>$F$</td>
<td>Create a BLT-set from family $F$. See Table 12.7 for possibilities for $F$.</td>
</tr>
</tbody>
</table>

Table 12.6: Commands for creating BLT-sets

### 12.4 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 to many other objects in combinatorics and finite geometry. They exist whenever $q$ is odd. BLT-sets have been defined in [4]. It is an interesting problem to classify BLT-sets of $Q(4,q)$ under the orthogonal group. Some references are Law [42], Penttila-Royle [56], Penttila-Law [43, 44], Betten [8], AlAzemi-Betten-Chowdhury [1].

Orbiter can be used to create members of known families of BLT-sets and sets from a catalogue of BLT-sets over small fields. Besides that, Orbiter can be used to classify all BLT-sets for a given value of $q$. We will see how we create known examples of BLT-sets either from the catalogue or from known families. Afterwards, we will consider the problem of classification.

Table 12.6 shows options to create known BLT-sets. Table 12.7 shows options for known families or sporadic sets. For instance, the command

\[ \text{BLT}_11.0: \]
> $(\text{ORBITER}) -v 2 \$
> $ -\text{define F } -\text{finite_field } -q 11 \ -\text{end } \$
> $ -\text{define O } -\text{orthogonal_space } 0 5 F \ -\text{end } \$
> $ -\text{with O } -\text{do } -\text{orthogonal_space_activity } \$
> $ -\text{create_BLT_set } -\text{catalogue } 0 \ -\text{end } \$
> $ -\text{end } \$
> \#pdflatex 0_1.6.2_report.tex
> \#open 0_1.6.2_report.pdf

creates the BLT-set #0 in $Q(4,11)$. The command

\[ \text{BLT}_11\text{.Mondello:} \]
> $(\text{ORBITER}) -v 2 \$
> $ -\text{define F } -\text{finite_field } -q 11 \ -\text{end } \$
> $ -\text{define O } -\text{orthogonal_space } 0 5 F \ -\text{end } \$
> $ -\text{with O } -\text{do } -\text{orthogonal_space_activity } \$

391
<table>
<thead>
<tr>
<th>Command</th>
<th>Condition</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td></td>
<td>Linear BLT-set.</td>
</tr>
<tr>
<td>Fisher</td>
<td></td>
<td>Fisher BLT-set [27].</td>
</tr>
<tr>
<td>Mondello</td>
<td>$q \equiv \pm 1 \mod 10$</td>
<td>Mondello BLT-set due to Penttila [55].</td>
</tr>
<tr>
<td>FTWKB</td>
<td>$q \equiv \pm 2 \mod 3$</td>
<td>Fisher, Thas, Walker [69], Kantor, Betten [13] BLT-set.</td>
</tr>
<tr>
<td>Kantor1</td>
<td></td>
<td>$q = p^e, e &gt; 1$ Kantor’s first family.</td>
</tr>
<tr>
<td>Kantor2</td>
<td></td>
<td>$q \equiv \pm 2 \mod 5$ Kantor’s second family.</td>
</tr>
<tr>
<td>LP_37_72</td>
<td>$q = 37$</td>
<td>BLT-set for $q = 37$ with ago=72 due to Law and Penttila [44].</td>
</tr>
<tr>
<td>LP_37_41a</td>
<td>$q = 37$</td>
<td>First BLT-set for $q = 37$ with ago=4, due to Law and Penttila [44].</td>
</tr>
<tr>
<td>LP_37_41b</td>
<td>$q = 37$</td>
<td>Second BLT-set for $q = 37$ with ago=4, due to Law and Penttila [44].</td>
</tr>
<tr>
<td>LP_71</td>
<td>$q = 71$</td>
<td>BLT-set for $q = 71$ due to Law and Penttila [44].</td>
</tr>
</tbody>
</table>

Table 12.7: Families of BLT-sets

```
▷▷▷ -create_BLT_set -family "Mondello" -end 
▷▷▷ -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:
<table>
<thead>
<tr>
<th>$i$</th>
<th>Rank</th>
<th>Point</th>
<th>$(a, b, c)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>846</td>
<td>$(1, 6, 4, 10, 3)$</td>
<td>$(22, 11, 1)$</td>
</tr>
<tr>
<td>1</td>
<td>851</td>
<td>$(1, 5, 7, 10, 3)$</td>
<td>$(22, 110, 1)$</td>
</tr>
<tr>
<td>2</td>
<td>1234</td>
<td>$(1, 5, 1, 7, 7)$</td>
<td>$(37, 11, 1)$</td>
</tr>
<tr>
<td>3</td>
<td>613</td>
<td>$(1, 6, 10, 5, 1)$</td>
<td>$(73, 110, 1)$</td>
</tr>
<tr>
<td>4</td>
<td>1307</td>
<td>$(1, 1, 3, 8, 5)$</td>
<td>$(59, 36, 1)$</td>
</tr>
<tr>
<td>5</td>
<td>1418</td>
<td>$(1, 3, 9, 6, 10)$</td>
<td>$(95, 36, 1)$</td>
</tr>
<tr>
<td>6</td>
<td>1022</td>
<td>$(1, 9, 5, 10, 2)$</td>
<td>$(99, 96, 1)$</td>
</tr>
<tr>
<td>7</td>
<td>835</td>
<td>$(1, 2, 6, 3, 3)$</td>
<td>$(99, 36, 1)$</td>
</tr>
<tr>
<td>8</td>
<td>950</td>
<td>$(1, 10, 8, 2, 9)$</td>
<td>$(95, 96, 1)$</td>
</tr>
<tr>
<td>9</td>
<td>789</td>
<td>$(1, 8, 2, 4, 4)$</td>
<td>$(59, 96, 1)$</td>
</tr>
<tr>
<td>10</td>
<td>611</td>
<td>$(1, 7, 7, 5, 1)$</td>
<td>$(73, 11, 1)$</td>
</tr>
<tr>
<td>11</td>
<td>1236</td>
<td>$(1, 4, 4, 7, 7)$</td>
<td>$(37, 110, 1)$</td>
</tr>
</tbody>
</table>

Plane intersection type is $4^{18} 3^{148}$
Plane invariant is too big (18 planes)

\[
\begin{array}{c|c}
\rightarrow & 18_1 \\
\hline
12_0 & 6 \\
\end{array}
\quad \begin{array}{c|c}
\downarrow & 18_1 \\
\hline
12_0 & 4 \\
\end{array}
\]

$C_0 = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}_{12}$
$C_1 = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17\}_{18}$

\[
\begin{array}{c|c}
\rightarrow & 18_1 \\
\hline
12_0 & 6 \\
\end{array}
\quad \begin{array}{c|c}
\downarrow & 18_1 \\
\hline
12_0 & 4 \\
\end{array}
\]

$C_0 = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}_{12}$
$C_1 = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17\}_{18}$

The classification of BLT-sets is a difficult problem. For recent contributions, see [1, 8, 42].
One approach is by means of the poset of partial BLT-sets. The following command classifies
the poset of partial BLT-sets in $Q(4, 13)$:

**BLT\_13\_deep\_search:**
\[\begin{align*}
&\text{$($ORBITER$) -v 2 \$} \\
&\quad \text{define F \ finite_field \ -q 13 \ -end} \\
&\quad \text{define O \ orthogonal_space \ 0 \ 5 \ F \ -end} \\
&\quad \text{define C \ -BLT_set_classifier \ 0 \ -starter_size \ 14 \ -end} \\
\end{align*}\]
The poset of partial BLT-sets is too big, and there are too many orbits. The technique of classification via substructure can help. Here is an example. We consider the same problem of BLT-sets of order 13. In the beginning, we classify all partial BLT-sets of size 5, and then create colored graphs for each of them:

```
$ORBITER -v 2 \n  define F -finite_field -q 13 -end \n  define O -orthogonal_space 0 5 F -end \n  define C -BLT_set_classifier 0 -starter_size 5 -end \n  with C -do -BLT_set_classify_activity \n     compute_starter \n     problem_label BLT_q13 \n     -W -depth 5 \n  -end \n  with C -do -BLT_set_classify_activity \n     create_graphs \n  -end
```

In the next step, we compute all rainbow cliques in each of the graphs:

```
$ORBITER -v 2 \n  loop L 0 38 1 \n     define G -graph -load BLT_q13_graph_5_%L.bin -end \n     with G -do \n     -graph_theoretic_activity \n     find_cliques -rainbow -target_size 9 -end \n  -end \n```

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:
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 0 -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:

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 0 -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
Then, we compute the stabilizer orbits, which are also known as the flag orbits:

```
BLT_13_isomorph_stabilizer_orbits:
  ▶ $(ORBITER) -v 2 \n  ▶ ▶ -define F -finite_field -q 13 -end \n  ▶ ▶ -define 0 -orthogonal_space 0 5 F -end \n  ▶ ▶ -define C -BLT_set_classifier 0 -starter_size 5 -end \n  ▶ ▶ -with C -do -BLT_set_classify_activity \n  ▶ ▶ ▶ -compute_starter \n  ▶ ▶ ▶ ▶ -problem_label BLT_q13 \n  ▶ ▶ ▶ ▶ -W -depth 5 \n  ▶ ▶ ▶ ▶ -end \n  ▶ ▶ ▶ -end \n  ▶ ▶ -with C -do -BLT_set_classify_activity \n  ▶ ▶ ▶ -isomorph \n  ▶ ▶ ▶ ▶ -prefix_iso "./BLT_q13" \n  ▶ ▶ ▶ ▶ -use_database_for_starter \n  ▶ ▶ ▶ ▶ -compute_orbits \n  ▶ ▶ ▶ ▶ -list_of_cases BLT_q13_list_of_cases_5_0_1.csv \n  ▶ ▶ ▶ ▶ -solution_prefix "" \n  ▶ ▶ ▶ ▶ -base_fname "BLT_q13_graph" \n  ▶ ▶ ▶ ▶ -end \n  ▶ ▶ ▶ -end
```

Finally, we perform isomorphism testing:

```
BLT_13_isomorph_testing:
  ▶ $(ORBITER) -v 4 \n  ▶ ▶ -define F -finite_field -q 13 -end \n  ▶ ▶ -define 0 -orthogonal_space 0 5 F -end \n  ▶ ▶ -define C -BLT_set_classifier 0 -starter_size 5 -end \n  ▶ ▶ -with C -do -BLT_set_classify_activity \n  ▶ ▶ ▶ -compute_starter \n  ▶ ▶ ▶ ▶ -problem_label BLT_q13 \n  ▶ ▶ ▶ ▶ -W -depth 5 \n  ▶ ▶ ▶ ▶ -end \n  ▶ ▶ ▶ ▶ -end \n  ▶ ▶ -with C -do -BLT_set_classify_activity \n  ▶ ▶ ▶ -isomorph \n  ▶ ▶ ▶ ▶ -prefix_iso "./BLT_q13" \n  ▶ ▶ ▶ ▶ -use_database_for_starter \n  ▶ ▶ ▶ ▶ -isomorph_testing \n  ▶ ▶ ▶ ▶ -solution_prefix "" \n  ▶ ▶ ▶ ▶ -base_fname "BLT_q13_graph" \n  ▶ ▶ ▶ ▶ -end \n```

396
This last command results in three isomorphism types of BLT-sets of order 13.
Chapter 13

Graph Theory

13.1 Creating Graphs

Tables 13.1-13.2 show Orbiter commands to create graphs.

For instance, the command

\[
\text{Cycle\_graph\_13:}
\]

\[
\begin{align*}
\text{\& (ORBITER) -v 2 } & \\
\text{\&\& -define Gamma -graph } & \\
\text{\&\&\& -cycle 13 } & \\
\text{\&\&\&\& -end } & 
\end{align*}
\]

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:

\[
\text{TRIANGLE\_GRAPH} = "0,1,1\n1,0,1\n1,1,0"
\]

And here is the command to create the csv file from the makefile variable and to create the graph from the csv file:

\[
\text{make\_triangle\_graph:}
\]

\[
\begin{align*}
\text{\& echo $(TRIANGLE\_GRAPH) >triangle\_graph.csv } & \\
\text{\& (ORBITER) -v 6 } & \\
\text{\&\& -define G -graph } & \\
\text{\&\&\& -load_csv_no_border } & \\
\text{\&\&\&\& triangle\_graph.csv } & 
\end{align*}
\]
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-load</td>
<td>filename</td>
<td>Read a graph from file in Orbiter format.</td>
</tr>
<tr>
<td>-load_csv_no_border</td>
<td>filename</td>
<td>Read a graph from a csv file. Ignore the first row and first column.</td>
</tr>
<tr>
<td>-load_dimacs</td>
<td>filename</td>
<td>Read a graph from file in dimacs format.</td>
</tr>
<tr>
<td>-edge_list</td>
<td>n list-of-edges</td>
<td>Create a graph on n vertices from a list of edges as ranked pairs.</td>
</tr>
<tr>
<td>-edges_as_pairs</td>
<td>n edges-as-pairs</td>
<td>Create a graph on n vertices from a list of edges as pairs.</td>
</tr>
<tr>
<td>-cycle</td>
<td>n</td>
<td>Cycle graph on n vertices.</td>
</tr>
<tr>
<td>-Hamming</td>
<td>n q</td>
<td>Hamming graph $H(n, q)$</td>
</tr>
<tr>
<td>-Johnson</td>
<td>n k s</td>
<td>Johnson graph</td>
</tr>
<tr>
<td>-Paley</td>
<td>q</td>
<td>Paley graph</td>
</tr>
<tr>
<td>-Sarnak</td>
<td>p q</td>
<td>Lubotzky-Phillips-Sarnak graph [46]</td>
</tr>
<tr>
<td>-Schlaefli</td>
<td>q</td>
<td>Schlaefli graph</td>
</tr>
<tr>
<td>-Shrikhande</td>
<td></td>
<td>Shrikhande graph</td>
</tr>
<tr>
<td>-Winnie_Li</td>
<td>q i</td>
<td>Winnie-Li graph [45]</td>
</tr>
<tr>
<td>-Grassmann</td>
<td>n k q r</td>
<td>Grassmann graph</td>
</tr>
<tr>
<td>-coll_orthogonal</td>
<td>$\epsilon$ d q</td>
<td>Collinearity graph of $O^\epsilon(d, q)$</td>
</tr>
<tr>
<td>-trihedral_pair_disjointness_graph</td>
<td></td>
<td>Trihedral pair disjointness graph</td>
</tr>
<tr>
<td>-non_attacking_queens_graph</td>
<td>n</td>
<td>Create the graph for non-attacking queens on a $n \times n$ chess board.</td>
</tr>
</tbody>
</table>

Table 13.1: Orbiter commands to define graphs (Part 1)
### Command Table

<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-subset</td>
<td>label labeltex subset</td>
<td>Define vertex coloring with two colors based on a subset of vertices.</td>
</tr>
<tr>
<td>-disjoint_sets_graph</td>
<td>fname</td>
<td>Define a graph on a system of sets. Adjacency is when two sets are disjoint. The sets are taken from the given file.</td>
</tr>
<tr>
<td>-orbital_graph</td>
<td>$G i$</td>
<td>Define orbital graph from the $i$-th orbit of the group $G$ acting on pairs.</td>
</tr>
<tr>
<td>-collinearity_graph</td>
<td>inc-matrix</td>
<td>Collinearity graph of the given incidence matrix.</td>
</tr>
<tr>
<td>-chain_graph</td>
<td>P1 P2</td>
<td>Chain graph with respect to the partitions P1 and P2.</td>
</tr>
<tr>
<td>-Cayley_graph</td>
<td>$G$ gens</td>
<td>Cayley graph with respect to group $G$ and generating set $gens$.</td>
</tr>
</tbody>
</table>

Table 13.2: Orbiter commands to define graphs (Part 2)

```
>>> -end
```

This will create the three-cycle graph.

The command

**Chain_232:**
```
>>> $(ORBITER) -v 2 \
>>> -define P1 -vector -dense 2,3,2 -end \n>>> -define P2 -vector -dense 2,3,2 -end \n>>> -define Gamma -graph \n>>> -chain_graph P1 P2 \n>>> -end
```

creates the chain graph with respect to the partitions (2, 3, 2) and (2, 3, 2).

The command

**Paley_13_graph:**
```
>>> $(ORBITER) -v 2 \
>>> -define Gamma -graph -Paley 13 -end 
```

creates the Paley graph on 13 vertices.

The command
triheiral_pair_graph:
\[\text{\$ORBITER) -v 2 \ -define Gamma \ -graph -triheiral_pair_disjointness_graph \ -end}\]

creates the graph of triheiral pairs. Two vertices are adjacent if the associated triheiral pairs are line-disjoint.

The command

small_graph:
\[\text{\$ORBITER) -v 2 \ -define G \ -graph -edges_as_pairs \ -end}\]

creates a graph by listing the edges in pairs. In this case, the graph

![Graph Image]

is created.

The command

petersen:
\[\text{\$ORBITER) -v 2 \ -define G \ -graph -Johnson 5 2 0 -end}\]

creates the Petersen graph.

The command
Johnson_6_2_0:
▷ $(\text{ORBITER}) -v 2 \$
▷ ▷ -define G -graph -Johnson 6 2 0 -end

creates the Johnson graph $J(6, 2, 0)$.

The command

Hamming_graph_3:
▷ $(\text{ORBITER}) -v 2 \$
▷ ▷ -define G -graph -Hamming 3 2 -end

creates the Hamming graph of order 3.

There is a graph on 315 vertices that arises from the Cohen-Tits near octagon (see [15]). The graph was first constructed in [18] 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 [6]. 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:

HJ_graph:
▷ $(\text{ORBITER}) -v 6 \$
▷ ▷ -define G -graph \$
▷ ▷ ▷ -load_csv_no_border \$
▷ ▷ ▷ halljanko315.csv \$
▷ ▷ ▷ -end

In Section 15.7, 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:

HJ315_orbital_graph_3: halljanko315_gens.csv
▷ $(\text{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 \$

403
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-complement</td>
<td></td>
<td>Complementary graph.</td>
</tr>
<tr>
<td>-distance_2</td>
<td></td>
<td>Distance two graph: Two vertices are adjacent if and only if they have distance two in the original graph.</td>
</tr>
</tbody>
</table>

Table 13.3: Orbiter commands to modify graphs

Table 13.3 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 $\Gamma$, the distance 2 graph $\Delta$ has the same vertices as $\Gamma$, and two vertices in $\Delta$ are adjacent if and only if the distance in $\Gamma$ is two. The following command creates the distance 2 graph of the Cohen-Tits graph.

```
HJ_d2_graph:
 $\langle$ORBITER$\rangle$ -v 6 \
  $\langle$define G -graph \ 
  $\langle$define F -finite_field -q 11 -end \ 
  $\langle$define S -vector -dense \ 
  $\langle$"1,1, 1,4, 1,7, 1,10" -end \ 
  $\langle$define G -linear_group -AGL 1 F \ 
  $\langle$define Gamma -graph \ 
  $\langle$Cayley_graph G S \ 
  $\langle$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 $(a,b)$.

```
Cayley_Z11_1mod3:
 $\langle$ORBITER$\rangle$ -v 2 \
  $\langle$define F -finite_field -q 11 -end \ 
  $\langle$define S -vector -dense \ 
  $\langle$"1,1, 1,4, 1,7, 1,10" -end \ 
  $\langle$define G -linear_group -AGL 1 F \ 
  $\langle$define subgroup_by_generators "Z11" 11 1 "1,1" \ 
  $\langle$end \ 
  $\langle$define Gamma -graph \ 
  $\langle$Cayley_graph G S \ 
  $\langle$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:

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:

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
```
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-find_cliques</td>
<td>options</td>
<td>Find all cliques. See Section 13.4.</td>
</tr>
<tr>
<td>-export_magma</td>
<td></td>
<td>Export to Magma [14].</td>
</tr>
<tr>
<td>-export_maple</td>
<td></td>
<td>Export to Maple [49].</td>
</tr>
<tr>
<td>-export_csv</td>
<td></td>
<td>Export to csv-file.</td>
</tr>
<tr>
<td>-export_graphviz</td>
<td></td>
<td>Export to graphviz-file.</td>
</tr>
<tr>
<td>-print</td>
<td></td>
<td>Print the graph.</td>
</tr>
<tr>
<td>-sort_by_colors</td>
<td></td>
<td>Sort the vertices by color classes.</td>
</tr>
<tr>
<td>-split</td>
<td>file</td>
<td>Split the graph into subgraphs.</td>
</tr>
<tr>
<td>-split_by_starters</td>
<td>file</td>
<td>Split the graph into subgraphs according to starters.</td>
</tr>
<tr>
<td>-split_by_clique</td>
<td>label clique</td>
<td>Compute the neighborhood graph of the given clique.</td>
</tr>
<tr>
<td>-save</td>
<td></td>
<td>Save the graph to file in binary format.</td>
</tr>
<tr>
<td>-automorphism_group</td>
<td></td>
<td>Compute the automorphism group and write a report. See Section 15.7.</td>
</tr>
<tr>
<td>-properties</td>
<td></td>
<td>Compute properties of the graph.</td>
</tr>
<tr>
<td>-eigenvalues</td>
<td></td>
<td>Compute the eigenvalues of the graph.</td>
</tr>
<tr>
<td>-draw</td>
<td></td>
<td>Draw the graph.</td>
</tr>
</tbody>
</table>

Table 13.4: Graph Theoretic Activities

### 13.2 Graph Theoretic Activities

Graph theoretic activities allow us to perform tasks on graphs. Table 13.4 shows the commands for graph theoretic activities. These are activities that can be applied to objects of type graph.

Continuing the example of the three-cycle, the command

```
triangle_graph_properties:
  ▶ echo $(TRIANGLE_GRAPH) >triangle_graph.csv
  ▶ $(ORBITER) -v 6 \
  ▶ ▶ -define G -graph \n  ▶ ▶ ▶ -load_csv_no_border \n  ▶ ▶ ▶ triangle_graph.csv \n  ▶ ▶ -end \n  ▶ ▶ -with G -do \n  ▶ ▶ ▶ -graph_theoretic_activity -properties \n  ▶ ▶ ▶ -end
```

406
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 sequence is printed as $2^3$.

We can export the adjacency matrix and create a bitmap drawing like so:

```bash
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:

```bash
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:
The energy is 11.5175
Eigenvalues: $\lambda_i$
Laplace eigenvalues: $\theta_i$

<table>
<thead>
<tr>
<th>$i$</th>
<th>$\lambda_i$</th>
<th>$\theta_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>3.87939</td>
</tr>
<tr>
<td>1</td>
<td>1.53209</td>
<td>3.87939</td>
</tr>
<tr>
<td>2</td>
<td>1.53209</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>0.347296</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>0.347296</td>
<td>1.6527</td>
</tr>
<tr>
<td>5</td>
<td>−1</td>
<td>1.6527</td>
</tr>
<tr>
<td>6</td>
<td>−1</td>
<td>0.467911</td>
</tr>
<tr>
<td>7</td>
<td>−1.87939</td>
<td>0.467911</td>
</tr>
<tr>
<td>8</td>
<td>−1.87939</td>
<td>−2.26243e−16</td>
</tr>
</tbody>
</table>

The command

Paley_13_draw:
$\$(ORBITER) -v 2 \$
$\$ -define Gamma -graph -Paley 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 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 13.1 using the external tool graphviz.

Let us consider the Cayley graphs from Section 13.1. Here is a command that draws the first graph and computes the eigenvalues:

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 \$

408
The drawing is shown in Figure 13.1. Let us draw the Cayley graph of Sym(5) with respect to the Coxter generators:

```plaintext
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
```

Figure 13.1: The Cayley graph in $\mathbb{Z}_{11}$
The drawing is shown in Figure 13.2.

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:

```
PGO_5_2_collinearity_graph: 0_5_2_incidence_matrix.csv
```

```
$ (ORBITER) -v 3 \
  -define Inc -vector -file 0_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 collinear with 6 other points.
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-girth</td>
<td>$d$</td>
<td>Girth at least $d$</td>
</tr>
<tr>
<td>-regular</td>
<td>$r$</td>
<td>Regular of degree $r$</td>
</tr>
<tr>
<td>-no_transmitter</td>
<td></td>
<td>Tournament without transmitter (requires -tournament)</td>
</tr>
</tbody>
</table>

Table 13.5: Options for classifying graphs

### 13.3 Classification of Graphs and Tournaments

Table 13.5 lists the Orbiter commands to classify graphs and tournaments. The following command classifies all graphs on 5 vertices:

```plaintext
graph_classify_5:
  $(ORBITER) -v 2 \
  -orbiter_path $(ORBITER_PATH) \
  -define GC -graph_classification \n  -n 5 \n  -poset_classification_control \n  -problem_label graphs_v5 \n  -depth 10 \n  -draw_options -radius 250 \n  -embedded \n  -end \n  -end \n  -with GC -do \n  -graph_classification_activity \n  -list_graphs_at_level 5 5 \n  -end \n  -with GC -do \n  -graph_classification_activity \n  -draw_options \n  -radius 300 -nodes_empty \n  -line_width 1.5 \n  -scale 0.1 \n  -end \n  -draw_graphs_at_level 5 \n  -end \n  -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 graph drawings are shown in Figure 13.3.

The next command classifies all tournaments on 4 vertices:

tournament\_classify\_4:
\begin{verbatim}
  $(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 \\
  -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
\end{verbatim}
pdflatex tournament\_4\_level\_6\_reps.tex
open tournament\_4\_level\_6\_reps.pdf

There are four tournaments. The graph drawings are shown in Figure 13.3.

The next command classifies all cubic graphs on 8 vertices:

graph\_classify\_8\_r3:
\begin{verbatim}
  $(ORBITER) -v 3 \\
\end{verbatim}
There are six cubic graphs. The graph drawings are shown in Figure 13.5.
### Table 13.6: Clique Finding Options

<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-rainbow</td>
<td></td>
<td>Find all rainbow cliques. The size of the cliques is the number of vertex colors.</td>
</tr>
<tr>
<td>-target_size</td>
<td>s</td>
<td>Find all cliques of size $s$.</td>
</tr>
<tr>
<td>-weighted</td>
<td>$s$</td>
<td>Find weighted cliques.</td>
</tr>
<tr>
<td>-Sajeeb</td>
<td></td>
<td>Use the implementation by Sajeeb Chowdhury.</td>
</tr>
<tr>
<td>-output_file</td>
<td>fname</td>
<td>Write cliques to the named file.</td>
</tr>
<tr>
<td>-restrictions</td>
<td>$l$ $r$ $m$</td>
<td>Restricted search: At level $l$, restrict to all branches congruent to $r$ modulo $m$. Here, $0 \leq r &lt; m$.</td>
</tr>
</tbody>
</table>

#### 13.4 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 $\Gamma$. Finding and classifying cliques in graphs is important for many applications of graph theory. Orbiter can help. The command `-find_cliques` command from Table 13.4 can be used to find all cliques in a graph. Additional options for this command are shown in Table 13.6. For instance, the cliques of size 3 in the graph `graph_v5_e7.colored_graph` from Section 13.1 can be found using:

```
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
```

This command finds three cliques of size 3.

It is also possible to classify all cliques under the automorphism group of the graph. This is a multi-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:

```
Paley_13_aut:
  ▶ $(ORBITER) \ -v \ 2 \$
  ▶ ▶ -define Gamma -graph -Paley 13 \ -end $
```

415
The command writes a file `Paley_13_group.makefile`, shown below:

```
Paley_13:
  $(ORBITER_PATH)orbiter.out -v 2 \
  -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 \

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:

```
Paley_13_cliques_classify:
  $(ORBITER) -v 4 \
  -define Control -poset_classification_control \
  -W \
  -problem_label Paley13_cliques \
  -clique_test Gamma \
  -depth 5 \
  -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 13 -end \
  -define Orb -orbits -group G \
  -on_subsets 5 Control \
  -end
```

For comparison, the command

```
Paley_13_cliques_all:
  $(ORBITER) -v 10 \
  -define Gamma -graph -Paley 13 -end \
  -with Gamma -do \
  -graph_theoretic_activity \
```
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 13.1. 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:

HJ64_cliques5:

```bash
$ (ORBITER) -v 6
> -define Gamma -graph
> -load
> GroupPerm315.Orbital3.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:

HJ64_cliques5_classify:

```bash
$ (ORBITER) -v 6
> -define Control -poset_classification_control
> -W
> -problem_label HJ64_cliques
> -clique_test Gamma
> -depth 5
> -end
> -define Gamma -graph
> -load
> GroupPerm315.Orbital3.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:

```
PGO 5.2 cliques: 0.5.2.incidence_matrix.csv
▷ $(ORBITER) -v 3 \\
▷ ▷ -define Inc -vector -file 0.5.2.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.
Chapter 14

Combinatorial Objects

14.1 Combinatorial Objects

Combinatorial objects are objects that are defined by means of a finite group action. The
isomorphism problem for combinatorial objects is the question to decide whether two objects
of the same type belong to the same orbit under the relevant group action. Orbiter offers
a unified treatment of such questions for various types of objects. The main tool is the
computation of a canonical form, as well as the automorphism group.

Combinatorial objects are coded as sequences of integers. Each type of object has its own
coding. Coding of objects as integer sequences allows easy handling of objects. For instance,
objects can be specified in a command line argument, or they can be stored in a file. Large
numbers of objects can be stored in files.

In order to apply Orbiter commands, an input stream is defined. An input stream is a
sequence of objects, all of the same kind. The objects can be defined using any of the
commands listed in Table 14.1. The file types will be discussed in more detail in the next
section. Here are some examples. First, we create the Hirschfeld surface. Since the Hirschfeld
surface is a cubic surface, the object is defined using point ranks in the relevant projective
space as described in Section 4.2. For instance, the Hirschfeld surface in PG(3,4) is defined
as 45 points, coded as 45 integers which are point ranks. A makefile variable is employed to
define the set. The makefile variable is then used to define a set-object:

HIRSCHFELD_SURFACE_Q4_SET_OF_POINTS="0,1,2,3,4,5,6,7,8,9,\n10,11,12,13,14,23,26,27,30,31,34,35,38,39,42,47,48,51,52,\n53,54,59,60,61,62,67,68,69,70,75,76,79,80,81,82"

Hirschfeld_q4_from_set:
▷ $(ORBITER) -v 4 \n▷ ▷ -define H -set -here \n▷ ▷ ▷ $(HIRSCHFELD_SURFACE_Q4_SET_OF_POINTS) \n▷ ▷ ▷ -end \n▷ ▷ -define C -combinatorial_objects \

419
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-set_of_points</td>
<td></td>
<td>A set consisting of points.</td>
</tr>
<tr>
<td>-set_of_lines</td>
<td></td>
<td>A set consisting of lines.</td>
</tr>
<tr>
<td>-set_of_points_and_lines</td>
<td></td>
<td>A set consisting of points and a second set consisting of lines.</td>
</tr>
<tr>
<td>-set_of_packing</td>
<td></td>
<td>A set of packings.</td>
</tr>
<tr>
<td>-file_of_points</td>
<td></td>
<td>A set consisting of points read from file.</td>
</tr>
<tr>
<td>-file_of_lines</td>
<td></td>
<td>A set consisting of lines read from file.</td>
</tr>
<tr>
<td>-file_of_packings</td>
<td></td>
<td>A set consisting of packings read from file.</td>
</tr>
<tr>
<td>-file_of_packings_through_spread_table</td>
<td></td>
<td>A file of packings.</td>
</tr>
<tr>
<td>-file_of_point_set</td>
<td></td>
<td>A file containing point sets.</td>
</tr>
<tr>
<td>-file_of_designs</td>
<td></td>
<td>A file containing designs or large sets.</td>
</tr>
<tr>
<td>-file_of_incidence_geometries</td>
<td>v b f</td>
<td>A file of incidence geometries defined by their set of flags. Here, ( v ) is the number of points, ( b ) is the number of blocks and ( f ) is the number of flags.</td>
</tr>
<tr>
<td>-file_of_incidence_geometries_by_row_ranks</td>
<td></td>
<td>A file describing incidence geometries defined by their row ranks.</td>
</tr>
<tr>
<td>-incidence_geometry</td>
<td>flags v b f</td>
<td>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.</td>
</tr>
<tr>
<td>-incidence_geometry_by_row_ranks</td>
<td></td>
<td>An incidence geometry defined by row ranks.</td>
</tr>
<tr>
<td>-from_parallel_search</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 14.1: Defining Combinatorial Objects
The next example creates the two hyperovals in PG(2,16). The hyperovals are stored in makefile variables:

\[
\begin{align*}
\text{HYPEROVAL}_{16,144} &= "0, 1, 2, 3, 52, 67, 89, 106, 126, \ldots, 141, 159, 176, 184, 199, 220, 235, 245, 262" \\
\text{HYPEROVAL}_{16,16320} &= "0, 1, 2, 3, 52, 70, 83, 109, 127, \ldots, 139, 156, 174, 186, 199, 217, 229, 256, 264"
\end{align*}
\]

hyperoval_16_create:
\[
\begin{align*}
&\text{\$\{ORBITER\} -v 2 \ \\
&\quad -\text{define C -combinatorial\_objects} \ \\
&\quad -\text{set\_of\_points } \text{\$\{HYPEROVAL\}_{16,16320}} \ \\
&\quad -\text{set\_of\_points } \text{\$\{HYPEROVAL\}_{16,144}} \ \\
&\quad -\text{end} \\
\end{align*}
\]

In the next example, we read the points of an elliptic curve from file (see Section 4.2):

\[
\begin{align*}
\text{EC\_read: } &\text{elliptic\_curve\_b1\_c3\_q11.txt} \\
&\text{\$\{ORBITER\} -v 4 \ \\
&\quad -\text{define C -combinatorial\_objects} \ \\
&\quad -\text{file\_of\_points elliptic\_curve\_b1\_c3\_q11.txt} \ \\
&\quad -\text{end} \\
\end{align*}
\]

In the next example, we read a packing, using a previously defined table of spreads, stored in a csv file.

\[
\begin{align*}
\text{PG\_3.5\_assume\_31\_read:} \\
&\text{\$\{ORBITER\} -v 2 \ \\
&\quad -\text{define C -combinatorial\_objects} \ \\
&\quad -\text{file\_of\_packings\_through\_spread\_table} \ \\
&\quad -\text{H31\_packings.csv} \ \\
&\quad -\text{SPREAD\_TABLES\_5\_REG/spread\_25\_spreads.csv} \ \\
&\quad -\text{5} \ \\
&\quad -\text{end} \\
\end{align*}
\]

The following command reads a file of large sets of designs:

421
LS_AG_2_3_read:
  ▶ $\text{(ORBITER)} -v 2 \n  ▶ ▶ -define C -combinatorial_objects \n  ▶ ▶ ▶ -file_of_designs \n  ▶ ▶ ▶ solutions.csv 9 84 3 12 \n  ▶ ▶ end

The next command reads incidence geometries from a file containing the flags:

geo_7_3_read:
  ▶ $(\text{ORBITER}) -v 10 \n  ▶ ▶ -draw.incidence_structure_description \n  ▶ ▶ ▶ -width 60 -with_10 6 -end \n  ▶ ▶ -define C -combinatorial_objects \n  ▶ ▶ ▶ -file_of_incidence_geometries \n  ▶ ▶ ▶ ▶ 7_3.inc 7 7 21 \n  ▶ ▶ -end

The next command creates incidence geometries from a file containing row-ranks:

Desargues_path_lex_least_read:
  ▶ echo $(\text{DESARGUES_PATH_LEX_LEAST}) >Desargues_path_lex_least.inc
  ▶ $(\text{ORBITER}) -v 10 \n  ▶ ▶ -draw.incidence_structure_description \n  ▶ ▶ ▶ -width 60 -with_10 6 -end \n  ▶ ▶ -define C -combinatorial_objects \n  ▶ ▶ ▶ -file_of_incidence_geometries_by_row_ranks \n  ▶ ▶ ▶ ▶ Desargues_path_lex_least.inc 10 10 3 \n  ▶ ▶ -end
14.2 Encoding Combinatorial Objects

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 in a numerical form. One number is one incidence. Another way is by describing the incidence matrix in a row-by-row fashion, using ranks of $k$-subsets. This assumes that the number of incidences per row is constant over all rows. Yet another way is by listing the columns of the incidence matrix, again using ranks of $k$-subsets. This version requires that the column sums of the incidence matrix are constant. Let us go over some of these file formats, using small examples to illustrate the ideas informally.

Suppose we want to work with the Pasch configuration. This is the configuration of 6 points and 4 lines shown in Figure 14.1. 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 shown in Figure 14.2. Row labels are on the left, column labels are on top. The $(i, j)$-entry is one if $P_i$ lies on $\ell_j$, and it is zero otherwise. 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 (see Figure 14.3). 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
<table>
<thead>
<tr>
<th></th>
<th>$\ell_0$</th>
<th>$\ell_1$</th>
<th>$\ell_2$</th>
<th>$\ell_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_0$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$P_1$</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$P_2$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$P_3$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$P_4$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$P_5$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 14.2: The incidence matrix of the Pasch configuration

<table>
<thead>
<tr>
<th></th>
<th>$\ell_0$</th>
<th>$\ell_1$</th>
<th>$\ell_2$</th>
<th>$\ell_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_0$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>$P_1$</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>$P_2$</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>$P_3$</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>$P_4$</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
</tr>
<tr>
<td>$P_5$</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
</tr>
</tbody>
</table>

Figure 14.3: Row-major ordering of the flag space

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

```
geo_pasch_read:
  $\$(ORBITER) -v 10 \n  $\$ $\$ -define C -combinatorial_objects \n  $\$ $\$ $\$ -file_of_incidence_geometries \n  $\$ $\$ $\$ $\$ pasch.inc 6 4 12 \n  $\$ $\$ $\$ -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:
geo_pasc_he_given:
▷ $(ORBITER) -v 10 \\
▷ ▷ -define C -combinatorial_objects \\
▷ ▷ ▷ -incidence_geometry \\
▷ ▷ ▷ ▷ "0,1,4,6,8,11,13,14,17,19,22,23" \\
▷ ▷ ▷ ▷ 6 4 12 \\
▷ ▷ -end
Chapter 15

Canonical Forms with Nauty

15.1 Overview of Canonical Forms

What is a combinatorial object? For the purposes of Orbiter, it is any kind of object that has a representation as a set of sets, all drawn from an underlying finite set. We allow colorings of the elements of the underlying set and of the sets in the set-system. The representation is functorial. Isomorphisms between the combinatorial objects must correspond to color preserving bijections of the set-representation and vice-versa. Under these conditions, the isomorphisms between combinatorial objects and automorphisms from one object to itself correspond to the mappings between the associated set representations.

The set-representation of combinatorial objects can help us computationally approach the isomorphism problem. We simply search for color-preserving bijections that take the set-representation of the object to the set-representation of the other object. Automorphisms can be found by mapping the set-representation of the object to itself.

Canonical labelings can be used to eliminate the need to do pairwise isomorphism testing. This is particularly helpful if the number of objects to test is large. If we have $N$ objects, say, then pairwise isomorphism testing requires $\binom{N}{2}$ tests. With canonical forms, we only need $N$ canonical forms computations.

Sets of sets are incidence structures. The Levi graph of an incidence structure is the bipartite graph whose two classes correspond to rows and columns of the incidence matrix. The partition of the set system (underlying point set and set of sets) reduces to a coloring of the vertices of the graph. Two combinatorial objects are isomorphic if and only if the associated colored Levi graphs are isomorphic in the sense of graph isomorphism. This allows to settle many questions associated with combinatorial object, such as isomorphism testing and determining the automorphism group.

A canonical labeling of a graph is a bijection of the vertices. The property is that if two graphs are isomorphic, then the graphs become identical if the canonical labeling permutation is applied (each graph has its own canonical labeling). It is therefore important to compute canonical forms. If there is a vertex coloring, we implicitly assume that the canonical labeling preserves the coloring.
The graph theory package Nauty [51] provides a canonical form algorithm for graphs. Using the Levi graph construction, this technique allows to solve the isomorphism problem for combinatorial objects in the more general sense just defined.

The technique of isomorphism testing can be lifted to combinatorial objects in projective spaces or other types of finite incidence geometries. For instance, arcs in projective planes have been classified this way (cf. [2]).

Table 15.1 list Orbiter commands related to canonical labelings of combinatorial objects.
15.2 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 referer to the file containing the Orbiter point ranks of points on the curve.

```
EC.canon: elliptic_curve_b1_c3_q11.txt
  $(ORBITER) -v 3 \n  -define C -combinatorial_objects \n  -file_of_points elliptic_curve_b1_c3_q11.txt \n  -end \n  -define F -finite_field -q 11 -end \n  -define P -projective_space -n 2 -field F -v 0 -end \n  -with C -do \n  -combinatorial_object_activity \n  -canonical_form_PG P \n  -classification_prefix EC \n  -label EC \n  -save_ago \n  -max_TDO_depth 4 \n  -end \n  -report \n  -prefix EC \n  -export_flag_orbits \n  -show_TDO \n  -show_TDA \n  -dont_show_incidence_matrices \n  -export_group_GAP \n  -end \n  -end
```

```
pdflatex EC_classification.tex
open EC_classification.pdf
$(ORBITER) -v 2 -draw_matrix \n -input_csv_file EC_object0_TDA_flag_orbits.csv \n -secondary_input_csv_file EC_object0_TDA.csv \n -box_width 20 -bit_depth 24 \n -end
open EC_object0_TDA_flag_orbits_draw.bmp
```

Orbiter shows that the curve has a collineation stabilizer of order 6, generated by
The following example computes the canonical form and the automorphism group of the Hirschfeld surface in \(\text{PG}(3, 4)\). Using indexing of points in \(\text{PG}(3, 4)\), 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.

```
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
```

```
Hirschfeld_q4_c: Hirschfeld_surface_q4.txt
  $(ORBITER) -v 6 \n  -define C -combinatorial_objects \n  -file_of_points Hirschfeld_surface_q4.txt \n  -end \n  -define F -finite_field -q 4 -end \n  -define P -projective_space -n 3 -field F -v 0 -end \n  -with C -do \n  -combinatorial_object_activity \n  -canonical_form_PG P \n  -classification_prefix Hirschfeld_surface_q4 \n  -save_ago \n  -max_TDO_depth 10 \n  -end \n  -report \n  -prefix Hirschfeld_surface_q4 \n  -export_flag_orbits \n  -show_TDO \n  -show_TDA \n  -dont_show_incidence_matrices \n  -export_group_GAP \n  -end \n  -end

pdflatex Hirschfeld_surface_q4_classification.tex
open Hirschfeld_surface_q4_classification.pdf
```

Hirschfeld_q4_set_c:
In the next example, we compute the canonical form of the two hyperovals in PG(2, 16).

hyperoval_16Canonical_form:
$\$(ORBITER) -v 2 \$
$\$ -define C -combinatorial_objects \$
$\$ -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 \$
$\$ -classification_prefix hyperoval_q16 \$
$\$ -label hyperoval_q16 \$
$\$ -save_ago \$
$\$ -save_transversal \$
$\$ -max_TDO_depth 10 \$
$\$ pdflatex hyperoval_q16_classification.tex$
$\$ open hyperoval_q16_classification.pdf
In the next example, we compute the set stabilizers of orbits of $\text{PGL}(4,2)$ on subsets of $\text{PG}(3,2)$, as computed earlier in Section 6.3, using the command $\text{PG}_3_2\_subsets$. These orbits are relevant for Section 7.5. Concerning the work in Dickson [24] only subsets whose size is odd are relevant, so we restrict to those subsets:

**Dickson_sets_stabilizer:**

```
$\text{ORBITER} -v 3 \\
$\text{define C -combinatorial_objects} \\
$\text{-set_of_points "0,1,2,5,6"} \\
$\text{-set_of_points "0,1,2,3,6"} \\
$\text{-set_of_points "0,1,2,3,4"} \\
$\text{-set_of_points "0,1,2,3,8"} \\
$\text{-set_of_points "0,1,2,5,6,7,8"} \\
$\text{-set_of_points "0,1,2,3,5,6,7"} \\
$\text{-set_of_points "0,1,2,3,5,6,9"} \\
$\text{-set_of_points "0,1,2,3,5,6,10"} \\
$\text{-set_of_points "0,1,2,3,5,6,4"} \\
$\text{-set_of_points "0,1,2,3,8,11,13"} \\
$\text{-set_of_points "3,6,9,7,10,12,8,11,13,14,4"} 
```
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:

`ovoid_q8_canon: ovoid_q8.txt

```
$(ORBITER) -v 6 \\
  -define C -combinatorial_objects \\
  -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 \\
  -classification_prefix ovoid \\
  -label ovoid \\
  -save_ago \\
  -max_TDO_depth 4 \\
  -end \\
  -report \\
  -prefix ovoid \\
  -show_TDO \\
  -show_TDA \\
  -end \\
  -end
```

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:
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 \( q = 8 \).

```bash
ovoid_ST_q8_canon: ovoid_ST_q8.txt
  $(ORBITER) -v 6 \
  -define C -combinatorial_objects \
  -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 \
  -classification_prefix ovoid_ST \
  -label ovoid_ST \
  -save_ago \
  -max_TDO_depth 4 \
  -end \
  -report \
  -prefix ovoid_ST \
  -show_TDO \
  -show_TDA \
  -dont_show_incidence_matrices \
  -export_group_GAP \
  -end \
  -end
```

We can store the generators in a makefile variable as follows:

```makefile
SUZUKI_8_GENERATORS="\n  1,0,0,0,0,0,1,0,0,0,0,1,0,0,0,1,1, \n  1,0,0,0,0,0,6,0,0,0,0,2,0,0,0,0,3,0, \n  1,0,0,0,1,1,1,0,0,0,1,0,1,0,0,1,0,1,0, \n```

434
We can now recover the Suzuki group using the command:

```plaintext
Suzuki_8:
▷ $\text{ORBITER} -v 6 \\n▷ ▷ -define F -finite_field -q 8 -end \n▷ ▷ -define gens -vector -field F \n▷ ▷ ▷ -compact $(SUZUKI_8_GENERATORS) -end \n▷ ▷ -define G -linear_group -PGGL 4 8 \n▷ ▷ -subgroup_by_generators "Sz8" "87360" 5 gens \n▷ ▷ ▷ -end \n▷ ▷ -with G -do \n▷ ▷ -group_theoretic_activity \n▷ ▷ ▷ -report \n▷ ▷ -end
▷ pdflatex PGGL_4_8_Subgroup_Sz8_87360_report.tex
▷ open PGGL_4_8_Subgroup_Sz8_87360_report.pdf
```
15.3 Canonical Forms of Incidence Geometries

Let us consider system 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 geometry. 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 \( v, b, k \) 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 geometry.

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 \( \text{PG}(2, 2) \), which is a configuration 7_3.

```
geo_7_3.c:
▷ $(ORBITER) -v 10 \n▷ ▷ -draw.incidence_structure_description \n▷ ▷ ▷ -width 60 -with_10 6 -end \n```
A bitmap drawing is produced, as shown in Figure 15.1. The command also produces the following report of the geometry:

### Summary of Orbits

<table>
<thead>
<tr>
<th>Rep</th>
<th># Ago</th>
<th>Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Ago : 168
Figure 15.1: Incidence matrix of the projective plane PG(2, 2)

**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:
\[ g_1 = (3, 5)(4, 6)(8, 9)(12, 13) \text{ of order 2 and with 6 fixed points.} \]
\[ g_2 = (3, 4)(5, 6)(10, 11)(12, 13) \text{ of order 2 and with 6 fixed points.} \]
\[ g_3 = (1, 2)(5, 6)(10, 12)(11, 13) \text{ of order 2 and with 6 fixed points.} \]
\[ g_4 = (1, 3)(2, 4)(7, 8)(11, 12) \text{ of order 2 and with 6 fixed points.} \]
\[ g_5 = (0, 1)(4, 5)(8, 10)(9, 11) \text{ of order 2 and with 6 fixed points.} \]

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:

\[ 21 \quad 1 \]

Anti-Flag orbits:
orbit length : number of orbits of that length:

\[ 28 \quad 1 \]

The following command computes the canonical form and a report of the affine plane \( \text{AG}(2, 3) \), which is a configuration \( 9_4 12_3 \).

\[ \text{AG}_2.3.c : \text{AG}_2.3.inc \]
\[ \text{AG}_2.3.c : \text{AG}_2.3.inc \]

\[ $(ORBITER) -v 2 \]
\[ $(ORBITER) -v 2 \]
\[ -define C -combinatorial_objects \]
\[ -file_of_incidence_geometries \]
\[ -AG_2.3.inc 9 12 36 \]
\[ -end \]
\[ -with C -do \]
\[ -combinatorial_object_activity \]
\[ -canonical_form \]
\[ -classification_prefix AG_2.3 \]
\[ -label AG_2.3 \]
A bitmap drawing is produced, shown in Figure 15.2. Because the geometry is flag transitive, there is only one color being used for the incidence. In fact, all incidences are in black. The geometry is also anti-flag transitive. This can be seen from the fact that there is only one color in the picture for the smaller boxes, which represent anti-flags. Orbiter also produces the following report of the geometry:
Summary of Orbits

<table>
<thead>
<tr>
<th>Rep</th>
<th>#</th>
<th>Ago</th>
<th>Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>432</td>
</tr>
</tbody>
</table>

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:
\[
\begin{array}{c|c}
\rightarrow & 12_1 \\
9_0 & 4 \\
\downarrow & 12_1 \\
9_0 & 3
\end{array}
\]
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.

```
geo_10_3.c:
  $(ORBITER) -v 10 \
  -draw_incidence_structure_description \n  -width 60 -with_10 6 -end \n  -define C -combinatorial_objects \n  -file_of_incidence_geometries 10_3.inc 10 10 30 \n  -end \n  -with C -do \n  -combinatorial_object_activity \n  -canonical_form \n```
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 form it is given as well as the tactical decomposition induced by the orbits of the automorphism group.
Summary of Orbits

<table>
<thead>
<tr>
<th>Rep</th>
<th>#</th>
<th>Ago</th>
<th>Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>9</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>1</td>
<td>24</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>1</td>
<td>120</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>1</td>
<td>6</td>
</tr>
</tbody>
</table>

Ago :2, 3², 4², 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:
g₁ = (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:

1013112141516181719

Canonical labeling:
canonical row = 5
canonical orbit number = 1
Flags: 0,1,2,16,17,18,25,27,29,34,38,40,43,45,51,53,56,62,63,64,70,74,77,82,86,89,91,95,98,

16181719151214101311

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:

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.

```
FILE_24_3_TFC_INC="24 24 72\n10161119121317141815
0  6  1  9  3  8  4  7 10 16 11 19 13 17 14 18
1  2  4  5  9  6  8  7 10 12 11 19 13 14 17 18
2  3  4  6  7  8  9  5 12 10 11 16 14 15 17 18
3  4  5  6  7  8  9 10 11 12 16 13 14 15 17 18
4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22
8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33
19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34
20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37
23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38
24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41
27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42
28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43
29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44
30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45
31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46
32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48
34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51
37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52
38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53
39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54
40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55
41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56
42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57
43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58
44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59
45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61
47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62
48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63
"```
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 in Figure 15.3. Because the geometry is flag transitive, there is only one color being used for the incidence. In fact, all incidences are in black.
15.4 Canonical Forms of Objects from Design Theory

In Section 11.5, designs have been created. In order to compute properties of the design, we export the incidence matrix 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 $\text{PG}(2,3)$:

design $\text{PG}_2(3)_\text{canonical}$:
\begin{verbatim}
$\text{ORBITER}$ -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 \\
$\text{ORBITER}$ -v 3 \\
  -draw_incidence_structure_description \\
  -width 60 -with_10 6 -end \\
  -define C -combinatorial_objects \\
  -file_of_incidence_geometries PG_2_3_inc.txt 13 13 52 \\
  -end \\
  -with C -do \\
    -combinatorial_object_activity \\
    -canonical_form \\
    -classification_prefix PG_2_3 \\
    -label PG_2_3 \\
    -save_ago \\
    -save_transversal \\
  -end \\
  -report \\
  -prefix PG_2_3 \\
  -export_flag_orbits \\
  -show_incidence_matrices \\
  -export_group_GAP \\
  -end \\
end
\end{verbatim}

\begin{verbatim}
pdflatex PG_2_3_classification.tex 
open PG_2_3_classification.pdf 
$\text{ORBITER}$ -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
\end{verbatim}

449
The command

```
$ (ORBITER) -v 10 \n   -draw_incidence_structure_description \n   -width 60 -with 10 6 -end \n   -define C -combinatorial_objects \n   -file_of_incidence_geometries \n   -wreath_product_designs_n4_k2_inc.txt \n   -8 12 24 \n   -end \n   -with C -do \n   -combinatorial_object_activity \n   -canonical_form \n   -classification_prefix wreath_4_2 \n   -label wreath_4_2 \n   -save_ago \n   -save_transversal \n   -end \n   -report \n   -prefix wreath_4_2 \n   -export_flag_orbits \n   -show_incidence_matrices \n   -export_group_GAP \n   -end \n```

computes the automorphism group of the design on 8 points created in Section 11.5. The group is $\text{Sym}(4) \wr \text{Sym}(2)$. The command

```
$ (ORBITER) -v 10 \n   -draw_incidence_structure_description \n   -width 60 -with 10 6 -end \n   -define C -combinatorial_objects \n   -file_of_incidence_geometries \n   -wreath_product_designs_n8_k6_inc.txt \n   -16 3920 23520 \n   -end \n```

450
computes the automorphism group of the design on 16 points created in Section 11.5. The group is Sym(8) \wr Sym(2).

In Section 11.6, some large sets of AG(2, 3) were constructed. The final isomorphism classification is performed using the Nauty interface. A list of combinatorial objects is created, and the \texttt{-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 AG(2, 3).

\texttt{LS\_AG\_2\_3\_solutions\_classify:}
\begin{verbatim}
  $ (ORBITER) -v 2 \n  -draw_incidence_structure_description \n  -width 20 -width_10 2 -end \n  -define C -combinatorial_objects \n  -file_of_designs \n  solutions.csv 9 84 3 12 \n  -end \n  -with C -do \n  -combinatorial_object_activity \n  -canonical_form \n  -save_ago \n  -save_transversal \n  -classification_prefix LS\_AG\_2\_3 \n  -label LS\_AG\_2\_3 \n  -max_TDO_depth 10 \n  -end \n  -report \n  -prefix LS\_AG\_2\_3 \n\end{verbatim}
It turns out that there are exactly two isomorphism types, with automorphism groups of order 54 and 42, respectively.
15.5 Canonical Forms of Linear Codes

Orbiter can compute canonical forms and automorphism groups of codes using Nauty. For linear codes, the semilinear automorphism group can be computed.

Consider the $[3, 2, 2]$ code generated by

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

The semilinear automorphism group can be computed using the following command:

code_3_2_aut:
  $(ORBITER) -v 20 \$
  $(ORBITER) -v 20 \$
  $-define F -finite_field -q 2 -end \$
  $-define F -finite_field -q 2 -end \$
  $-define genma -vector -field F -format 2 \$
  $-define genma -vector -field F -format 2 \$
  $-dense $(CODE_N3_K2_Q2_GENMA) \$
  $-dense $(CODE_N3_K2_Q2_GENMA) \$
  $-end \$
  $-end \$
  $-define P -projective_space -n 1 -field F -v 0 -end \$
  $-define P -projective_space -n 1 -field F -v 0 -end \$
  $-with P -do \$
  $-with P -do \$
  $-projective_space_activity \$
  $-projective_space_activity \$
  $-canonical_form_of_code \$
  $-canonical_form_of_code \$
  $"3_2" genma -save_ago -label "3_2" \$
  $"3_2" genma -save_ago -label "3_2" \$
  $-classification_prefix "3_2" \$
  $-classification_prefix "3_2" \$
  $-end \$
  $-end \$
  $-end \$
  $-end \$
  $pdflatex 3_2_classification.tex$
  $pdflatex 3_2_classification.tex$
  $open 3_2_classification.pdf$
  $open 3_2_classification.pdf$
  $(ORBITER) -v 2 -draw_matrix \$
  $(ORBITER) -v 2 -draw_matrix \$
  $-input_csv_file 3_2_object0_TDA_flag_orbits.csv \$
  $-input_csv_file 3_2_object0_TDA_flag_orbits.csv \$
  $-secondary_input_csv_file 3_2_object0_TDA.csv \$
  $-secondary_input_csv_file 3_2_object0_TDA.csv \$
  $-box_width 16 -bit_depth 24 \$
  $-box_width 16 -bit_depth 24 \$
  $-end \$
  $-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:

<table>
<thead>
<tr>
<th>Rep</th>
<th># Ago</th>
<th>Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1 6 0</td>
</tr>
</tbody>
</table>

Summary of Orbits
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)

<table>
<thead>
<tr>
<th>i</th>
<th>Rank</th>
<th>Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>(1, 0)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>(0, 1)</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>(1, 1)</td>
</tr>
</tbody>
</table>

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:
Decomposition by automorphism group:

\[
\begin{array}{c|c}
40 & 2 \\
\hline
21 & 2 \\
21 & 3 \\
\end{array}
\]

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

```bash
CODE_RM_3_1_GENMA="\n11111111\n01010101\n00110011\n00001111"
```

RM_3_1_group:
- $(ORBITER) -v 2 \
- -define F -finite_field -q 2 -end \n- -define genma -vector -field F -format 4 \n- -compact $(CODE_RM_3_1_GENMA) \n- -end \

455
computes the automorphism group of the Reed-Muller code, of order 1344. It is the affine group AGL(3, 2). 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:

```
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
```

```bash
$(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
```

```bash
$(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
```

456
Figure 15.4: PG(3, 2) with the Reed-Muller code distinguished

▷ open RM.3.1_object0_TDA_flag_orbits_draw.bmp

The drawing in Figure 15.4 is created.

The command

```bash
RS_6_4_7_group:
▷ $(ORBITER) -v 20 \n▷ ▷ -define F -finite_field -q 7 -end \n▷ ▷ -define genma -vector -field F -format 4 \n▷ ▷ ▷ -compact $(CODE_RS_6_4_7) \n▷ ▷ -end \n▷ ▷ -define P -projective_space -n 3 -field F -v 0 -end \n▷ ▷ -with P -do \n▷ ▷ -projective_space_activity \n▷ ▷ ▷ -canonical_form_of_code \n▷ ▷ ▷ ▷ "RS_6" genma -save_ago -label "RS_6" \n▷ ▷ ▷ ▷ -classification_prefix "RS_6" \n▷ ▷ ▷ -end \n▷ ▷ -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)\}
<table>
<thead>
<tr>
<th>i</th>
<th>Rank</th>
<th>Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>(1,0,0,0)</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>(5,1,0,0)</td>
</tr>
<tr>
<td>2</td>
<td>51</td>
<td>(6,5,1,0)</td>
</tr>
<tr>
<td>3</td>
<td>344</td>
<td>(0,6,5,1)</td>
</tr>
<tr>
<td>4</td>
<td>253</td>
<td>(0,0,4,1)</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>(0,0,0,1)</td>
</tr>
</tbody>
</table>

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,

\[
\begin{array}{c|cc}
\rightarrow & 2850 & 1_2 \\
401_0 & 57 & 1
\end{array}
\]

The command

```
GV_n15_k6_d5_group:
- $\text{ORBITER) -v 20}
- $\text{-define F -finite_field -q 2 -end}
- $\text{-define genma -vector -field F -format 6}
- $\text{-compact $(CODE GV_N15_K6)$}
- $\text{-end}
- $\text{-define P -projective_space -n 5 -field F -v 0 -end}
- $\text{-with P -do}
```

458
computes the automorphism group of the Gilbert-Varshamov code from Section 10.8. It has order 12.
15.6 Canonical Forms of General Codes

The command

```
HAMMING_CODE_CODEWORDS="0, 67, 37, 102, 22, 85, 51, 112, 15, 76, 42, 105, 25, 90, 60, 127"
```

Hamming_graph_7_with_Hamming_code:

```
▷ $(ORBITER) \text{-v} 2 \\
▷ \quad -define G -graph -Hamming 7 2 \\
▷ \quad \quad -subset "_Hamming_code" "\_with\_Hamming\_code" \\
▷ \quad \quad $(HAMMING_CODE_CODEWORDS) -end \\
▷ \quad -with G -do \\
▷ \quad -graph_theoretic_activity -export_csv -end \\
▷ \quad -with G -do \\
▷ \quad -graph_theoretic_activity -export_graphviz -end \\
▷ \quad -with G -do \\
▷ \quad -graph_theoretic_activity -save -end \\
▷ \quad -with G -do \\
▷ \quad -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 $2688 = 16 \cdot 168$. 
15.7 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:

```
Cycle_13.aut:
  $(ORBITER) -v 2 \n  -define Gamma -graph -cycle 13 -end \n  -with Gamma -do \n  -graph_theoretic_activity -automorphism_group \n  -end \n```

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 $(2, 3, 2)$.

```
Chain_232.aut:
  $(ORBITER) -v 2 \n  -define P1 -vector -dense 2,3,2 -end \n  -define P2 -vector -dense 2,3,2 -end \n  -define Gamma -graph \n  -chain_graph P1 P2 \n  -end \n  -with Gamma -do \n  -graph_theoretic_activity -automorphism_group \n  -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:

$$(12, 13), $$

$$(3, 4), $$

$$(2, 3),$$
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 -\texttt{load\_dimacs} command. For instance, the following command computes the automorphism group of the Levi graph of the desarguesian projective plane of order 16:

\begin{verbatim}
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 \\
\end{verbatim}

The command shows a group of order 34217164800. As a measurement of the complexity, the number of backtrack nodes in Nauty is recorded:

\begin{verbatim}
nb_backtrack1 = 6
nb_backtrack2 = 134
nb_backtrack3 = 134
nb_backtrack4 = 1
\end{verbatim}

Here, \texttt{nb\_backtrack1} is the number of calls to \texttt{firstpathnode}, \texttt{nb\_backtrack2} is the number of calls to \texttt{othernode}, \texttt{nb\_backtrack3} is the number of calls to \texttt{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:

```
JK_graph_pp16_2:
  ▶ $(ORBITER) -v 2 \n  ▶ ▶ -define Gamma -graph -load_dimacs \n  ▶ ▶ ../JUNTTILA_KASKI/benchmarks/pp/pp16-2 \n  ▶ ▶ -end \n  ▶ ▶ -with Gamma -do \n  ▶ ▶ -graph_theoretic_activity -save -end \n  ▶ ▶ -with Gamma -do \n  ▶ ▶ -graph_theoretic_activity -automorphism_group -end \n```

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 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.

```
JK_graph_sts_13:
  ▶ $(ORBITER) -v 2 \n  ▶ ▶ -define Gamma -graph -load_dimacs \n  ▶ ▶ ▶ ../JUNTTILA_KASKI/benchmarks/srg/sts-13 \n  ▶ ▶ -end \n  ▶ ▶ -with Gamma -do \n  ▶ ▶ -graph_theoretic_activity -save -end \n  ▶ ▶ -with Gamma -do \n  ▶ ▶ -graph_theoretic_activity -automorphism_group -end
  ▶ make ORBITER_PATH=$(ORBITER_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:

```
halljanko315_gens.csv:
$\text{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:

```
HJ_group_and_orbits:
$\text{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:

```
HJ_orbital_graph_3:
$\text{ORBITER} -v 2 \ 
  > -define gens -vector -file \ 
  >  halljanko315_gens.csv -end \ 
```

464
The graph is regular of degree 64.

The next command computes the automorphism group of the collinearity graph of the $Q(4, 2)$ quadric.

```
PGO5_2_graph_group: 0_5_2_incidence_matrix.csv
   $\$(ORBITER) -v 3 \
   -define Inc -vector -file 0_5_2_incidence_matrix.csv -end \n   -define Gamma -graph -collinearity_graph Inc -end \n   -with Gamma -do \n   -automorphism_group \n   -end \n   -with Gamma -do \n   -graph_theoretic_activity \n   -eigenvalues \n   -end
```

dltext collinearity_graph_eigenvalues.tex
open collinearity_graph_eigenvalues.pdf

The group is 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.
15.8 Canonical Forms of Quartic Curves

We wish to study the automorphism groups of certain quartic curves introduced by Edge. We start by creating a cheat sheet of the field $\mathbb{F}_{17}$.

\begin{verbatim}
F_{17}_edge:
  $(ORBITER) -v 3 \\n  -define F -finite_field -q 17 -end \\
  -with F -do -finite_field_activity \\
  -cheat_sheet_GF -end
  pdflatex GF
  open GF_{17}.pdf
\end{verbatim}

Next, we compute the canonical form of the Edge quartic. This command also computes generators for the automorphism group of the curve.

\begin{verbatim}
Edge_curve_{17}_nauty:
  $(ORBITER) -v 3 \\
  -define C -combinatorial_objects \\
  -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 \\
  -classification_prefix Edge_curve_q17 \\
  -label Edge_curve_q17 \\
  -save_ago \\
  -save_transversal \\
  -max_TDO_depth 10 \\
  -end \\
  -report \\
  -prefix Edge_curve_q17 \\
  -export_flag_orbits \\
  -show_TDO \\
  -show_TDA \\
  -dont_show_incidence_matrices \\
  -export_group_GAP \\
  -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
\end{verbatim}
Using the generators that have just been computed, we can recreate the group of the quartic curve:

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_activities \\
▶ ▶ -print_elements_tex \\
▶ ▶ -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 16

Interfaces

16.1 Graphical Output

Orbiter can produce graphical output in a variety of formats:

1. TikZ / Latex [67],
2. Metapost [34],
3. Bitmap files (.bmp) [70],
4. Povray, see Section 16.2.

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 16.1. Suppose we want to create a graphical representation of the addition table of the finite field $F_7$. The following command sequence first creates the addition and multiplication tables of the field, and then produces a bitmap graphic for the addition table:

<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-input_csv_file</code></td>
<td>csv-file</td>
<td>Specify the input csv-file</td>
</tr>
<tr>
<td><code>-partition</code></td>
<td>$w R C$</td>
<td>Specify a partition $R$ of rows and $C$ of columns. Use separating lines of with $w$.</td>
</tr>
<tr>
<td><code>-box_width</code></td>
<td>$w$</td>
<td>Use $w$ pixels per matrix entry.</td>
</tr>
<tr>
<td><code>-bit_depth</code></td>
<td>$d$</td>
<td>Use color bit depth of $d$ bits ($d = 8$ or $d = 24$).</td>
</tr>
<tr>
<td><code>-invert_colors</code></td>
<td></td>
<td>Use an inverted color scheme.</td>
</tr>
</tbody>
</table>

Table 16.1: Commands to Create Bitmap Graphics
The finite field activity \texttt{-cheat_sheet_GF} creates the file \texttt{GF\_q7\_addition\_table.csv} which is used as the input for the second command. The file content is:

\begin{verbatim}
Row,C0,C1,C2,C3,C4,C5,C6
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
END
\end{verbatim}

The second command creates the diagram in Figure 16.1. The \texttt{-partition} command is used to define an outline of width 3 pixes. The all-in-one partition 7 is used as both row-partition and column-partition.

The planes \(\text{PG}(2,q)\) admit a cyclic automorphism group known as the Singer cycle. The command

\begin{verbatim}
PG\_2\_4\_cyclic\_incma:
\end{verbatim}
Figure 16.1: Addition table of $\mathbb{F}_7$

```
0 1 2 3 4 5 6
1 2 3 4 5 6 0
2 3 4 5 6 0 1
3 4 5 6 0 1 2
4 5 6 0 1 2 3
5 6 0 1 2 3 4
6 0 1 2 3 4 5
```

produces a cyclically ordered incidence matrix of the plane $\text{PG}(2, 4)$, shown in Figure 16.2. The Singer cycle is the projectivity defined by the companion matrix of an irreducible polynomial. We may pick the irreducible polynomial $X^2 + X + \omega$ over $\mathbb{F}_4$. The associated Singer cycle is the projectivity defined by the companion matrix

$$
\begin{bmatrix}
0 & 1 & 0 \\
0 & 0 & 1 \\
2 & 1 & 1
\end{bmatrix}.
$$

The poset classification algorithm from Sections 6.3 and 6.4 computes partially ordered sets. The posets are created using the `-draw_poset` option in the poset classification control command package, see Table 6.3. 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 16.2 and Table 16.3 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 $\text{PGL}(4, 2)$ in its action on the wedge product. The command...
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-file</td>
<td>fname</td>
<td>Use the given file name for output files.</td>
</tr>
<tr>
<td>-xin</td>
<td>a</td>
<td>Assume input $x$-coordinates are in the interval $[0, a]$. Default value: 10000.</td>
</tr>
<tr>
<td>-yin</td>
<td>a</td>
<td>Assume input $y$-coordinates are in the interval $[0, a]$. Default value: 10000.</td>
</tr>
<tr>
<td>-xout</td>
<td>a</td>
<td>Assume output $x$-coordinates are in the interval $[0, a]$. Default value: 1000000.</td>
</tr>
<tr>
<td>-yout</td>
<td>a</td>
<td>Assume output $y$-coordinates are in the interval $[0, a]$. Default value: 1000000.</td>
</tr>
<tr>
<td>-spanning_tree</td>
<td></td>
<td>Place nodes according to a spanning tree. Default value: off.</td>
</tr>
<tr>
<td>-circle</td>
<td></td>
<td>Circle all nodes. Default value: on.</td>
</tr>
<tr>
<td>-corners</td>
<td></td>
<td>Draw corners at the outside of the drawing. Default value: off.</td>
</tr>
<tr>
<td>-rad</td>
<td>r</td>
<td>Use radius $r$ for drawing circles around nodes. Default value: 50.</td>
</tr>
<tr>
<td>-embedded</td>
<td></td>
<td>Create latex headers for stand-alone latex files. Default value: off.</td>
</tr>
<tr>
<td>-sideways</td>
<td></td>
<td>Create latex figure sideways. Default value: off.</td>
</tr>
<tr>
<td>-label_edges</td>
<td></td>
<td>Label the edges in Schreier trees. Default value: off.</td>
</tr>
<tr>
<td>-x_stretch</td>
<td>s</td>
<td>Apply $x$-axis scaling by a factor of $s$. Default value: $s = 1.0$. This option does not affect the drawing of Schreier trees.</td>
</tr>
<tr>
<td>-y_stretch</td>
<td>s</td>
<td>Apply $y$-axis scaling by a factor of $s$. Default value: $s = 1.0$. This option does not affect the drawing of Schreier trees.</td>
</tr>
</tbody>
</table>

Table 16.2: Drawing Options for Layered Graph Files (Part 1)
Figure 16.2: A cyclic ordering of the incidence matrix of PG(2, 4)

<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-scale</td>
<td>$s$</td>
<td>Use tikz global scale-factor of $s$. Default value: $s = 0.45$.</td>
</tr>
<tr>
<td>-line_width</td>
<td>$s$</td>
<td>Set tikz line width to $s$. Default value: $s = 1.5$.</td>
</tr>
<tr>
<td>-nodes_empty</td>
<td></td>
<td>Draw nodes empty. Do not label. Default value: off.</td>
</tr>
<tr>
<td>-select_layers</td>
<td>$S$</td>
<td>Draw layers whose index is given in the list $S$ only.</td>
</tr>
<tr>
<td>-paths_in_between</td>
<td>$l_1 \ i_1 \ l_2 \ i_2$</td>
<td>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 $l_2$.</td>
</tr>
</tbody>
</table>

Table 16.3: Drawing Options for Layered Graph Files (Part 2)
Figure 16.3: The first basic orbit of PGL(4, 2) as a subgroup of PGO⁺(6, 2)

PGL_4_2_Wedge_4_0_graphical_output:
▷ $(ORBITER) -v 4 \n ▷  ▷ -define G -linear_group -PGL 4 2 \n ▷  ▷  ▷ -wedge_detached \n ▷  ▷  ▷ -end \n ▷  ▷ -with G -do \n ▷  ▷  ▷ -group_theoretic_activity \n ▷  ▷  ▷ -report \n ▷  ▷ -end
▷ pdflatex PGL_4_2_Wedge_4_2_detached_report.tex
▷ open PGL_4_2_Wedge_4_2_detached_report.pdf

produces a report about this group action. Figure 16.3 shows the first basic orbit in the stabilizer chain of the group in that action.

The command

schreier_tree_graphical_output:
▷ $(ORBITER) -v 4 \n ▷  ▷ -draw_options \n ▷  ▷  ▷ -yout 500000 \n ▷  ▷  ▷ -radius 15 -nodes_empty \n ▷  ▷  ▷ -line_width 0.5 -y_stretch 0.25 \n ▷  ▷  ▷ -embedded \n ▷  ▷ -end \n ▷  ▷ -define G -linear_group -PGL 4 2 -end \n ▷  ▷ -define Orb -orbits -group G 

474
Figure 16.4: A Schreier tree in the action on polynomials

draws the 6th Schreier tree in the classification of orbits of PGL(4, 2) on homogeneous polynomials of degree 3 in 4 variables. The drawing is shown in Figure 16.4. This particular orbit has length 420, so there are that many nodes in the tree.
16.2 The Povray Interface

Orbiter can be used to create raytracing 3D-graphics. Orbiter serves as a front end for the raytracing software Povray [59]. 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 16.3). By default, an rotational animation is produced.

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 16.4-16.5 list the commands to control the 3D-povray frontend. The main part in a 3D graphics is the scene description. This tells the system what will be in the picture. A scene is composed of 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 Hilbert, Cohn-Vossen 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 make drawing easier, it is possible to group objects. All objects in a group must have the same type. One group of object can be drawn with one command. Tables 16.6 and 16.7 summarize the Orbiter commands to build objects of a 3D scene. Building the scene itself does not create any graphical output. To this end, the commands in Table 16.8 are used. Each of these commands applies to a group of objects of the same kind. Groups of objects are created using the commands in Table 16.7 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:

cube:

```plaintext
$ORBITER -v 2 -povray -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

-obj_file cube_centered.obj
-vertex "0, 1"
-vertex "0, 2"
-vertex "0, 4"
-vertex "1, 3"
-vertex "1, 5"
-vertex "2, 3"
-vertex "2, 6"
-vertex "3, 7"
-vertex "4, 5"
```

476
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-do_not_rotate</td>
<td></td>
<td>No rotation. By default, the animation consists of a full rotation around a vertical axis.</td>
</tr>
<tr>
<td>-rotate_about_z_axis</td>
<td></td>
<td>Rotate around z-axis.</td>
</tr>
<tr>
<td>-rotate_about_111</td>
<td></td>
<td>Rotate around (1,1,1)-axis (default).</td>
</tr>
<tr>
<td>-rotate_about_custom_axis</td>
<td>axis</td>
<td>Rotate around a custom axis. The axis is specified as a vector of length 3.</td>
</tr>
<tr>
<td>-boundary_none</td>
<td></td>
<td>Remove the clipping.</td>
</tr>
<tr>
<td>-boundary_box</td>
<td></td>
<td>Clip using a box shape.</td>
</tr>
<tr>
<td>-boundary_sphere</td>
<td></td>
<td>Clip using a sphere (default).</td>
</tr>
<tr>
<td>-font_size s</td>
<td></td>
<td>Set font size to s.</td>
</tr>
<tr>
<td>-stroke_width s</td>
<td></td>
<td>Set text depth to s.</td>
</tr>
<tr>
<td>-omit_bottom_plane</td>
<td></td>
<td>Remove the bottom plane.</td>
</tr>
<tr>
<td>-W w</td>
<td></td>
<td>Set output dimension to w pixels wide.</td>
</tr>
<tr>
<td>-H h</td>
<td></td>
<td>Set output dimension to h pixels height.</td>
</tr>
<tr>
<td>-nb_frames n</td>
<td></td>
<td>Set number of frames to n. One revolution around the axis is split into n frames.</td>
</tr>
<tr>
<td>-zoom r a_s a_t c_s c_t</td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>-pan r F T C</td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>-pan_reverse r F T C</td>
<td></td>
<td>Same as -pan, but camera movement is in opposite order.</td>
</tr>
<tr>
<td>-no_background</td>
<td></td>
<td>Remove background.</td>
</tr>
<tr>
<td>-no_bottom_plane r</td>
<td></td>
<td>Remove bottom plane in round r.</td>
</tr>
<tr>
<td>-camera r S C L</td>
<td></td>
<td>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.</td>
</tr>
</tbody>
</table>

Table 16.4: Options for Orbiter 3D-graphics (Part 1)
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-clipping</td>
<td>( r \ c )</td>
<td>In round ( r ), set clipping radius to ( c ).</td>
</tr>
<tr>
<td>-text</td>
<td>( r \ a \ text )</td>
<td>In round ( r ), produce running text \text{text} with sustain value ( a ).</td>
</tr>
<tr>
<td>-label</td>
<td>( r \ s \ a \ g \ text )</td>
<td>In round ( r ), produce running text \text{text} with start value ( s ), sustain ( s ) and gravity ( g ).</td>
</tr>
<tr>
<td>-latex</td>
<td>( r \ s \ a \ praemable \ g \ text \ l \ fname )</td>
<td>In round ( r ), produce running latex text \text{text} with start value ( s ), sustain ( s ) and gravity ( g ). Put \text{praemable} in the latex source code. Use \text{fname} for the latex file names (no extension).</td>
</tr>
<tr>
<td>-global_picture_scale</td>
<td>( d )</td>
<td>Set scaling factor to ( d ).</td>
</tr>
<tr>
<td>-picture</td>
<td>( r \ d \ fname \ options )</td>
<td>In round ( r ), place picture \text{fname} scaled by ( d ) using options.</td>
</tr>
<tr>
<td>-picture</td>
<td>( r \ d \ fname \ options )</td>
<td>In round ( r ), place picture \text{fname} scaled by ( d ) using options.</td>
</tr>
<tr>
<td>-look_at</td>
<td>( L )</td>
<td>Override camera look-at value to ( L ). ( L ) is a three-dimensional vector.</td>
</tr>
<tr>
<td>-default_angle</td>
<td>( a )</td>
<td>Set default camera angle to ( a ).</td>
</tr>
<tr>
<td>-clipping_radius</td>
<td>( f )</td>
<td>Set default clipping radius to ( f ).</td>
</tr>
<tr>
<td>-scale_factor</td>
<td>( s )</td>
<td>Set default scale factor to ( s ).</td>
</tr>
<tr>
<td>-line_radius</td>
<td>( s )</td>
<td>Set default line radius to ( s ).</td>
</tr>
</tbody>
</table>

Table 16.5: Options for Orbiter 3D-graphics (Part 2)
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-cubic_lex</td>
<td>coeffs</td>
<td>Cubic surface given by 20 coefficients in lexicographic ordering</td>
</tr>
<tr>
<td>-cubic_orbiter</td>
<td>coeffs</td>
<td>Cubic surface given by 20 coefficients in Orbiter ordering</td>
</tr>
<tr>
<td>-cubic_Goursat</td>
<td>A B C</td>
<td>Cubic surface with tetrahedral symmetry given by 3 Goursat coefficients as $Axyz + B(x^2 + y^2 + z^2) + C = 0$</td>
</tr>
<tr>
<td>-quadric_lex_10</td>
<td>coeffs</td>
<td>Quadric surface given by 10 coefficients in lexicographic ordering</td>
</tr>
<tr>
<td>-quartic_lex_35</td>
<td>coeffs</td>
<td>Quartic surface given by 35 coefficients in lexicographic ordering</td>
</tr>
<tr>
<td>-octic_lex_165</td>
<td>coeffs</td>
<td>Octic surface given by 165 coefficients in lexicographic ordering</td>
</tr>
<tr>
<td>-point</td>
<td>coeffs</td>
<td>Point given by three coordinates</td>
</tr>
<tr>
<td>-point_list_from_csv_file</td>
<td>fname</td>
<td>List of points with coordinates given in a csv file</td>
</tr>
<tr>
<td>-line_through_two_points_recentered_from_csv_file</td>
<td>fname</td>
<td>List of lines through two points with point coordinates given in a csv file</td>
</tr>
<tr>
<td>-line_through_two_points_from_csv_file</td>
<td>fname</td>
<td>List of lines through two points with point coordinates given in a csv file</td>
</tr>
<tr>
<td>-point_as_intersection_of_two_lines</td>
<td>$i_1 ; i_2$</td>
<td>Create a point from the intersection of two lines $i_1$ and $i_2$</td>
</tr>
<tr>
<td>-edge</td>
<td>$i_1 ; i_2$</td>
<td>Create an edge (line segment) between points $i_1$ and $i_2$</td>
</tr>
<tr>
<td>-text</td>
<td>$i_1 ; s$</td>
<td>Create a label $s$ located at the point $i_1$</td>
</tr>
<tr>
<td>-triangular_face_given_by_three_lines</td>
<td>$i_1 ; i_2 ; i_3$</td>
<td>Create a triangular face give by three lines $i_1, i_2, i_3$</td>
</tr>
<tr>
<td>-face</td>
<td>pts</td>
<td>Create a face through the vertices pts, ordered cyclically</td>
</tr>
<tr>
<td>-quadric_through_three_skew_lines</td>
<td>$i_1 ; i_2 ; i_3$</td>
<td>Create a quadric through three skew lines</td>
</tr>
<tr>
<td>-plane_defined_by_three_points</td>
<td>$i_1 ; i_2 ; i_3$</td>
<td>Create a plane through three noncollinear points</td>
</tr>
<tr>
<td>-line_through_two_points_recentered</td>
<td>pt-coords</td>
<td>Create a line through two points given by 6 coordinates, recentered</td>
</tr>
</tbody>
</table>

Table 16.6: Scene definition commands (part 1)
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-line_through_two_points</td>
<td>pt-coords</td>
<td>Create a line through two points given by 6 coordinates</td>
</tr>
<tr>
<td>-line_through_two_existing_points</td>
<td>i_1 i_2</td>
<td>Create a line through two points</td>
</tr>
<tr>
<td>-line_through_point_with_direction</td>
<td>x y z u_x u_y u_z</td>
<td>Create a line through a point (x, y, z) with a given direction (u_x, u_y, u_z)</td>
</tr>
<tr>
<td>-plane_by_dual_coordinates</td>
<td>a b c d</td>
<td>Create the plane (ax + by + cz + d = 0) given in dual coordinates</td>
</tr>
<tr>
<td>-dodecahedron</td>
<td></td>
<td>Create a Dodecahedron centered at the origin (20 points, 30 edges, 12 faces)</td>
</tr>
<tr>
<td>-Hilbert_Cohn_Vossen_surface</td>
<td></td>
<td>Create the Hilbert, Cohn-Vossen surface (1 cubic surface, 45 tritangent planes, 27 lines)</td>
</tr>
<tr>
<td>-obj_file</td>
<td>fname</td>
<td>Read points and faces from the given .obj file</td>
</tr>
<tr>
<td>-group_of_things</td>
<td>list</td>
<td>Create a group of things from the given list</td>
</tr>
<tr>
<td>-group_of_things_with_offset</td>
<td>list offset</td>
<td>Create a group of things from the given list, each value is increase by offset</td>
</tr>
<tr>
<td>-group_of_things_as_interval</td>
<td>a b</td>
<td>Create a group of things indexed by the interval (a, \ldots, a + b - 1)</td>
</tr>
<tr>
<td>-group_of_things_as_interval_with_exceptions</td>
<td>a b ex</td>
<td>Create a group of things indexed by the interval (a, \ldots, a + b - 1) with the exceptional elements in the list ex removed</td>
</tr>
<tr>
<td>-group_of_all_points</td>
<td></td>
<td>Create a group of things from all points currently defined</td>
</tr>
<tr>
<td>-group_of_all_faces</td>
<td></td>
<td>Create a group of things from all faces currently defined</td>
</tr>
<tr>
<td>-group_subset_at_random</td>
<td>i f</td>
<td>Create a group of things from the existing group (i) by picking a random subset with probability (f)</td>
</tr>
<tr>
<td>-create_regulus</td>
<td>i N</td>
<td>Create a regulus for quadric (i) with (N) lines</td>
</tr>
</tbody>
</table>

Table 16.7: Scene definition commands (part 2)
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-spheres</td>
<td>$i \ r \ \text{prop}$</td>
<td>For each element in point group $i$, create a sphere of radius $r$ with given Povray properties.</td>
</tr>
<tr>
<td>-cylinders</td>
<td>$i \ r \ \text{prop}$</td>
<td>For each element in edge group $i$, create a cylinder of radius $r$ with given Povray properties.</td>
</tr>
<tr>
<td>-prisms</td>
<td>$i \ d \ \text{prop}$</td>
<td>For each element in face group $i$, create a prism of half-thickness $d$ with given Povray properties.</td>
</tr>
<tr>
<td>-planes</td>
<td>$i \ \text{prop}$</td>
<td>For each element in plane group $i$, create a plane with given Povray properties.</td>
</tr>
<tr>
<td>-lines</td>
<td>$i \ r \ \text{prop}$</td>
<td>For each element in line group $i$, create a line of radius $r$ with given Povray properties.</td>
</tr>
<tr>
<td>-cubics</td>
<td>$i \ \text{prop}$</td>
<td>For each element in group $i$ of cubics, create a surface with given Povray properties.</td>
</tr>
<tr>
<td>-quadrics</td>
<td>$i \ \text{prop}$</td>
<td>For each element in group $i$ of quadrics, create a surface with given Povray properties.</td>
</tr>
<tr>
<td>-quartics</td>
<td>$i \ \text{prop}$</td>
<td>For each element in group $i$ of quartics, create a surface with given Povray properties.</td>
</tr>
<tr>
<td>-octics</td>
<td>$i \ \text{prop}$</td>
<td>For each element in group $i$ of octics, create a surface with given Povray properties.</td>
</tr>
<tr>
<td>-texts</td>
<td>$i \ d \ s \ \text{prop}$</td>
<td>For each element in group $i$ of labels, create a text element with half-thickness $d$ and size $s$ with given Povray properties.</td>
</tr>
</tbody>
</table>

Table 16.8: Graphical output commands
This command instructs Orbiter to create 30 povray files (extension .pov), one for each frame of a rotating scene. The scene contains 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:

![Image of rotating cube](image)

The coordinates of the cube are stored in an object file `cube_centered.obj`. The content of this file is:

```plaintext
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
```
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.

MONKEY_SADDLE_CUBIC="1,0,0,0,-3,0,0,0,0,0,0,0,0,0,0,0,0,0,-1,0"

monkey:
\> $(ORBITER) -v 2 -povray \n\> \> -round 0 -nb_frames_default 30 \n\> \> -output_mask monkey_%d_%03d.pov \n\> \> -video_options -W 1024 -H 768 \n\> \> -global_picture_scale 0.8 \n\> \> -default_angle 75 \n\> \> -clipping_radius 0.8 \n\> \> -camera 0 "0,0,1" "1,1,0.5" "0,0,0" \n\> \> -rotate_about_z_axis \n\> \> -end \n\> \> -scene_objects \n\> \> \> -cubic_lex $(MONKEY_SADDLE_CUBIC) \n\> \> \> -plane_by_dual_coordinates "0,0,1,0" \n\> \> \> -group_of_things "0" \n\> \> \> -group_of_things "0" \n\> \> \> -cubics 0 $(COLOR_GOLD) \n\> \> \> -planes 1 $(COLOR_BLUE) \n\> \> -scene_objects_end \n\> \> -povray_end \n\> - rm -rf POV \n\> mkdir POV \n\> mv monkey_0*.pov POV \n\> mv makefile_animation POV
Here is one of the frames that are created:

![Eckardt surface plot](image)

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 Schlafli labeling of the lines is indicated.

**Eckardt:**
```bash
$(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_Vosson_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 \\
  -lines 1 0.02 $(COLOR_RED_SHINY) \\
  -lines 2 0.02 $(COLOR_BLUE_SHINY) \\
  -lines 3 0.02 $(COLOR_YELLOW_SHINY) \\
  -label 0 "a1"
```
Figure 16.5 shows the final product.

The Endrass octic [26] is the algebraic surface given by the equation

\[
X_8 := 64 \left( -w^2 + x^2 \right) \left( -w^2 + y^2 \right) \left( (x+y)^2 - 2w^2 \right) \left( (x-y)^2 - 2w^2 \right) - \left( -4 \left( 1 + \sqrt{2} \right) \left( 2w^2 \right) + \left( 2 + 7\sqrt{2} \right) w^2 \right) \left( x^2 + y^2 \right) - 16z^4 + 8 \left( 1 - 2 \sqrt{2} \right) z^2 w^2 - \left( 1 + 12 \sqrt{2} \right) w^4
\]

The following Orbiter command creates a povray graphics of the octic, shown in Figure 16.6:
Figure 16.5: The Eckardt surface
enadrass8:

$ (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

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,-687.529,0,0,1582.59,0,1186.94,0,0,0,-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,-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,-309.019,\ 
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,-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,-93.2548,0,0,527.529,0,395.647,\ 
0,0,0,0,-1055.06,0,-1582.59,0,-593.47,0,0,0,0,0,0,0,0,0,0,0,-2110.12,0,-3165.17,0,-1186.94,0,0,0,0,0,\ 
1677.92,0,343.362,0,0,0,0,0,0,0,0,0,0,-256.0,-468.077,0,-789.019,0,\ 
-525.726,0,0.941125"
Figure 16.6: The Endrass Octic

```
mv endrass_octic_0_*.pov POV
mv makefile_animation POV
```

This illustration includes coordinate axes and the $x, y$-plane.
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>s l mask</td>
<td>Specify the input file names by running a printf command with the given mask applied to the index $i$ where $i$ goes from $s$ to $s + l - 1$. This option can be repeated.</td>
</tr>
<tr>
<td>-step</td>
<td>s</td>
<td>Increment the index in steps of size $s$.</td>
</tr>
<tr>
<td>-o</td>
<td>mask</td>
<td>Create the output file using the given mask.</td>
</tr>
<tr>
<td>-output_starts_at</td>
<td>i</td>
<td>Start output file indices at $i$ (default is 0).</td>
</tr>
</tbody>
</table>

Table 16.9: Prepare frames commands

16.3 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 16.9. For instance, the command

```
monkey_video:
▷ - rm -r FRAMES
▷ - mkdir FRAMES
▷ - rm monkey.mp4
▷ $(ORBITER) \n▷ ▷ -prepare_frames \n▷ ▷ ▷ -i 0 30 monkey_0_0%03d.png \n▷ ▷ ▷ -output_starts_at 0 \n▷ ▷ ▷ -o FRAMES/frame%04d.png \n▷ ▷ -end
▷ ffmpeg -r 5 -f image2 -i FRAMES/frame%04d.png \n▷ ▷ -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_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 [39].
16.4 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:

- \( \sin(x) \mapsto \) push x sin,
- \( a + b \mapsto \) push a push b add,
- \( a \cdot b \mapsto \) push a push b mult.

The coordinate functions are enclosed between \(-\text{code}\) and \(-\text{code}_\text{end}\) 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 \(-\text{const}\) and \(-\text{const}_\text{end}\) keywords. Likewise, variables are declared between the \(-\var\) and \(-\var\_\text{end}\) keywords. Picking \(a = 3, b = 2, c = \pi/2\) and \(r = 7\), the function is computed using

```
lissajous:
  > $(ORBITER) -v 2 \n  > -smooth_curve "lissajous" 0.07 2000 15 0 18.85 \n  > -var a 3 b 2 c 1.57 r 7 -const_end \n  > -const a 3 b 2 c 1.57 r 7 -const_end \n  > -code \n  > -code \n  > push t push a mult push c add sin push r mult \n  > push t push b mult sin push r mult \n  > -code_end \n```

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

\[-\text{const} a 3 b 2 c 1.57 r 7 -\text{const}_\text{end}\]
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 $\epsilon$ apart. The code to produce the plot is

```
lissajous_plot:
   $(ORBITER) -v 2 -povray \n   -round 0 -nb_frames_default 1 \n   -output_mask lissajous.%d.%03d.pov \n   -video_options -W 1024 -H 768 \n   -global_picture_scale 0.40 \n   -default_angle 45 \n   -clipping_radius 5 \n   -omit_bottom_plane \n   -camera 0 "0,-1,0" "0,0,12" "0,0,0" \n   -rotate_about_z_axis \n   -end \n   -scene_objects \n   -line_through_two_points_recentered_from_csv_file \n   coordinate_grid.csv \n   -group_of_things "0" \n   -group_of_things "1" \n   -group_of_things "2" \n   -lines 0 0.09 "texture{ pigment{ color Yellow } }" \n   -lines 1 0.09 "texture{ pigment{ color Yellow } }" \n   -lines 2 0.09 "texture{ pigment{ color Yellow } }" \n   -group_of_things_as_interval 3 39 \n   -point_list_from_csv_file \n   function_lissajous_N2000_points.csv \n   -group_of_things_as_interval 0 6524\n   -spheres 4 0.1 "texture{ pigment{ color Red } }" finish { diffuse 0.9 phong 1})"\n   -plane_by_dual_coordinates "0,0,1,0" \n   -group_of_things "0" \n   -planes 5 "texture{ pigment{ color Blue*0.5 \n   transmit 0.5 } }"\n   -scene_objects_end \n```

491
Figure 16.7: Lissajous figure

The plot is shown in Figure 16.7.

We can turn it into a 3D plot by using the $t$ value for the $z$ coordinate. The function is computed using the command

```
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

```
lissajous_3d.plot:
  $(ORBITER) -v 2 -povray \
  -round 0 -nb_frames_default 30 \
```

492
The 3D curve is shown in Figure 16.8.
Figure 16.8: Lissajous Spacecurve
Chapter 17

Miscellaneous

17.1 Miscellaneous

Table 17.1 lists 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:

$$\alpha^0, \alpha^1, \alpha^2, \alpha^3, \alpha^4, \alpha^5.$$ 

After that, we pull the rows and columns corresponding to even powers $\alpha^0, \alpha^2, \alpha^4$.

```
misc_select:
  ▶ $(ORBITER) -v 3 \n  ▶ -define F -finite_field -q 7 -end \n  ▶ -with F -do -finite_field_activity -cheat_sheet_GF -end
  ▶ $(ORBITER) -v 4 -csv_file_select_rows_and-cols \n  ▶ GF_q7_multiplication_table_reordered.csv \n  ▶ "0,2,4" "0,2,4"
```

The even powers of $\alpha$ create a multiplicative subgroup. Figure 17.1 shows the table of the multiplicative group $\mathbb{F}_7^*$ and the subgroup of squares (compare with Figure 3.2 in Section 3.2). Here is the file `GF_q7_multiplication_table_reordered.csv`

```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
```
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-csv_file_select_rows</td>
<td>fname $R$</td>
<td>Selects rows listed in $R$ from the csv-file fname.</td>
</tr>
<tr>
<td>-csv_file_select_cols</td>
<td>fname $R$</td>
<td>Selects columns listed in $R$ from the csv-file fname.</td>
</tr>
<tr>
<td>-csv_file_select_rows_and_cols</td>
<td>fname $R C$</td>
<td>Selects rows listed in $R$ and columns listed in $C$ from the csv-file fname.</td>
</tr>
<tr>
<td>-csv_file_join</td>
<td>fname col-label</td>
<td>Joins csv file fname according to column with label col-label. This option is given once for each file that should be joined.</td>
</tr>
<tr>
<td>-csv_file_latex</td>
<td>fname</td>
<td>Produces a latex table from the given csv-file.</td>
</tr>
<tr>
<td>-store_as_csv_file</td>
<td>fname $m \times n ; L$</td>
<td>Stores the data in $L$ to a csv file. The data is an $m \times n$ matrix in row-major ordering.</td>
</tr>
</tbody>
</table>

Table 17.1: Miscellaneous Orbiter Commands

Figure 17.1: Cyclic multiplication table of $\mathbb{F}_7$ and subgroup of squares
and next the file that is created by selecting rows and columns 0, 2, 4:

<table>
<thead>
<tr>
<th>Row</th>
<th>&quot;C0&quot;, &quot;C2&quot;, &quot;C4&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>&quot;1&quot;, &quot;2&quot;, &quot;4&quot;</td>
</tr>
<tr>
<td>1</td>
<td>&quot;2&quot;, &quot;4&quot;, &quot;1&quot;</td>
</tr>
<tr>
<td>2</td>
<td>&quot;4&quot;, &quot;1&quot;, &quot;2&quot;</td>
</tr>
</tbody>
</table>
17.2 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.
Chapter 18

Orbiter on Windows

18.1 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 illustrate 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 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

Orbiter - compile

• After cloning orbiter, run the command
  • dir

• You will find a new directory created called “orbiter”

• Change directory to “orbiter”
  • cd orbiter
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

![Compilation Command]

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 
    - :wa + 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 19

The Makefile

19.1 The Makefile

1  #MY_PATH=../orbiter
2  MY_PATH="/DEV.22/orbiter"
3  #MY_PATH=/scratch/betten/COMPILE/orbiter
4
5  # uncomment exactly one of the following two lines.
6  # uncomment the first if you want to run orbiter through docker.
7  # uncomment the second if you have an installed copy of orbiter and you want to run it directly
8  #ORBITER_PATH=docker run -it --volume ${PWD}:/mnt -w /mnt abetten/orbiter
9  ORBITER_PATH=$(MY_PATH)/src/apps/orbiter/
10  ORBITER=$(ORBITER_PATH)orbiter.out
11  SANDBOX=$(MY_PATH)/src/apps/sandbox/sandbox.out
12
13  ###############################################################################
14  # additional configurations for when you want to
15  # compile automatically generated code
16  ###############################################################################
17  SRC=$(MY_PATH)/src
18  MY_CPP = g++
19  MY_CC = gcc
20  CPPFLAGS = -Wall -I../../DEV.22/orbiter/src/lib -std=c++14
21  LIB = $(SRC)/lib/liborbiter.a -lpthread
22  LFLAGS = -lM -Wl,-rpath -Wl,usr/local/gcc-8.2.0/lib64
23  
24  ###############################################################################
25  # End of configuration part
26  ###############################################################################
27
28
29
30
31
GINAC_PATH=$(MY_PATH)/src/apps/ginac
Sandbox_PATH=$(MY_PATH)/src/apps/sandbox

update:
  cd $(ORBITER_PATH); make clean;
  cd $(MY_PATH); make cleana; git pull; make

update_all:
  cd $(MY_PATH); make clean; git pull; make

sandbox:
  $(SANDBOX_PATH)/sandbox.out

###############################################################################
# overall test:
###############################################################################

test:
  make test_{2..4}
  make test_{2..5}
  make test_{4..2}
  make test_{4..3}
  make test_{4..5}
  make test_{4..7}
  make test_{4..8}
  make test_{5..1}
  make test_{5..2}
  make test_{5..3}
  make test_{5..4}
  make test_{5..5}
  make test_{5..6}
  make test_{5..7}
  make test_{6..1}
  make test_{6..2}
  make test_{6..3}
  make test_{6..4}
  make test_{6..5}
  make test_{6..6}
  make test_{6..7}
  make test_{7..1}
  make test_{7..2}
  make test_{7..3}
  make test_{7..4}
78  ▷ make test_7.5
79  ▷ make test_7.6
80  ▷ make test_8.1
81  ▷ make test_8.2
82  ▷ make test_9.1
83  ▷ make test_9.2
84  ▷ make test_9.3
85  ▷ make test_10.1
86  ▷ make test_10.2
87  ▷ make test_10.3
88  ▷ make test_10.4
89  ▷ make test_10.5
90  ▷ make test_10.6
91  ▷ make test_10.7
92  ▷ make test_10.8
93  ▷ make test_10.9
94  ▷ make test_11.1
95  ▷ make test_11.2
96  ▷ make test_11.3
97  ▷ make test_11.4
98  ▷ make test_11.5
99  ▷ make test_11.6
100 ▷ make test_12.1
101 ▷ make test_12.2
102 ▷ make test_12.3
103 ▷ make test_12.4
104 ▷ make test_13.1
105 ▷ make test_13.2
106 ▷ make test_13.3
107 ▷ make test_13.4
108 ▷ make test_14.1
109 ▷ make test_14.2
110 ▷ make test_15.2
111 ▷ make test_15.3
112 ▷ make test_15.4
113 ▷ make test_15.5
114 ▷ make test_15.6
115 ▷ make test_15.7
116 ▷ make test_15.8
117 ▷ make test_16.1
118 ▷ make test_16.2
119 ▷ make test_16.3
120 ▷ make test_17.1
121 ▷ make test_4.4

test_problem:
>>> make test_4.6
>>> make test_4.9
>>> make test_4.10

# Makefile Variables

#MAGMA
PATH=/usr/local/magma
MAGMA_PATH=""

V7_VANDERMONDE_EXTENDED="\n1,0,0,0,0,0,0,1,0,0,0,0,0,0, \n1,1,1,1,1,1,0,1,0,0,0,0,0,0, \n1,2,4,1,2,4,1,0,0,1,0,0,0,0, \n1,3,2,6,4,5,1,0,0,0,1,0,0,0, \n1,4,2,1,4,2,1,0,0,0,1,0,0,0, \n1,5,4,6,2,3,1,0,0,0,0,0,1,0, \n1,6,1,6,1,6,1,0,0,0,0,0,0,1"

DOILY="Row,C0,C1,C2\n\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"

#Co3 from Conway et al., 1985 (ATLAS)
CONWAY GEN1="
  1101100010000101010000
  1111011110100001010000
  0000001000000100010101
  1111100110110001001110
  0101010000000100111101
  0000000000000100010100
  0001000011000000111111
  1110100100110100010011
  0000000000000100010100
  0000000000000100010111
  0000000000000100010001"
  0101000010111010111111
  0110010100011110110000
  001101000011111010111
  0001101110001011010011
  1010010001000010111100
  1100100000010101000011
  1100101010001111010101
  1000110100110101010101
  0100110001010000001111
  1100000010100101000101
  0101110110011100000101
  0101111010100111110001
  1000101010101010000101
  0010100001111001001111
  0011010010110011111010
  1101011001111011001111
  0100101001001000000101"
# large sets of PG(2,3):

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)

GENERATORS_N5="
0,1,2,3,4,9,5,6,7,8,10,11,12,
0,1,2,3,4,5,6,7,8,9,10,12,11,
0,4,3,2,1,5,9,8,7,6,10,11,12,
0,2,4,1,3,5,7,9,6,8,10,11,12,
0,1,2,3,4,5,6,7,8,9,11,10,12,
1,2,3,4,0,6,7,8,9,5,10,11,12,
5,9,8,7,6,0,4,3,2,1,10,11,12"

GENERATORS_C13="11, 0, 10, 12, 5, 3, 7, 4, 2, 8, 6, 9, 1"

# (0,11,9,8,2,10,6,7,4,5,3,12,1)

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

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"

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

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"

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)
GENERATORS_HESSE_GROUP="\\ 3000300030 \\
2000201230 \\
1000100111 \\
1000220200 \\
1002312010 \\
0331003211 \\
2200011331"
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, \\
-1,0,-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"
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"
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"
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"
HYPEROVAL_16_144="0, 1, 2, 3, 52, 67, 89, 106, 126, \\
141, 159, 176, 184, 199, 220, 235, 245, 262"
HYPEROVAL_16.16320="0, 1, 2, 3, 52, 70, 83, 109, 127, \
139, 156, 174, 186, 199, 217, 229, 256, 264"

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 \
367 379 381 392 398 400 417 428 429 443 450 466 471 \
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 \
132 146 157 158 171 175 183 195 201 207 220 226 232 244 \
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 \
479 494 499 503 519 521 525 544 547 550 570 572 575 \
144"
\nn-1 3"

ELEMENTARY_SYMMETRIC_3_1="x0 + x1 + x2"

ELEMENTARY_SYMMETRIC_3_2="x0*x1 + x0*x2 + x1*x2"

ELEMENTARY_SYMMETRIC_3_3="x0*x1*x2"

ELEMENTARY_SYMMETRIC_4_1="x0 + x1 + x2 + x3"

ELEMENTARY_SYMMETRIC_4_2="x0*x1 + x0*x2 + x0*x3 + x1*x2 + x1*x3 + x2*x3"

ELEMENTARY_SYMMETRIC_4_3="x0*x1*x2 + x0*x1*x3 + x0*x2*x3 + x1*x2*x3"

ELEMENTARY_SYMMETRIC_4_4="x0*x1*x2*x3"

CODE_5.2.3_CODEWORDS="0,7,25,30"

SURFACE_F7.15LINES_POINTS="0,1,2,3,5,6,7,8,9,10,12,18,20,26,28,34,36,42,44,45,50,
53, 58, 59, 60, 61, 62, 63, 64, 67, 77, 80, 90, 93, 103, 107, 115, 118, 122, 125, 127, 142, 147, 155, 157, 162, 165, 170, 172, 204, 208, 219, 229, 240, 244, 246, 251, 253, 259, 277, 278, 281, 286, 298, 300, 302, 303, 310, 312, 316, 340, 343, 351, 354, 358, 365, 369, 372, 373, 379, 384, 386, 388, 393, 399

```
SURFACE_F7_15LINES_MCKEAN_POINTS="0,1,2,3,4,5,6,7,8,9,10,16,17,28,35,36,59,60,61,62,63,64,65,76,80,91,95,106,107,111,119,121,122,130,138,139,141,146,150,154,155,158,170,175,184,186,199,203,204,206,226,231,234,239,240,252,253,254,278,279,282,287,299,301,302,319,320,330,338,343,345,350,351,357,364,370,371,376,378,382,385,388,392,394,399"

SURFACE_MCKEAN_15_LINES="1,5,-1,16,-2,17,1,10,2,18,1,8,1,9,-1,11,-1,19,-1,14"

CODE_RS_6_4_7="\n621000\n006210\n000621"

CODE_RS_10_8_11="\n7,2,1,0,0,0,0,0,0,0,\n0,7,2,1,0,0,0,0,0,0,\n0,0,7,2,1,0,0,0,0,0,\n0,0,0,7,2,1,0,0,0,0,\n0,0,0,0,7,2,1,0,0,0,\n0,0,0,0,0,7,2,1,0,0,\n0,0,0,0,0,0,7,2,1,0,\n0,0,0,0,0,0,0,7,2,1"

# Normal form for cubic surfaces with 15 rational lines:

F_ALPHA_BETA_GAMMA_DELTA="beta*(gamma + 1)*x0*x0*x2 \n+ (alpha*delta - beta*gamma + alpha - beta - delta - 1)*x0*x1*x2 \n-1*(alpha*beta -alpha*delta + delta)*(gamma + 1)*x0*x1*x3 \n+ (0-alpha*delta + alpha*gamma -beta*gamma -beta + delta -gamma)*x0*x2*x2 \n-(alpha*delta +beta -delta)*(gamma +1)*x0*x2*x3 \n-(delta + 1)*(alpha - 1)*x1*x1*x2 \n-(delta + 1)*(alpha - 1)*x1*x1*x3 \n+(alpha*delta - alpha*gamma + beta*gamma + beta - delta + gamma)*x1*x2*x2 \n+(alpha*beta*gamma + alpha*beta + alpha*delta \n- alpha*gamma + beta*gamma + beta - delta + gamma)*x1*x2*x3"
+ alpha*beta*(gamma + 1)*x1*x3*x3"

# general normal form for cubic surfaces with 27 rational lines:

F_abcd_eqn_no_exponents="-(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*(b - d)*X0*X2 \n+ (a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*(a + b - c - d)*X0*X1*X2 \n+ (a*a*c - a*a*d - a*c*c + b*c*c + a*d - b*c)*(b - d)*X0*X1*X3 \n- (a*d - b*c)*(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*X0*X2*X3 \n- (a - c)*(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*X1*X1*X2 \n- (a - c)*(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*X1*X1*X3 \n+ (a*d - b*c)*(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*X1*X2*X2 \n+ ((1+1)*a*a*b*c*d - a*a*b*d*d - (1+1)*a*a*c*d*d - (1+1+1)*a*b*c*d - a*c*c*d + a*d - b*c)*KNECHT_13_1_ASPAIRS="1,0,1,1,1,2,12,9"
+ c*a*(a*d - b*c - a + b + c - d)*(b - d)*X1*X3*X3"

# general normal form for cubic surfaces with 27 rational lines:

F_abcd_eqn="-(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*(b - d)*X0^2*X2 \n+ (a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*(a + b - c - d)*X0*X1*X2 \n+ (a^2*c - a^2*d - a*c^2 + b*c^2 + a*d - b*c)*(b - d)*X0*X1*X3 \n- (a*d - b*c)*(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*X0*X2*X2 \n- (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 \n- (a - c)*(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*X1^2*X2 \n- (a - c)*(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*X1^2*X3 \n+ (a*d - b*c)*(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*X1*X2^2 \n+ ((1+1)*a*a*b*c*d - a*a*b*d*d - (1+1+1+1)*a*b*c*d - a*c*c*d - a*d - b*c)*KNECHT_13_1_ASVETOR="1,1,1,0,0, 0,0,0,0,12, 0,0,0,0,0"
+ c*a*(a*d - b*c - a + b + c - d)*(b - d)*X1*X3^2"

KNECHT_13_2_ASPAIRS="1,0,1,1,1,2,8,9,8,10,8,11"
# coding theory

CRC4="1,4,1,2,1,1,1,0"
CRC7="1,7,1,3,1,0"
CRC8_ATM="1,8,1,2,1,1,1,0"
CRC16_CCITT="1,16,1,12,1,5,1,0"
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"
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"
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"
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"

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"

# [23, 12, 8]
# 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 132, 913, 1460, 1750, 1898, 2518, 2787, 2874, 3320, 3357, 3662

# [24, 12, 8]
# 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 132, 913, 1460, 1750, 1898, 2518, 2787, 2874, 3320, 3357, 3662, 4004

CODE_RM_3_1_GENMA="\n
CODE_RM_4.1_GENMA="\n1111111111111111\n0101010101010101\n0011001100110011\n0000111000111001\n0000000011111111"

CODE_RS_8="\n561000 \n056100 \n005610 \n000561"

CODE_RS_11_RREF="\n1,0,0,0,0,0,0,0,7,2,\n0,1,0,0,0,0,0,0,8,3,\n0,0,1,0,0,0,0,0,1,2,\n0,0,0,1,0,0,0,0,8,8,\n0,0,0,0,1,0,0,0,10,3,\n0,0,0,0,1,0,0,0,14,\n0,0,0,0,0,1,0,0,5,4,\n0,0,0,0,0,0,1,0,1,5,8"

RS_8_reduced="\n010001100000000000000000\n001110010000000000000000\n110011001000000000000000\n000010001100000000000000\n000011010001000000000000\n000110011001000000000000\n000000010001100000000000\n000000001110010000000000\n000000000111010000000000\n000000000011100100000000\n000000000001110100000000\n000000000111100100000000\n000000000000011110010000\n00000000000000111001100\n00000000000000011101001\n000000000000000011001101"
CODE_21_15_4="\n1100010000000000000000 \n1101000100000000000000 \n1011000100000000000000 \n0111000001000000000000 \n1101000001000000000000 \n1010100000010000000000 \n0110100000010000000000 \n1001100000001000000000 \n11111000000001000000 \n1100010000000001000000 \n1010010000000000010000 \n0110100000000100000000 \n0011100000000100000000 \n1111100000000100000000 \n1100010000000001000000 \n1010010000000000100000 \n0110010000000000100000 \n1001010000000000010000"

# there are 5 [15,6,6]

# ago=12
CODE_15_6_6_A="\n1111111111000000 \n1111100000100000 \n1110011000010000 \n1101010100001000 \n1010101100000100 \n1010101100000100 \n101101001000010001"

# ago=12
CODE_15_6_6_B="\n1111111111000000 \n1111100000100000 \n1110011000010000 \n1101010100001000 \n1010101100000100 \n1010101100000100 \n011011001000001"

# ago=720:
CODE_15_6_6_C="\n1111111111100000 \n1111100000100000 \n1110011000010000 \n1101010100001000 \n1010101100000100 \n1010101100000100 \n10010111000001"

521
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611

612
613

#ago=96:
CODE 15 6 6 D="\
111111111100000 \
111110000010000 \
111001100001000 \
110101010000100 \
101011001000010 \
011001011000001"
#ago=360
CODE 15 6 6 E="\
111111111100000 \
111110000010000 \
111001100001000 \
100111010000100 \
010101110000010 \
010110101000001"

BCH 21 15 PROJ=" 0, 1, 19, 37, 113, 420, 1651, 6577, \
26284, 105115, 420442, 1681753, 6727000, 26907991, \
107631958, 27874647, 111498582, 43341143, 173364566, \
156587350, 14 "

BCH
0,
0,
1,
0,
0,
0,
1,
0,
0,
0,
1,

21 15 GENERATOR
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,

BCH
0,
0,
1,
1,
"

21 6 GENERATOR MATRIX=" 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 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, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 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,
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

MATRIX="1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 0, 1, 1, 1, 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, 1, 1,
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, 1, 0, 1, 0, 1, 1, 1, 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,
1, 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, 0, 0, 1, 0, 1, 0, 1, 1, 1, 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, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 1, 1, 1, 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, 1, 1, 1"

522


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, 24, 17, 248, 18, 174, 28, 26, 109, 27, 174, 28, 253, 29, 16, 132, 21, 22, 212, 23, 112, 24, 144, 25, 97, 26, 109, 27, 174, 28, 253, 29, 1, 30"

POLY_Q256_DEG30_DENSE="1, 26, 24, 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, 18, 174, 28, 26, 109, 27, 174, 28, 253, 29, 1, 30"

# created in the combinatorics section:

ELEMENTARY_SYMMETRIC_8_1="x0 + x1 + x2 + x3 + x4 + x5 + x6 + x7"

ELEMENTARY_SYMMETRIC_8_2="x0*x1 + x0*x2 + x0*x3 + x0*x4 + x0*x5 + x0*x6 + x0*x7 + x1*x2 + x1*x3 + x1*x4 + x1*x5 + x1*x6 + x1*x7 + x2*x3 + x2*x4 + x2*x5 + x2*x6 + x2*x7 + x3*x4 + x3*x5 + x3*x6 + x3*x7 + x4*x5 + x4*x6 + x4*x7 + x5*x6 + x5*x7 + x6*x7"

ELEMENTARY_SYMMETRIC_8_3="x0*x1*x2 + x0*x1*x3 + x0*x1*x4 + x0*x1*x5 + x0*x1*x6 + x0*x1*x7 + x0*x2*x3 + x0*x2*x4 + x0*x2*x5 + x0*x2*x6 + x0*x2*x7 + x0*x3*x4 + x0*x3*x5 + x0*x3*x6 + x0*x3*x7 + x0*x4*x5 + x0*x4*x6 + x0*x4*x7 + x0*x5*x6 + x0*x5*x7 + x0*x6*x7 + x1*x2*x3 + x1*x2*x4 + x1*x2*x5 + x1*x2*x6 + x1*x2*x7 + x1*x3*x4 + x1*x3*x5 + x1*x3*x6 + x1*x3*x7 + x1*x4*x5 + x1*x4*x6 + x1*x4*x7 + x1*x5*x6 + x1*x5*x7 + x1*x6*x7 + x1*x7*x8 + x2*x3*x4 + x2*x3*x5 + x2*x3*x6 + x2*x3*x7 + x2*x4*x5 + x2*x4*x6 + x2*x4*x7 + x2*x5*x6 + x2*x5*x7 + x2*x6*x7 + x2*x7*x8 + x3*x4*x5 + x3*x4*x6 + x3*x4*x7 + x3*x5*x6 + x3*x5*x7 + x3*x6*x7 + x3*x7*x8 + x4*x5*x6 + x4*x5*x7 + x4*x5*x8 + x4*x6*x7 + x4*x6*x8 + x4*x7*x8 + x5*x6*x7 + x5*x6*x8 + x5*x7*x8 + x6*x7*x8"

ELEMENTARY_SYMMETRIC_8_4="x0*x1*x2*x3 + x0*x1*x2*x4 + x0*x1*x2*x5 + x0*x1*x2*x6 + x0*x1*x2*x7 + x0*x1*x3*x4 + x0*x1*x3*x5 + x0*x1*x3*x6 + x0*x1*x3*x7 + x0*x1*x4*x5 + x0*x1*x4*x6 + x0*x1*x4*x7 + x0*x1*x5*x6 + x0*x1*x5*x7 + x0*x1*x6*x7 + x0*x2*x3*x4 + x0*x2*x3*x5 + x0*x2*x3*x6 + x0*x2*x3*x7 + x0*x2*x4*x5 + x0*x2*x4*x6 + x0*x2*x4*x7 + x0*x2*x5*x6 + x0*x2*x5*x7 + x0*x2*x6*x7 + x0*x2*x7*x8 + x0*x3*x4*x5 + x0*x3*x4*x6 + x0*x3*x4*x7 + x0*x3*x5*x6 + x0*x3*x5*x7 + x0*x3*x6*x7 + x0*x3*x7*x8 + x0*x4*x5*x6 + x0*x4*x5*x7 + x0*x4*x5*x8 + x0*x4*x6*x7 + x0*x4*x6*x8 + x0*x4*x7*x8 + x0*x5*x6*x7 + x0*x5*x6*x8 + x0*x5*x7*x8 + x0*x6*x7*x8 + x1*x2*x3*x4 + x1*x2*x3*x5 + x1*x2*x3*x6 + x1*x2*x3*x7 + x1*x2*x4*x5 + x1*x2*x4*x6 + x1*x2*x4*x7 + x1*x2*x5*x6 + x1*x2*x5*x7 + x1*x2*x6*x7 + x1*x2*x7*x8 + x1*x3*x4*x5 + x1*x3*x4*x6 + x1*x3*x4*x7 + x1*x3*x5*x6 + x1*x3*x5*x7 + x1*x3*x6*x7 + x1*x3*x7*x8 + x1*x4*x5*x6 + x1*x4*x5*x7 + x1*x4*x5*x8 + x1*x4*x6*x7 + x1*x4*x6*x8 + x1*x4*x7*x8 + x1*x5*x6*x7 + x1*x5*x6*x8 + x1*x5*x7*x8 + x1*x6*x7*x8 + x2*x3*x4*x5 + x2*x3*x4*x6 + x2*x3*x4*x7 + x2*x3*x5*x6 + x2*x3*x5*x7 + x2*x3*x6*x7 + x2*x3*x7*x8 + x2*x4*x5*x6 + x2*x4*x5*x7 + x2*x4*x5*x8 + x2*x4*x6*x7 + x2*x4*x6*x8 + x2*x4*x7*x8 + x2*x5*x6*x7 + x2*x5*x6*x8 + x2*x5*x7*x8 + x2*x6*x7*x8 + x3*x4*x5*x6 + x3*x4*x5*x7 + x3*x4*x5*x8 + x3*x4*x6*x7 + x3*x4*x6*x8 + x3*x4*x7*x8 + x3*x5*x6*x7 + x3*x5*x6*x8 + x3*x5*x7*x8 + x3*x6*x7*x8 + x4*x5*x6*x7 + x4*x5*x6*x8 + x4*x5*x7*x8 + x4*x6*x7*x8"
x0*x1*x2*x5*x7 + x0*x1*x2*x6*x7 + x0*x1*x3*x4*x5 + x0*x1*x3*x4*x6 + x0*x1*x3*x4*x
7 + x0*x1*x3*x5*x6 + x0*x1*x3*x5*x7 + x0*x1*x3*x6*x7 + x0*x1*x4*x5*x6 + x0*x1*x4*
x5*x7 + x0*x1*x4*x6*x7 + x0*x1*x5*x6*x7 + x0*x2*x3*x4*x5 + x0*x2*x3*x4*x6 + x0*x2
*x3*x4*x7 + x0*x2*x3*x5*x6 + x0*x2*x3*x5*x7 + x0*x2*x3*x6*x7 + x0*x2*x4*x5*x6 + x
0*x2*x4*x5*x7 + x0*x2*x4*x6*x7 + x0*x2*x5*x6*x7 + x0*x3*x4*x5*x6 + x0*x3*x4*x5*x7
+ x0*x3*x4*x6*x7 + x0*x3*x5*x6*x7 + x0*x4*x5*x6*x7 + x1*x2*x3*x4*x5 + x1*x2*x3*x
4*x6 + x1*x2*x3*x4*x7 + x1*x2*x3*x5*x6 + x1*x2*x3*x5*x7 + x1*x2*x3*x6*x7 + x1*x2*
x4*x5*x6 + x1*x2*x4*x5*x7 + x1*x2*x4*x6*x7 + x1*x2*x5*x6*x7 + x1*x3*x4*x5*x6 + x1
*x3*x4*x5*x7 + x1*x3*x4*x6*x7 + x1*x3*x5*x6*x7 + x1*x4*x5*x6*x7 + x2*x3*x4*x5*x6
+ x2*x3*x4*x5*x7 + x2*x3*x4*x6*x7 + x2*x3*x5*x6*x7 + x2*x4*x5*x6*x7 + x3*x4*x5*x6
*x7"
638
639

640
641

642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664

ELEMENTARY SYMMETRIC 8 6="x0*x1*x2*x3*x4*x5 + x0*x1*x2*x3*x4*x6 + x0*x1*x2*x3*x4*
x7 + x0*x1*x2*x3*x5*x6 + x0*x1*x2*x3*x5*x7 + x0*x1*x2*x3*x6*x7 + x0*x1*x2*x4*x5*x
6 + x0*x1*x2*x4*x5*x7 + x0*x1*x2*x4*x6*x7 + x0*x1*x2*x5*x6*x7 + x0*x1*x3*x4*x5*x6
+ x0*x1*x3*x4*x5*x7 + x0*x1*x3*x4*x6*x7 + x0*x1*x3*x5*x6*x7 + x0*x1*x4*x5*x6*x7
+ x0*x2*x3*x4*x5*x6 + x0*x2*x3*x4*x5*x7 + x0*x2*x3*x4*x6*x7 + x0*x2*x3*x5*x6*x7 +
x0*x2*x4*x5*x6*x7 + x0*x3*x4*x5*x6*x7 + x1*x2*x3*x4*x5*x6 + x1*x2*x3*x4*x5*x7 +
x1*x2*x3*x4*x6*x7 + x1*x2*x3*x5*x6*x7 + x1*x2*x4*x5*x6*x7 + x1*x3*x4*x5*x6*x7 + x
2*x3*x4*x5*x6*x7"
ELEMENTARY SYMMETRIC 8 7="x0*x1*x2*x3*x4*x5*x6 + x0*x1*x2*x3*x4*x5*x7 + x0*x1*x2*
x3*x4*x6*x7 + x0*x1*x2*x3*x5*x6*x7 + x0*x1*x2*x4*x5*x6*x7 + x0*x1*x3*x4*x5*x6*x7
+ x0*x2*x3*x4*x5*x6*x7 + x1*x2*x3*x4*x5*x6*x7"
ELEMENTARY SYMMETRIC 8 8="x0*x1*x2*x3*x4*x5*x6*x7"

PG 3 5 DESARGUESIAN SPREAD="0, 805, 36, 108, 72, 144, \
581, 509, 686, 415, 639, 758, 285, 722, 332, 343, 202, \
592, 473, 238, 675, 379, 166, 545, 249, 451"
# elements of order 2:
# conjugacy class reps:
# elt order, class size, centralizer order
#2A: 2 48960 40320 Baer involution
#2B: 2 5355 368640

one block of 10,11

#2C: 2 64260 30720 two blocks of 10,11 (problem group)

#. pdflatex PGGL 4 4 classes out.tex
#. open PGGL 4 4 classes out.pdf

524


# elements of order 2:
# conjugacy class reps:
# elt order, class size, centralizer order

#2 48960 40320 Baer involution

#2 5355 368640 one block of 10,11

#2 64260 30720 two blocks of 10,11 (problem group)

CLASS_2A=-centralizer_of_element "1,0,0,0, 0,1,0,0, 0,0,1,0, 0,0,0,1, 1" -label "2A"

CLASS_2B=-centralizer_of_element "1,0,0,0, 1,1,0,0, 0,0,1,0, 0,0,0,1, 0" -label "2B"

CLASS_2C=-centralizer_of_element "1,0,0,0, 1,1,0,0, 0,0,1,0, 0,0,1,1, 0" -label "2C"

# problem group

# 3 classes of elements of order 3
# 4 classes of elements of order 4

# Baer involution:

PGGL_4_4_SUBGROUP_2A=-PGGL 4 4 -subgroup_by_generators "2A" 2 1

PGGL_4_4_SUBGROUP_2A_NORMALIZER=-PGGL 4 4 -subgroup_by_generators "centralizer_2A" "40320" 10
# the problem group, two blocks of 10,11:

PGGL_4_4_SUBGROUP_2C=-PGGL 4 4

- subgroup_by_generators "2C" 2 1
  "1,0,0,0, 1,1,0,0, 0,0,1,0, 0,0,1,1, 0"

PGGL_4_4_SUBGROUP_2C_NORMALIZER=-PGGL 4 4

- subgroup_by_generators "centralizer_2C" "30720" 9
  "1,0,0,0,0,1,0,0,0,0,1,0,0,0,1,1,0, 1,0,0,0,0,1,0,0,0,0,1,0,0,0,1,1,0, 1,0,0,0,0,1,0,0,0,0,1,0,0,0,1,1,0, 1,0,0,0,0,1,0,0,0,0,1,0,0,0,1,1,0, 1,0,3,0,1,1,0,0,0,2,0,0,0,0,2,1,0,"
elementary abelian subgroups of order 4 with 3 elements of class 2C:
# 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, 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, 0,1,0,0,1,0,0,0,0,4,0,0,0,0,4,"

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,"

# subgroup of order 31 for the construction of regular packings in PG_3_5:

PGL_4_5_SUBGROUP_31=-PGL 4 5\
    -subgroup_by_generators "31" 31 1\
    "2,0,0,0, 0,0,1,0, 0,0,0,1, 0,3,0,4"

PGL_4_5_SUBGROUP_31_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,"
#372:
"1,0,0,0,4,0,0,0,0,4,0,0,0,0,4, "
"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,"

#Exterior square roots:

e#elt of order 3:
#the exterior square root of f is X=
#[1 0 0 0]
#[0 1 0 0]
#[0 0 2 2]
#[0 0 4 2]

#elt of order 31:
#the exterior square root of g is Z=
#[1 0 0 0]
#[0 3 4 3]
#[0 3 3 4]
#[0 3 2 3]

#Michael

HAMMING_CODE_CODEWORDS="0, 67, 37, 102, 22, 85, \
51, 112, 15, 76, 42, 105, 25, 90, 60, 127"

SIMPLEX_CODE_GENERATOR="\
1,0,1,0,1,0,1, \\
0,1,1,0,0,1,1, \\
0,0,0,1,1,1,1"

HAMMING_CODE_GENERATOR="\
1,0,0,0,0,1,1, \\
0,1,0,0,1,0,1, \\
0,0,0,1,0,1,0, \\
0,0,0,1,1,1,1"
GOLAY23_CODE_GENERATOR="\n1,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,1,0,0,1,0,\n0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,1,1,0,1,\n0,0,1,0,0,0,0,0,0,0,1,1,0,1,0,1,0,0,0,1,1,1,\n0,0,0,0,1,0,0,0,0,0,0,1,0,1,1,0,1,0,1,1,1,0,\n0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,0,1,0,1,1,1,0,\n0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,1,0,1,0,1,1,0,\n0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,1,0,1,0,1,1,0,\n0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,1,1,1,1,\n0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,1,1,1,1,0,\n0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1"

HAMMING_CODE_ROWS_IN_BINARY_RANKS="67,37,22,15"

SIMPLEX_CODE_GENMA_CYCLIC="\n1,0,0,1,1,1,0, \
0,1,0,1,1,1,1, \n0,0,1,1,1,0,1,0"

CODE_GV_N15_K6="\n111111111100000\n11110000010000\n11001100001000\n11010101000010\n10101011000010\n10110100100001"

CODE_GV_N15_K6_CREATE="\n100000000111111\n01000000111100\n00100000011101\n00010000011010\n00001000101010\n00000010100110\n00000001100001"
947  REED_MULLER_3_1_CODEWORDS="0,255,170,85,204,51,102,\n153,240,15,90,165,60,195,150,105"
948  REED_MULLER_3_1_BASIS_IN_BINARY="255,170,204,240"
949  REED_MULLER_4_1_COLUMNS_OF_PARITY_CHECK="1,3,5,7,9,11,13,\n15,17,19,21,23,25,27,29,31"
950  #-nearest_codeword "8,16,32,24,40,48,56,1,2,4,3,5,6,7,9,18,36,27,45,54,63,33,42,2\n0,11,53,62,31"
951  AG.2.3_BLOCKS="0,13,22,27,35,41,47,53,55,59,71,76"
952  LARGE_SET_AG.2.3_NEIGHBOR_SET="129,130,133,134,136,139,141,142,153,154,156,160,16\n5,166,178,179,183,184,185,190,192,194,197,203,204,206,218,221,222,225,227,231,248\n,251,252,255,256,259,261,262,272,277,279,283,285,287,299,301,303,305,306,309,313,\n315,319,320,323,325,341,342,343,344,345,349,368,371,375,378,381,383,392,393,397,4\n02,403,405,416,419,421,422,425,426,429,430,440,443,447,449,453,454,464,467,468,47\n3,474,479,490,493,494,497,500,503,513,517,518,520,523,527,536,539,541,542,544,547\n,548,551,563,566,567,571,572,573,585,589,590,593,595,596,600,601,603,611,614,615,\n625,629,631,635,637,638,657,659,661,667,688,691,693,705,706,709,710,712,715,717,718,720,723,724,729,733,735,747,748,750,752,754,757,777,780,78\n1,784,790,791,802,804,807,808,811,814,824,827,828,831,832,835,837,839"\n953  TEST_SYSTEM="\n954  0,1,0,1,0,0, \n955  0,0,1,0,1,0, \n956  1,0,1,0,0,0, \n957  0,1,0,1,0,1, \n958  1,0,0,0,0,1, \n959  1,0,1,0,0,0, \n960  0,1,0,1,1,1"\n961  TEST_RHS="1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1"\n962  PP4= -d1 1 -q1 3 -d2 1 -q2 7 -K 5 -search_control -W -end -problem_label PP4\n963  PP4_GROUP1=-subgroup "1,1,1,1, " "21" -group_label "cyclic21"
964  PP4_MASK1="\n965  531"
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

DELANDTSHEER_DOYEN_PROBLEM_COLBOURN_COLBOURN_7.13_GROUP1=-subgroup "1,1,1,1, " "9 1" -group_label "cyclic91"

DELANDTSHEER_DOYEN_PROBLEM_COLBOURN_COLBOURN_7.13_MASK1=

DELANDTSHEER_DOYEN_PROBLEM_27.53= -d1 1 -q1 27 -d2 1 -q2 53 -K 11 -DDx 2 -DDy 1 -
-search_control -W -end

DELANDTSHEER_DOYEN_PROBLEM_27.53_GROUP1=-subgroup \\
"1,1,1,0, 1,3,1,0, 1,9,1,0, 1,0,1,1, -2,0,-4,0" "18603" -group_label "group1"

# mask 1:
# XX.
# X.X+

DELANDTSHEER_DOYEN_PROBLEM_27.53_MASK1=

DELANDTSHEER_DOYEN_PROBLEM_3.7= -d1 1 -q1 3 -d2 1 -q2 7 -K 5 -DDx 3 -DDy 1 -
-search_control -W -end

DELANDTSHEER_DOYEN_PROBLEM_3.7_GROUP1=-subgroup \\
"1,1,1,0, 1,0,1,1 " "21" -group_label "group_cyclic"

DELANDTSHEER_DOYEN_PROBLEM_3.7_MASK1= -mask_label "mask1" -depth 5

PENTTILA_WILLIAMS_PRINCE_REG_PACKING_0="444,43313,154402,46682,108254,\n75363,27729,32139,5244,60442,142811,111115,94209,120678,89533,13798,\n
532
1027 103994,129953,82168,136838,19253,23017,145985,134996,54705,36267,\
1028 55066,117542,96699,69154,72460"
1029
1030 PENTTILA_WILLIAMS_PRINCE_REG_PACKING.1="616,42728,152655,48576,105431,\
1031 79607,28634,32817,9799,62356,141176,110085,92557,122136,86312,13975,\
1032 101942,126869,81478,139352,18028,24325,147284,130370,52074,36843,\
1033 55602,118454,95973,69642,"
1034
1035 PENTTILA_WILLIAMS_PRINCE_REG_PACKING.0_DUAL="3938, 66740, 56555, \
1036 93538, 107785, 64917, 47567, 54483, 141012, 138602, 18308, 6880, \
1037 131351, 88788, 125484, 102075, 21234, 99392, 119149, 80640, 124839, \
1038 148843, 71862, 11468, 35950, 27050, 75338, 113337, 40002, 154102, \
1039 30567"
1040
1041 PG_3_5_PACKING.0_WITH_AG03="0,5201,60427,86602,11453,121452,46663,\
1042 19716,32921,108680,23456,91963,68386,26921,74601,57067,36188,42312,\
1043 78780,53117,118488,114700,83960,99669,104791,126662,130960,145179,\
1044 137230,150626,140216"
1045
1046
1047 PG_3_5_PACKING.0_WITH_AG03_FIXP444="444,5001,12957,18194,23485,26817,\
1048 34667,38299,41249,47472,50450,56601,62638,68986,71833,75369,80805,\
1049 87025,92577,95676,104509,109718,114948,116333,124391,127498,133240,\
1050 137711,144777,148059,150175"
1051
1052
1053
1054
1055
1056 # consider the binary code with generator matrix:
1057 # 1 0 1
1058 # 0 1 1
1059 CODE_N3_K2_Q2_GENMA="1,0,1, 0,1,1"
1060
1061 CODE_N6_K3_Q2_GENMA="\n1062 111100\n1063 110010\n1064 101001"
1065
1066
1067
1068 TRIANGLE_GRAPH="0,1,1\n1,0,1\n1,1,0\n"
1069
1070
1071
1072 # q=17:
1073 # 3 is p.e. mod 17.
# so we pick \( f = 3 \).
then, \( 2f^2 = 18 = 1 \)
\( 4f = 12 \)

\[ X^4 - Y^4 - Z^4 + 2f^2 Y^2 Z^2 + 4f X^2 Y Z \]

\((1, -1, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2f^2, 4f, 0, 0)\)

```
EDGE\nCURVE\nQ17\nEQUATION="1,16,16,0,0,0,0,0,0,0,1,12,0,0"
EDGE\nCURVE\nQ17\nAS\nPOINTS="4, 7, 16, 19, 20, 23, 32, 35, 89, 100, 244, 251"
FILE\nQ17="orbit,curve,pts_on.curve,bitangents,go"
\n0,"$(EDGE\nCURVE\nQ17\nEQUATION)"","$(EDGE\nCURVE\nQ17\nAS\nPOINTS)"",","",-1\n\nEND"
```

```
DESARGUES_PATH\nLEX\nLEAST="10 10 3\n0\n2 0 15\n3 0 15\n26\n4 0 15\n26 46\n5 0 15\n26 46 56\n6 0 15\n26 46 56 72\n7 0 15\n26 46 56 72 80\n8 0 15\n26 46 56 72 80 93\n9 0 15\n26 46 56 72 80 93 106\n10 0 15\n26 46 56 72 80 93 106 119\n-1"
```

```
SPREADS_27_ISO_0="\n0, 33879, 1339, 2678, 3994, 7671, 10180, 5862, 9524, 6852, 22243, \n12745, 24295, 11062, 13615, 23894, 15056, 29367, 16429, 31521, 17726, \n31103, 18887, 26333, 19566, 28400, 21531, 27228"
```

```
SPREADS_27_ISO_1="\n0, 33879, 1339, 2678, 3994, 7671, 10182, 5761, 6796, 9327, 15339, \n31914, 24415, 12713, 22748, 11666, 13353, 23555, 30103, 16395, 17827, \n30790, 18254, 26422, 20046, 28112, 20900, 26801"
```

```
SPREADS_27_ISO_2="\n0, 33879, 1339, 2678, 3994, 7671, 10182, 5817, 6796, 9276, 23891, \n15368, 11666, 22124, 12713, 24415, 13353, 29619, 15910, 31914, 17030, \n30931, 19213, 26422, 19905, 28112, 21217, 27545"
```

```
SPREADS_27_ISO_3="\n0, 33879, 1339, 2678, 3994, 7671, 10625, 6590, 9476, 5576, 24688, \n23043, 10996, 22124, 12723, 13522, 15421, 29894, 16532, 32442, 17997, \n31015, 18311, 26109, 19807, 28113, 21220, 27195"
```
# Povray:

```plaintext
# 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 } "
```
COLOR_BLACK_SHINY="\n  "texture{ pigment{ color Black } \n  finish { diffuse 0.9 phong 1}}"

COLOR_RED_SEE_THROUGH="\n  "texture{ pigment{ color Red transmit 0.5 } \n  finish { diffuse 0.9 phong 1}}"

COLOR_GREEN_SEE_THROUGH="\n  "texture{ pigment{ color Green transmit 0.5 } \n  finish { diffuse 0.9 phong 1}}"

COLOR_BLUE_SEE_THROUGH="\n  "texture{ pigment{ color Blue transmit 0.5 } \n  finish { diffuse 0.9 phong 1}}"

COLOR_YELLOW_SEE_THROUGH="\n  "texture{ pigment{ color Yellow transmit 0.5 } \n  finish { diffuse 0.9 phong 1}}"

COLOR_YELLOW_THICK="\n  "texture{ pigment{ color Yellow } \n  finish { diffuse 0.9 phong 1}}"

COLOR_BLACK_NO_SHADOW="\n  "texture{ pigment{Black} } no_shadow"

SURFACE_COLOR="\n  "texture{ pigment{ White*0.5 } \n  finish {ambient 0.4 diffuse 0.5 roughness 0.001 \n  reflection 0.1 specular .8} }"

SURFACE_COLOR_SEETHROUGH="\n  "texture{ pigment{ White*0.5 transmit 0.5 } \n  finish {ambient 0.4 diffuse 0.5 roughness 0.001 \n  reflection 0.1 specular .8} }"

COLOR_GOLD="\n  "texture{ pigment{ Gold } finish \n  {ambient 0.4 diffuse 0.5 roughness 0.001 \n  reflection 0.1 specular .8} }"

COLOR_TURQUOISE="\n  "texture{ pigment{Cyan*1.3} \n  finish {ambient 0.4 diffuse 0.6 roughness 0.001 \n  reflection 0.1 specular .8} }"
 reflection 0 specular .8} }")

MONKEY_SADDLE_CUBIC="1,0,0,0,-3,0,0,0,0,0,0,0,0,0,0,0,-1,0"

ECKARDT_CUBIC_DEFORM1_LEX="0, 10, 0, -8, 10, 25, 2, 0, -20, -8, -20, -10, -24, 10 , -2, 12, 0, -8, 8, 16"

ECKARDT_CUBIC_DEFORM2_LEX="0, -5, 0, -5, -5, 10, -1, 0, 10, 4, 10, 5, 3, -5, 1, -6, 0, -5, -4, 1"

KUMMER_QUARTIC_LEX_35="-2,0,0,0,2,0,2,0,2,0,0,0,-2,0,2,0,-2"


ENDRASS_OCTIC_LEX_165="-93.2548,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-309.019,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-2110.12,0,-3165.17,0,-1186.94,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
# Section 2.3: Makefiles and Shell Scripts

SECTION MAKENFILES AND SHELL SCRIPTS:

# Section 2.4: Objects and Activities

SECTION OBJECTS AND ACTIVITIES:

test_2.4:
  make example_set
  make object_F_2
  make object_PG_3_2
  make vector_ex

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

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

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

define vector_ex:
  $(ORBITER) -v 2 \
  -define F -finite_field -q 5 -end \
  -define v -vector -field F -dense "0,1,2,3,4" -end \
  -print_symbols
## Section 2.5: Mathematical Data

### test_2.5:
- `make create_BLT_5_1`
- `make create_surface_4_0`

### create_BLT_5_1:
```
$ORBITER -v 2
DEFINE F -finite_field -q 5 -end
DEFINE 0 -orthogonal_space 0 5 F -end
WITH 0 -do -orthogonal_space_activity
CREATE_BLT_SET -catalogue 1 -end
```

### 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
```

## Section 2.6: Set Builder

### set_of_primes:
```
$ORBITER -v 2
DEFINE S -set -here "2,3,5,7,11,13" -end
PRINT_SYMBOLS
```

### set_interval:
```
$ORBITER -v 2 -define S -set -loop 0 64 1 -end
PRINT_SYMBOLS
```
# Section 2.7: Vector Builder

**SECTION VECTOR_BUILDER:**

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

### Vector Example 2:
```
$ (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
```

### 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
```

### Vector Example Repeat:
```
$ (ORBITER) -v 2 \
  >  > -define v -vector -repeat "0,1,2,3" 11 -end \
  >  > -print_symbols
```

### Vector Example All One 11:
```
$ (ORBITER) -v 2 \
  >  > -define v -vector -repeat 1 11 -end \
  >  > -print_symbols
```

### Matrix Example 1:
```
$ (ORBITER) -v 2 \
  >  > -define F -finite_field -q 2 -end \
  >  > -define v -vector -field F -format 4 \ 
  >  >  > -dense $(HAMMING_CODE_GENERATOR) -end \
  >  > -print_symbols
```
matrix_example.co_1:

-define F -finite_field -q 2 -end 
-define v -vector -field F -format 22 
-compact $(CONWAY_GEN1) -end 
-print_symbols

# Section 2.8: Formula Builder

SECTION_FORMULA_BUILDER:

TEST_FORMULA="(-a+b*b)*x*x+a*b*x"

formula_example:

-define f -formula 
"test_formula" "test\_formula" "" 
$(TEST_FORMULA) 

dot -Tpng test_formula.gv >test_formula.png

open test_formula.png

formula_evaluate:

-define F -finite_field -q 5 -end 
-define f -formula 
"test_formula" "test\_formula" "" 
$(TEST_FORMULA) 

-with F -do -finite_field_activity 
-evaluate f "a=2,b=3,x=4" -end

# should evaluate to 1, since (-2+3*3)*4\*4+2*3*4=2+4=6=1 mod 5
# but: problem with the leading minus sign

# Chapter 3 - Basic Algebra

SECTION_BASIC_NUMBER_THEORY:
1446
1447
1448
1449 PR29:
1450 ▶ $(ORBITER) -v 1 -smallest_primitive_root 29
1451
1452 PR31:
1453 ▶ $(ORBITER) -v 1 -smallest_primitive_root 31
1454
1455 PR37:
1456 ▶ $(ORBITER) -v 1 -smallest_primitive_root 37
1457
1458 PR_100:
1459 ▶ $(ORBITER) -v 1 -smallest_primitive_root.interval 2 100
1460
1461
1462
1463
1464 # randomized algo:
1465
1466 PR_915839:
1467 ▶ $(ORBITER) -v 5 -primitive_root 915839
1468
1469 # a primitive root modulo 915839 is 43085
1470
1471 PR_915839_check:
1472 ▶ $(ORBITER) -v 5 -power_mod 43085 49842 915839
1473
1474 # the power of 43085 to the 49842 mod 915839 is 487320
1475
1476 DL_915839:
1477 ▶ $(ORBITER) -v 5 -discrete_log 487320 43085 915839
1478
1479
1480 # The discrete log is 49842 since 487320 = 43085^49842 mod 915839, time: 0:22
1481
1482 IM_723:
1483 ▶ $(ORBITER) -v 5 -inverse_mod 723 4060
1484
1485
1486
1487 IM_3.19:
1488 ▶ $(ORBITER) -v 5 -inverse_mod 3 19
1489
1490
1491 IM:
1492 ▶ $(ORBITER) -v 5 -inverse_mod 1865025205 2147483647
IMgcd:
$(\text{ORBITER}) -v 5 -extended\_gcd 1865025205 2147483647

PM3a:
$(\text{ORBITER}) -v 5 -power\_mod 16807 1073741823 2147483647

sqrt\_mod:
$(\text{ORBITER}) -v 2 -square\_root\_mod 33 41

sqrt\_5\_mod\_11:
$(\text{ORBITER}) -v 2 -square\_root\_mod 5 11

sqrt\_5\_mod\_19:
$(\text{ORBITER}) -v 2 -square\_root\_mod 5 19

sqrt\_mod\_20\_31:
$(\text{ORBITER}) -v 2 -square\_root\_mod 20 31

order\_of\_2\_mod\_n:
$(\text{ORBITER}) -v 3 -order\_of\_q\_mod\_n 2 3 151

order\_of\_q\_mod\_n\_q\_2\_3\_151.csv
pdflatex order\_of\_q\_mod\_n\_q\_2\_3\_151.tex
open order\_of\_q\_mod\_n\_q\_2\_3\_151.pdf

Eulerfunction\_150:
$(\text{ORBITER}) -v 1 -eulerfunction\_interval 1 150

Eulerfunction\_900:
$(\text{ORBITER}) -v 1 -eulerfunction\_interval 900 1150

543
Eulerfunction_10000:

$\text{(ORBITER)} -v 1 -eulerfunction\_interval 10000 10150$

$\text{(ORBITER)} -v 1 -csv\_file\_latex 1 \$

$\text{pdflatex table\_eulerfunction\_10000\_10150.tex}$

open table\_eulerfunction\_10000\_10150.pdf

PR_1000:

$\text{(ORBITER)} -v 1 -smallest\_primitive\_root\_interval 2 1000$

$\text{(ORBITER)} -v 1 -csv\_file\_latex 1 \$

$\text{pdflatex primitive\_element\_table\_2.1000.csv}$

open primitive\_element\_table\_2.1000.pdf

PE\_number\_1000:

$\text{(ORBITER)} -v 1 -number\_of\_primitive\_roots\_interval 2 1000$

$\text{(ORBITER)} -v 1 -csv\_file\_latex 1 \$

$\text{pdflatex table\_number\_of\_pe\_2.1000.csv}$

open table\_number\_of\_pe\_2.1000.pdf

number\_of\_primitive\_roots\_10000:

$\text{(ORBITER)} -v 1 -number\_of\_primitive\_roots\_interval 10000 10001$

power\_function\_2\_mod\_11:

$\text{(ORBITER)} -v 5 -power\_function\_mod\_n 2 11$

$\text{(ORBITER)} -v 1 -csv\_file\_latex 1 \$

$\text{pdflatex power\_function\_k2\_n11.csv}$

open power\_function\_k2\_n11.pdf

draw\_mod\_13:

$\text{(ORBITER)} -v 2 \$

$\text{pdflatex mod\_13\_draw.tex}$

open mod\_13\_draw.pdf

Chinese\_remainders\_A:

$\text{(ORBITER)} -v 2 \$

$\text{-define R -vector -dense "2,2,5" -end \$

$\text{-define M -vector -dense "5,6,7" -end \$

$\text{-Chinese\_remainders R M}$
Chinese_remainders_B:

```
$ORBITER -v 2 \n DEFINE R -VECTOR -DENSE "38,2" -END \n DEFINE M -VECTOR -DENSE "74,27" -END \n -CHINESE Remainders R M \n```

Chinese_remainders_C2:

```
$ORBITER -v 2 \n DEFINE R -VECTOR -DENSE "2,3" -END \n DEFINE M -VECTOR -DENSE "2147483647,5915587277" -END \n -CHINESE Remainders R M \n```

The solution is 5684294357108828365 modulo 12703626939758759219 (computed in longinteger)

# checking with Maple:

```
#5684294357108828365 mod 2147483647;
# 2
```

```
#5684294357108828365 mod 5915587277;
# 3
```

Chinese_remainders_C3:

```
$ORBITER -v 2 \n DEFINE R -VECTOR -DENSE "2,3,4" -END \n DEFINE M -VECTOR -DENSE "2147483647,5915587277,3267000013" -END \n -CHINESE Remainders R M \n```

The solution is 31431541759324477327451572539 modulo 41502749377339016585336869847 (computed in longinteger)

# checking with Maple:

```
#31431541759324477327451572539 mod 2147483647;
# 2
```

```
#31431541759324477327451572539 mod 5915587277;
# 3
```

```
#31431541759324477327451572539 mod 3267000013;
# 4
```
# Section 3.2: Prime Fields

SECTION PRIME_FIELDS:

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

F_3:
$(ORBITER) -v 3 \
   -define F -finite_field -q 3 -end \
   -with F -do -finite_field_activity -cheat_sheet_GF -end
   #pdflatex GF_3.tex
   #open GF_3.pdf

F_5:
$(ORBITER) -v 3 \
   -define F -finite_field -q 5 -end \
   -with F -do -finite_field_activity -cheat_sheet_GF -end
   pdflatex GF_5.tex
   open GF_5.pdf

F_5_add_table:
$(ORBITER) -v 3 \
   -define F -finite_field -q 5 -end \
   -with F -do -finite_field_activity -cheat_sheet_GF -end \
   -draw_matrix -input_csv_file GF_q5_addition_table.csv \
   -box_width 40 -bit_depth 24 -partition 3 5 5 -end 

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

F_127:
$\text{ORBTER} -v 3$

-define F -finite_field -q 127 -end

-with F -do -finite_field_activity -cheat_sheet_GF -end

$\text{ORBTER} -v 3$

-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

$\text{ORBTER} -v 3$

-define F -finite_field -q 7 -end

-with F -do -finite_field_activity

-Vandermonde_matrix

-end

$\text{ORBTER} -v 3$

-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

$\text{ORBTER} -v 3$

-define F -finite_field -q 1021 -without_tables -end

-with F -do -finite_field_activity -cheat_sheet_GF -end

# Section 3.3: Extension Fields

SECTION EXTENSION_FIELDS:

$\text{ORBTER} -v 3$

-define F -finite_field -q 4 -end
F_4_tables:

F_16:

F_16_tables:
Section 3.4: Linear Algebra over Finite Fields

SECTION LINEAR_ALGEBRA:

RREF:

$\begin{verbatim}
\define F -finite_field -q 2 -end \\
\define v -vector -field F -format 2 \\
\end{verbatim}$

$\begin{verbatim}
\dense "1,1,1,0,1,0,0,1" \\
\end{verbatim}$

$\begin{verbatim}
\with F -do -finite_field_activity \\
\RREF v -normalize_from_the_right \\
\end{verbatim}$

$\begin{verbatim}
RREF_V7:
\define F -finite_field -q 7 -end \\
\define V -vector -format 7 \\
\dense $(V7\_VANDERMONDE\_EXTENDED)$ \\
\end{verbatim}$

$\begin{verbatim}
\with F -do -finite_field_activity \\
\RREF V \\
\end{verbatim}$

nullspace:

$\begin{verbatim}
\define F2 -finite_field -q 2 -end \\
\define v -vector -field F2 -format 2 \\
\end{verbatim}$

$\begin{verbatim}
\dense "1,1,1,0,1,0,0,1" \\
\end{verbatim}$

$\begin{verbatim}
\with F2 -do \\
-finite_field_activity \\
-nullspace v \\
-normalize_from_the_right \\
\end{verbatim}$
eigenstuff:

\$\text{ORBITER} -v 6 \$

\$\text{ORBITER} -v 7 \$

\$\text{ORBITER} -v 2 \$

\$\text{ORBITER} -v 2 \$

\$\text{ORBITER} -v 2 \$

RREF demo 4 4 q5:

RREF demo 4 6 q7:
### Section 3.5: Advanced Topics in finite fields

#### RREF.demo_4_8_q7:

```latex
\$(ORBITER) -v 2 \\
-define F -finite_field -q 8 -end \\
-define v -vector -field F -format 4 \\
-compact $(CODE RS 6 4 7) \\
-end \\
-with F -do \\
-finite_field_activity -RREF_random_matrix 4 8 -end
```

#### RREF.demo_4_8_q8:

```latex
\$(ORBITER) -v 2 \\
-define F -finite_field -q 8 -end \\
-define v -vector -field F -format 4 \\
-compact $(CODE RS 6 4 7) \\
-end \\
-with F -do \\
-finite_field_activity -RREF_random_matrix 4 8 -end
```

#### RREF.demo_6_4_7:

```latex
\$(ORBITER) -v 2 \\
-define F -finite_field -q 7 -end \\
-define v -vector -field F -format 4 \\
-compact $(CODE RS 6 4 7) \\
-end \\
-with F -do \\
-finite_field_activity -RREF v -end
```

# Section 3.5: Advanced Topics in finite fields
normal_basis_2.3:
$\$(ORBITER) -v 2 \\  
 DEFINE F \-finite\_field \-q 2 \-end \\  
 WITH F \-do \-finite\_field\_activity \\  
 NORMAL\_basis 3 \-end

normal_basis_2.6:
$\$(ORBITER) -v 2 \\  
 DEFINE F \-finite\_field \-q 2 \-end \\  
 WITH F \-do \-finite\_field\_activity \\  
 NORMAL\_basis 6 \-end

F8_over_F2\_field\_reduction:
$\$(ORBITER) -v 2 \\  
 DEFINE F \-finite\_field \-q 8 \-end \\  
 LOOP L 0 8 1 \\  
 WITH F \-do \\  
\-finite\_field\_activity \\  
\-field\_reduction "F8\_red\_\%L" 2 1 1 "\%L" \\  
\-end \\  
END\_loop \\  
$\$(ORBITER) -v 2 \-loop L 0 8 1 \\  
 \-draw\_matrix \-input\_csv\_file F8\_red\_\%L.csv \\  
 \-box\_width 40 \-bit\_depth 24 \-partition 4 3 3 \-end \\  
 END\_loop \\  
#pdflatex field\_reduction\_Q8\_q2.5.7.tex

F64_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" \-end \\  
 open F64\_over\_F8\_draw.bmp
1961  ▶  #pdflatex field_reduction_Q64.q8_8_8.tex
1962  ▶  #open field_reduction_Q64.q8_8_8.pdf
1963
1964
1965  F_64_over_F4_field_reduction:
1966  ▶  $(ORBITER) -v 2 \  
1967  ▶  ▶  -define F -finite_field -q 64 -end \  
1968  ▶  ▶  -define elts -vector -field F -loop 0 64 1 -end \  
1969  ▶  ▶  -with F -do \  
1970  ▶  ▶  -finite_field_activity \  
1971  ▶  ▶  ▶  -field_reduction "F64_over_F4" 4 8 8 elts -end
1972  ▶  $(ORBITER) -v 2 -draw_matrix \  
1973  ▶  ▶  -input_csv_file F64_over_F4.csv \  
1974  ▶  ▶  -box_width 40 -bit_depth 24 \  
1975  ▶  ▶  -partition 4 "3,3,3,3,3,3,3,3" "3,3,3,3,3,3,3,3" -end
1976  ▶  open F64_over_F4_draw.bmp
1977  ▶  #pdflatex field_reduction_Q64.q4_8_8.tex
1978  ▶  #open field_reduction_Q64.q4_8_8.pdf
1979
1980
1981  F_64_over_F2_field_reduction:
1982  ▶  $(ORBITER) -v 2 \  
1983  ▶  ▶  -define F -finite_field -q 64 -end \  
1984  ▶  ▶  -define elts -vector -field F -loop 0 64 1 -end \  
1985  ▶  ▶  -with F -do \  
1986  ▶  ▶  -finite_field_activity \  
1987  ▶  ▶  ▶  -field_reduction "F64_over_F2" 2 8 8 elts -end
1988  ▶  $(ORBITER) -v 2 -draw_matrix \  
1989  ▶  ▶  -input_csv_file F64_over_F2.csv \  
1990  ▶  ▶  -box_width 40 -bit_depth 24 \  
1991  ▶  ▶  -partition 4 "6,6,6,6,6,6,6,6" "6,6,6,6,6,6,6,6" -end
1992  ▶  open F64_over_F2_draw.bmp
1993  ▶  #pdflatex field_reduction_Q64.q2_8_8.tex
1994  ▶  #open field_reduction_Q64.q2_8_8.pdf
1995
1996
1997
1998
1999
2000  F_8_Nth_roots_21:
2001  ▶  $(ORBITER) -v 3 \  
2002  ▶  ▶  -define F -finite_field -q 8 -override_polynomial 11 -end \  
2003  ▶  ▶  -with F -do -coding_theoretic_activity \  
2004  ▶  ▶  ▶  -nth_roots 21 \  
2005  ▶  ▶  ▶  -end
2006  ▶  pdflatex Nth_roots_q8_n21.tex
2007  ▶  open Nth_roots_q8_n21.pdf
2008
2009
2010
2011
2012 F_8.vandermonde:
2013 $\diamond$ $(\text{ORBITER}) -v 3 \ \$
2014 $\diamond$ $\diamond$ -define F -finite_field -q 8 -end \ 
2015 $\diamond$ $\diamond$ -with F -do -finite_field_activity \ 
2016 $\diamond$ $\diamond$ $\diamond$ -Vandermonde_matrix \ 
2017 $\diamond$ $\diamond$ -end
2018
2019
2020
2021 F_1024.vandermonde:
2022 $\diamond$ $(\text{ORBITER}) -v 3 \ \$
2023 $\diamond$ $\diamond$ -define F -finite_field -q 1024 -end \ 
2024 $\diamond$ $\diamond$ -with F -do -finite_field_activity \ 
2025 $\diamond$ $\diamond$ $\diamond$ -Vandermonde_matrix \ 
2026 $\diamond$ $\diamond$ -end
2027 $\diamond$ rm Vandermonde_1024.csv
2028 $\diamond$ rm Vandermonde_inv_1024.csv
2029
2030 #User time: 0:46
2031
2032
2033
2034
2035 NTT_k4_q17.cpp:
2036 $\diamond$ $(\text{ORBITER}) -v 3 \ \$
2037 $\diamond$ $\diamond$ -define F -finite_field -q 17 -end \ 
2038 $\diamond$ $\diamond$ -with F -do -coding_theoretic_activity \ 
2039 $\diamond$ $\diamond$ $\diamond$ -NTT 4 17 \ 
2040 $\diamond$ $\diamond$ -end
2041
2042 F_17_NTT_compile: NTT_k4_q17.cpp
2043 $\diamond$ $(\text{MYCPP})$ NTT_k4_q17.cpp $(CPPFLAGS) \ 
2044 $\diamond$ $\diamond$ $(\text{LIB})$ $(LFLAGS)$ -o NTT_k4_q17.out
2045 $\diamond$ $\diamond$ ./NTT_k4_q17.out
2046
2047
2048
2049 ####################################################################################################################################
2050 # Section 3.6: Basic Ring Theory
2051
2052
2053
2054 SECTION_BASIC_RING_THEORY:
Polynomial ring:
\[ \text{define F -finite_field -q 4 -end} \]
\[ \text{define R -polynomial_ring -field F} \]
\[ \text{number of variables 4} \]
\[ \text{homogeneous of degree 3} \]
\[ \text{variables } x_0, x_1, x_2, x_3 \]
\[ \text{end} \]

# Chapter 4 - Geometry

# Section 4.1: Finite Projective Spaces

SECTION_FINITE_PROJECTIVE_SPACES:

PG_3_easy:
\[ \text{define F -finite_field -q 2 -end} \]
\[ \text{define P -projective_space -n 3 -field F -end} \]

PG_1_16:
\[ \text{define F -finite_field -q 16 -end} \]
\[ \text{define P -projective_space -n 1 -field F -v 0 -end} \]
\[ \text{with P -do -projective_space_activity} \]
\[ \text{-cheat_sheet} \]
\[ \text{-end} \]
\[ \text{pdflatex PG_1_16.tex} \]
\[ \text{open PG_1_16.pdf} \]

PG_2_4:
\[ \text{define F -finite_field -q 2 -end} \]
\begin{itemize}
\item \texttt{-define F -finite_field -q 4 -end}
\item \texttt{-define P -projective_space -n 2 -field F -v 0 -end}
\item \texttt{-define P -projective_space -n 2 -field F -v 0 -end}
\item \texttt{-define P -projective_space -n 2 -field F -v 0 -end}
\item \texttt{-define P -projective_space -n 2 -field F -v 0 -end}
\item \texttt{-define P -projective_space -n 2 -field F -v 0 -end}
\item \texttt{-define P -projective_space -n 2 -field F -v 0 -end}
\item \texttt{-define P -projective_space -n 2 -field F -v 0 -end}
\item \texttt{-define P -projective_space -n 2 -field F -v 0 -end}
\item \texttt{-define P -projective_space -n 2 -field F -v 0 -end}
\item \texttt{-define P -projective_space -n 2 -field F -v 0 -end}
\item \texttt{-define P -projective_space -n 2 -field F -v 0 -end}
\end{itemize}
2149 PG_3_4:
2150 $\text{(ORBITER)} -v 2 \$
2151 $\text{define F -finite_field -q 4 -end}$
2152 $\text{define P -projective_space -n 3 -field F -v 0 -end}$
2153 $\text{with P -do -projective_space_activity}$
2154 $\text{cheat_sheet}$
2155 $\text{-end}$
2156 pdflatex PG_3_4.tex
2157 open PG_3_4.pdf
2158
2159 PG_3_5:
2160 $\text{(ORBITER)} -v 2 \$
2161 $\text{define F -finite_field -q 5 -end}$
2162 $\text{define P -projective_space -n 3 -field F -v 0 -end}$
2163 $\text{with P -do -projective_space_activity}$
2164 $\text{cheat_sheet}$
2165 $\text{-end}$
2166 pdflatex PG_3_5.tex
2167 open PG_3_5.pdf
2168
2169 PG_3_7:
2170 $\text{(ORBITER)} -v 2 \$
2171 $\text{define F -finite_field -q 7 -end}$
2172 $\text{define P -projective_space -n 3 -field F -v 0 -end}$
2173 $\text{with P -do -projective_space_activity}$
2174 $\text{cheat_sheet}$
2175 $\text{-end}$
2176 pdflatex PG_3_7.tex
2177 open PG_3_7.pdf
2178
2179 PG_3_8:
2180 $\text{(ORBITER)} -v 2 \$
2181 $\text{define F -finite_field -q 8 -end}$
2182 $\text{define P -projective_space -n 3 -field F -v 0 -end}$
2183 $\text{with P -do -projective_space_activity}$
2184 $\text{cheat_sheet}$
2185 $\text{-end}$
2186 pdflatex PG_3_8.tex
2187 open PG_3_8.pdf
2188
2189 PG_3_16:
2190 $\text{(ORBITER)} -v 2 \$
2191
557
\define F -finite_field -q 16 -end \\n\define P -projective_space -n 3 -field F -v 0 -end \\n\with P -do -projective_space_activity \ opens cheat sheet \\n\end

pdflatex PG_3.16.tex
open PG_3.16.pdf

PG_3.25:
\define F -finite_field -q 25 -end \\n\define P -projective_space -n 3 -field F -v 0 -end \\n\with P -do -projective_space_activity \\n\open cheat sheet \\n\end
pdflatex PG_3.25.tex
open PG_3.25.pdf

PG_4.3:
\define F -finite_field -q 3 -end \\n\define P -projective_space -n 4 -field F -v 0 -end \\n\with P -do -projective_space_activity \\n\open cheat sheet \\n\end
pdflatex PG_4.3.tex
open PG_4.3.pdf

PG_8.2:
\define F -finite_field -q 2 -end \\n\define P -projective_space -n 8 -field F -v 0 -end \\n\with P -do -projective_space_activity \\n\open cheat sheet \\n\end
pdflatex PG_8.2.tex
open PG_8.2.pdf

558
# Section 4.2: Indexing Points

SECTION_INDEXING_POINTS:

test 4.2:

test 4.2: make PG_2_4_rank_point

test 4.2: make PG_2_4_unrank_point

test 4.2: make elliptic_curve_b1_c3_q11.txt

test 4.2: make PG_2_2.incidence_matrix

test 4.2: make PG_2_4.incidence_matrix

test 4.2: make PG_2_8.incidence_matrix

PG_2_4.rank_point:

PG_2_4.unrank_point:

elliptic_curve_b1_c3_q11.txt:

PG_2_2.incidence_matrix:
define P -projective_space -n 2 -field F -v 0 -end 
-define all_one -vector -repeat 1 7 -end 
-draw_matrix 
-input_csv_file PG_n2_q2_incidence_matrix.csv 
-box_width 20 -bit_depth 8 
-partition 3 
-all_one all_one 
-end 
open PG_n2_q2_incidence_matrix_draw.bmp

PG_2.4_incidence_matrix:
$(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 
-export_point_line_incidence_matrix 
-end 
-input_csv_file PG_n2_q4_incidence_matrix.csv 
-box_width 20 -bit_depth 8 
-partition 3 
-all_one all_one 
-end 
open PG_n2_q4_incidence_matrix_draw.bmp

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 
-input_csv_file PG_n2_q8_incidence_matrix.csv 

# writes PG_n2_q4_incidence_matrix.csv
# Section 4.3: Finite Desarguesian Projective Planes

**SECTION FINITE DESARGUESIAN PROJECTIVE PLANES:**

```
2354 test_4_3:
2355  ▶ make PG_2_16
2356  ▶ make PG_2_4_with_decomposition
2357  ▶ make PG_2_4_incma_cyclic
2358  ▶ make PG_2_4_incma_singer_sub_3
2359  ▶ make PG_2_4_incma_singer_sub_7
2360
2361 PG_2_16:
2362  ▶ $(ORBITER) -v 2 \\
2363  ▶  ▶ -draw_options -xin 20000 -yin 20000 \\
2364  ▶  ▶ -radius 200 -line_width 0.3 -nodes_empty -end \\
2365  ▶  ▶ -define F -finite_field -q 16 -end \\
2366  ▶  ▶ -define P -projective_space -n 2 -field F -v 0 -end \\
2367  ▶  ▶ -with P -do -projective_space_activity \\
2368  ▶  ▶  ▶ -cheat_sheet \\
2369  ▶  ▶ -end
2370
2371  ▶ pdflatex PG_2_16.tex
2372  ▶ open PG_2_16.pdf
2373
2374
2375
2376
2377 PG_2_4_with_decomposition:
2378  ▶ $(ORBITER) -v 2 \\
2379  ▶  ▶ -define F -finite_field -q 4 -end \\
2380  ▶  ▶ -define P -projective_space -n 2 -field F -v 0 -end \\
2381  ▶  ▶ -with P -do -projective_space_activity \\
2382  ▶  ▶  ▶ -cheat_sheet_for_decomposition_by_element_PG \\
```

561
2384 \triangleright \triangleright \triangleright 1 \"0,1,0,0,0,1,2,1,1,0\" \ \\
2385 \triangleright \triangleright \triangleright "PG_2.4_singer" \ \\
2386 \triangleright \triangleright -end \ \\
2387 \triangleright \pdfLaTeX{} PG_2.4_singer.tex \ \\
2388 \triangleright \text{open PG}_2.4_singer.pdf \ \\
2389 \ \\
2390 \ \\
2391 \#PG_2.4_singer_incma_cyclic.csv \ \\
2392 \#PG_2.4_singer_incma_subgroup_index_3.csv \ \\
2393 \#PG_2.4_singer_incma_subgroup_index_7.csv \ \\
2394 \ \\
2395 \ \\
2396 \ \\
2397 PG_2.4.incma.cyclic: \ \\
2398 \triangleright \$\text{(ORBITER)} -v 2 \ \\
2399 \triangleright \triangleright -list_arguments \ \\
2400 \triangleright \triangleright -define R -vector -repeat 1 21 -end \ \\
2401 \triangleright \triangleright -define C -vector -repeat 1 21 -end \ \\
2402 \triangleright \triangleright -draw_matrix \ \\
2403 \triangleright \triangleright -input_csv_file PG_2.4_singer_incma_cyclic.csv \ \\
2404 \triangleright \triangleright -box_width 40 -bit_depth 24 \ \\
2405 \triangleright \triangleright -partition 3 R C \ \\
2406 \triangleright \triangleright -end \ \\
2407 \triangleright \text{open PG}_2.4_singer_incma_cyclic_draw.bmp \ \\
2408 \ \\
2409 \ \\
2410 PG_2.4.incma_singer_sub_3: \ \\
2411 \triangleright \$\text{(ORBITER)} -v 2 \ \\
2412 \triangleright \triangleright -list_arguments \ \\
2413 \triangleright \triangleright -define R -vector -repeat 3 7 -end \ \\
2414 \triangleright \triangleright -define C -vector -repeat 3 7 -end \ \\
2415 \triangleright \triangleright -draw_matrix \ \\
2416 \triangleright \triangleright -input_csv_file PG_2.4_singer_incma_subgroup_index_3.csv \ \\
2417 \triangleright \triangleright -box_width 40 -bit_depth 24 \ \\
2418 \triangleright \triangleright -partition 3 R C \ \\
2419 \triangleright \triangleright -end \ \\
2420 \triangleright \text{open PG}_2.4_singer_incma_subgroup_index_3_draw.bmp \ \\
2421 \ \\
2422 PG_2.4.incma_singer_sub_7: \ \\
2423 \triangleright \$\text{(ORBITER)} -v 2 \ \\
2424 \triangleright \triangleright -draw_matrix \ \\
2425 \triangleright \triangleright -input_csv_file PG_2.4_singer_incma_subgroup_index_7.csv \ \\
2426 \triangleright \triangleright -box_width 20 -bit_depth 24 \ \\
2427 \triangleright \triangleright -partition 3 3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3 -end \ \\
2428 \triangleright \text{open PG}_2.4_singer_incma_subgroup_index_7_draw.bmp \ \\
2429 \ \\
2430
# Section 4.4: The Grassmannian

SECTION GRASSMANNIAN:

---

test 4.4:

$\text{make GR}_3 2$

$\text{make PG}_3 3$

$\text{rank lines}$

$\text{make PG}_3 5$

$\text{unrank lines}$

$\text{make planes in pencil}$

---

# ToDo

---

GR 3 2 2:

$\text{\verb!(ORBITER)! -v 2 \ $

$\text{\verb!-define F -finite_field -q 2 -end \ $

$\text{\verb!-with F -do -finite_field_activity \ $

$\text{\verb!-cheat_sheet_Gr 3 2 -end \ $

$\text{\verb!pdflatex Gr 3 2 2.tex \ $

$\text{\verb!open Gr 3 2 2.pdf \ $

---

PG 3 3 rank lines:

$\text{\verb!(ORBITER)! -v 2 \ $

$\text{\verb!-define v1 -vector -format 3 \ $

$\text{\verb!-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" \ $

$\text{\verb!-end \ $

$\text{\verb!-define v2 -vector -format 3 \ $

$\text{\verb!-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" \ $

$\text{\verb!-end \ $

$\text{\verb!-define F -finite_field -q 3 -end \ $

$\text{\verb!-define P -projective_space -n 3 -field F -v 0 -end \ $

$\text{\verb!-with P -do \ $

---
# Section 4.5: Algebraic Sets

SECTION ALGEBRAIC SETS:

test 4.5:
make EC_11.txt
make Hirschfeld_surface_q4.txt
make Hirschfeld_surface_q16.txt
EC_11.txt:

```bash
$ (ORBITER) -v 2
```

```bash
> -define F -finite_field -q 11 -end
```

```bash
> -define R -polynomial_ring -field F
```

```bash
> -number_of_variables 3
```

```bash
> -homogeneous_of_degree 3
```

```bash
> -end
```

```bash
> -define P -projective_space -n 2 -field F -v 0 -end
```

```bash
> -define EC -geometric_object P
```

```bash
> -projective_variety R
```

```bash
> -EC_11 "EC\_11"
```

```bash
> $(EC_11_EQUATION)
```

```bash
> -end
```

```bash
> -with EC -do -combinatorial_object_activity -save
```

```bash
> -end
```

Hirschfeld_surface_q4.txt:

```bash
$ (ORBITER) -v 2
```

```bash
> -define F -finite_field -q 4 -end
```

```bash
> -define R -polynomial_ring -field F
```

```bash
> -number_of_variables 4
```

```bash
> -homogeneous_of_degree 3
```

```bash
> -end
```

```bash
> -define P -projective_space -n 3 -field F -v 0 -end
```

```bash
> -define H4 -geometric_object P
```

```bash
> -projective_variety R
```

```bash
> "Hirschfeld_surface_q4"
```

```bash
> "Hirschfeld\_surface\_q4"
```

```bash
> $(HIRSCHFELD_SURFACE_EQUATION)
```

```bash
> -end
```

```bash
> -with H4 -do -combinatorial_object_activity -save
```

```bash
> -end
```

# creates Hirschfeld_surface_q4.txt

Hirschfeld_surface_q16.txt:

```bash
$ (ORBITER) -v 2
```

```bash
> -define F -finite_field -q 16 -end
```

```bash
> -define R -polynomial_ring -field F
```

```bash
> -number_of_variables 4
```

```bash
> -homogeneous_of_degree 3
```

```bash
> -end
```

```bash
> -define P -projective_space -n 3 -field F -v 0 -end
```

565
2572 \define H16 \geometric_object P \projective Variety R \\Hirschfeld\_surface\_q16 \\Hirschfeld\_surface\_q16 \$(HIRSCHFELD\_SURFACE\_EQUATION) \end\
2579 \with H16 \do \combinatorial\_object\_activity \save \\
2582 # the coefficient vector is given as a list of pairs. 
2583 # 165 = binomial(11,3) 

2587 # Section 4.6: The Klein Quadric and Pluecker coordinates 
2591 SECTION\_KLEIN\_QUADRIC\_AND\_PLUECKER\_COORDINATES: 
2595 test 4 6: 
2600 \define F \finite_field -q 2 \end 
2602 \with F \do \finite_field\_activity \end 
2604 \cheat\_sheet Gr 4 2 -end 
2605 pdflatex Gr 4 2 2.tex 
2606 open Gr 4 2 2.pdf 

2608 # Section 4.7: Orthogonal spaces 
2612 SECTION\_ORTHOGONAL\_SPACES: 
2615 test 4 7: 
2616 \make Op 4 2 
2617 \make 0 5 2\_incidence\_matrix.csv 
2618 \make Op 6 2
\[ $(ORBITER) -v 2 \]
\[ -define F -finite_field -q 2 -end \]
\[ -define O -orthogonal_space 1 4 F -without_group -end \]
\[ -with 0 -do -orthogonal_space_activity \]
\[ -cheat_sheet_orthogonal -end \]
\[ pdflatex 0.1_4.2_report.tex \]
\[ open 0.1_4.2_report.pdf \]

\[ $(ORBITER) -v 2 \]
\[ -define F -finite_field -q 2 -end \]
\[ -define O -orthogonal_space 0 5 F -without_group -end \]
\[ -with 0 -do -orthogonal_space_activity \]
\[ -export_point_line_incidence_matrix \]
\[ -end \]

\[ $(ORBITER) -v 2 \]
\[ -define all_one_r -vector -repeat 1 15 -end \]
\[ -define all_one_c -vector -repeat 1 15 -end \]
\[ -draw_matrix \]
\[ -input_csv_file 0.5_2_incidence_matrix.csv \]
\[ -box_width 20 -bit_depth 8 \]
\[ -partition 2 \]
\[ all_one_r all_one_c \]
\[ -end \]
\[ open 0.5_2_incidence_matrix_draw.bmp \]

#O(5,2) projectively = Q(4,2) = (dual of) W(3,2) = W(3,2)
# recall that W(3,2) and Q(4,q) are self dual if q is even
2666 0p_6_2:
2667 ▶ $(ORBITER) -v 2 \
2668 ▶ ▶ -define F -finite_field -q 2 -end \n2669 ▶ ▶ -define 0 -orthogonal_space 1 6 F -without_group -end \n2670 ▶ ▶ -with 0 -do -orthogonal_space_activity \n2671 ▶ ▶ ▶ -cheat_sheet_orthogonal -end
2672 ▶ pdflatex 0_1_6_2_report.tex
2673 ▶ open 0_1_6_2_report.pdf
2674
2675
2676 0p_6_2.incidence_matrix.csv:
2677 ▶ $(ORBITER) -v 2 \
2678 ▶ ▶ -define F -finite_field -q 2 -end \n2679 ▶ ▶ -define 0 -orthogonal_space 1 6 F -without_group -end \n2680 ▶ ▶ -with 0 -do -orthogonal_space_activity \n2681 ▶ ▶ ▶ -export_point_line_incidence_matrix \n2682 ▶ ▶ -end
2683 ▶ ▶ $(ORBITER) -v 2 \
2684 ▶ ▶ ▶ -define all_one_r -vector -repeat 1 35 -end \n2685 ▶ ▶ ▶ -define all_one_c -vector -repeat 1 105 -end \n2686 ▶ ▶ ▶ -draw_matrix \n2687 ▶ ▶ ▶ ▶ -input_csv_file Op_6_2.incidence_matrix.csv \n2688 ▶ ▶ ▶ ▶ -box_width 20 -bit_depth 8 \n2689 ▶ ▶ ▶ ▶ -partition 2 \n2690 ▶ ▶ ▶ ▶ ▶ all_one_r all_one_c \n2691 ▶ ▶ ▶ ▶ -end
2692 ▶ ▶ open Op_6_2.incidence_matrix_draw.bmp
2693
2694
2695 0p_6_2.with_group:
2696 ▶ $(ORBITER) -v 2 \
2697 ▶ ▶ -define F -finite_field -q 2 -end \n2698 ▶ ▶ -define 0 -orthogonal_space 1 6 F -end \n2699 ▶ ▶ -with 0 -do -orthogonal_space_activity \n2700 ▶ ▶ ▶ -cheat_sheet_orthogonal -end
2701 ▶ pdflatex 0_1_6_2_report.tex
2702 ▶ open 0_1_6_2_report.pdf
2703
2704 # problem:
2705 # error message:
2706 #stabilizer_chain_base.data::allocate_base.data degree is too large
2707
2708
2709 0p_6_8:
2710 ▶ $(ORBITER) -v 2 \
2711 ▶ ▶ -define F -finite_field -q 8 -end \n2712 ▶ ▶ -define 0 -orthogonal_space 1 6 F -without_group -end \n
Op 8.2:
$\text{(ORBITER)} -v 2 \$
-define F -finite_field -q 2 -end 
-define O -orthogonal_space 1 8 F -without_group -end 
-with O -do -orthogonal_space_activity 
-cheat_sheet_orthogonal -end
\pdflatex O168_report.tex
open O168_report.pdf

Op 6.64:
$\text{(ORBITER)} -v 2 \$
-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 0_1_6_64_report.tex
open 0_1_6_64_report.pdf

# problem, because we are trying to create PGL(6,64):

Op 6.64_line_rank_problem:
$\text{(ORBITER)} -v 4 \$
-define F -finite_field -q 64 -end 
-define O -orthogonal_space 1 6 F -end 
-with O -do -orthogonal_space_activity 
-unrank_line_through_two_points 15447347 15225451 
-end

# use option -without_group to skip the group. This will work:

Op 6.64_line_rank:
$\text{(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
# this will create a basic report without the group:

Op_64_report:

-define F -finite_field -q 64 -end \\
-with 0 -do -orthogonal_space_activity \\
-define O -orthogonal_space 1 6 F -without_group -end \\
-with O -do -orthogonal_space_activity \\
-cheat_sheet_orthogonal \\
-end

pdflatex Op_64_report.tex

open Op_64_report.pdf

elliptic_quadric_subspace:

-define F -finite_field -q 5 -end \\
-define v -vector -format 4 \\
-dense "1,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,1,1" \\
-end

-define O -orthogonal_space 1 6 F -end \\
-with 0 -do -orthogonal_space_activity \\
-intersect_with_subspace v \\
-end

#We found that the intersection contains 26 points:


BLT_database_7_1:

-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

# writes BLT_7.1.txt

BLT_database_7_1_print:

-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 \

570
\begin{verbatim}
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 \$

# the output defines doily.csv
\end{verbatim}

~~~

# Section 4.8: Hermitian varieties

SECTION_HERMITIAN_VARIETIES:

\begin{verbatim}
H_2_4:
\$ (ORBITER) -v 2 \$
\$ -define F -finite_field -q 4 -end \$
\$ -with F -do -finite_field_activity \$

\end{verbatim}

571
H_2.9:
$(ORBITER) -v 2 \ 
-define F -finite_field -q 9 -end \ 
-with F -do -finite_field_activity \ 
-define F -finite_field -q 9 -end \ 
-define F -finite_field -q 9 -end \
-define F -finite_field -q 9 -end \
-define F -finite_field -q 9 -end \
-define F -finite_field -q 9 -end \
-define F -finite_field -q 9 -end \
-define F -finite_field -q 9 -end \
-define F -finite_field -q 9 -end \
-define F -finite_field -q 9 -end \
-define F -finite_field -q 9 -end \
-define F -finite_field -q 9 -end \
-define F -finite_field -q 9 -end \
-define F -finite_field -q 9 -end 

H_3.4:
$(ORBITER) -v 2 \ 
-define F -finite_field -q 4 -end \ 
-with F -do -finite_field_activity \ 
-define F -finite_field -q 4 -end 

# 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

# H_3.4 = the Hirschfeld surface

# Section 4.9: Projective Space Advanced Topics

SECTION_PROJECTIVE_SPACE_ADVANCED_TOPICS:

test_4.9:
make fix_structure_2A
make fix_structure_2B
make fix_structure_2C
make trans
make del_Pezzo_F13ab_report
make del_Pezzo_F13a_points.txt
make del_Pezzo_169
fix_structure_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 
-cheat_sheet_for_decomposition_by_element_PG 1 
"1,0,0,0, 0,1,0,0, 0,0,1,0, 0,0,0,1, 1" 
fix_structure_2A 
-end
pdflatex fix_structure_2A.tex
open fix_structure_2A.pdf

fix_structure_2B:
$(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 
-cheat_sheet_for_decomposition_by_element_PG 1 
"1,0,0,0, 1,1,0,0, 0,0,1,0, 0,0,0,1, 0" 
fix_structure_2B 
-end
pdflatex fix_structure_2B.tex
open fix_structure_2B.pdf

fix_structure_2C:
$(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 
-cheat_sheet_for_decomposition_by_element_PG 1 
"1,0,0,0, 1,1,0,0, 0,0,1,0, 0,0,1,1, 0" 
fix_structure_2C 
-end
pdflatex fix_structure_2C.tex
open fix_structure_2C.pdf

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 \\
de_pezzo_F13ab_report: \\
$(ORBITER) -v 3 \\
#define F -finite_field -q 13 -end \\
#define P -projective_space -n 3 -field F  -v 0 -end \\
#define f3 -formula \\
"del_Pezzo\_F13a" "del\_Pezzo\_F13a" "x,y,z" \\
"x*x*x*x+y*y*y*y+z*z*z*z+8*x*x*y*y+8*x*x*z*z+8*y*y*z*z" \\
define f4 -formula \\
"del_Pezzo\_F13b" "del\_Pezzo\_F13b" "x,y,z" \\
"x*x*x*x+y*y*y*y+z*z*z*z-x*x*y*y" \\
define del_Pezzo13 -collection "f3,f4" \\
-with P -do \\
-projective_space_activity \\
-analyze_del_Pezzo_surface del_Pezzo13 "" \\
end \\
open del_Pezzo_F13b_report.pdf \\
del_Pezzo_F13a_points.txt: \\
$(ORBITER) -v 3 \\
define f1 -formula "del\_Pezzo\_F9" \\
"x\_x\_x\_x+y\_y\_y\_y+z\_z\_z\_z" \\
define f2 -formula "del\_Pezzo\_F11" \\
"x\_y\_z" \\
define f3 -formula "del\_Pezzo\_F13a" \\
define f4 -formula "del\_Pezzo\_F13b" \\
define f5 -formula "del\_Pezzo\_F13c" \\
define del_Pezzo13 -collection "f3,f4,f5" \\
-with P -do \\
-projective_space_activity \\
analyze_del_Pezzo_surface del_Pezzo13 "" \\
end \\
open del_Pezzo_F13b_report.pdf
# Section 4.10: Geometric Objects

############################################################################

# takes long time to initialize the polarity in the projective space object

###projective_space::projective_space_init N_points=4855540

```plaintext
del_Pezzo_169:
```
SECTION_GEOMETRIC_OBJECTS:

test_4_10:

make elliptic_quadric_ovoid_q8
make ovoid_ST_q8
make Baer_PG_2_4
make Baer_PG_3_4
make BLT_database_5_0
make BLT_database_7_0
make BLT_database_67_4
make Doily_W_2
make Doily_disjoint_sets_graph_cliques_3
make Doily_disjoint_sets_graph_cliques_5
make PG_3_2_test
make Edge_curve_17
make Edge_curve_17_line_type
make Edge_curve_q23_line_type

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 \
define elliptic_quadric_ovoid \
with O -do -combinatorial_object_activity -save \
end

#ovoid_q8.txt
# 65 points

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 \
with O -do -combinatorial_object_activity -save \
end

#ovoid_ST_q8.txt
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 
-validate O -geometric_object P 
--Baer_substructure 
-end 
-with 0 -do -combinatorial_object_activity -save 
-end

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 
-validate O -geometric_object P 
--Baer_substructure 
-end 
-with 0 -do -combinatorial_object_activity -save 
-end

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 
-validate O -geometric_object P 
--BLT_database 0 
-end 
-with 0 -do -combinatorial_object_activity -save 
-end

# writes BLT_5.0.txt

BLT_database_7.0:

$\$(ORBITER) -v 2 \$

-define F -finite_field -q 7 -end 
-define P -projective_space -n 4 -field F -v 0 -end 
-validate O -geometric_object P 
--BLT_database 0 
-end 
-with 0 -do -combinatorial_object_activity -save 
-end

# writes BLT_7.0.txt
BLT_database_67_4:
 $(ORBITER) -v 2 \\n -define F -finite_field -q 67 -end \\
 -define P -projective_space -n 4 -field F -v 0 -end \\
 -define Obj -geometric_object P \\
 -BLT_database 4 \\
 -with Obj -do -combinatorial_object_activity -save \\
 -end \\
 -define 0 -orthogonal_space 0 5 F -without_group -end \\
 -define BLT_67_4 -set -file_orbiter_format BLT_67_4.txt -end \\
 -with 0 -do -orthogonal_space_activity \\
 -print_points BLT_67_4 -end \\
 pdflatex BLT_67_4_set_report.tex \\
 open BLT_67_4_set_report.pdf

Doily_disjoint_sets_graph_cliques_3:
 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 3 \\
 -output_file doily_cliques \\
 -end \\
 -print_symbols \\
 $(ORBITER) -v 2 \\
 -union doily.csv doily_cliques.txt doily_cliques_union.csv \\
 # 80 cliques

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 -print_symbols

# 6 cliques doily_cliques_5.csv

PG_3.2_test:
$ORBITER -v 2
$ORBITER -v 2
$ORBITER -v 2
$ORBITER -v 2
$ORBITER -v 2
$ORBITER -v 2
$ORBITER -v 2
$ORBITER -v 2
$ORBITER -v 2
$ORBITER -v 2
$ORBITER -v 2
$ORBITER -v 2
$ORBITER -v 2
$ORBITER -v 2
$ORBITER -v 2
$ORBITER -v 2
$ORBITER -v 2

Edge_curve_17:
$(ORBITER) -v 2
$(ORBITER) -v 2
$(ORBITER) -v 2
$(ORBITER) -v 2
$(ORBITER) -v 2
$(ORBITER) -v 2
$(ORBITER) -v 2
$(ORBITER) -v 2

#Edge_q17.txt
#combinatorial_object_create::init created a set of size 12
#( 4, 7, 16, 19, 20, 23, 32, 35, 89, 100, 244, 251 )
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 \
  -define P -projective_space -n 2 -field F -v 0 -end \
  -define E -geometric_object P \
  -set $(EDGE_CURVE_Q17 AS POINTS) \
  -with E -do \
  -combinatorial_object_activity \
  -save \
  -end \
  -with C -do \
  -combinatorial_object_activity \
  -line_type \
  -print_symbols

#( 4^6, 2^30, 1^132, 0^139 )

Edge_curve_q23_line_type:
  $(ORBITER) -v 2 \
  -define F -finite_field -q 23 -end \
  -define P -projective_space -n 2 -field F -v 0 -end \
  -define E -geometric_object P \
  -set $(EDGE_CURVE_Q23 AS POINTS) \
  -end \
  -with E -do \
  -combinatorial_object_activity \
  -save \
  -end \
  -with E -do \
  -combinatorial_object_activity \
  -line_type \
  -print_symbols
Section 5.1: Permutation groups

SECTION
PERMUTATION
GROUPS:

test_5.1:
▷ make Cyclic_6
▷ make Cyclic_6_group_table
▷ make Symmetric_3
▷ make Symmetric_3_group_table
▷ make Symmetric_3_elements
▷ make Symmetric_3_long
▷ make Symmetric_4
▷ make Symmetric_4_group_table
▷ make Symmetric_4_long

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

Cyclic_6_group_table:
▷ $(ORBITER) -v 3 \$
▷ ▷ -define G -permutation_group -cyclic_group 6 -end \$
▷ ▷ -with G -do \$

# Chapter 5 - Group Theory
3322  ▶  ▶  -group_theoretic_activity \\
3323  ▶  ▶  ▶  -export_group_table \\
3324  ▶  ▶  -end \\
3325  ▶  ▶  $(ORBITER) -v 2 \\
3326  ▶  ▶  ▶  -define all_one_r -vector -repeat 1 6 -end \\
3327  ▶  ▶  ▶  -define all_one_c -vector -repeat 1 6 -end \\
3328  ▶  ▶  ▶  -draw_matrix \\
3329  ▶  ▶  ▶  ▶  -input_csv_file C_6_group_table.csv \\
3330  ▶  ▶  ▶  ▶  -box_width 50 -bit_depth 24 \\
3331  ▶  ▶  ▶  ▶  -partition 3 all_one_r all_one_c \\
3332  ▶  ▶  ▶  -end \\
3333  ▶  ▶  ▶  ▶  open C_6_group_table_draw.bmp \\
3334 \\
3335 \\
3336  Symmetric_3: \\
3337  ▶  ▶  $(ORBITER) -v 3 \\
3338  ▶  ▶  ▶  -define G -permutation_group -symmetric_group 3 -end \\
3339  ▶  ▶  ▶  -with G -do \\
3340  ▶  ▶  ▶  -group_theoretic_activity \\
3341  ▶  ▶  ▶  ▶  -report \\
3342  ▶  ▶  ▶  -end \\
3343  ▶  ▶  ▶  ▶  pdflatex Sym_3_report.tex \\
3344  ▶  ▶  ▶  ▶  open Sym_3_report.pdf \\
3345 \\
3346 \\
3347  Symmetric_3_group_table: \\
3348  ▶  ▶  $(ORBITER) -v 3 \\
3349  ▶  ▶  ▶  -define G -permutation_group -symmetric_group 3 -end \\
3350  ▶  ▶  ▶  -with G -do \\
3351  ▶  ▶  ▶  -group_theoretic_activity \\
3352  ▶  ▶  ▶  ▶  -export_group_table \\
3353  ▶  ▶  ▶  -end \\
3354  ▶  ▶  ▶  $(ORBITER) -v 2 \\
3355  ▶  ▶  ▶  ▶  -define all_one_r -vector -repeat 1 6 -end \\
3356  ▶  ▶  ▶  ▶  -define all_one_c -vector -repeat 1 6 -end \\
3357  ▶  ▶  ▶  ▶  -draw_matrix \\
3358  ▶  ▶  ▶  ▶  ▶  -input_csv_file Sym_3_group_table.csv \\
3359  ▶  ▶  ▶  ▶  ▶  -box_width 50 -bit_depth 24 \\
3360  ▶  ▶  ▶  ▶  ▶  -partition 3 all_one_r all_one_c \\
3361  ▶  ▶  ▶  ▶  -end \\
3362  ▶  ▶  ▶  ▶  open Sym_3_group_table_draw.bmp \\
3363 \\
3364  Symmetric_3_elements: \\
3365  ▶  ▶  $(ORBITER) -v 3 \\
3366  ▶  ▶  ▶  -define G -permutation_group -symmetric_group 3 -end \\
3367  ▶  ▶  ▶  -with G -do \\
3368  ▶  ▶  ▶  -group_theoretic_activity \\

582
3369 ▶ ▶ ▶ -save_elements_csv "Symmetric3_elts.csv" \\
3370 ▶ ▶ -end \\
3371 ▶ $(ORBITER) -v 2 \\
3372 ▶ ▶ -define Sym3_elts -vector -load_csv.data_column \\
3373 ▶ ▶ ▶ Symmetric3_elts.csv 1 -end \\
3374 ▶ ▶ -save_matrix_csv Sym3_elts \\
3375 ▶ $(ORBITER) -v 2 \\
3376 ▶ ▶ -define all_one_r -vector -repeat 1 6 -end \\
3377 ▶ ▶ -define all_one_c -vector -repeat 1 3 -end \\
3378 ▶ ▶ -draw_matrix \\
3379 ▶ ▶ ▶ -input_csv_file Sym3_elts_matrix.csv \\
3380 ▶ ▶ ▶ -box_width 50 -bit_depth 8 \\
3381 ▶ ▶ ▶ -partition 3 \\
3382 ▶ ▶ ▶ ▶ all_one_r all_one_c \\
3383 ▶ ▶ ▶ -end \\
3384 ▶ ▶ open Sym3_elts_matrix_draw.bmp \\
3385 \\
3386 Symmetric_3_long: \\
3387 ▶ $(ORBITER) -v 3 \\
3388 ▶ ▶ -define G -permutation_group -symmetric_group 3 -end \\
3389 ▶ ▶ -with G -do \\
3390 ▶ ▶ -group_theoretic_activity \\
3391 ▶ ▶ ▶ -export_orbiter \\
3392 ▶ ▶ ▶ -end \\
3393 ▶ ▶ -with G -do \\
3394 ▶ ▶ -group_theoretic_activity \\
3395 ▶ ▶ ▶ -print_elements_tex \\
3396 ▶ ▶ ▶ -end \\
3397 ▶ ▶ -with G -do \\
3398 ▶ ▶ -group_theoretic_activity \\
3399 ▶ ▶ ▶ -report \\
3400 ▶ ▶ ▶ -end \\
3401 ▶ ▶ -with G -do \\
3402 ▶ ▶ -group_theoretic_activity \\
3403 ▶ ▶ ▶ -save_elements_csv "Symmetric3_elts.csv" \\
3404 ▶ ▶ ▶ -end \\
3405 ▶ $(ORBITER) -v 3 \\
3406 ▶ ▶ -draw_options \\
3407 ▶ ▶ ▶ -nodes \\
3408 ▶ ▶ ▶ ▶ -embedded -radius 250 \\
3409 ▶ ▶ ▶ ▶ -xin 10000 -yin 10000 \\
3410 ▶ ▶ ▶ ▶ -xout 1000000 -yout 600000 \\
3411 ▶ ▶ ▶ ▶ -scale 0.3 -line_width 1.0 \\
3412 ▶ ▶ ▶ ▶ -end \\
3413 ▶ ▶ ▶ -tree_draw -file Sym3_elements_tree.txt -end \\
3414 ▶ $(ORBITER) -v 2 \\
3415 ▶ ▶ -define M -vector -load_csv.data_column \\

583
Symmetric3: elts.csv 1-end \
 save_matrix.csv M 
 $(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 M_matrix.csv 
 box_width 50 -bit_depth 8 
 partition 3 \n all_one_r all_one_c 
 end 
 latex Sym elements tree draw.tex 
 open Sym elements tree draw.pdf 
 latex Perm report.tex 
 open Perm report.pdf 

Symmetric 4: 
 $(ORBITER) -v 3 \
 define G -permutation_group -symmetric_group 4 -end \
 with G -do \
 group_theoretic_activity 
 report 
 end 
 latex Sym elements tree draw.tex 
 open Sym elements tree draw.pdf 

Symmetric 4_group_table: 
 $(ORBITER) -v 3 \
 define G -permutation_group -symmetric_group 4 -end \
 with G -do \
 group_theoretic_activity \
 report 
 end 
 $(ORBITER) -v 2 \
 define all_one_r -vector -repeat 1 24 -end \
 define all_one_c -vector -repeat 1 24 -end \
 draw_matrix \
 input_csv_file Sym group table.csv \
 box_width 50 -bit_depth 24 \
 partition 3 all_one_r all_one_c 
 end 
 open Sym group table draw.bmp
Symmetric

$\text{Symmetric\_4\_long:}$

-define G -permutation_group -symmetric_group 4 -end \ 
-group_theoretic_activity \ 
-export_orbiter \ 
-end \ 
-with G -do \ 
-group_theoretic_activity \ 
-export_group_table \ 
-end \ 
-with G -do \ 
-group_theoretic_activity \ 
-export_inversion_graphs "Sym\_4\_inversion\_graphs.csv" \ 
-end \ 
-with G -do \ 
-group_theoretic_activity \ 
-save_elements_csv "Sym\_4\_elts.csv" \ 
-end \ 
-with G -do \ 
-group_theoretic_activity \ 
-
-report \ 
-end \ 
-with G -do \ 
-group_theoretic_activity \ 
-save_elements_csv "Sym\_4\_elts.csv" \ 
-end \ 
-with G -do \ 
-group_theoretic_activity \ 
-
-report \ 
-end \ 
-with G -do \ 
-group_theoretic_activity \ 
-
-report \ 
-end \ 
-with G -do \ 
-group_theoretic_activity \ 
-
-report \ 
-end \ 
-with G -do \ 
-group_theoretic_activity \ 
-
-report \ 
-end \ 
-with G -do \ 
-group_theoretic_activity \ 
-
-report \ 
-end \ 

$\text{Symmetric\_4\_long:}$

$\text{Symmetric\_4\_long:}$

$\text{Symmetric\_4\_long:}$

$\text{Symmetric\_4\_long:}$

$\text{Symmetric\_4\_long:}$
3510 ▶ ▶ ▶  -input_csv_file M_matrix.csv \\
3511 ▶ ▶ ▶  -box_width 50 -bit_depth 8 \\
3512 ▶ ▶ ▶  -partition 3 \\
3513 ▶ ▶ ▶ ▶ all_one_r all_one.c \\
3514 ▶ ▶  -end \\
3515 ▶  pdflatex Sym_4_elements_tree_draw.tex \\
3516 ▶  #open Sym_4_elements_tree_draw.pdf \\
3517 3518 3519 3520 3521 3522

# Section 5.2: Linear Groups
3523
3524 SECTION LINEAR_GROUPS:
3525
3526 test_5.2:
3527  make PGL_3_2
3528  make PGL_4_2
3529  make PGL_4_2_export
3530  make PGL_4_2_generated
3531  make L_5_3
3532  make L_4_5
3533  make PGL_4_5
3534  make PGGL_3_4
3535  make PGGL_3_8
3536  make PGGL_3_8_report
3537  make AGL_1_27
3538  make SP_4_2
3539  make PSP_4_4
3540  make PG0_5_2
3541  make PGG0_5_4
3542  make PGOp_6_2
3543  make PGOp_6_2
3544  make PSP_6_2
3545  make PG0m_6_2
3546  make PGOm_6_2
3547  make PGO_7_2
3548
3549
3550  PGL_3_2:
3551  $(ORBITER) -v 2 \\
3552  ▶  -define F -finite_field -q 2 -end \\
3553  ▶  -define G -linear_group -PGL 3 F -end \\
3554  ▶  ▶  -with G -do \\

586
PGL_3_2:
\$(ORBITER) -v 2
\define F -finite_field -q 2
\define G -linear_group -PGL 4 F
\with G
\group_theoretic_activity
\report
\end
\pdflatex PGL_3_2_report.tex
\open PGL_3_2_report.pdf

PGL_4_2:
\$(ORBITER) -v 2
\define F -finite_field -q 2
\define G -linear_group -PGL 4 F
\with G
\group_theoretic_activity
\report
\end
\pdflatex PGL_4_2_report.tex
\open PGL_4_2_report.pdf

# created by PGL_4_2_export

PGL_4_2_generated:
\$(ORBITER) -v 2
\define gens -vector -file PGL_4_2_gens.csv
\define G -permutation_group
\bigs -bsgs PGL_4_2 "{(\rm PGL}(4,2)" 15 20160 "0,1,2,3" 6 gens

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

#PSL(5,3): Order 237783237120 = 121 \* 120 \* 117 \* 108 \* 81 \* 16 

L_4_5: 
-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

#PSL(4,5): Order 7254000000 

PGL_4_5: 
-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

PGGL_3_4: 
-define F -finite_field -q 3 -end \
-define G -linear_group -PGGL 3 4 -end \

$\mathrm{ORBITER}$ -v 2
-define G -linear_group -PGGL 3 8 -end
-group_theoretic_activity
-report
-sylow
-classes
-end
-pdflatex PGGL_3.4_report.tex
-open PGGL_3.4_report.pdf

PGGL_3.8:
$\mathrm{ORBITER}$ -v 3
-define G -linear_group -PGGL 3 8 -end
-group_theoretic_activity
-report
-end
-pdflatex PGGL_3.8_report.tex
-open PGGL_3.8_report.pdf

AGL_{1,27}:
$\mathrm{ORBITER}$ -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

# -group_table

SP_{4,2}:
$\mathrm{ORBITER}$ -v 2
-define F -finite_field -q 2 -end
-define G -linear_group -GL 4 F
-symplectic_group
-end
3698  ▷ ▷ -with G -do \\
3699  ▷ ▷ -group_theoretic_activity \\
3700  ▷ ▷ ▷ -report \\
3701  ▷ ▷ ▷ -end \\
3702  ▷ pdflatex GL_4.2_Sp_4.2_report.tex \\
3703  ▷ open GL_4.2_Sp_4.2_report.pdf \\
3704  \\
3705  ▷ # order 720 \\
3706  \\
3707  \\
3708  ▷ PSP_4.4: \\
3709  ▷ $(ORBITER) -v 5 \\
3710  ▷ ▷ -define F -finite_field -q 4 -end \\
3711  ▷ ▷ -define G -linear_group -PGL 4 F \\
3712  ▷ ▷ ▷ -symplectic_group \\
3713  ▷ ▷ ▷ -end \\
3714  ▷ ▷ -with G -do \\
3715  ▷ ▷ -group_theoretic_activity \\
3716  ▷ ▷ ▷ -report \\
3717  ▷ ▷ ▷ -end \\
3718  ▷ pdflatex PGL_4.4_Sp_4.4_report.tex \\
3719  ▷ open PGL_4.4_Sp_4.4_report.pdf \\
3720  \\
3721  ▷ #order 979200 \\
3722  \\
3723  \\
3724  \\
3725  ▷ PGO_5.2: \\
3726  ▷ $(ORBITER) -v 2 \\
3727  ▷ ▷ -define F -finite_field -q 2 -end \\
3728  ▷ ▷ -define G -linear_group -PGO 5 F -end \\
3729  ▷ ▷ -with G -do \\
3730  ▷ ▷ -group_theoretic_activity \\
3731  ▷ ▷ ▷ -report \\
3732  ▷ ▷ ▷ -end \\
3733  ▷ pdflatex PGO_5.2_report.tex \\
3734  ▷ open PGO_5.2_report.pdf \\
3735  \\
3736  ▷ PGGO_5.4: \\
3737  ▷ $(ORBITER) -v 2 \\
3738  ▷ ▷ -define F4 -finite_field -q 4 -end \\
3739  ▷ ▷ -define G -linear_group -PGGO 5 F4 -end \\
3740  ▷ ▷ -with G -do \\
3741  ▷ ▷ -group_theoretic_activity \\
3742  ▷ ▷ ▷ -report \\
3743  ▷ ▷ ▷ -end \\
3744  ▷ pdflatex PGGO_5.4_report.tex
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

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

# the following two groups are isomorphic:

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

# group order 1451520, acting on 63 points
# group order 1451520, acting on 63 points

# Section 5.3: Subgroups

SECTION_SUBGROUPS:

code:
```
code
```
3839 ▶ $(ORBITER) -v 2 \
3840 ▶ ▶ -define gens -vector -dense $(GEN_C13) -end \
3841 ▶ ▶ -define G -permutation_group \
3842 ▶ ▶ ▶ -bsgs C13 C.{13} 13 13 0 1 \
3843 ▶ ▶ ▶ ▶ gens \
3844 ▶ ▶ ▶ -end \
3845 ▶ ▶ ▶ with G -do \
3846 ▶ ▶ ▶ -group_theoretic_activity \
3847 ▶ ▶ ▶ ▶ -export_orbiter \
3848 ▶ ▶ ▶ -end \
3849 ▶ ▶ ▶ with G -do \
3850 ▶ ▶ ▶ -group_theoretic_activity \
3851 ▶ ▶ ▶ ▶ -export_group_table \
3852 ▶ ▶ ▶ -end \
3853 ▶ ▶ ▶ with G -do \
3854 ▶ ▶ ▶ -group_theoretic_activity \
3855 ▶ ▶ ▶ ▶ -report \
3856 ▶ ▶ ▶ -end \
3857 ▶ ▶ ▶ with G -do \
3858 ▶ ▶ ▶ -group_theoretic_activity \
3859 ▶ ▶ ▶ ▶ -save_elements_csv "C13_elts.csv" \
3860 ▶ ▶ ▶ -end 
3861 ▶ pdflatex C13_report.tex
3862 ▶ open C13_report.pdf
3863
3864
3865 C13_generated:
3866 ▶ $(ORBITER) -v 2 \
3867 ▶ ▶ -define gens -vector -file C13 gens.csv -end \
3868 ▶ ▶ -define G -permutation_group \
3869 ▶ ▶ ▶ -bsgs C13 "C.{13}" 13 13 "0" 1 gens -end \
3870
3871
3872 C13_as_subgroup:
3873 ▶ $(ORBITER) -v 2 \
3874 ▶ ▶ -define G -permutation_group -symmetric_group 13 \
3875 ▶ ▶ ▶ -subgroup_by_generators C13 13 1 $(GEN_C13) -end \
3876 ▶ ▶ ▶ with G -do \
3877 ▶ ▶ ▶ -group_theoretic_activity \
3878 ▶ ▶ ▶ ▶ -export_orbiter \
3879 ▶ ▶ ▶ -end \
3880 ▶ ▶ ▶ with G -do \
3881 ▶ ▶ ▶ -group_theoretic_activity \
3882 ▶ ▶ ▶ ▶ -report \
3883 ▶ ▶ ▶ -end \
3884 ▶ ▶ ▶ with G -do \
3885 ▶ ▶ ▶ -group_theoretic_activity \

593
J1:

\%(ORBITER) -v 2 \n\%(ORBITER) -define G -linear_group -PGL 7 11 -Janko1 -end \n\%(ORBITER) -with G -do \n\%(ORBITER) -group_theoretic_activity \n\%(ORBITER) -do \n\%(ORBITER) -report \n\%(ORBITER) -end

pdflatex PGL_7_11_Subgroup_Janko1_report.tex
open PGL_7_11_Subgroup_Janko1_report.pdf

J3_11_singer:

\%(ORBITER) -v 2 \n\%(ORBITER) -define G -linear_group -PGL 3 11 -singer 19 -end \n\%(ORBITER) -with G -do \n\%(ORBITER) -group_theoretic_activity \n\%(ORBITER) -do \n\%(ORBITER) -report \n\%(ORBITER) -end

pdflatex PGL_3_11_Singer_3_11_19_report.tex
open PGL_3_11_Singer_3_11_19_report.pdf

PG_3_11_singer_and_frobenius:

\%(ORBITER) -v 2 \n\%(ORBITER) -define G -linear_group -PGL 3 11 -singer_and_frobenius 19 -end \n\%(ORBITER) -with G -do \n\%(ORBITER) -group_theoretic_activity \n\%(ORBITER) -do \n\%(ORBITER) -report \n\%(ORBITER) -end

pdflatex PGL_3_11_Singer_and_Frob3_11_19_report.tex
open PGL_3_11_Singer_and_Frob3_11_19_report.pdf

PG_2_4_order_21:

\%(ORBITER) -v 2 \n\%(ORBITER) -define G -linear_group -PGL 3 4 -end \n\%(ORBITER) -with G -do \n\%(ORBITER) -group_theoretic_activity \n\%(ORBITER) -do \n\%(ORBITER) -search_element_of_order 21 \n\%(ORBITER) -end
quaternion:
$\triangleright$ $(\text{ORBITER}) -v 2 \$
$\triangleright$ $\triangleright$ $\triangleright$ $\text{-define G -linear_group -SL 2 3}$
$\triangleright$ $\triangleright$ $\triangleright$ $\text{-subgroup_by_generators "quaternion" "8" 3}$
$\triangleright$ $\triangleright$ $\triangleright$ $\triangleright$ $\text{"1,1,1,2, 2,1,1,1, 0,2,1,0"}$
$\triangleright$ $\triangleright$ $\text{-end}$
$\triangleright$ $\triangleright$ $\text{-with G -do}$
$\triangleright$ $\triangleright$ $\text{-group_theoretic_activity}$
$\triangleright$ $\triangleright$ $\text{-print_elements_tex}$
$\triangleright$ $\triangleright$ $\text{-report_group_table}$
$\triangleright$ $\triangleright$ $\text{-report}$
$\triangleright$ $\triangleright$ $\text{-end}$
$\triangleright$ $\text{pdflatex GL_2_3_Subgroup_quaternion_8_elements.tex}$
$\triangleright$ $\text{open GL_2_3_Subgroup_quaternion_8_elements.pdf}$
$\triangleright$ $\text{pdflatex GL_2_3_Subgroup_quaternion_8_report.tex}$
$\triangleright$ $\text{open GL_2_3_Subgroup_quaternion_8_report.pdf}$
$cube_group:$
$\triangleright$ $(\text{ORBITER}) -v 2 \$
$\triangleright$ $\triangleright$ $\triangleright$ $\text{-define gens -vector -dense}$
$\triangleright$ $\triangleright$ $\triangleright$ $\text{"0,1,0,2,0,0,0,0,1,1,0,1,1,0,1,0,2,0,0,0,0,0,0,0,0,1\"}$
$\triangleright$ $\triangleright$ $\text{-end}$
$\triangleright$ $\triangleright$ $\text{-define G -linear_group -GL 3 3}$
$\triangleright$ $\triangleright$ $\text{-subgroup_by_generators "cube" "48" 3}$
$\triangleright$ $\triangleright$ $\text{-print_elements_tex}$
$\triangleright$ $\triangleright$ $\text{-report}$
$\triangleright$ $\triangleright$ $\text{-end}$
$\triangleright$ $\text{pdflatex GL_3_3_Subgroup_cube_48_report.tex}$
$\triangleright$ $\text{open GL_3_3_Subgroup_cube_48_report.pdf}$
$\triangleright$ $\text{pdflatex GL_3_3_Subgroup_cube_48_elements.tex}$
$\triangleright$ $\text{open GL_3_3_Subgroup_cube_48_elements.pdf}$
tetra_group:
$\triangleright$ $(\text{ORBITER}) -v 3 \$
$\triangleright$ $\triangleright$ $\text{-define G -linear_group -GL 3 3}$
$\triangleright$ $\triangleright$ $\text{-subgroup_by_generators "tetra" "12" 2}$
Hesse group:

```plaintext
#Hesse group:
1,0,0,0,0,1,0,0,0,1,0,3,2,3,2,0, 
1,0,0,0,0,1,0,0,0,1,0,3,2,3,0,0, 
1,0,0,0,0,1,1,0,0,1,1,0,0,0,0,1,1, 
1,0,0,0,0,1,1,0,0,1,1,0,0,0,0,1,0, 
1,0,0,0,0,1,1,0,0,1,1,0,0,0,0,1,0, 
1,0,0,0,0,1,1,0,0,1,1,0,0,0,0,1,0, 
1,0,0,0,0,1,1,0,0,1,1,0,0,0,0,1,0, 
1,0,0,0,0,0,0,0,3,0,0,0,0,0,0,1,0, 
```

Weyl_E8:

```plaintext
#Weyl_E8:
```

596
subgroup_by_generators

"Weyl_E8" "696729600" 2 gens

end

-with G -do

-end

-Weyl E8" 696729600 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


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

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

coset_reps_read:
4073  \(\text{-symplectic\_group} \) \n4074  \(\text{-end} \) \n4075  \(\text{-define CR -vector\_ge -action G2} \) \n4076  \(\text{-read\_csv} \) \n4077  \(\text{PGOp\_6\_2\_coset\_reps.csv Element} \) \n4078  \(\text{-end} \) \n4079  \(\text{SP\_6\_2\_point\_stab\_subgroup:} \) \n4080  \(\text{pdflatex PGL\_6\_2\_report.tex} \) \n4081  \(\text{open PGL\_6\_2\_report.pdf} \) \n4090  \(\text{PGOp\_6\_2\_report:} \) \n4091  \(\text{pdflatex PGOp\_6\_2\_report.tex} \) \n4092  \(\text{open PGOp\_6\_2\_report.pdf} \) \n4101  \(\text{PGOp\_6\_2\_point\_stab\_subgroup:} \)
# group of order 1152

PGOp_6_2_GENS="\n 1,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,1,0,\n 1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,1,0,1,0,\n 1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,1,0,1,0,\n 1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,1,0,1,0,\n 1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,1,0,1,0,\n 0,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,1,0,0,0,0,0,0,1,0,0,1,0,\n 0,0,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,1,0,1,0,1,0,0,0,0,0,0,1,0,0,1,1,0,\n 0,0,0,0,0,0,1,0,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,1,1,1,0,"\n
PGOp_6_2_linear:
\$\text{ORBITER} -v 2 \$
\$\text{define F} -\text{finite}\_\text{field} -q 2 \text{-end} \$
$\text{define G} -\text{linear}\_\text{group} -\text{PGL 6 F} \$
\$\text{subgroup}\_\text{by}\_\text{generators} \text{PGOp}_6_2 \$
\$\text{group}\_\text{theoretic}\_\text{activity}\$
\$\text{report} \$
\$\text{end} \$
\$\text{with G} \text{-do} \$
\$\text{end} \$
pdflatex PGOp_6_2_report.tex
open PGOp_6_2_report.pdf

PGOp_6_2_linear\_stab\_6:
\$\text{ORBITER} -v 2 \$
$\text{define F} -\text{finite}\_\text{field} -q 2 \text{-end} \$
$\text{define G} -\text{linear}\_\text{group} -\text{PGL 6 F} \$
\$\text{subgroup}\_\text{by}\_\text{generators} \text{PGOp}_6_2 \$
\$\text{group}\_\text{theoretic}\_\text{activity}\$
\$\text{report} \$
\$\text{end} \$
pdflatex PGL_6_2_Subgroup_PGOp_6_2_40320_report.tex
open PGL_6_2_Subgroup_PGOp_6_2_40320_report.pdf

PGOp_6_2_linear\_stab\_6:
\$\text{ORBITER} -v 2 \$
$\text{define F} -\text{finite}\_\text{field} -q 2 \text{-end} \$
$\text{define G} -\text{linear}\_\text{group} -\text{PGL 6 F} \$
\$\text{subgroup}\_\text{by}\_\text{generators} \text{PGOp}_6_2 \$
\$\text{group}\_\text{theoretic}\_\text{activity}\$
\$\text{report} \$
\$\text{end} \$
$\text{define G6} -\text{modified}\_\text{group} -\text{from G} \$
test_5_4:
make U_3_3
make PGL_2_3
make Co3
make Ree_27

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

# ToDo -embedded for group table
PGL_2_3:

$(ORBITER) -v 3 

-define G -linear_group -PGL 2 3 -end 

-with G -do 

-group_theoretic_activity 

-report 

-report_group_table 

-end 

#pdflatex PGL_2_3_group_table_order_24.tex 

#open PGL_2_3_group_table_order_24.pdf 

#Co3 from Conway et al., 1985 (ATLAS) 

#order = 495766656000 

#Co3 from the paper by Suleiman and Wilson 1997 

#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 -concatenate 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 

# needs a lot of memory to run! 

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 -concatenate g2 -end 

601
\begin{verbatim}
4261  \\
4262  \\
4263  \\
4264  \\
4265  \\
4266  \\
4267  \\
4268  \\
4269  \\
4270  \\
4271  \\
4272  \\
4273  \\
4274  \\
4275  \\
4276  \\
4277  \\
4278  \\
4279  \\
4280  \\
4281  \\
4282  \\
4283  \\
4284  \\
4285  \\
4286  \\
4287  \\
4288  \\
4289  \\
4290  \\
4291  \\
4292  \\
4293  \\
4294  \\
4295  \\
4296  \\
4297  \\
4298  \\
4299  \\
4300  \\
4301  \\
4302  \\
4303  \\
4304  \\
4305  \\
4306  \\
4307  \\
\end{verbatim}

Section 5.5: Induced Actions

\begin{verbatim}
4276 SECTION_INDUCED_ACTIONS:  
4277  
4278 test_5.5:  
4279  
4280  
4281  
4282  
4283  
4284  
4285  
4286  
4287  
4288  
4289  
4290  
4291  
4292  
4293  
4294  
4295  
4296  
4297  
4298  
4299  
4300  
4301  
4302  
4303  
4304  
4305  
4306  
\end{verbatim}

Symmetric_4_on_pairs:
\begin{verbatim}
4295  \\
4296  \\
4297  \\
4298  \\
4299  \\
4300  \\
4301  \\
4302  \\
4303  \\
4304  \\
4305  \\
4306  \\
\end{verbatim}

# Section 5.5: Induced Actions

# needs a lot of memory to run!

Section 5.5: Induced Actions

test_5.5:

Symmetric_4_on_pairs:

\texttt{pdflatex Sym\_4\_on\_2\_subsets\_report.tex}

\texttt{open Sym\_4\_on\_2\_subsets\_report.pdf}
surface_q13_Eckardt:
$\text{ORBITER} \ -v \ 3 \$
$\text{-define F -finite_field -q 13 -end}$
$\text{-define P -projective_space -n 3 -field F -v 0 -end}$
$\text{-define S -cubic_surface -space P -arc_lifting "0,1,2,3,43,113" -end}$
$\text{-with S -do}$
$\text{-cubic_surface_activity}$
$\text{-report}$
$\text{-end}$
$\text{-with S -do}$
$\text{-cubic_surface_activity}$
$\text{-export something tritangent_planes}$
$\text{-end}$
$\text{pdflatex surface_arc_lifting_trihedral_q13_arc0_1_2_3_43_113_report.tex}$
$\text{open surface_arc_lifting_trihedral_q13_arc0_1_2_3_43_113_report.pdf}$

SURFACE_Q13_STABGENS="1,0,0,0,9,12,0,10,0,12,0,9,0,0,12, 1,0,0,0,12,12,6,6,0,0,7,1,0,2,0, 0,1,1,7,3,9,9,11,2,10,10,3,9,9,1,11"

surface_q13_Eckardt_on_tritangent_planes:
$\text{ORBITER} \ -v \ 2 \$
$\text{-orbiter_path $(ORBITER\_PATH)$}$
$\text{-draw_options -embedded -end}$
$\text{-define F -finite_field -q 13 -end}$
$\text{-define gens -vector -field F -dense $(SURFACE\_Q13\_STAB\_GENS)$ -end}$
$\text{-define TriP -set -file}$
$\text{arc_lifting_trihedral_q13_arc0_1_2_3_43_113_tritangent_planes.csv}$
$\text{-end}$
$\text{-define G -linear_group -PGL 4 F}$
$\text{-subgroup_by_generators "stab"}$
$\text{24 3 gens}$
$\text{-end}$
$\text{-define G_on_planes -modified_group -from G}$
$\text{-on_k_subspaces 3}$
$\text{-end}$
$\text{-define Gr -modified_group -from G_on_planes}$
$\text{-restricted_action TriP}$
$\text{-end}$
$\text{-with Gr -do}$
$\text{-group_theoretic_activity}$
$\text{-report}$
$\text{-end}$
\begin{verbatim}
4355  >>  -define Orb -orbits -group Gr \ 
4356  >>  >>  > -on_points \ 
4357  >>  >>  >>  -end \ 
4358  >>  >>  > -with Orb -do -orbits_activity \ 
4359  >>  >>  >>  -report \ 
4360  >>  >>  >>  -end \ 
4361  >>  >>  > -with Orb -do -orbits_activity \ 
4362  >>  >>  >>  >>  > -draw_tree 0 \ 
4363  >>  >>  >>  -end \ 
4364  >>  >>  > -with Orb -do -orbits_activity \ 
4365  >>  >>  >>  >>  > -draw_tree 1 \ 
4366  >>  >>  >>  -end \ 
4367  >>  >>  > -with Orb -do -orbits_activity \ 
4368  >>  >>  >>  >>  > -stabilizer 36 \ 
4369  >>  >>  >>  -end \ 
4370  >>  pdflatex PGL_4.13_Gr_4.3_res45_orbits_report.tex \ 
4371  >>  open PGL_4.13_Gr_4.3_res45_orbits_report.pdf \ 
4372 \ 
4373 \ 
4374  >>  #pdflatex PGL_4.13_Gr_4.3_res45_orbit_0_tree.tex \ 
4375  >>  #open PGL_4.13_Gr_4.3_res45_orbit_0_tree.pdf \ 
4376  >>  #pdflatex PGL_4.13_Gr_4.3_res45_orbit_1_tree.tex \ 
4377  >>  #open PGL_4.13_Gr_4.3_res45_orbit_1_tree.pdf \ 
4378  >>  #pdflatex PGL_4.13_Gr_4.3_res45_stab_orb_0_report.tex \ 
4379  >>  #open PGL_4.13_Gr_4.3_res45_stab_orb_0_report.pdf \ 
4380 \ 
4381 \ 
4382 \ 
4383 \ 
4384  # related to planes_in_pencil: \ 
4385  # we are computing the action on the planes through the line 0. \ 
4386 \ 
4387 \ 
4388 \ 
4389  on_planes: \ 
4390  >>  $(ORBITER) -v 2 \ 
4391  >>  >>  -define F -finite_field -q 8 -end \ 
4392  >>  >>  > -define P -projective_space -n 3 -field F -v 0 -end \ 
4393  >>  >>  >>  -define G -linear_group -PGL 4 F -end \ 
4394  >>  >>  > -define G_on_planes -modified_group -from G \ 
4395  >>  >>  >>  > -on_k_subspaces 3 \ 
4396  >>  >>  >>  -end \ 
4397  >>  >>  > -with G_on_planes -do \ 
4398  >>  >>  >>  > -group_theoretic_activity \ 
4399  >>  >>  >>  >>  > -apply "0,8,1,6,4,3,7,2,5" \ 
4400  >>  >>  >>  > "1,0,0,0, 0,1,0,0, 0,0,0,2, 0,0,1,1" \ 
4401  >>  >>  >>  -end \ 
\end{verbatim}
T3 on tensors:

```
$(ORBITER) -v 2 \
define G \nlinear_group -GL_d_q_wr_Sym_n 2 2 3 \non_tensors -end \nwith G -do \ngroup_theoretic_activity \nreport \nend
```

```
pdflatex GL_2_2_wreath_Sym3_report.tex
open GL_2_2_wreath_Sym3_report.pdf
```

T3r1:

```
$(ORBITER) -v 4 \
define G \nlinear_group -GL_d_q_wr_Sym_n 2 2 3 \non_rank_one_tensors -end \nwith G -do \ngroup_theoretic_activity \nreport \nend
```

```
pdflatex GL_2_2_wreath_Sym3_report.tex
open GL_2_2_wreath_Sym3_report.pdf
```

T4 on tensors:

```
$(ORBITER) -v 4 \
define G \nlinear_group -GL_d_q_wr_Sym_n 2 2 4 \non_tensors -end \nwith G -do \ngroup_theoretic_activity \nreport \nend
```

```
pdflatex GL_2_2_wreath_Sym4_report.tex
open GL_2_2_wreath_Sym4_report.pdf
```

605
T4r1:

\$(\text{ORBITER}) -v 4 \$

-define G \
-\text{linear\_group} -GL\_d\_q\_wr\_Sym\_n \ 2 \ 2 \ 4 \$
-on\_rank\_one\_tensors -end \
-with G -do \
-group\_theoretic\_activity \
-report \
-end \

\text{pdflatex} \ GL\_2\_2\_wreath\_Sym4\_report.tex \
\text{open} \ GL\_2\_2\_wreath\_Sym4\_report.pdf \\

# ToDo \\

PGGL\_2\_8\_on\_conic: \\

\$(\text{ORBITER}) -v 4 \$

-define G -linear\_group \
-PGL 2 8 -PGL2OnConic \
-end \
-with G -do \
-group\_theoretic\_activity \
-report \
-end \

\text{pdflatex} \ PGGL\_2\_8\_OnConic\_2\_8\_report.tex \
\text{open} \ PGGL\_2\_8\_OnConic\_2\_8\_report.pdf \\

PGL\_4\_2\_wd: \\

\$(\text{ORBITER}) -v 3 \$

-define G -linear\_group -PGL 4 2 -wedge\_detached -end \
-with G -do \
-group\_theoretic\_activity \
-report \
-end \

\text{pdflatex} \ PGL\_6\_2\_Wedge\_4\_2\_detached\_report.tex \
\text{open} \ PGL\_6\_2\_Wedge\_4\_2\_detached\_report.pdf \\

PGL\_4\_2\_wd\_reverse: \\

\$(\text{ORBITER}) -v 3 \$
define G -linear_group -PGL 4 2 -wedge_detached -end \
-group_theoretic_activity \
-reverse_isomorphism_exterior_square \ 
-end

# Section 5.6: Group Theoretic Activities

SECTION_GROUP_THEORETIC.ACTIVITIES:

test_5.6:

make PGL_3.2_elements
make Sym_3.elements
make Cycle_13.power
make Cycle_12.power
make PGL_3.4.singer
make GL_2.8.multiply
make GL_2.7.multiply
make GL_2.7.inv
make GL_2.7.power
make PGL_3.2_classes
make PGL_4.2_classes_based_on_normal_form
make PGL_10.2_classes_based_on_normal_form
make normal_forms_PGL_4.4
make PGL_4.2A_rank
make PGL_4.2A_unrank
make PGL_4.5.3B_rank
make PGL_4.5.3B_unrank
make normal_forms_PGL_4_5
make on_planes

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 PGL_3_2_elements.csv

Sym_3_elements:

```bash
$ (ORBITER) -v 3 \
  -define G -permutation_group -symmetric_group 3 -end \
  -with G -do \n  -group_theoretic_activity \n  -print_elements.tex \n  -end
```

```
$ (ORBITER) -v 2 \
  -draw_options \n  -nodes \n  -embedded -radius 250 \n  -xin 10000 -yin 10000 \n  -xout 1000000 -yout 600000 \n  -scale 0.3 -line_width 1.0 \n  -end \n  -tree_draw -file Sym_3_elements.tree.txt -end
```

```
pdflatex Sym_3_elements_tree_draw.tex
open Sym_3_elements_tree_draw.pdf
```

Cycle_13_power:

```
$ (ORBITER) -v 5 \
  -define G -permutation_group -symmetric_group 13 -end \
  -with G -do \n  -group_theoretic_activity \n  -consecutive_powers \n  "1,2,3,4,5,6,7,8,9,10,11,12,0" 13 \n  -end
```

```
pdflatex Sym_13_all_powers.tex
open Sym_13_all_powers.pdf
```

Cycle_12_power:

```
$ (ORBITER) -v 5 \
  -define G -permutation_group -symmetric_group 12 -end \
  -with G -do \n  -group_theoretic_activity \n  -consecutive_powers \n  "1,2,3,4,5,6,7,8,9,10,11,0" 12 \n  -end
```

```
pdflatex Sym_12_all_powers.tex
open Sym_12_all_powers.pdf
```

608
```latex
\begin{verbatim}
4590
4591
4592
4593  PGL_3.4_singer:
4594  \$ (ORBITER) -v 5 \ 
4595  \$ -define G -linear_group -PGL 3 4 -end \ 
4596  \$ -with G -do \ 
4597  \$ -group_theoretic_activity \ 
4598  \$ -find_singer_cycle \ 
4599  \$ -end
4600
4601
4602  GL_2.8_multiply:
4603  \$ (ORBITER) -v 5 \ 
4604  \$ -define G -linear_group -GL 2 8 -end \ 
4605  \$ -with G -do \ 
4606  \$ -group_theoretic_activity \ 
4607  \$ -multiply "0,1,2,3" "4,5,6,7" \ 
4608  \$ -end
4609  \$ pdflatex GL_2.8_mult.tex
4610  \$ open GL_2.8_mult.pdf
4611
4612
4613  GL_2.7_multiply:
4614  \$ (ORBITER) -v 5 \ 
4615  \$ -define G -linear_group -GL 2 7 -end \ 
4616  \$ -with G -do \ 
4617  \$ -group_theoretic_activity \ 
4618  \$ -multiply "0,1,2,3" "4,5,6,0" \ 
4619  \$ -end
4620  \$ pdflatex GL_2.7_mult.tex
4621  \$ open GL_2.7_mult.pdf
4622
4623
4624  GL_2.7_inv:
4625  \$ (ORBITER) -v 5 \ 
4626  \$ -define G -linear_group -GL 2 7 -end \ 
4627  \$ -with G -do \ 
4628  \$ -group_theoretic_activity \ 
4629  \$ -inverse "0,1,2,3" \ 
4630  \$ -end
4631  \$ pdflatex GL_2.7_inv.tex
4632  \$ open GL_2.7_inv.pdf
4633
4634  GL_2.7_power:
4635  \$ (ORBITER) -v 5 \ 
4636  \$ -define G -linear_group -GL 2 7 -end \ 
\end{verbatim}
```
PGL_3_2_classes:
$\text{(ORBITER)} -v 3 $
-define G -linear_group -PGL 3 2 -end
-with G -do 
-group_theoretic_activity 
-classes_based_on_normal_form 
-end
pdflatex PGL_3_2_classes_normal_form.tex
open PGL_3_2_classes_normal_form.pdf
#pdflatex PGL_3_2_classes_out.tex
#open PGL_3_2_classes_out.pdf

PGL_4_2_classes_based_on_normal_form:
$\text{(ORBITER)} -v 3 $
-define G -linear_group -PGL 4 2 -end
-with G -do 
-group_theoretic_activity 
-classes_based_on_normal_form 
-end
pdflatex PGL_4_2_classes_normal_form.tex
open PGL_4_2_classes_normal_form.pdf

PGL_10_2_classes_based_on_normal_form:
$\text{(ORBITER)} -v 3 $
-define G -linear_group -PGL 10 2 -end
-with G -do 
-group_theoretic_activity 
-classes_based_on_normal_form 
-end
pdflatex PGL_10_2_classes_normal_form.tex
open PGL_10_2_classes_normal_form.pdf

normal_forms_PGL_4_4:
\$(ORBITER) -v 7 \ndefine G -linear_group -PGGL 4 4 -end \nwith G -do \ngroup_theoretic_activity \nclasses_based_on_normal_form \nend

\pdflatex\ PGGL_4_4_classes_normal_form.tex
open PGGL_4_4_classes_normal_form.pdf

PGL_4_4_2A_rank:
\define G -linear_group -PGGL 4 4 -end \nwith G -do \ngroup_theoretic_activity \nelement_rank \nelement_unrank "1,0,0,0, 0,1,0,0, 0,0,1,0, 0,0,0,1, 1" \nend

PGL_4_4_2A_unrank:
\define G -linear_group -PGGL 4 4 -end \nwith G -do \ngroup_theoretic_activity \nelement_unrank "1"
end

element_unrank

PGL_4_5_3B_rank:
\define G -linear_group -PGL 4 5 -end \nwith G -do \ngroup_theoretic_activity \nelement_rank "0,0,0,1, 2,3,0,1, 0,3,4,4, 0,1,2,1" \nend

element_rank

PGL_4_5_3B_unrank:
\define G -linear_group -PGL 4 5 -end \nwith G -do \ngroup_theoretic_activity \nelement_unrank "1"
end

element_unrank


4731 ▷ ▷ -define G -linear_group -PGL 4 5 -end \ 
4732 ▷ ▷ -with G -do \ 
4733 ▷ ▷ -group_theoretic_activity \ 
4734 ▷ ▷ ▷ -element_unrank "701459351" \ 
4735 ▷ ▷ -end 
4736 ▷ ▷ 
4737 
4738 
4739 
4740 normal_forms_PGL_4_5: 
4741 ▷ $(ORBITER) -v 7 \ 
4742 ▷ ▷ -define G -linear_group -PGL 4 5 -end \ 
4743 ▷ ▷ -with G -do \ 
4744 ▷ ▷ -group_theoretic_activity \ 
4745 ▷ ▷ ▷ -classes_based_on_normal_form \ 
4746 ▷ ▷ -end 
4747 ▷ pdflatex PGL_4_5_classes_normal_form.tex 
4748 ▷ open PGL_4_5_classes_normal_form.pdf 
4749 
4750 
4751 
4752 
4753 
4754 
4755 # Section 5.7: Group Theoretic Activities Based on Magma 
4756 
4757 
4758 
4759 SECTION_GROUP_THEORETIC_ACTIVITIES_BASED_ON_MAGMA: 
4760 
4761 
4762 test_5_7: 
4763 ▷ make PGGL_2_4_classes 
4764 ▷ make PGL_7_2_classes 
4765 ▷ make PGL_8_2_classes 
4766 ▷ make PGL_10_2_classes 
4767 ▷ make PGGL_2_4_cent_2A 
4768 ▷ make Normalizer_of_H5 
4769 ▷ make PGGL_3_4_classes 
4770 ▷ make classes_PGGL_4_4 
4771 ▷ make subgroups_PGL_4_5 
4772 ▷ make classes_PGL_4_5 
4773 ▷ make PGL_4_5_3B_class_again 
4774 ▷ make search_primitive_poly_q5_deg3 
4775 ▷ make GL_3_5_singer_power 
4776 ▷ make PGL_4_5_norm_31 
4777 ▷ make Normalizer_of_Z22_in_PGL_2_9
classes:  

\begin{verbatim}
$(ORBITER) -v 3 \\
  -define G \\
  -linear_group -PGGL 2 4 \\
  -end \\
  -with G -do \\
  -group_theoretic_activity \\
  -classes \\
  -end
\end{verbatim}

\begin{verbatim}
$(MAGMA_PATH)magma PGGL_2_4_classes.magma
\end{verbatim}

\begin{verbatim}
 $(ORBITER) -v 3 \\
  -define G \\
  -linear_group -PGGL 2 4 \\
  -end \\
  -with G -do \\
  -group_theoretic_activity \\
  -classes \\
  -end
\end{verbatim}

\begin{verbatim}
pdflatex PGGL_2_4_classes.out.tex
\end{verbatim}

\begin{verbatim}
open PGGL_2_4_classes.out.pdf
\end{verbatim}

\begin{verbatim}
open PGGL_2_4_classes.out.csv
\end{verbatim}

\begin{verbatim}
PGL_7_2_classes:  

$(ORBITER) -v 3 \\
  -define G \\
  -linear_group -PGL 7 2 \\
  -end \\
  -with G -do \\
  -group_theoretic_activity \\
  -classes \\
  -end
\end{verbatim}

\begin{verbatim}
$(MAGMA_PATH)magma PGL_7_2_classes.magma
\end{verbatim}

\begin{verbatim}
 $(ORBITER) -v 3 \\
  -define G \\
  -linear_group -PGL 7 2 \\
  -end \\
  -with G -do \\
  -group_theoretic_activity \\
  -classes \\
  -end
\end{verbatim}

\begin{verbatim}
pdflatex PGL_7_2_classes.out.tex
\end{verbatim}

\begin{verbatim}
open PGL_7_2_classes.out.pdf
\end{verbatim}

\begin{verbatim}
open PGL_7_2_classes.out.csv
\end{verbatim}

\begin{verbatim}
PGL_8_2_classes:  

 $(ORBITER) -v 3 \\
\end{verbatim}

613
-define G
-linear_group -PGL 8 2
-end
-with G -do
-group_theoretic_activity
-classes
-end
$(MAGMA_PATH)magma PGL_8_2_classes.magma

-define G
-linear_group -PGL 8 2
-end
-with G -do
-group_theoretic_activity
-classes
-end
$(ORBITER) -v 3

PGL_10_2_classes:
-define G
-linear_group -PGL 10 2
-end
-with G -do
-group_theoretic_activity
-classes
-end
$(MAGMA_PATH)magma PGL_10_2_classes.magma

-define G
-linear_group -PGL 10 2
-end
-with G -do
-group_theoretic_activity
-classes
-end
$(ORBITER) -v 3

PGGL_2_4_cent_2A:
-define G
-linear_group -PGGL 2 4 -end
-with G -do
-group_theoretic_activity

# ToDo:
Normalizer of H5:

 PGGL_3_4_classes:
classes_PGGL_4_4:

$\text{\texttt{(ORBITER)} -v 3 \> -magma_path $(MAGMA_PATH) \> -define G -linear_group -PGGL 4 4 -end \> -with G -do \> -group_theoretic_activity \> -classes \> -end}$

# group order $1974067200 = 2^{13} \times 3^4 \times 5^2 \times 7 \times 17$

# the -find_subgroup command is too specialized

subgroups_PGGL_4_5:

$\text{\texttt{(ORBITER)} -v 6 \> -define G \> -linear_group -PGL 4 5 -end \> -with G -do \> -group_theoretic_activity \> -find_subgroup 3 \> -end}$

pdflatex PGL_4_5_report.tex
open PGL_4_5_report.pdf

classes_PGGL_4_5:

$\text{\texttt{(ORBITER)} -v 6 \> -define G \> -linear_group -PGL 4 5 -end \> -with G -do \> -group_theoretic_activity \> -classes \> -end}$

pdflatex PGL_4_5_classes_out.tex
open PGL_4_5_classes_out.pdf

# 163 classes

# two classes of elements of order 3
#Order of element = 3 Class size = 310000 Centralizer order = 93600 Normalizer order = 187200
# of order 3 and with 0 fixed points.
#0,1,0,2,0,1,2,1,4,2,3,1,2,0,4,3,

#Class size = 10075000 Centralizer order = 2680 Normalizer order = 5760
# of order 3 and with 6 fixed points.
#0,0,0,1,2,3,0,1,0,3,4,0,1,2,1,

PGL_4_5_3B_class_again:

⊿ $(ORBITER)$ -v 6 \ $\define$ G -linear_group -PGL 4 5 -end \ $\with$ G -do \ $\group.theoretic.activity$ \ $\conjugacy.class.of$ \
$\"0,0,0,1,2,3,0,1,0,3,4,0,1,2,1\"$ \ $\end$ \ $\end$

search_primitive_poly_q5_deg3:
$\define$ G -linear_group -GL 3 5 -end \ $\with$ G -do \ $\group.theoretic.activity$ \ $\raise.to.the.power$ \ $\"0,1,0,0,0,1,0,3,0,4\"$ 31 \ $\end$ \ $\end$

⊿ $(ORBITER)$ -v 6 \ $\define$ G -linear_group -GL 3 5 -end \ $\with$ G -do \ $\group.theoretic.activity$ \ $\raise.to.the.power$ \ $\"0,1,0,0,1,0,3,0,4\"$ 31 \ $\end$ \ $\end$
pdflatex GL_3_5_power.tex
open GL_3_5_power.pdf

# ToDo:

PGL_4_5_norm_31:
$\define$ G -linear_group -PGL 4 5 -end \ $\with$ G -do \ $\group.theoretic.activity$ \ $\normalizer.of.cyclic.subgroup$ "31" \ $\"2,0,0,0,0,0,1,0,0,0,1,0,3,0,4\"$ \ $\end$ \ $\end$
Normalizer of $\mathbb{Z}_2^2$ in $\text{PGL}_2(9)$:

```
$\text{(ORBITER)} \ -v 2 \ -define \ G \ -linear \ group \ -PGL \ 2 \ 9 \ -subgroup \ by \ generators \ Z22 \ 4 \ 2 \ "2,0,0,1, 0,1,1,0" \ -end \ -with \ G \ -do \ -group_theoretic_activity \ -normalizer \ -end$
```

`pdflatex PGL_{2,9}Subgroup_Z22_4_normalizer.tex`

`open PGL_{2,9}Subgroup_Z22_4_normalizer.pdf`

# Chapter 6 - Orbit Algorithms

# Section 6.1: Orbit Algorithms

**SECTION** ORBIT ALGORITHMS SCHREIER TREES:

**test_6_1:**

```
make orbits_PGL_4_2_on_points_draw_tree
make orbits_PGL_4_2_on_points_export_trees
make T3r1_orbits
make T3r1_orbits_draw
make 2C_orbit_under_PGGL_4_4_elements_coded.csv
make PGGL_4_4_subgroups_of_type_2C_2C
make orbits_on_conics_q13
make orbits_cubic_curves_q2
make orbits_cubic_curves_q2_with_draw_tree
make poly_orbits_d3_n3_q2.csv
make poly_orbits_d3_n3_q2_get_ranks
make T4_orbits
make T4r1_orbits
make T4r1_orbits_draw
```
5059 ▷ make T4r1_orbits_4
5060 ▷ make PGGL_2_8_on_conic_orbits
5061 ▷ make PGGL_7_8_orbits
5062
5063
5064
5065 orbits_PGL_4_2_on_points.draw_tree:
5066 ▷ $(ORBITER) -v 4 \
5067 ▷ ▷ -draw_options -embedded -end \n5068 ▷ ▷ -define G -linear_group -PGL 4 2 -end \n5069 ▷ ▷ -define Orb -orbits -group G \n5070 ▷ ▷ ▷ -on_points \n5071 ▷ ▷ ▷ -end \n5072 ▷ ▷ ▷ -with Orb -do -orbits_activity \n5073 ▷ ▷ ▷ ▷ -report \n5074 ▷ ▷ ▷ ▷ -end \n5075 ▷ ▷ ▷ -with Orb -do -orbits_activity \n5076 ▷ ▷ ▷ ▷ -export_something "orbit" 0 \n5077 ▷ ▷ ▷ ▷ -end \n5078 ▷ ▷ ▷ -with Orb -do -orbits_activity \n5079 ▷ ▷ ▷ ▷ -draw_tree 0 \n5080 ▷ ▷ ▷ ▷ -end \n5081 ▷ pdflatex PGL_4_2.orbits_report.tex
5082 ▷ open PGL_4_2.orbits_report.pdf
5083
5084
5085 orbits_PGL_4_2_on_points.export_trees:
5086 ▷ $(ORBITER) -v 4 \
5087 ▷ ▷ -draw_options -embedded -end \n5088 ▷ ▷ -define G -linear_group -PGL 4 2 -end \n5089 ▷ ▷ -define Orb -orbits -group G \n5090 ▷ ▷ ▷ -on_points \n5091 ▷ ▷ ▷ -end \n5092 ▷ ▷ ▷ -with Orb -do -orbits_activity \n5093 ▷ ▷ ▷ ▷ -report \n5094 ▷ ▷ ▷ ▷ -end \n5095 ▷ ▷ ▷ ▷ -with Orb -do -orbits_activity \n5096 ▷ ▷ ▷ ▷ -export_trees \n5097 ▷ ▷ ▷ ▷ -end \n5098 ▷ ▷ ▷ ▷ $(ORBITER) -v 3 \n5099 ▷ ▷ ▷ ▷ -draw_layered_graph \n5100 ▷ ▷ ▷ ▷ orbit_PGL_4_2_0.layered_graph \n5101 ▷ ▷ ▷ ▷ -radius 500 -spanning_tree -embedded \n5102 ▷ ▷ ▷ ▷ ▷ -line_width 1.1 -x_stretch 1.4 -scale 0.25 \n5103 ▷ ▷ ▷ ▷ ▷ -end \n5104 ▷ pdflatex orbit_PGL_4_2_0.draw.tex
5105 ▷ open orbit_PGL_4_2_0.draw.pdf

619
T3r1.orbits:

\$(\text{ORBITER}) -v 4 \$

\$ -\text{draw\_options} -\text{embedded} -\text{end} \$

\$ -\text{define} G \$

\$ -\text{linear\_group} -\text{GL}_d_q_{wr\_Sym} n 2 2 3 \$

\$ -\text{on\_rank\_one\_tensors} -\text{end} \$

\$ -\text{define} \text{Orb} -\text{orbits} -\text{group} G \$

\$ -\text{on\_points} \$

\$ -\text{end} \$

\$ -\text{with} \text{Orb} -\text{do} -\text{orbits\_activity} \$

\$ -\text{report} \$

\$ -\text{end} \$

\$ -\text{with} \text{Orb} -\text{do} -\text{orbits\_activity} \$

\$ -\text{draw\_tree} 0 \$

\$ -\text{end} \$

pdflatex GL_2 2 wreath Sym3.orbit_0.tree.tex

open GL_2 2 wreath Sym3.orbit_0.tree.pdf

# ToDo

T3r1.orbits.draw:

\$(\text{ORBITER}) -v 3 \$

\$ -\text{draw\_layered\_graph} \$

\$ -\text{GL}_d_q_{wr\_Sym} n res27 0.layered\_graph \$

\$ -\text{radius} 500 -\text{spanning\_tree} -\text{embedded} \$

\$ -\text{line\_width} 1.1 -x\text{\_stretch} 1.4 -\text{scale} 0.25 \$

\$ -\text{end} \$

#pdflatex GL_2 2 wreath Sym3_report.tex

#open GL_2 2 wreath Sym3_report.pdf

pdflatex GL_2 2 wreath Sym3.res27 0.draw.tex

open GL_2 2 wreath Sym3.res27 0.draw.pdf

# write GL_2 2 wreath Sym3.res27 0.layered\_graph

2C_orbit\_under\_PGGL 4 4 elements\_coded.csv:

\$(\text{ORBITER}) -v 6 \$

\$ -\text{define} G -\text{linear\_group} -\text{PGGL} 4 4 -\text{end} \$

\$ -\text{with} G -\text{do} \$

\$ -\text{group\_theoretic\_activity} \$

\$ -\text{conjugacy\_class\_of\_element} \$

#write GL_2 2 wreath Sym3.res27 0.layered\_graph

620
# class of size 64260
# creates:
# 2C orbit under PGGL 4 4.csv
# 2C orbit under PGGL 4 4.txt
# 2C orbit under PGGL 4 4 elements coded.csv
# 2C orbit under PGGL 4 4 transporter.csv

# ToDo, the file 2C orbit under PGGL 4 4 transporter.csv is missing

PGGL_4_4_subgroups_of_type_2C_2C: 2C orbit under PGGL 4 4 elements coded.csv

# The distribution of orbit lengths is: ( 1, 2, 15, 20, 24^3, 30, 40, 240, 256, 480, 512, 960^2, 1280, 1920^2, 2560^4, 3840, 5120, 6144^3, 7680, 10240 )
# group_theoretic_activity::do orbits on group elements under conjugation after Cl asses.compute all point orbits
# found 29 conjugacy classes
# User time: 0:57
orbits_on_conics_q13:
(orbit) -v 4
-define G -linear_group -PGL 3 13 -end
-define Orb -orbits -group G
-define Orb -orbits -group G
-on_polynomials 2
-end
#pdflatex poly_orbits_d2_n2_q13.tex
#open poly_orbits_d2_n2_q13.pdf

orbits_cubic_curves_q2:
(orbit) -v 4
-define G -linear_group -PGL 3 2 -end
-define G -linear_group -PGL 3 2 -end
-define Orb -orbits -group G
-define Orb -orbits -group G
-on_polynomials 3
-end
#pdflatex poly_orbits_d3_n3_q2.tex
#open poly_orbits_d3_n3_q2.pdf

orbits_cubic_curves_q2_with_draw_tree:
(orbit) -v 4
-draw_options -yout 500000 -radius 150 -nodes_empty
-line_width 0.5 -y_stretch 0.25 -embedded -end
-define G -linear_group -PGL 3 2 -end
-define G -linear_group -PGL 3 2 -end
-on_polynomials 3
-end
-with Orb -do -orbits_activity
-draw_tree 6
-end
dflatex PGL3_2_orbit_6_tree.tex
open PGL3_2_orbit_6_tree.pdf

#ToDo: problem in report
#projective_space_with_action::compute_group_of_set done
#make: *** [poly_orbits_d3_n3_q2.csv] Segmentation fault: 11

poly_orbits_d3_n3_q2.csv:
(orbit) -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 \non-polynomials 3 \end
with Orb -do -orbits_activity \report \end
with Orb -do -orbits_activity \draw_tree 6 \end
poly_orbits_d3_n3_q2_get_ranks:
$(ORBITER) -v 4 
csv_file_select_cols poly_orbits_d3_n3_q2.csv 0
#pdflatex poly_orbits_d3_n3_q2.tex
#open poly_orbits_d3_n3_q2.pdf
T4_orbits:
$(ORBITER) -v 4 
define G \linear_group -GL_d_q_wr_Sym_n 2 2 4 \on_tensors -end \define Orb -orbits -group G \on_points \end
#pdflatex GL_2_2_wreath_Sym4_res65535_orbits.tex
#open GL_2_2_wreath_Sym4_res65535_orbits.pdf
#pdflatex GL_2_2_wreath_Sym4_report.tex
#open GL_2_2_wreath_Sym4_report.pdf
# ToDo

T4r1_orbits:
$(ORBITER) -v 4 \define G -linear_group -GL_d_q_wr_Sym_n 2 2 4 \on_rank_one_tensors -end \define Orb -orbits -group G \on_points \end
with Orb -do -orbits_activity \export_trees \end
T4r1.orbits.draw:

$(ORBITER) -v 3 \n-draw_layered_graph \norbit_GL_2.2.wreath_Sym4_res81_0.layered_graph \nradius 400 -spanning_tree -embedded \n-line_width 1.1 -x_stretch 2.5 -scale 0.15 \n-end

pdflatex orbit_GL_2.2.wreath_Sym4_res81_0_draw.tex
open orbit_GL_2.2.wreath_Sym4_res81_0_draw.pdf

T4r1.orbits_4:

$(ORBITER) -v 4 \norbit_path $(ORBITER_PATH) \n-define Control -poset_classification_control \n-problem_label T4r1 -W \ndraw_options -end \n-end

-define G -linear_group -GL_d_q_wr_Sym_n 2 2 4 \non_rank_one_tensors -end \ndefine Orb -orbits -group G \non_subsets 4 Control \n-end \n-with Orb -do -orbits_activity \n-report \n-report_options -draw_poset -end \n-end

pdflatex T4r1.poset.tex
open T4r1.poset.pdf

# ToDo

PGGL_2.8_on_conic_orbits:

$(ORBITER) -v 4 \ndefine G \n-linear_group -PGGL 2 8 -PGL2OnConic -end \ndefine Orb -orbits -group G \non_points \n-end

# example from the Fining manual, page 107:

PGGL_7.8_orbits:

$(ORBITER) -v 4 \n
SECTION POSET CLASSIFICATION:

# 1 min 31 sec on Mac

SECTION_POSET_CLASSIFICATION:

SECTION 6.2: Poset Classification

poset_of_4subsets:

poset_of_4subsets:

poset_of_5subsets:

poset_of_5subsets:

Symmetric_4_on_pairs_poset:

V_3.2_trivial:

V_4.2_trivial:

# 6.2: Poset Classification

test_6.2:

# 1 min 31 sec on Mac

SECTION POSET CLASSIFICATION:
poset_of_4subsets_draw:
  $(ORBITER) -v 3 \n  -draw_layered_graph \n  poset_4_poset_lvl_4.layered_graph \n  -radius 300 -embedded -line_width 1.1 \n  -y_stretch 0.9 -scale 0.25 \n  -end
  pdflatex poset_4_poset_lvl_4_draw.tex
  open poset_4_poset_lvl_4_draw.pdf

poset_of_5subsets:
  $(ORBITER) -v 3 \n  -orbiter_path $(ORBITER_PATH) \n  -define Control -poset_classification_control \n  -problem_label poset_5 \n  -W -depth 5 \n  -draw_options -radius 150 -end \n  -end \n  -define G -permutation_group -identity_group 5 -end \n  -define Orb -orbits -group G \n  -on_subsets 5 Control \n  -end \n  -with Orb -do -orbits_activity \n  -report \n  -report_options -draw_poset -end \n  -end
  pdflatex poset_5_poset.tex
  open poset_5_poset.pdf

poset_of_5subsets_draw:
  $(ORBITER) -v 3 \n  -draw_layered_graph \n  poset_5_poset_lvl_5.layered_graph \n  -radius 300 -embedded \n  -line_width 1.1 -y_stretch 0.9 \n  -scale 0.25 \n  -end
  pdflatex poset_5_poset_lvl_5_draw.tex
  open poset_5_poset_lvl_5_draw.pdf

Symmetric_4_on_pairs_poset:
  $(ORBITER) -v 3 \n
626
V3_2_trivial:

V4_2_trivial:
define Control -poset_classification_control \  
problem_label V_4_2.trivial \  
-W -depth 3 \  
draw_options \  
-radius 200 -embedded \  
-end \  
define G -linear_group -PGL 4 2 -identity_group -end \  
define Orb -orbits -group G \  
on_subspaces 4 Control \  
-end \  
-with Orb -do -orbits_activity \  
-report \  
-report_options -draw_poset -end \  
-end

# Section 6.3: Orbits on Subsets

SECTION ORBITS_ON_SUBSETS:

test_6.3:

make PG_2_2_subsets
make PG_3_2_subsets
make PGL_3_2_singer
make PGL_3_2_on_lines
make PGL_2_5_on_subsets
make PGL_2_7_on_subsets
make PGGL_2_8_on_subsets
make PGGL_2_9_on_subsets
make PGL_2_11_on_subsets
make PGGL_2_16_on_subsets
make PGGL_2_32_on_subsets
make PG_3_4_subsets
make PGGL_2_9_orbits
make PG_5_2_on_subsets

PG_2_2_subsets:

$(ORBITER) -v 3 \  
-orbiter_path $(ORBITER_PATH) \  
-define Control -poset_classification_control \  

628
5527 \problem_label PGL_3_2 \\
5528 \depth 7 \\
5529 \draw_options \\
5530 \draw Options \radius 200 -embedded \\
5531 \end \\
5532 \end \\
5533 \define F \finite_field \q 2 \end \\
5534 \define G \linear_group \PGL 3 F \end \\
5535 \define Orb \orbits \group G \\
5536 \on_subsets 7 Control \\
5537 \end \\
5538 \with Orb -do -orbits_activity \\
5539 \report \\
5540 \report_options -draw_poset -end \\
5541 \end \\
5542 \pdflatex PGL_3_2_poset_lvl_7.tex \\
5543 \open PGL_3_2_poset_lvl_7.pdf \\
5544 \pdflatex PGL_3_2_poset.tex \\
5545 \open PGL_3_2_poset.pdf \\
5546 \pdflatex PGL_3_2_poset_detailed_lvl_7.tex \\
5547 \open PGL_3_2_poset_detailed_lvl_7.pdf \\
5548 \\
5549 \\
5550 # PG(3,2) has \(2^3+2^2+2^1+1 = 15\) points: \\
5551 # PG(3,3) has \(3^3+3^2+3^1+1 = 27 + 9 + 3 + 1 = 40\) points. \\
5552 \\
5553 \\
5554 \\
5555 \\
5556 \\
5557 PG_3_2_subsets: \\
5558 \orbiter $\$ (ORBITER) -v 3 \$
5559 \orbiter_path $\$ (ORBITER_PATH) \$
5560 \define Control \poset_classification_control \$
5561 \problem_label PGL_4_2 \$
5562 \depth 15 \$
5563 \draw_options \$
5564 \draw Options \radius 200 -embedded \$
5565 \end \$
5566 \end \$
5567 \define F \finite_field \q 2 \end \$
5568 \define G \linear_group \PGL 4 F \end \$
5569 \define Orb \orbits \group G \$
5570 \on_subsets 15 Control \$
5571 \end \$
5572 \with Orb -do -orbits_activity \$
5573 \report \$

629
PGL_3_2_singer:

\$(ORBITER) -v 3 \\
-define Control -population PGL_3_2 -singer 1 -end \\
-define Orb -orbits -group G \\
-on_subsets 7 Control \\
-extend \\
-with Orb -do -orbits_activity \\
-report \\
-report_options -draw_poset -end \\
-end \\
pdflatex PGL_3_2_singer_1_poset.tex \\
open PGL_3_2_singer_1_poset.pdf \\

5601

PGL_3_2_on_lines:

\$(ORBITER) -v 3 \\
-define G -linear_group -PGL_3 2 -singer 1 -end \\
-define Orb -orbits -group G \\
-on_subsets 7 Control \\
-on_subsets 7 Control \\
-end \\
pdflatex PGL_3_2_lines_poset.tex
open PGL_3_2_lines_poset.pdf

PGL_2_5_on_subsets:
$\langle ORBITER \rangle -v 5 \$

- orbiter_path $(ORBITER_PATH) \$

-define Control -poset_classification_control \$

-problem_label PGL_2_5 \$

-W -depth 6 \$

-end \$

-define G -linear_group -PGL_2_5 -end \$

-define Orb -orbits -group G \$

-on_subsets 6 Control \$

-end \$

-with Orb -do -orbits_activity \$

-report \$

-report_options -draw_poset -end \$

-end \$

pdflatex PGL_2_5_poset.tex

open PGL_2_5_poset.pdf

PGL_2_7_on_subsets:
$\langle ORBITER \rangle -v 10 \$

- orbiter_path $(ORBITER_PATH) \$

-define Control -poset_classification_control \$

-problem_label PGL_2_7 \$

-W -depth 8 \$

-end \$

-define G -linear_group -PGL_2_7 -end \$

-define Orb -orbits -group G \$

-on_subsets 8 Control \$

-end \$

-with Orb -do -orbits_activity \$

-report \$

-report_options -draw_poset -end \$

-end \$

pdflatex PGL_2_7_poset.tex

open PGL_2_7_poset.pdf

PGGL_2_8_on_subsets:
$\langle ORBITER \rangle -v 10 \$

- orbiter_path $(ORBITER_PATH) \$

-define Control -poset_classification_control \$

-problem_label PGGL_2_8 \$

-W -depth 9 \$

-end \$

pdflatex PGGL_2_8_poset.tex

open PGGL_2_8_poset.pdf
define G -linear_group -PGGL 2 8 -end \
define Orb -orbits -group G \
on_subsets 9 Control \
end \
with Orb -do -orbits_activity \
report \
end \
define Orb -orbits -group G \
on_subsets 10 Control \
end \
with Orb -do -orbits_activity \
report \
end \
define G -linear_group -PGGL 2 9 -end \
define Orb -orbits -group G \
on_subsets 10 Control \
end \
with Orb -do -orbits_activity \
report \
end \
define G -linear_group -PGGL 2 9 -end \
define Orb -orbits -group G \
on_subsets 10 Control \
end \
with Orb -do -orbits_activity \
report \
end \
define G -linear_group -PGGL 2 11 -end \
define Orb -orbits -group G \
on_subsets 12 Control \
end \
with Orb -do -orbits_activity \
report \
end \
define G -linear_group -PGGL 2 11 -end \
define Orb -orbits -group G \
on_subsets 12 Control \
end \
with Orb -do -orbits_activity \
report \
end 

PGL_2.9 on subsets: 
$(ORBITER) -v 10 \n-orbiter_path $(ORBITER_PATH) \ndefine Control -poset_classification_control \nproblem_label PGGL_2.9 \n-W -depth 10 \nend 
define G -linear_group -PGGL 2 9 -end 
define Orb -orbits -group G 
on_subsets 10 Control 
end 
with Orb -do -orbits_activity 
report 
end

PGL_2.11 on subsets: 
$(ORBITER) -v 10 \norbiter_path $(ORBITER_PATH) \ndefine Control -poset_classification_control \nproblem_label PGL_2.11 \n-W -depth 12 \nend 
define G -linear_group -PGL 2 11 -end 
define Orb -orbits -group G 
on_subsets 12 Control 
end 
with Orb -do -orbits_activity 
report 
end 
define G -linear_group -PGL 2 11 -end 
define Orb -orbits -group G 
on_subsets 12 Control 
end 
with Orb -do -orbits_activity 
report 
end

\pdflatex PGGL_2.8_poset.tex 
open PGGL_2.8_poset.pdf 
\pdflatex PGGL_2.9_poset.tex 
open PGGL_2.9_poset.pdf 
\pdflatex PGL_2.9_poset.pdf 
\pdflatex PGL_2.11_poset.tex
open PGL_2_11_poset.pdf

PGGL_2_16_on_subsets:
$\text{ORBITER} \text{-v 3 \ }
\text{-orbiter_path $(ORBITER_PATH) \ }
\text{-define Control -poset_classification_control \ }
\text{-problem_label PGGL_2_16 \ }
\text{-W -depth 10 \ }
\text{-end \ }
\text{-define G -linear_group -PGGL 2 16 -end \ }
\text{-define Orb -orbits -group G \ }
\text{-on_subsets 10 Control \ }
\text{-end \ }
\text{-with Orb -do -orbits_activity \ }
\text{-report \ }
\text{-report_options -draw_poset -end \ }
\text{-end }
pdflatex PGGL_2_16_poset.tex
open PGGL_2_16_poset.pdf

PGGL_2_32_on_subsets:
$\text{ORBITER} \text{-v 3 \ }
\text{-orbiter_path $(ORBITER_PATH) \ }
\text{-define Control -poset_classification_control \ }
\text{-problem_label PGGL_2_32 \ }
\text{-W -depth 8 \ }
\text{-end \ }
\text{-define G -linear_group -PGGL 2 32 -end \ }
\text{-define Orb -orbits -group G \ }
\text{-on_subsets 8 Control \ }
\text{-end \ }
\text{-with Orb -do -orbits_activity \ }
\text{-report \ }
\text{-report_options -draw_poset -end \ }
\text{-end }
pdflatex PGGL_2_32_poset.tex
open PGGL_2_32_poset.pdf

PG_3_4_subsets:
$\text{ORBITER} \text{-v 3 \ }
\text{-orbiter_path $(ORBITER_PATH) \ }
\text{-define Control -poset_classification_control \ }
\text{-problem_label PGGL_4_4 \ }

633
PGLGL_2_9.orbits:

PGGL_2_9.on_subsets:

PGO_5_2.on_subsets:
5809 ▷ ▷ -define G -linear_group -PGO 5 F -end \n5810 ▷ ▷ -define Orb -orbits -group G \n5811 ▷ ▷ ▷ -on_subsets 15 Control \n5812 ▷ ▷ -end \n5813 ▷ ▷ -with Orb -do -orbits_activity \n5814 ▷ ▷ ▷ -report \n5815 ▷ ▷ ▷ -report_options -draw_poset -end \n5816 ▷ ▷ -end \n5817 ▷ pdflatex PGO.5_2.poset.tex \n5818 ▷ open PGO.5_2.poset.pdf \n5819 \n5820 \n5821 \n5822 \n5823 # Section 6.4: Orbits on Subspaces \n5827 SECTION_ORBITS_ON_SUBSPACES: \n5830 test_6.4: \n5831 ▷ make subspaces_0p_4_2 \n5832 ▷ make PGL.4_2_on_subspaces \n5833 ▷ make PGL.4_2_singer_on_subspaces \n5834 ▷ make PGL.8_2_singer_on_subspaces \n5835 ▷ make Op.6_2_orbits_on_subspaces \n5836 ▷ make Op.6_3_orbits_on_subspaces \n5837 ▷ make Op.6.11_orbits_on_subspaces \n5838 ▷ make Op.8_2_orbits_on_subspaces \n5839 ▷ make PGO.7_2_on_subspaces \n5840 \n5841 subspaces_0p_4_2: \n5842 ▷ $(ORBITER) -n 5 \n5843 ▷ ▷ -orbiter_path $(ORBITER_PATH) \n5844 ▷ ▷ -define Control -poset_classification_control \n5845 ▷ ▷ ▷ -draw_options -radius 200 -end \n5847 ▷ ▷ ▷ -problem_label Op.4_2 -W -depth 4 \n5848 ▷ ▷ -end \n5849 ▷ ▷ ▷ -define G -linear_group -PGL 4 2 -orthogonal 1 -end \n5850 ▷ ▷ ▷ -define Orb -orbits -group G \n5851 ▷ ▷ ▷ -on_subspaces 4 Control \n5852 ▷ ▷ -end \n5853 ▷ ▷ -with Orb -do -orbits_activity \n5854 ▷ ▷ ▷ -report \n5855 ▷ ▷ ▷ -report_options -draw_poset -end \n
635
PGL\_4\_2\_on\_subspaces:

```latex
\texttt{\$\text{ORBITER} -v 5 \$
\texttt{-orbiter\_path $\text{ORBITER\_PATH}$ \}
\texttt{-define Control \text{-poset\_classification\_control} \}
\texttt{-problem\_label PGL\_4\_2 \}
\texttt{-W -depth 4 \}
\texttt{-end \}
\texttt{-define G \text{-linear\_group -PGL 4 2 -end \}
\texttt{-define Orb \text{-orbits -group G} \}
\texttt{-on\_subspaces 4 Control \}
\texttt{-end \}
\texttt{-with Orb \text{-do -orbits\_activity} \}
\texttt{-report \}
\texttt{-report\_options -draw\_poset -end \}
\texttt{-end}
\texttt{pdf\_latex PGL\_4\_2\_poset\_tex}
\texttt{open PGL\_4\_2\_poset.pdf}
\texttt{PGL\_4\_2\_singer\_on\_subspaces:
\texttt{\$\text{ORBITER} -v 5 \$
\texttt{-orbiter\_path $\text{ORBITER\_PATH}$ \}
\texttt{-define Control \text{-poset\_classification\_control} \}
\texttt{-draw\_options -end \}
\texttt{-problem\_label PGL\_4\_2\_singer \}
\texttt{-W -depth 4 \}
\texttt{-end \}
\texttt{-define G \text{-linear\_group -PGL 4 2 -singer 1 -end \}
\texttt{-define Orb \text{-orbits -group G} \}
\texttt{-on\_subspaces 4 Control \}
\texttt{-end \}
\texttt{-with Orb \text{-do -orbits\_activity} \}
\texttt{-report \}
\texttt{-report\_options -draw\_poset -end \}
\texttt{-end}
\texttt{pdf\_latex PGL\_4\_2\_Singer\_4\_2\_1\_poset\_tex}
\texttt{open PGL\_4\_2\_Singer\_4\_2\_1\_poset.pdf}
```

636
PGL_8_2_singer_on_subspaces:

```
> $(ORBITER) -v 5 \
>   -orbiter_path $(ORBITER_PATH) "\n>   -define Control -poset_classification_control "\n>   -draw_options -radius 150 -end "\n>   -problem_label PGL_8_2_singer "\n>   -W -depth 8 "\n>   -end "

> > -define G -linear_group -PGL 8 2 -singer 1 -end "
> > -define Orb -orbits -group G "
> > -on_subspaces 8 Control "
> > -end "
> > -with Orb -do -orbits_activity "
> > -report "
> > -report_options -draw_poset -end "
> > -end "
```

# May 7, 2020: 16 sec on Mac

```
Op_6_2_orbits_on_subspaces:

```

```
Op_6_3_orbits_on_subspaces:

```
Op_6.11_orbits_on_subspaces:
  $(ORBITER) -v 5 \n  -orbiter_path $(ORBITER_PATH) \n  -define Control -poset_classification_control \n  -draw_options -radius 200 -end \n  -problem_label Op_6.11 -W -depth 6 \n  -end \n  -define G -linear_group -PGL 6 11 -orthogonal 1 -end \n  -define Orb -orbits -group G \n  -on_subspaces 6 Control \n  -end

Op_8.2_orbits_on_subspaces:
  $(ORBITER) -v 5 \n  -orbiter_path $(ORBITER_PATH) \n  -define Control -poset_classification_control \n  -draw_options -radius 200 -end \n  -problem_label Op_8.2 -W -depth 8 \n  -end \n  -define G -linear_group -PGL 8 2 -orthogonal 1 -end \n  -define Orb -orbits -group G \n  -on_subspaces 8 Control \n  -end

PGO_7.2_on_subspaces:
  $(ORBITER) -v 20 \n  -orbiter_path $(ORBITER_PATH) \n  -define Control -poset_classification_control \n  -draw_options -radius 200 -end \n  -problem_label 0_7.2 \n  -W -depth 7 \n  -end \n  -define F -finite_field -q 2 -end \n  -define G -linear_group -PGL 7 F -orthogonal 0 -end \n
638
 SECTION ORBITS ON SET PARTITIONS:

 test 6.5:

 make C6_on_partition
 make PGL_2.17_on_partition

 C6_on_partition:

 make C6_on_partition

 PGL_2.17_on_partition:

 make PGL_2.17_on_partition
SECTION_ARCS_AND_CAPS_IN_PROJECTIVE_SPACES:

test_6.6:
> make PGL_3_27
> make AGGL_2_27
> make hyperoval_4_classify
> make hyperoval_8_classify
> make frame_stabilizer_PGGL
> make hyperoval_16_classify
> make hyperoval_16_1_conic_type
> make hyperoval_16_1_nonconical_type
> make hyperoval_16_2_nonconical_type
> make hyperoval_16_stab_0_disjoint_sets_graph
> make nc_arcs_16
> make nc_arcs_32_E13
> make F64_work
> make F64_frob
> make surface_64_0
> make nc_arcs_128
> make nc_arcs_256_E13
> make Example_F64
> make six_arcs_4_nbE13
> make six_arcs_8_nbE13
> make six_arcs_16_nbE13
> make six_arcs_32_nbE13
> make six_arcs_64_nbE13
> make six_arcs_128_nbE13
> make six_arcs_256_nbE13
> make five_arcs_q13

PGL_3_27:
> $(ORBITER) -v 5 \
> -define G \n> -linear_group -PGL 3 27 -end \n> -with G -do \n
640
$\text{AGGL}_2.27$: 

```
$(\text{ORBITER}) -v 5 \$
```

```
define G 
```

```
-linear_group -AGGL 2 27 -end 
```

```
with G -do 
```

```
-group.theoretic_activity \ 
```

```
-report \ 
```

```
-end 
```

```
pdflatex AGGL_2.27_report.tex 
```

```
open AGGL_2.27_report.pdf 
```

```

```
hyperoval_4_classify: 
```

```
$(\text{ORBITER}) -v 4 \$
```

```
-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 hyperoval_q4 \ 
```

```
-W -depth 6 \ 
```

```
-end \ 
```

```
-with P -do \ 
```

```
-projective_space_activity \ 
```

```
-classify_arcs \ 
```

```
-control Control \ 
```

```
-target_size 6 \ 
```

```
-d 2 \ 
```

```
-end \ 
```

```
-end 
```

```
pdflatex hyperoval_q4_poset.tex 
```

```
open hyperoval_q4_poset.pdf 
```

```

```
hyperoval_8_classify: 
```

```
$(\text{ORBITER}) -v 4 \$
```

```
-orbiter_path $(\text{ORBITER\_PATH}) \ 
```

```
-define F -finite_field -q 8 -end \ 
```

```
-orbiter_path $(ORBITER_PATH) \
-define F -finite_field -q 16 -end \n-define P -projective_space -n 2 -field F -v 0 -end \n-define Control -poset_classification_control \n-problem_label hyperoval_q16 \\
-W -depth 18 \n-draw_options \n-radius 200 \n-end \\
-with P -do \n-projective_space_activity \\
-classify_arcs \\
-control Control \\
-target_size 18 \\
-d 2 \n-end \\
-end \\
#pdflatex hyperoval_q16_poset.tex 
#open hyperoval_q16_poset.pdf 

hyperoval_16_1_conic_type:
$(ORBITER) -v 2 \\
-define F -finite_field -q 16 -end \\
-define P -projective_space -n 2 -field F -v 0 -end \\
-define H_16_1 -geometric_object P \\
-set $(HYPEROVAL_16_144) \\
-end \\
-with H_16_1 -do \\
-combinatorial_object_activity \\
-save \\
-end \\
-with H_16_1 -do \\
-combinatorial_object_activity \\
-conic_type 6 \\
-end \\
-print_symbols 

hyperoval_16_1_nonconical_type:
$(ORBITER) -v 2 \\
-define F -finite_field -q 16 -end \

643
define P -projective_space -n 2 -field F -v 0 -end 
define H_16.1 -geometric_object P 

\[ \text{define H}_{16.1} \text{ -geometric_object P} \]

\[ \text{set } $(\text{HYPEROVAL}_{16.144}) \text{ -end} \]

\[ \text{end} \]

\[ \text{with } H_{16.1} \text{ -do} \]

\[ \text{non_conical_type} \]

\[ \text{end} \]

\[ \text{save} \]

\[ \text{end} \]

\[ \text{non_conical_type} \]

\[ \text{end} \]

\[ \text{print} \]

\[ \text{symbols} \]

\[
\begin{align*}
\text{We found 17028 non-conical 6 subsets} \\
\text{Eckardt point number distribution : } &13^{252}, \text{,} \, 9^{720}, \text{,} \, 5^{2304}, \text{,} \, 3^{13752} \\
\text{We found 6188 non-conical 6 subsets} \\
\text{Eckardt point number distribution : } &45^{68}, \text{,} \, 13^{2040}, \text{,} \, 5^{4080} \\
\end{align*}
\]

\[
\begin{align*}
\text{neighbors of } 0 \text{ with 4 removed.csv} \\
\text{Row,C0,C1,C2,C3} \\
0,2,3,9,10 \\
1,1,3,7,8 \\
2,10,12,13,15 \\
3,1,5,10,11 \\
4,3,5,6,13 \\
\end{align*}
\]
# ToDo neighbors of 0 with 4 removed.csv is missing

hyperoval_16_stab_0.disjoint_sets_graph:

```
$ (ORBITER) -v 2 \ 
-define G -graph -disjoint_sets_graph \ 
neighbors_of_0_with_4_removed.csv \ 
-with G -do \ 
-graph_theoretic_activity \ 
-find_cliques \ 
-target_size 4 \ 
-end \ 
-end \ 
-print_symbols
```

# 5 cliques of size 4

```
#ROW,C0,C1,C2,C3
#0,0,6,15,16
#1,1,2,13,14
#2,3,9,12,18
#3,4,5,7,10
#4,8,11,17,19
#END
```

clique 0:

```
#0,2,3,9,10
#6,7,11,13,17
#15,5,8,15,16
```
#16,1,6,12,14
# partition: (1,6,12,14|2,3,9,10|5,8,15,16|7,11,13,17)
# 4 is missing, it is the nucleus
# 0 is missing is the chosen point

# nonconical 6-arcs are used for classifying cubic surfaces:

cnc_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

#User time: 0:00

# ToDo:
poset_orbit_node::compute_flag_orbits before schreier_forest

#make: *** [nc_arcs_32_E13] Segmentation fault: 11

nc_arcs_32_E13:
$\text{(ORBITER) -v 4 \ 
-orbiter_path $(ORBITER_PATH) \ 
\text{-define F -finite_field -q 32 -end \n\text{-define P -projective_space -n 2 -field F -v 0 -end \n\text{-define Control -poset_classification_control \n\text{-problem_label nc_arcs_q32_d2 \n\text{-W -depth 6 \n\text{-draw_options -end \n\text{-end \n\text{-with P -do \n\text{-projective_space_activity \n\text{-classify_arcs \n\text{-control Control \n\text{-target_size 6 \n\text{-test_nb_Eckardt_points 13 \n\text{-d 2 \n\text{-conic_test \n\text{-end \n\text{-end \n#pdflatex nc_arcs_q32_d2_poset.tex
#open nc_arcs_q32_d2_poset.pdf

#User time: 0:00

F64_work:
$\text{(ORBITER) -v 3 \n\text{-define F -finite_field -q 64 -end \n\text{-define f -formula "f" "f" " " "a*a+a" \n\text{-with F -do -finite_field_activity \n\text{-evaluate f "a=2" -end

F64_frob:
$\text{(ORBITER) -v 3 \n\text{-define F -finite_field -q 64 -end \n\text{-define f -formula "f" "f" " " "a*a*a*a*a*a*a*a" \n\text{-with F -do -finite_field_activity \n\text{-evaluate f "a=61" -end

# surfaces with 13 Eckardt points have OCN=0,98,99

surface_64_0:
$\text{(ORBITER) -v 3 \n\text{-define F -finite_field -q 64 -end \n
647
6419 \define P -projective_space -n 3 -field F -v 0 -end \ 
6420 \define S -cubic_surface -space P -catalogue 0 -end \ 
6421 \with S -do \ 
6422 \cubic_surface_activity \ 
6423 \report \ 
6424 \end
6425 \pdflatex surface\_catalogue\_q64\_iso0\_report.tex
6426 \open surface\_catalogue\_q64\_iso0\_report.pdf

6431 \#makes it slow:
6432 \test nb\_Eckardt\_points 13 \ 
6433 \report -select_orbits\_by\_level 6 -select_orbits\_by\_stabilizer\_order\_multiple\_of 24 -end \ 
6434 \#User time: 0:3
6435
6436
6437
6438 nc\_arcs\_128:\n6439 \$\$(\text{ORBITER}) -v 4 \ 
6440 \define F -finite\_field -q 128 -end \ 
6441 \define P -projective\_space -n 2 -field F -use\_projectivity\_subgroup -v 0 -end \ 
6442 \define Control -poset\_classification\_control \ 
6443 \problem\_label nc\_arcs\_q128\_d2 -W -depth 6 \ 
6444 \end \ 
6445 \with P -do \ 
6446 \projective\_space\_activity \ 
6447 \classify\_arcs \ 
6448 \control Control \ 
6449 \target\_size 6 \ 
6450 \d 2 \ 
6451 \conic\_test \ 
6452 \end \ 
6453 \end
6454 \pdflatex nc\_arcs\_q128\_d2\_poset.tex
6455 \open nc\_arcs\_q128\_d2\_poset.pdf
6456
6457 \report -select_orbits\_by\_level 6 \ 
6458 \report -select_orbits\_by\_stabilizer\_order\_multiple\_of 24 \ 
6459
6460 \#User time: 0:52
6461
6462
6463
nc_arcs_256_E13:

\$\text{ORBITER} -v 8 \$

\$\text{define } F \text{ -finite_field -q 256 -end } \$

\$\text{define } P \text{ -projective_space -n } 2 \text{ -field } F \text{ -use_projectivity_subgroup -v 0 -end } \$

\$\text{define Control -poset_classification_control } \$

\$\text{-problem_label } nc\text{arcs}_q256_d2 \text{-W -depth 6 } \$

\$\text{-define P -projective -n 2 -field } F \text{ -v 0 -end } \$

\$\text{--control Control } \$

\$\text{-target_size 6 } \$

\$\text{-test_nb_Eckardt_points 13 } \$

\$\text{-d 2 } \$

\$\text{-conic_test } \$

\$\text{-end } \$

\$\text{-end } \$

\$\text{pdflatex } nc\text{arcs}_q256_d2\text{poset.tex} \$

\$\text{open } nc\text{arcs}_q256_d2\text{poset.pdf} \$

##

Example_F64:

\$\text{ORBITER} -v 3 \$

\$\text{define F -finite_field -q 64 -end } \$

\$\text{define P -projective_space -n 3 -field } F \text{ -v 0 -end } \$

\$\text{define S64_abcd}_{52,8,8,52} \text{-cubic_surface } \$

\$\text{space P -family_general_abcd } 52 8 8 52 \$

\$\text{-end } \$

\$\text{with S64_abcd}_{52,8,8,52} \text{-do } \$

\$\text{-cubic_surface_activity } \$

\$\text{-report } \$

\$\text{-end } \$

\$\text{pdflatex surface_family_general_abcd_q64_a52_b8_c8_d52_report.tex} \$

\$\text{open surface_family_general_abcd_q64_a52_b8_c8_d52_report.pdf} \$

six_arcs_4_nbE13:

\$\text{ORBITER} -v 3 \$

649
define F -finite_field -q 4 -end 
define P -projective_space -n 2 -field F -v 0 -end 
with P -do 
projective_space_activity 
control_six_arcs -problem_label sixarcs_q4 -end 
six_arcs_not_on_conic -filter_by_nb_Eckardt_points 13 -end

six_arcs_8_nbE13:
$\text{ORBITER}) -v 3 \$
define F -finite_field -q 8 -end 
define P -projective_space -n 2 -field F -v 0 -end 
with P -do 
projective_space_activity 
control_six_arcs -problem_label sixarcs_q8 -end 
six_arcs_not_on_conic -filter_by_nb_Eckardt_points 13 -end

six_arcs_16_nbE13:
$\text{ORBITER}) -v 3 \$
define F -finite_field -q 16 -end 
define P -projective_space -n 2 -field F -v 0 -end 
with P -do 
projective_space_activity 
control_six_arcs -problem_label sixarcs_q16 -end 
six_arcs_not_on_conic -filter_by_nb_Eckardt_points 13 -end

six_arcs_32_nbE13:
$\text{ORBITER}) -v 3 \$
define F -finite_field -q 32 -end 
define P -projective_space -n 2 -field F -v 0 -end 
with P -do 
projective_space_activity 
control_six_arcs -problem_label sixarcs_q32 -end 
six_arcs_not_on_conic -filter_by_nb_Eckardt_points 13 -end

six_arcs_64_nbE13:
$\text{ORBITER}) -v 3 \$
define F -finite_field -q 64 -end 
define P -projective_space -n 2 -field F -v 0 -end 
with P -do 
projective_space_activity 
control_six_arcs -problem_label sixarcs_q64 -end 
six_arcs_not_on_conic -filter_by_nb_Eckardt_points 13 -end

# 9 arcs: ago: 4, 8, 24^5, 48^2
\begin{verbatim}
 six_arcs_128_nbE13:
 $(ORBITER) -v 3 \n -define F -finite_field -q 128 -end \n -define P -projective_space -n 2 -field F -v 0 -end \n -with P -do \n -projective_space_activity \n -control_six_arcs -problem_label sixarcs_q128 -end \n -six_arcs_not_on_conic -filter_by_nb_Eckardt_points 13 -end

# 1 min 39 sec
# 12 arcs, ago: 4^3, 24^9

six_arcs_256_nbE13:
 $(ORBITER) -v 3 \n -define F -finite_field -q 256 -end \n -define P -projective_space -n 2 -field F -v 0 -end \n -with P -do \n -projective_space_activity \n -control_six_arcs -problem_label sixarcs_q256 -end \n -six_arcs_not_on_conic -filter_by_nb_Eckardt_points 13 -end

# 27 minutes on ripoff
#User time: 29:11 on ripoff 7/30/21

five_arcs_q13:
 $(ORBITER) -v 4 \n -define F -finite_field -q 13 -end \n -define P -projective_space -n 2 -field F -v 0 -end \n -define Control -poset_classification_control \n -problem_label five_arcs_q13 -W -depth 5 \n -end \n -with P -do \n -projective_space_activity \n -classify_arcs \n -control Control \n -target_size 5 \n -d 2 \n -end \n -end

#pdflatex five_arcs_q13_poset.tex
#open five_arcs_q13_poset.pdf
\end{verbatim}
# Section 6.7: Cubic Curves

SECTION_CUBIC_CURVES:

test_6.7:

`make cubic_curves_PG_2.4`

`make cubic_curves_PG_2.4_draw`

`make cubic_curves_PG_2.8`

`make cubic_curves_PG_2.8_draw`

cubic_curves_PG_2.4:

`$(ORBITER) -v 3 \`

`-orbiter_path $(ORBITER_PATH) \`

`-define F -finite_field -q 3 -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 \`

`-with P -do \`

`-projective_space_activity \`

`-classify_cubic_curves \`

`-control Control \`

`-target_size 9 -d 3 \`

`-end \`

`#pdflatex cc_4_poset.tex`

`#open cc_4_poset.pdf`

`#pdflatex cc_4_poset_lvl_9.tex`

`#open cc_4_poset_lvl_9.pdf`

`#pdflatex Cubic_curves_q4.tex`

`#open Cubic_curves_q4.pdf`

`#ToDo`
#graphical_output::draw_layered_graph_from_file file cc_4_poset_lvl_9.layered_graph does not exist

cubic_curves_PG_2_4.draw:
$\text{\textdollar}(\text{ORBITER}) -v 3 \$

$\text{\textdollar} -\text{draw}_\text{layered}_\text{graph} \text{ cc}_4\text{_poset}_\text{lvl}_9.\text{layered}_\text{graph} \$

$\text{\textdollar} -\text{radius} \text{ 300 -embedded -line_width 1.1} \$

$\text{\textdollar} -\text{y}\_\text{stretch} \text{ 0.9 -scale 0.25} \$

$\text{\textdollar} -\text{paths}_\text{in}_\text{between} \text{ 6 4 9 0} \$

$\text{\textdollar} -\text{end} \$

$\text{\textdollar} \text{pdflatex cc}_4\text{_poset}_\text{lvl}_9.\text{draw}.\text{tex} \$

$\text{\textdollar} \text{open cc}_4\text{_poset}_\text{lvl}_9.\text{draw}.\text{pdf} \$


cubic_curves_PG_2_8:
$\text{\textdollar}(\text{ORBITER}) -v 3 \$

$\text{\textdollar} -\text{define} \text{ F -finite_field -q 8 -end} \$

$\text{\textdollar} -\text{define} \text{ P -projective_space -n 2 -field F -v 0 -end} \$

$\text{\textdollar} -\text{define} \text{ Control -poset_classification_control} \$

$\text{\textdollar} -\text{problem_label cc}_8\text{ -\textit{W} -depth 9} \$

$\text{\textdollar} -\text{draw}_\text{options} -\text{radius} 200 -\text{embedded} -\text{end} \$

$\text{\textdollar} -\text{end} \$

$\text{\textdollar} -\text{with} \text{ P -do} \$

$\text{\textdollar} -\text{projective_space_activity} \$

$\text{\textdollar} -\text{classify}_\text{cubic}_\text{curves} \$

$\text{\textdollar} -\text{control} \text{ Control} \$

$\text{\textdollar} -\text{target}_\text{size} \text{ 9 -d 3} \$

$\text{\textdollar} -\text{end} \$

$\text{\textdollar} $(\text{ORBITER}) -v 2 -\text{draw}_\text{matrix} \$

$\text{\textdollar} -\text{input}_\text{csv}_\text{file cc}_8\text{_KM}_6.9.csv \$

$\text{\textdollar} -\text{box}_\text{width} 50 -\text{bit}_\text{depth} 8 -\text{end} \$

$\text{\textdollar} \$

$\text{\textdollar} \$

$\text{\textdollar} \# \text{pdflatex Cubic_curves_q8.tex} \$

$\text{\textdollar} \# \text{open Cubic_curves_q8.pdf} \$

$\text{\textdollar} \# \text{pdflatex cc}_8\text{_tree}_\text{lvl}_9.\text{tex} \$

$\text{\textdollar} \# \text{open cc}_8\text{_tree}_\text{lvl}_9.\text{pdf} \$

$\text{\textdollar} \$

$\text{\textdollar} \$

$\text{\textdollar} \# \text{the 6-set is orbit 7} \$

$\text{\textdollar} \# \text{the 9-set is orbit 1} \$

$\text{\textdollar} \#\# -\text{recognize }0,1,2,3,35,28 \$

$\text{\textdollar} \#\# -\text{recognize }1,2,3,51,28,61,46,71,40 \$

$\text{\textdollar} \$

$\text{\textdollar} \$

$\text{\textdollar} \# \text{todo:} \$

653
cubic_curves_PG_2.8.draw:

$\text{ORBITER} -v 3 \text{-draw_layered_graph }
\text{cc}_8\text{poset_lvl}_9\text{.layered_graph }
\text{-radius 2 -embedded -line_width 0.01 }
\text{-y_stretch 1.3 -scale 0.5 }
\text{-paths_in_between 6 7 9 1 }
\text{-end}

\text{pdflatex cc}_8\text{poset_lvl}_9\text{draw.tex}
\text{open cc}_8\text{poset_lvl}_9\text{draw.pdf}

# Chapter 7 - Cubic Surfaces

# Section 7.1: Cubic Surfaces Creation

SECTION_CUBIC_SURFACES_CREATION:

test_7.1:

make surface_4.0
make Hirschfeld_surface_get_incidence_matrix_40_40
make Hirschfeld_surface_incma_40_40_c
make surface_7.0
make Family_general_F7
make surface_eckardt_13_4.12
make surface_8.0_catalogue
make surface_8.0_clean
make surface_8.0_clean
make surface_13_0
make Eckardt_13
make surface_13.0
make surface_16.0
6743 ▷ make G13_8
6744 ▷ make F13_8
6745 ▷ make F13_16
6746 ▷ make F13_32
6747 ▷ make F13_64a
6748 ▷ make F13_64b
6749 ▷ make Colorado1
6750 ▷ make Colorado2
6751 ▷ make Colorado3
6752 ▷ make F13_128a
6753 ▷ make F13_128b
6754 ▷ make F13_128c
6755 ▷ make move_two_lines
6756 ▷ make F.alpha_beta_gamma_delta
6757 ▷ make surface_F_abcd_q7
6758 ▷ make surface_F_abcd_Eckardt_q31
6759 ▷ make surface_F_abcd_sweep_4_27_q7
6760 ▷ make surface_F_alpha_beta_gamma_delta_q7_override_group
6761 ▷ make surface_F_alpha_beta_gamma_delta_sweep_4_q3
6762 ▷ make surface_15lines_q7_1
6763 ▷ make surface_F_alpha_beta_gamma_delta_sweep_4_15_lines_q7
6764 ▷ make surface_F_alpha_beta_gamma_delta_q7_recognize
6765 ▷ make surface_49_recognize
6766 ▷ make surface_McKean_15lines_q7
6767 ▷ make surface_F_abcd_4433_q7
6768
6769
6770
6771  surface_4_0:
6772 ▷ $(ORBITER) -v 3 \
6773 ▷ ▷ -define F -finite_field -q 4 -end \
6774 ▷ ▷ -define P -projective_space -n 3 -field F -v 0 -end \
6775 ▷ ▷ -define S -cubic_surface -space P -catalogue 0 -end \
6776 ▷ ▷ -with S -do \ 
6777 ▷ ▷ ▷ -cubic_surface_activity \ 
6778 ▷ ▷ ▷ ▷ -report \ 
6779 ▷ ▷ ▷ ▷ -end \ 
6780 ▷ ▷ ▷ -with S -do \ 
6781 ▷ ▷ ▷ -cubic_surface_activity \ 
6782 ▷ ▷ ▷ ▷ -exportSomething "points" \ 
6783 ▷ ▷ ▷ ▷ -end \ 
6784 ▷ ▷ ▷ -with S -do \ 
6785 ▷ ▷ ▷ -cubic_surface_activity \ 
6786 ▷ ▷ ▷ ▷ -exportSomething "points_off" \ 
6787 ▷ ▷ ▷ ▷ -end \ 
6788 ▷ ▷ ▷ -with S -do \ 
6789 ▷ ▷ ▷ -cubic_surface_activity \ 

655
# 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

# points off:
#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

HIRSCHFELD_SURFACE_POINTS_OFF="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"

HIRSCHFELD_SURFACE_HESSE_PLANES="7,8,11,13,14,16,17,19,28,29,32,34,35,37,38,40,42,43,44,45,47,48,52,54,56,57,60,61,63,64,65,66,69,73,75,77,78,81,82"

Hirschfeld surface get incidence matrix 40 40:

$(ORBITER) -v 3 \\
> -define points -vector -dense $(HIRSCHFELD_SURFACE_POINTS_OFF) -end \\
> -define planes -vector -dense $(HIRSCHFELD_SURFACE_HESSE_PLANES) -end \\
> -define F -finite_field -q 4 -end \\
> -define P -projective_space -n 3 -field F -v 0 -end \\
> -with P -do \\
> -projective_space_activity \\
> -restricted_incidence_matrix 1 3 points planes "H_incma_40_40" \\
> -end \\

Hirschfeld_surface.incma.40.40.c:

$(ORBITER) -v 10 \\
> -draw_incidence_structure_description \\
> -width 60 -with_10 6 -end \\
> -define C -combinatorial_objects \\
> -file_of_incidence_geometries H_incma_40_40.inc 40 40 480 \\
> -end \\

656
-with C -do \
combinatorial_object_activity \
canonical_form \
-classification_prefix H.incma_40_40 \
-label H.incma_40_40 \
save_ago \
save_transversal \
-end \
-report \
-prefix H.incma_40_40 \
-export_flag_orbits \
-show_incidence_matrices \
-export_group_GAP \
-end \
$\text{(ORBITER)} -v 2 -draw \matrix \n-input_csv_file H.incma_40_40_object0_TDA_flag_orbits.csv \n-secondary_input_csv_file H.incma_40_40_object0_TDA.csv \n-box_width 32 -bit_depth 24 \
-end \
pdflatex H.incma_40_40_classification.tex 
open H.incma_40_40_classification.pdf 

surface_4_0:plane_type_of_lines_on_Klein_quadric: 
$\text{(ORBITER)} -v 6 \n\define L -vector -file surface_catalogue_q4_iso0_lines.csv -end \n\define F -finite_field -q 4 -end \n\define P -projective_space -n 3 -field F -v 0 -end \n-with P -do \n-projective_space_activity \n-plane_intersection_type_of_klein_image 4 L \n-end 

#The plane intersection type is (0^244496, 1^112914, 2^15120, 3^4005, 5^270) 
The plane intersection type is (5^270) 
L_highest_weight_objects.csv 

surface_7: 
$\text{(ORBITER)} -v 3 \n\define F -finite_field -q 7 -end \n\define P -projective_space -n 3 -field F -v 0 -end \n\define S -cubic_surface -space P -catalogue 0 -end \n-with S -do \ncubic_surface_activity \n-report \n
Family general F7:

```bash
> $(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 \
> -with S7_abcd_2_3_3_4 -do \
> -cubic_surface_activity \
> -report \ 
> -end \
> pdflatex surface_family_general_abcd_q7_a2_b3_c3_d4_report.tex \
> open surface_family_general_abcd_q7_a2_b3_c3_d4_report.pdf
```

# Fermat with 18 Eckardt points

# no automorphism group

```bash
> $(ORBITER) -v 3 \
> -define L7_0 -vector -file surface_catalogue_q7_iso0_lines.csv -end \
> -define F -finite_field -q 7 -end \
> -define P -projective_space -n 3 -field F -v 0 -end \
> -with P -do \
> -projective_space_activity \
> -plane_intersection_type_of_klein_image 4 L7_0 \
> -end
```

# Time: 4:50

The plane intersection type is (0^44526575, 1^3529170, 2^112563, 3^8541, 4^324, 5^27)

The plane intersection type is (4^324, 5^27)

L7_0_highest_weight_objects.csv

Family general F7:

```bash
> $(ORBITER) -v 6 \
> -define L7_0 -vector -file surface_catalogue_q7_iso0_lines.csv -end \
> -define F -finite_field -q 7 -end \
> -define P -projective_space -n 3 -field F -v 0 -end \
> -with P -do \
> -projective_space_activity \
> -plane_intersection_type_of_klein_image 4 L7_0 \
> -end
```

# Time: 4:50

The plane intersection type is (0^44526575, 1^3529170, 2^112563, 3^8541, 4^324, 5^27)

The plane intersection type is (4^324, 5^27)

L7_0_highest_weight_objects.csv

Family general F7:

```bash
> $(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 \
> -with S7_abcd_2_3_3_4 -do \
> -cubic_surface_activity \
> -report \ 
> -end \
> pdflatex surface_family_general_abcd_q7_a2_b3_c3_d4_report.tex \
> open surface_family_general_abcd_q7_a2_b3_c3_d4_report.pdf
```

# Fermat with 18 Eckardt points

# no automorphism group

```bash
> $(ORBITER) -v 3 \
> -define L7_0 -vector -file surface_catalogue_q7_iso0_lines.csv -end \
> -define F -finite_field -q 7 -end \
> -define P -projective_space -n 3 -field F -v 0 -end \
> -with P -do \
> -cubic_surface_activity \
> -export something "lines" \
> -end
```

# Fermat with 18 Eckardt points

# no automorphism group

```bash
> $(ORBITER) -v 3 \
> -define L7_0 -vector -file surface_catalogue_q7_iso0_lines.csv -end \
> -define F -finite_field -q 7 -end \
> -define P -projective_space -n 3 -field F -v 0 -end \
> -with P -do \
> -cubic_surface_activity \
> -export something "lines" \
> -end
```

# Fermat with 18 Eckardt points

# no automorphism group

```bash
> $(ORBITER) -v 3 \
> -define L7_0 -vector -file surface_catalogue_q7_iso0_lines.csv -end \
> -define F -finite_field -q 7 -end \
> -define P -projective_space -n 3 -field F -v 0 -end \
> -with P -do \
> -cubic_surface_activity \
> -export something "lines" \
> -end
```
-define F -finite_field -q 8 -end \n-define P -projective_space -n 3 -field F -v 0 -end \n-define S -cubic_surface -space P -catalogue 0 -end \n-with S -do \n-cubic_surface_activity \n-report \n-all_quartic_curves \n-end \n-with S -do \n-cubic_surface_activity \n-report \n-end \n
surface.8.0.plane.type_of_lines_on.Klein.quadric:
$\$(ORBITER) -v 6 \n-define L_8_0 -vector -file surface.catalogue.q8.iso0.lines.csv -end \n-define F -finite_field -q 8 -end \n-define P -projective_space -n 3 -field F -v 0 -end \n-with P -do \n-projective_space_activity \n-plane_intersection_type_of_klein_image 4 L_8_0 \n-end

#L_8.0.highest_weight.objects.csv
#intersection numbers: ( 0, 0, 0, 0, 192, 30 )
# time: 15:48

surface.8.0.catalogue:
$\$(ORBITER) -v 3 \n-define F -finite_field -q 8 -end \n-define P -projective_space -n 3 -field F -v 0 -end \n-define S8.0 -cubic_surface -space P -catalogue 0 -end \n-with S8.0 -do \n-cubic_surface_activity \n-report \n-end
-pdflatex surface.catalogue.q8.iso0.report.tex
-open surface.catalogue.q8.iso0.report.pdf

surface.8.0.clean:
\begin{verbatim}
$\text{(ORBITER) -v 3 \ }
\text{-define F -finite_field -q 8 -end \ }
\text{-define P -projective_space -n 3 -field F -v 0 -end \ }
\text{-define S8_0 -cubic_surface -space P -catalogue 0 \ }
\text{-select_double_six "15,11,22,19,24,5,16,10,23,20,25,4" \ }
\text{-select_double_six "3,2,1,0,5,4,9,8,7,6,11,10" \ }
\text{-transform_inverse "1,4,4,0,6,0,0,0,6,2,0,1,7,0,4,0,0" \ }
\text{-transform "4,4,0,0,0,0,1,0,1,0,0,0,0,0,1,0,0" \ }
\text{-transform_inverse "2,0,0,0,0,2,0,0,0,0,2,0,1,1,2,3,0" \ }
\text{-end \ }
\text{-with S8_0 -do \ }
\text{-cubic_surface_activity \ }
\text{-report \ }
\text{-end \ }
\text{pdflatex surface_catalogue_q8_iso0_report.tex}
\text{open surface_catalogue_q8_iso0_report.pdf}
\text{# clean equation for Tekirdag-1:}
\text{surface_9_0:}
\text{$\text{(ORBITER) -v 3 \ }
\text{-define F -finite_field -q 9 -end \ }
\text{-define P -projective_space -n 3 -field F -v 0 -end \ }
\text{-define S -cubic_surface -space P -catalogue 0 -end \ }
\end{verbatim}
surface_9_0_plane_type_of_lines_on_Klein_quadric:

$\#$intersection numbers: ( 0, 0, 0, 0, 180, 15 )

surface_9_1:

$\#$intersection numbers: ( 0, 0, 0, 0, 180, 15 )
7066 ▷ ▷ ▷ -plane_intersection_type_of_klein_image 4 L_9_1 \n7067 ▷ ▷ -end
7068
7069  # intersection numbers: ( 0, 0, 0, 0, 216 )
7070
7071  #44:36
7072
7073
7074  # Joel:
7075
7076  surface_eckardt_13_4_12:
7077  ▷ $(ORBITER) -v 6 \n7078  ▷  ▷ -define F -finite_field -q 13 -end \n7079  ▷  ▷ -define P -projective_space -n 3 -field F -v 0 -end \n7080  ▷  ▷ -define Eckardt_4_12 -cubic_surface \n7081  ▷  ▷  ▷ -space P -family_Eckardt 4 12 \n7082  ▷  ▷  ▷ -end \n7083  ▷  ▷ -with Eckardt_4_12 -do \n7084  ▷  ▷ -cubic_surface_activity \n7085  ▷  ▷  ▷ -report \n7086  ▷  ▷ -end
7087
7088
7089
7090
7091
7092
7093
7094  # 13_0 has 4 Eckardt points
7095  # 13_1 has 6 Eckardt points
7096  # 13_2 has 9 Eckardt points
7097  # 13_3 has 18 Eckardt points
7098  #
7099
7100
7101
7102
7103  Eckardt_13:
7104  ▷ $(ORBITER) -v 3 \n7105  ▷  ▷ -define F -finite_field -q 13 -end \n7106  ▷  ▷ -define P -projective_space -n 3 -field F -v 0 -end \n7107  ▷  ▷ -define Eckardt_3_1 -cubic_surface \n7108  ▷  ▷  ▷ -space P -family_Eckardt 3 1 \n7109  ▷  ▷  ▷ -end \n7110  ▷  ▷ -with Eckardt_3_1 -do \n7111  ▷  ▷ -cubic_surface_activity \n7112  ▷  ▷  ▷ -report \n
662
surface_13_0:
$$(ORBITER) -v 3 \n$\define F -\text{finite\_field} -q 13 \end \n$\define P -\text{projective\_space} -n 3 -\text{field} F -v 0 \end \n$\define S13\_0 -\text{cubic\_surface} -\text{space} P -\text{catalogue} 0 \end \n$\with S13\_0 -do \n$\text{cubic\_surface\_activity} \n$\text{report} \n$\end

# clean equation for Tekirdag-2:

surface_16_0:
$$(ORBITER) -v 3 \n$\define F -\text{finite\_field} -q 16 \end \n$\define P -\text{projective\_space} -n 3 -\text{field} F -v 0 \end \n$\define S16\_0 -\text{cubic\_surface} -\text{space} P -\text{catalogue} 0 \n$\transform "1,0,0,0,0,1,0,12,0,0,0,1,0" \n$\transform "15,11,4,0,0,0,12,0,0,0,0,0,1,3" \n$\end \n$\with S16\_0 -do \n$\text{cubic\_surface\_activity} \n$\text{report} \n$\end

# 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 )

# 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 )

# ai: 29181, 4677, 29950, 33, 62496, 429
# bi: 1, 9205, 37, 29964, 29364, 21501
Tekirdag-1:

G13.8:

$\$(ORBITER) -v 3 \
-define F -finite_field -q 8 -end \
-define P -projective_space -n 3 -field F -v 0 -end \
-define T1 -cubic_surface -space P -family_G13 2 -end \
-with T1 -do \
-cubic_surface_activity \
-report \
-end

Tekirdag-2:

F13.8:

$\$(ORBITER) -v 3 \
-define F -finite_field -q 8 -end \
-define P -projective_space -n 3 -field F -v 0 -end \
-define T1 -cubic_surface -space P -family_F13 2 -end \
-with T1 -do \
-cubic_surface_activity \
-report \
-end

Tekirdag-3:

F13.32:

$\$(ORBITER) -v 3 \
-define F -finite_field -q 32 -end \
-define P -projective_space -n 3 -field F -v 0 -end \
-define T3 -cubic_surface -space P -family_F13 2 -end \

664
# Kapadokya-1:

F13_64a:

$$(ORBITER) -v 3 \n\$$(define F -finite_field -q 64 -end \n\$$(define P -projective_space -n 3 -field F -v 0 -end \n\$$(define K1 -cubic_surface -space P -family_F13 2 -end \n\$$(with K1 -do \n\$$(cubic_surface_activity \n\$$(report \n\$$(end

# Kapadokya-2:

F13_64b:

$$(ORBITER) -v 3 \n\$$(define F -finite_field -q 64 -end \n\$$(define P -projective_space -n 3 -field F -v 0 -end \n\$$(define K2 -cubic_surface -space P -family_F13 18 -end \n\$$(with K2 -do \n\$$(cubic_surface_activity \n\$$(report \n\$$(end

# Colorado1:

$$(ORBITER) -v 3 \n\$$(define F -finite_field -q 128 -end \n\$$(define P -projective_space -n 3 -field F -v 0 -end \n\$$(define CO-1 -cubic_surface -space P -catalogue 0 \n\$$(transform_inverse "1,0,0,0,0,1,0,96,0,0,1,96,0,0,0,1,0" \n\$$(end \n\$$(with CO-1 -do \n\$$(cubic_surface_activity \n\$$(report \n\$$(end

# recognize the arcs from Colorado-1,2,3:
Colorado2:

```
$(ORBITER) -v 3 \
-def F -finite_field -q 128 -end \
-def P -projective_space -n 3 -field F -v 0 -end \
-def CO-2 -cubic_surface -space P -catalogue 926 \
-transform_inverse "1,0,0,0,1,0,32,0,0,1,32,0,0,0,1,0" \
-end \n-with CO-2 -do \n-cubic_surface_activity \n-report \n-end
```

Colorado3:

```
$(ORBITER) -v 3 \
-def F -finite_field -q 128 -end \
-def P -projective_space -n 3 -field F -v 0 -end \
-def CO-3 -cubic_surface -space P -catalogue 928 \
-transform inverse "1,0,0,0,1,0,59,0,0,1,59,0,0,0,1,0" \
-end \n-with CO-3 -do \n-cubic_surface_activity \n-report \n-end
```

# Colorado-1:

F13_128a:

```
$(ORBITER) -v 3 \
-def F -finite_field -q 128 -end \
-def P -projective_space -n 3 -field F -v 0 -end \
-def CO-1 -cubic_surface -space P -family F13 2 -end \
-with CO-1 -do \n-cubic_surface_activity \n-report \n-end
```

# Colorado-2:

F13_128b:

```
$(ORBITER) -v 3 \
-def F -finite_field -q 128 -end \
-def P -projective_space -n 3 -field F -v 0 -end \
-def CO-2 -cubic_surface -space P -family F13 6 -end \
-with CO-2 -do \n```
-cubic_surface_activity \  
-report \  
-end  

# Colorado-3:  
F13_128c:  
$ (ORBITER) -v 3 \  
-define F -finite_field -q 128 -end \  
-define P -projective_space -n 3 -field F -v 0 -end \  
-define CO-3 -cubic_surface -space P -family F13 14 -end \  
-with CO-3 -do \  
-cubic_surface_activity \  
-report \  
-end  

move_two_lines:  
$ (ORBITER) -v 5 \  
-define F -finite_field -q 8 -end \  
-with F -do -finite_field_activity \  
-move_two_lines_in_hyperplane_stabilizer \  
-65 4680 72 657 \  
-end  

F_alpha_beta_gamma_delta:  
$ (ORBITER) -v 3 \  
-define F -finite_field -q 7 -end \  
-with F -do -finite_field_activity \  
-parse_and_evaluate \  
"F_alpha_beta_gamma_delta" "x0,x1,x2,x3" \  
$(F_ALPHA_BETA_GAMMA_DELTA) \  
"alpha=2,beta=3,gamma=4,delta=5" \  
-end  

dot -Tpng F_alpha_beta_gamma_delta.gv >F_alpha_beta_gamma_delta.png  

surface_F_abcd_q7:  
$ (ORBITER) -v 3 \  
-define F -finite_field -q 7 -end \  
-with F -do \  
-finite_field_activity \  
-parse_and_evaluate "Fabcd" "X0,X1,X2,X3" \  
$(F_abcd_eqn_no_exponents) "a=4,b=2,c=2,d=4" 

7349 #dot -Tpng F_alpha_beta_gamma_delta.gv >F_alpha_beta_gamma_delta.png

7352 surface_F_abcd_Eckardt_q31:

7356 $(ORBITER) -v 3 \
7358 -define F -finite_field -q 31 -end \
7360 -define P -projective_space -n 3 -field F -v 0 -end \
7362 -define F_abcd -cubic_surface -space P \ 
7364 -by_equation "F_abcd" \ 
7366 "$\{a,b,c,d\}\" "X0,X1,X2,X3" \ 
7368 "$\{a=2,b=30,c=30,d=2\}\" \ 
7370 "$\{a=2,b=30,c=30,d=2\}\" \ 
7372 -end \
7374 -with F_abcd -do \ 
7376 -cubic_surface_activity \ 
7378 -report \ 
7380 -end \ 

7383 surface_F_abcd_sweep_4_27_q7:

7389 $(ORBITER) -v 3 \
7401 -define F -finite_field -q 7 -end \
7403 -define P -projective_space -n 3 -field F -v 0 -end \
7405 -projective_space_activity \ 
7407 -sweep 4_27 sweep_4_27_q7 -q 7 -by_equation "F_abcd" \ 
7409 "$\{a,b,c,d\}\" "X0,X1,X2,X3" \ 
7411 "$\{a=2,b=3,c=4,d=5\}\" \ 
7413 "$\{a=2,b=3,c=4,d=5\}\" \ 
7415 -end \
7416 -end \ 

7419 surface_F_alpha_beta_gamma_delta_q7_override_group:

7426 $(ORBITER) -v 3 \
7428 -define F -finite_field -q 7 -end \
7430 -define P -projective_space -n 3 -field F -v 0 -end \ 

668
define F_2345 -cubic_surface -space P \ 
by_equation "F_alpha_beta_gamma_delta" \ 
"x0,x1,x2,x3" \ 
$(F ALPHA BETA GAMMA DELTA) \ 
"alpha=2,beta=3,gamma=4,delta=5" \ 
"\alpha=2,\beta=3,\gamma=4,\delta=5" \ 
override_group 6 2 \ 
"1,5,0,0,3,6,0,0,1,1,3,0,5,5,0,3, \ 
1,0,2,5,0,1,6,1,0,0,3,5,0,0,4,4" \ 
end \ 
- with F_2345 -do \ 
-cubic_surface_activity \ 
-report \ 
-end \ 
-pdflatex surface_equation_F_alpha_beta_gamma_delta_q7_report.tex \ 
open surface_equation_F_alpha_beta_gamma_delta_q7_report.pdf \ 

# cubic surfaces with 15 lines: \ 

surface_F_alpha_beta_gamma_delta_sweep_4_q3: \ 
$(ORTBIER) -v 3 \ 
define F -finite_field -q 3 -end \ 
define P -projective_space -n 3 -field F -v 0 -end \ 
with P -do \ 
-projective_space_activity \ 
sweep_4_15_lines sweep_4_15_lines_q3 -q 3 \ 
by_equation "F_alpha_beta_gamma_delta" \ 
"x0,x1,x2,x3" \ 
$(F ALPHA BETA GAMMA DELTA) \ 
"alpha=2,beta=3,gamma=4,delta=5" \ 
"\alpha=2,\beta=3,\gamma=4,\delta=5" \ 
end \ 
end \ 

# cubic surfaces with 15 lines: \ 

surface_15lines_q7.1: \ 
$(ORTBIER) -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 \ 
by_equation "F_alpha_beta_gamma_delta" \ 
"\alpha=2,\beta=3,\gamma=4,\delta=5" \ 
end \ 
end \ 

669
7438 \> \> \> $(F_{\text{ALPHA_BETA_GAMMA_DELTA}}) \>
7439 \> \> \> "$\text{alpha}=6,\text{beta}=4,\text{gamma}=2,\text{delta}=2" \>
7440 \> \> \> "$\text{\textbackslash Dalpha}=6,\text{\textbackslash beta}=4,\text{\textbackslash gamma}=2,\text{\textbackslash delta}=2\text{D}" \>
7441 \> \> -end \>
7442 \> \> -with S -do \>
7443 \> \> -cubic_surface_activity \>
7444 \> \> \> -report \>
7445 \> \> -end \>
7446 \> \> pdflatex surface_equation_F_alpha_beta_gamma_delta_q7_report.tex \>
7447 \> \> open surface_equation_F_alpha_beta_gamma_delta_q7_report.pdf \>
7448 \>
7449 \>
7450 \>
7451 \> surface_F_alpha_beta_gamma_delta_sweep_4_15_lines.q7: \>
7452 \> $(\text{ORBITER}) -v 3 \>
7453 \> \> -define F -finite_field -q 7 -end \>
7454 \> \> -define P -projective_space -n 3 -field F -v 0 -end \>
7455 \> \> -with P -do \>
7456 \> \> -projective_space_activity \>
7457 \> \> -sweep_4_15_lines sweep_4_q7 -q 7 \>
7458 \> \> \> -by_equation \>
7459 \> \> \> "$F_{\text{\textbackslash alpha},\text{\textbackslash beta},\text{\textbackslash gamma},\text{\textbackslash delta}}" \>
7460 \> \> \> "$\text{\textbackslash D\textbackslash F}_{\text{\textbackslash alpha},\text{\textbackslash beta},\text{\textbackslash gamma},\text{\textbackslash delta}}\text{D}" \>
7461 \> \> \> "$x_0,x_1,x_2,x_3" \>
7462 \> \> \> $$(F_{\text{ALPHA_BETA_GAMMA_DELTA}}) \>
7463 \> \> \> "$\text{alpha}=2,\text{beta}=3,\text{gamma}=4,\text{delta}=5" \>
7464 \> \> \> "$\text{\textbackslash Dalpha}=2,\text{\textbackslash beta}=3,\text{\textbackslash gamma}=4,\text{\textbackslash delta}=5\text{D}" \>
7465 \> \> -end \>
7466 \>
7467 \>
7468 \> #User time: 0:30 \>
7469 \> # 348 parameter sets \>
7470 \> #F_alpha_beta_gamma_delta_q7_points.txt \>
7471 \> #F_alpha_beta_gamma_delta_q7_sweep.csv \>
7472 \> #F_alpha_beta_gamma_delta_q7_sweep4_15_data.csv \>
7473 \>
7474 \>
7475 \> surface_F_alpha_beta_gamma_delta_q7_recognize: \>
7476 \> $(\text{ORBITER}) -v 2 \>
7477 \> \> -define F -finite_field -q 49 -end \>
7478 \> \> -define P -projective_space -n 3 -field F -v 0 -end \>
7479 \> \> -define Control -W -end \>
7480 \> \> -with P -do \>
7481 \> \> -projective_space_activity \>
7482 \> \> -classify_surfaces_with_double_sixes Surf Control \>
7483 \> \> -end \>
7484 \>
670
-classification_of_cubic_surfaces_with_double_sixes_activity
-recognize 
-q 49 
-by_equation "F_alpha_beta_gamma_delta" 
-TF_{\alpha,\beta,\gamma,\delta}\"x0,x1,x2,x3" 
-(F_ALPHA_BETA_GAMMA_DELTA) 
"alpha=2,\beta=1,\gamma=1,\delta=2\" 
"Dalpha=2,\beta=1,\gamma=1,\delta=2D" 
-end 
-end 
-end 

7500  surface_49_recognize: 
7501  $(ORBITER) -v 3 
7502  -define F -finite_field -q 49 -end 
7503  -define P -projective_space -n 3 -field F -v 0 -end 
7504  -define Control -poset_classification_control -W -end 
7505  -with P -do 
7506  -projective_space_activity 
7507  -classify_surfaces_with_double_sixes Surf27 Control 
7508  -end 
7509  -with Surf27 -do 
7510  -classification_of_cubic_surfaces_with_double_sixes_activity 
7511  -recognize 
7512  -q 49 
7513  -by_coefficients "2,5,1,16,4,10,1,18,4,8,4,9,3,11,4,14" 
7514  -end 
7515  -end 
7516  -end 
7517  -print_symbols

7518
7519
7520
7521

7522  surface_McKean_15lines_q7: 
7523  $(ORBITER) -v 3 
7524  -define F -finite_field -q 7 -end 
7525  -define P -projective_space -n 3 -field F -v 0 -end 
7526  -define S -cubic_surface -space P 
7527  -by_coefficients $(SURFACE_MCKEAN_15_LINES) 
7528  -end 
7529  -with S -do 
7530  -cubic_surface_activity 
7531  -report 

671
7532 ▷ ▷ -end
7533 ▷ #pdflatex surface_by_coefficients_q7_report.tex
7534 ▷ #open surface_by_coefficients_q7_report.pdf
7535
7536 # 2 Eckardt points
7537
7538 surface_F_abcd_4433_q7:
7539 ▷ $(ORBITER) -v 3 \\n7540 ▷ ▷ -define F -finite_field -q 7 -end \\n7541 ▷ ▷ -define P -projective_space -n 3 -field F -v 0 -end \\n7542 ▷ ▷ -define S -cubic_surface -space P \\
7543 ▷ ▷ ▷ -by_equation \\n7544 ▷ ▷ ▷ "F_alpha_beta_gamma_delta" \\n7545 ▷ ▷ ▷ "DF_{\alpha,\beta,\gamma,\delta}\D" \\
7546 ▷ ▷ ▷ "x0,x1,x2,x3" \\
7547 ▷ ▷ ▷ $(F_ALPHA_BETA_GAMMA_DELTA) \\
7548 ▷ ▷ ▷ "alpha=4, beta=4, gamma=3, delta=3" \\
7549 ▷ ▷ ▷ "D\alpha=4, \beta=4, \gamma=3, \delta=3\D" \\
7550 ▷ ▷ ▷ -end \\
7551 ▷ ▷ ▷ -with S -do \\
7552 ▷ ▷ ▷ -cubic_surface_activity \\
7553 ▷ ▷ ▷ ▷ -report \\
7554 ▷ ▷ ▷ -end
7555 ▷ pdflatex surface_equation_F_alpha_beta_gamma_delta_q7_report.tex
7556 ▷ open surface_equation_F_alpha_beta_gamma_delta_q7_report.pdf
7557
7558 # the surface has 15 lines and 4 Eckardt points
7559
7560
7561
7562
7563
7564
7565 # Section 7.2: Cubic Surfaces and Quartic Curves
7566
7567
7568
7569 SECTION_CUBIC_SURFACES_AND_QUARTIC_CURVES:
7570
7571 test_7.2:
7572 ▷ make quartic_curve_9_0_report
7573 ▷ make quartic_curve_13_0_report
7574 ▷ make PG_2.13_rank_lines
7575 ▷ make PG_2.13_orbits_on_lines
7576 ▷ make quartic_curve_13_1_report
7577 ▷ make quartic_curves_19_report
7579 ▶ make_quartic_curve_q9_1
7580 ▶ make_quartic_curves_q7
7581 ▶ make_quartic_curves_q13
7582 ▶ make_quartic_curves_q13_combine
7583 ▶ make_quartic_curves_q13_classify
7584 ▶ make_quartic_curves_q19
7585 ▶ make_quartic_curves_q19_combine
7586 ▶ make_quartic_curves_q19_classify
7587
7588
7589 quartic_curve_9_0_report:
7590 ▶ $(ORBITER) -v 3 \n7591 ▶ ▶ -define F -finite_field -q 9 -end \n7592 ▶ ▶ -define P -projective_space -n 2 -field F -v 0 -end \n7593 ▶ ▶ -define C -quartic_curve -space P -catalogue 0 -end \n7594 ▶ ▶ ▶ -with C -do \n7595 ▶ ▶ ▶ ▶ -quartic_curve_activity \n7596 ▶ ▶ ▶ ▶ ▶ -report \n7597 ▶ ▶ ▶ -end
7598 ▶ ▶ pdflatex quartic_curve_catalogue_q9_iso0_report.tex
7599 ▶ ▶ open quartic_curve_catalogue_q9_iso0_report.pdf
7600
7601
7602
7603
7604 quartic_curve_13_0_report:
7605 ▶ $(ORBITER) -v 3 \n7606 ▶ ▶ -define F -finite_field -q 13 -end \n7607 ▶ ▶ -define P -projective_space -n 2 -field F -v 0 -end \n7608 ▶ ▶ -define C -quartic_curve -space P -catalogue 0 \n7609 ▶ ▶ ▶ -transform "10,4,1,11,5,11,4,1,11" \n7610 ▶ ▶ ▶ -transform_inverse "9,1,0,12,9,0,2,10,11" \n7611 ▶ ▶ ▶ -end \n7612 ▶ ▶ ▶ -with C -do \n7613 ▶ ▶ ▶ ▶ -quartic_curve_activity \n7614 ▶ ▶ ▶ ▶ ▶ -report \n7615 ▶ ▶ ▶ ▶ -end \n7616 ▶ ▶ ▶ -with C -do \n7617 ▶ ▶ ▶ ▶ -quartic_curve_activity \n7618 ▶ ▶ ▶ ▶ ▶ -extract_orbit_on_bitangents_by_length 4 \n7619 ▶ ▶ ▶ ▶ -end
7620 ▶ #pdflatex quartic_curve_catalogue_q13_iso0_report.tex
7621 ▶ #open quartic_curve_catalogue_q13_iso0_report.pdf
7622
7623 # 170, 111, 140, 2
7624
7625
PG_2.13_rank_lines:

$\text{(ORBITER) -v 2 ~}$

$\text{define v -vector -format 4 ~}$

$\text{dense "1,0,0, 0,1,0, 1,0,0, 0,0,1, 1,1,1, 0,1,0, 1,1,1, 0,0,1" ~}$

$\text{end ~}$

$\text{define F -finite_field -q 23 -end ~}$

$\text{define P -projective_space -n 2 -field F -v 0 -end ~}$

$\text{with P -do ~}$

$\text{define \text{projective_space_activity} ~}$

$\text{rank_lines_in_PG v ~}$

$\text{end ~}$

PG_2.13_orbits_on_lines:

$\text{(ORBITER) -v 5 ~}$

$\text{orbiter_path \text{(ORBITER_PATH) ~}}$

$\text{define Control -poset_classification_control ~}$

$\text{problem_label PGL_3.13 ~}$

$\text{depth 4 ~}$

$\text{draw_options -radius 200 -end ~}$

$\text{end ~}$

$\text{define G -linear_group -PGL 3 13 -end ~}$

$\text{define G_on_lines -modified_group -from G ~}$

$\text{on_k_subspaces 2 ~}$

$\text{end ~}$

$\text{define Orb -orbits -group G_on_lines ~}$

$\text{on_subsets 4 Control ~}$

$\text{end ~}$

$\text{with Orb -do -orbits_activity ~}$

$\text{recognize "170, 111, 140, 2" ~}$

$\text{recognize "0,23,24,47" ~}$

$\text{end ~}$

# pdflatex PGL_3.13.poset.tex

# open PGL_3.13.poset.pdf

# stabilizer of \{0,23,24,47\}

#1,0,0,7,9,0,9,5,3,

#1,3,0,12,0,10,9,2,

#1,11,7,9,6,10,5,2,

#1,4,11,12,1,8,10,4,2,

quartic_curve_13.1_report:

$\text{(ORBITER) -v 3 ~}$

$\text{define F -finite_field -q 13 -end ~}$

$\text{define P -projective_space -n 2 -field F -v 0 -end ~}$

$\text{define C -quartic_curve -space P -catalogue 1 -end ~}$

$\text{with C -do ~}$
\begin{verbatim}
7673 \textbf{\textit{quartic_curve_activity}} \end{verbatim}

\begin{verbatim}
7673 \textbf{\textit{-quartic_curve_activity}} \end{verbatim}

\begin{verbatim}
7674 \textbf{\textit{-report}} \end{verbatim}

\begin{verbatim}
7675 \textbf{\textit{-end}} \end{verbatim}

\begin{verbatim}
7676 pdflatex quartic_curve_catalogue_q13_iso1_report.tex
7677 open quartic_curve_catalogue_q13_iso1_report.pdf
7678
7679
7680 NB_QUARTIC_CURVES_Q19=14
7681
7682 quartic_curves.19_report:
7683 \textbf{\textit{\$\{ORBITER\} -v 3}} \end{verbatim}

\begin{verbatim}
7684 \textbf{\textit{-define F -finite_field -q 19 -end}} \end{verbatim}

\begin{verbatim}
7685 \textbf{\textit{-define P -projective_space -n 2 -field F -v 0 -end}} \end{verbatim}

\begin{verbatim}
7686 \textbf{\textit{-define L 0 \$(NB_QUARTIC_CURVES_Q19) 1}} \end{verbatim}

\begin{verbatim}
7687 \textbf{\textit{-define C -quartic_curve -space P -catalogue \%L -end}} \end{verbatim}

\begin{verbatim}
7688 \textbf{\textit{-with C -do}} \end{verbatim}

\begin{verbatim}
7689 \textbf{\textit{quartic_curve_activity}} \end{verbatim}

\begin{verbatim}
7690 \textbf{\textit{-report}} \end{verbatim}

\begin{verbatim}
7691 \textbf{\textit{-end}} \end{verbatim}

\begin{verbatim}
7692 \textbf{\textit{-end_loop}} \end{verbatim}

\begin{verbatim}
7693 pdflatex quartic_curve_catalogue_q19_iso0_report.tex
7694 pdflatex quartic_curve_catalogue_q19_iso1_report.tex
7695 pdflatex quartic_curve_catalogue_q19_iso2_report.tex
7696 pdflatex quartic_curve_catalogue_q19_iso3_report.tex
7697 pdflatex quartic_curve_catalogue_q19_iso4_report.tex
7698 pdflatex quartic_curve_catalogue_q19_iso5_report.tex
7699 pdflatex quartic_curve_catalogue_q19_iso6_report.tex
7700 pdflatex quartic_curve_catalogue_q19_iso7_report.tex
7701 pdflatex quartic_curve_catalogue_q19_iso8_report.tex
7702 pdflatex quartic_curve_catalogue_q19_iso9_report.tex
7703 pdflatex quartic_curve_catalogue_q19_iso10_report.tex
7704 pdflatex quartic_curve_catalogue_q19_iso11_report.tex
7705 pdflatex quartic_curve_catalogue_q19_iso12_report.tex
7706 pdflatex quartic_curve_catalogue_q19_iso13_report.tex
7707
7708
7709 quartic_curve_q9.1:
7710 \textbf{\textit{\$\{ORBITER\}/oribiter.out -v 3}} \end{verbatim}

\begin{verbatim}
7712 \textbf{\textit{-define F -finite_field -q 9 -end}} \end{verbatim}

\begin{verbatim}
7713 \textbf{\textit{-define P2 -projective_space -n 2 -field F -end}} \end{verbatim}

\begin{verbatim}
7714 \textbf{\textit{-define P3 -projective_space -n 3 -field F -end}} \end{verbatim}

\begin{verbatim}
7715 \textbf{\textit{-define S9.1 -cubic_surface -space P3 -catalogue 1 -end}} \end{verbatim}

\begin{verbatim}
7716 \textbf{\textit{-define C -quartic_curve -space P2 -from_cubic_surface S9.1 0 -end}} \end{verbatim}

\begin{verbatim}
7717 \textbf{\textit{-with C -do}} \end{verbatim}

\begin{verbatim}
7718 \textbf{\textit{quartic_curve_activity}} \end{verbatim}

\begin{verbatim}
7719 \textbf{\textit{-report}} \end{verbatim}

\end{verbatim}
#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 )

#eqn15:

# the 27 single points (of the surface) are:

NB_CUBIC_SURFACES_Q7=1

quartic_curves_q7:

quartic_curves_q13:

NB_CUBIC_SURFACES_Q13=4
\texttt{L} -loop L 0 $(\texttt{NB\_CUBIC\_SURFACES\_Q13}) 1 \ \\
\texttt{L} -with S_L -do \ \\
\texttt{L} -cubic_surface_activity \ \\
\texttt{L} -export_all_quartic_curves \ \\
\texttt{L} -end \ \\
\texttt{L} -end_loop \ \\
\texttt{L} -print_symbols \\
\texttt{L} quartic_curves_q13\_combine: \\
\texttt{L} $(\texttt{ORBITER\_PATH})\texttt{orbiter.out} -v 3 \ \\
\texttt{L} -csv_file_concatenate_from_mask $(\texttt{NB\_CUBIC\_SURFACES\_Q13}) \ \\
\texttt{L} -surface_catalogue_q13_iso%ld\_quartics.csv \ \\
\texttt{L} -quartics_q13.csv \\
\texttt{L} quartic_curves_q13\_classify: \\
\texttt{L} $(\texttt{ORBITER}) -v 3 \ \\
\texttt{L} -define F -finite_field -q 13 -end \ \\
\texttt{L} -define P -projective_space -n 2 -field F -v 0 -end \ \\
\texttt{L} -with P -do \ \\
\texttt{L} -projective_space_activity \ \\
\texttt{L} -classify_quartic_curves_with_substructure \ \\
\texttt{L} -quartics_q13.csv \ \\
\texttt{L} -quartic_curves_q13 \\
\texttt{L} -end \ \\
\texttt{L} -print_symbols \ \\
\texttt{L} NB\_CUBIC\_SURFACES\_Q19=10 \ \\
\texttt{L} quartic_curves_q19: \\
\texttt{L} $(\texttt{ORBITER\_PATH})\texttt{orbiter.out} -v 3 \ \\
\texttt{L} -define F -finite_field -q 19 -end \ \\
\texttt{L} -define P -projective_space -n 3 -field F -end \ \\
\texttt{L} -loop L 0 $(\texttt{NB\_CUBIC\_SURFACES\_Q19}) 1 \ \\
\texttt{L} -define S_L -cubic_surface -space P -catalogue %L -end \ \\
\texttt{L} -end_loop \ \\
\texttt{L} -print_symbols \ \\
\texttt{L} -loop L 0 $(\texttt{NB\_CUBIC\_SURFACES\_Q19}) 1 \ \\
\texttt{L} -with S_L -do \ \\
\texttt{L} -cubic_surface_activity \ \\
\texttt{L} -export_all_quartic_curves \ \\
\texttt{L} -end \ \\
\texttt{L} -end_loop \ \\
\texttt{L} -print_symbols \\
\texttt{L}
quartic_curves_q19_combine:
  $ (ORBITER_PATH) orbiter.out -v 3 \
  $csv_file_concatenate_from_mask $(NB_CUBIC_SURFACES_Q19) \
  surface_catalogue_q19_iso\ld_quartics.csv \
  quartics_q19.csv

quartic_curves_q19_classify:
  $ (ORBITER) -v 3 \
  -define F -finite_field -q 19 -end \n  -define P -projective_space -n 2 -field F -v 0 -end \
  -with P -do \
  -projective_space_activity \n  -classify_quartic_curves_with_substructure \n  quartics_q19.csv \n  1 4 4 quartic_curves_q19 \
  -end \
  -print_symbols

SECTION_CLASSIFICATION_OF_CUBIC_SURFACES_WITH_27_LINES:

# Section 7.3: Classification of Cubic Surfaces with 27 lines

test_7.3:
  make surface_classify_q4
  make surface_classify_q4_arc_lifting_two_lines
  make surface_classify_q7
  make surface_classify_q9
  make surface_classify_q13

surface_classify_q4:
  $ (ORBITER) -v 5 \
  -define F -finite_field -q 4 -end \
  -define P -projective_space -n 3 -field F -v 0 -end \
  -define Control -poset_classification_control -W -end \
  -with P -do \

678
surface_classify_q4.arc_lifting_two_lines:

$\text{(ORBITER)} -v 10$

$\text{define F -finite_field -q 4 -end}$

$\text{define P -projective_space -n 3 -field F -v 0 -end}$

$\text{with P -do}$

$\text{-projective_space_activity}$

$\text{control_six_arcs -problem_label six_arcs_q4 -end}$

$\text{-classification_of_cubic_surfaces_through_arcs_and_two_lines}$

$\text{-end}$

pdflatex surfaces.arc_lifting_4.tex

open surfaces.arc_lifting_4.pdf

surface_classify_q7:

$\text{(ORBITER)} -v 5$

$\text{define F -finite_field -q 7 -end}$

$\text{define P -projective_space -n 3 -field F -v 0 -end}$

$\text{define Control -poset_classification_control -W -end}$

$\text{with P -do}$

$\text{-projective_space_activity}$

$\text{-classify_surfaces_with_double_sixes Surf27 Control}$

$\text{-end}$

$\text{-with Surf27 -do}$

$\text{-classification_of_cubic_surfaces_with_double_sixes_activity}$

$\text{-report -end}$

$\text{-end}$

$\text{-print_symbols}$

pdflatex Surfaces.q7.tex

open Surfaces.q7.pdf
surface_classify_q9:
$$(\text{ORBITER}) -v 5 \backslash$
\begin{verbatim}
> > -define F -finite_field -q 9 -end \n> > -define P -projective_space -n 3 -field F -v 0 -end \n> > -define Control -poset_classification_control -W -end \n> > -with P -do \n> > -projective_space_activity \n> > -classify_surfaces_with_double_sixes Surf27 Control \n> > -end \n> > -with Surf27 -do \n> > -classification_of_cubic_surfaces_with_double_sixes_activity \n> > -report -end \n> > -end \n> > -print_symbols
\end{verbatim}

\texttt{pdflatex Surfaces_q9.tex}
\texttt{open Surfaces_q9.pdf}

surface_classify_q13:
$$(\text{ORBITER}) -v 5 \backslash$$
\begin{verbatim}
> > -define F -finite_field -q 13 -end \n> > -define P -projective_space -n 3 -field F -v 0 -end \n> > -define Control -poset_classification_control -W -end \n> > -with P -do \n> > -projective_space_activity \n> > -classify_surfaces_with_double_sixes C Control \n> > -end \n> > -with C -do \n> > -classification_of_cubic_surfaces_with_double_sixes_activity \n> > -report -end \n> > -end \n> > -print_symbols
\end{verbatim}

\texttt{pdflatex Surfaces_q13.tex}
\texttt{open Surfaces_q13.pdf}

# Section 7.4: Cubic Surfaces - Isomorphism Testing and Recognition
7953
7954  test_7_4:
7955  ▷ make surface_recognize_q7_abcd_2_3_3_4
7956  ▷ make surface_isomorph_16
7957  ▷ make surface_recognize_8
7958  ▷ make surface_recognize_F13_q4
7959  ▷ make surface_sweep_Cayley_13
7960  ▷ make F_sweep_15_q7
7961
7962
7963
7964
7965
7966  #ToDo:
7967
7968  surface_recognize_q7_abcd_2_3_3_4:
7969  ▷ $(ORBITER) -v 3 \
7970  ▷ ▷ -define F -finite_field -q 7 -end \
7971  ▷ ▷ -define P -projective_space -n 3 -field F -v 0 -end \
7972  ▷ ▷ -define Control -poset_classification_control -W -end \
7973  ▷ ▷ -with P -do \ 
7974  ▷ ▷ -projective_space_activity \ 
7975  ▷ ▷ ▷ -classify_surfaces_with_double_sixes Surf Control \ 
7976  ▷ ▷ ▷ -end \ 
7977  ▷ ▷ ▷ -with Surf -do \ 
7978  ▷ ▷ ▷ ▷ -classification_of_cubic_surfaces_with_double_sixes_activity \ 
7979  ▷ ▷ ▷ ▷ -recognize \ 
7980  ▷ ▷ ▷ ▷ ▷ -space P -family_general_abcd 2 3 3 4 -end \ 
7981  ▷ ▷ ▷ ▷ -end
7982
7983
7984  surface_isomorph_16:
7985  ▷ $(ORBITER) -v 3 \
7986  ▷ ▷ -define F -finite_field -q 16 -end \
7987  ▷ ▷ -define P -projective_space -n 3 -field F -v 0 -end \
7988  ▷ ▷ -define Control -poset_classification_control -W -end \
7989  ▷ ▷ -with P -do \ 
7990  ▷ ▷ -projective_space_activity \ 
7991  ▷ ▷ ▷ -classify_surfaces_with_double_sixes Surf27 Control \ 
7992  ▷ ▷ ▷ -end \ 
7993  ▷ ▷ ▷ -with Surf27 -do \ 
7994  ▷ ▷ ▷ ▷ -classification_of_cubic_surfaces_with_double_sixes_activity \ 
7995  ▷ ▷ ▷ ▷ -isomorphism_testing \ 
7996  ▷ ▷ ▷ ▷ ▷ -space P -by_coefficients \ 
7997  ▷ ▷ ▷ ▷ ▷ ▷ "1,5,1,8,1,9,1,10,1,11,1,12,6,14,6,15,7,18,7,19" -end \ 
7998  ▷ ▷ ▷ ▷ ▷ ▷ -space P -by_coefficients \ 
7999  ▷ ▷ ▷ ▷ ▷ ▷ "13,6,3,8,3,11,13,13,1,19" -end \
surface_recognize_8:
$\text{(ORBITER) -v 3 \ }
\text{-define F -finite_field -q 8 -end \ }
\text{-define P -projective_space -n 3 -field F -v 0 -end \ }
\text{-define Control -poset_classification_control -W -end \ }
\text{-with P -do \ }
\text{-projective_space_activity \ }
\text{-classify_surfaces_with_double_sixes Surf27 Control \ }
\text{-end \ }
\text{-with Surf27 -do \ }
\text{-classification_of_cubic_surfaces_with_double_sixes_activity \ }
\text{-recognize \ }
\text{-space P \ }
\text{-by.coefficients "1,6,1,8,1,11,1,13,1,19" \ }
\text{-end \ }
\text{-end \ }
\text{-print_symbols \ }

surface_recognize_F13_q4:
\$\text{(ORBITER) -v 3 \ }
\text{-define F -finite_field -q 4 -end \ }
\text{-define P -projective_space -n 3 -field F -v 0 -end \ }
\text{-define Control -poset_classification_control -W -end \ }
\text{-with P -do \ }
\text{-projective_space_activity \ }
\text{-classify_surfaces_with_double_sixes Surf27 Control \ }
\text{-end \ }
\text{-with Surf27 -do \ }
\text{-classification_of_cubic_surfaces_with_double_sixes_activity \ }
\text{-identify_F13 \ }
\text{-end \ }
\text{-print_symbols \ }

surface_sweep_Cayley_13:
\$\text{(ORBITER) -v 3 \ }
\text{-define F -finite_field -q 13 -end \ }
# Section 7.5: Cubic Surfaces of Dickson type

---

test_7_5:
8094 ▷ make orbits_cubic_surfaces_q3
8095 ▷ make orbits_cubic_curves_q2_again
8096 ▷ make orbits_cubic_curves_q3
8097 ▷ make poly_orbits_d3_n3_q2_F2.csv
8098 ▷ make Dickson_q2.analyze
8099 ▷ make poly_orbits_d3_n3_q2_F4.csv
8100 ▷ make Dickson_q4.analyze
8101 ▷ make poly_orbits_d3_n3_q2_F8.csv
8102 ▷ make Dickson_q8.analyze
8103 ▷ make poly_orbits_d3_n3_q2_F16.csv
8104 ▷ make Dickson_q16.analyze
8105
8106
8107
8108
8109 orbits_cubic_surfaces_q3:
8110 ▷ $(ORBITER) -v 4 \
8111 ▷ ▷ -define G -linear_group -PGL 4 3 -end \n8112 ▷ ▷ -define Orb -orbits -group G \n8113 ▷ ▷ ▷ -on_polynomials 3 \n8114 ▷ ▷ -end
8115 ▷ #pdflatex poly_orbits_d3_n3_q3.tex
8116 ▷ #open poly_orbits_d3_n3_q3.pdf
8117 ▷
8118 # this takes 3 days and about 150 GB memory on ripoff
8119
8120 orbits_cubic_curves_q2_again:
8121 ▷ $(ORBITER) -v 4 \
8122 ▷ ▷ -define G \n8123 ▷ ▷ -linear_group -PGL 3 2 \n8124 ▷ ▷ -end \n8125 ▷ ▷ -define Orb -orbits -group G \n8126 ▷ ▷ ▷ -on_polynomials 3 \n8127 ▷ ▷ -end
8128 ▷ #pdflatex poly_orbits_d3_n2_q2.tex
8129 ▷ #open poly_orbits_d3_n2_q2.pdf
8130
8131 orbits_cubic_curves_q3:
8132 ▷ $(ORBITER) -v 4 \
8133 ▷ ▷ -define G \n8134 ▷ ▷ -linear_group -PGL 3 3 \n8135 ▷ ▷ -end \n8136 ▷ ▷ -define Orb -orbits -group G \n8137 ▷ ▷ ▷ -on_polynomials 3 \n8138 ▷ ▷ -end
8139 ▷ #pdflatex poly_orbits_d3_n2_q3.tex
8140 ▷ #open poly_orbits_d3_n2_q3.pdf
# compute and analyze properties over $F_2$

```
poly_orbits_d3_n3_q2_F2.csv: poly_orbits_d3_n3_q2.csv

$(ORBITER)$ -v 4 \
-define F -finite_field -q 2 -end \
-define P -projective_space -n 3 -field F -v 0 -end \
-with P -do \
-projective_space_activity \
-table_of_cubic_surfaces_compute_properties \
-poly_orbits_d3_n3_q2.csv 2 0 \
-end
```

Dickson_q2_analyze: poly_orbits_d3_n3_q2_F2.csv

```
poly_orbits_d3_n3_q2_F2.csv: poly_orbits_d3_n3_q2.csv

$(ORBITER)$ -v 4 \
-define F -finite_field -q 2 -end \
-define P -projective_space -n 3 -field F -v 0 -end \
-with P -do \
-projective_space_activity \
-cubic_surface_properties_analyze \
-poly_orbits_d3_n3_q2_F2.csv 2 \
-end
```

pdflatex poly_orbits_d3_n3_q2_F2_report.tex

open poly_orbits_d3_n3_q2_F2_report.pdf

# compute and analyze properties over $F_4$

```
poly_orbits_d3_n3_q2_F4.csv: poly_orbits_d3_n3_q2.csv

$(ORBITER)$ -v 4 \
-define F -finite_field -q 4 -end \
-define P -projective_space -n 3 -field F -v 0 -end \
-with P -do \
-projective_space_activity \
-table_of_cubic_surfaces_compute_properties \
-poly_orbits_d3_n3_q2.csv 2 0 \
-end
```

Dickson_q4_analyze: poly_orbits_d3_n3_q2_F4.csv

```
poly_orbits_d3_n3_q2_F4.csv: poly_orbits_d3_n3_q2.csv

$(ORBITER)$ -v 4 \
-define F -finite_field -q 4 -end \
-define P -projective_space -n 3 -field F -v 0 -end \
-with P -do \
-projective_space_activity \
-cubic_surface_properties_analyze \
```
8188  ➤  ➤  ➤  poly_orbits_d3_n3_q2_F4.csv 2 \\
8189  ➤  ➤  -end
8190  ➤  pdflatex poly_orbits_d3_n3_q2_F4_report.tex
8191  ➤  open poly_orbits_d3_n3_q2_F4_report.pdf
8192
8193  # compute and analyze properties over F8
8194
8195  poly_orbits_d3_n3_q2_F8.csv: poly_orbits_d3_n3_q2.csv
8196  ➤  $(ORBITER) -v 4 \\
8197  ➤  ➤  -define F -finite_field -q 8 -end \\
8198  ➤  ➤  -define P -projective_space -n 3 -field F -v 0 -end \\
8199  ➤  ➤  -with P -do \\
8200  ➤  ➤  -projective_space_activity \\
8201  ➤  ➤  -table_of_cubic_surfaces.compute.properties \\
8202  ➤  ➤  ➤  poly_orbits_d3_n3_q2.csv 2 0 \\
8203  ➤  ➤  -end
8204
8205  Dickson_q8.analyze: poly_orbits_d3_n3_q2_F8.csv
8206  ➤  $(ORBITER) -v 4 \\
8207  ➤  ➤  -define F -finite_field -q 8 -end \\
8208  ➤  ➤  -define P -projective_space -n 3 -field F -v 0 -end \\
8209  ➤  ➤  -with P -do \\
8210  ➤  ➤  -projective_space_activity \\
8211  ➤  ➤  -cubic_surface_properties.analyze \\
8212  ➤  ➤  ➤  poly_orbits_d3_n3_q2_F8.csv 2 \\
8213  ➤  ➤  -end
8214  ➤  pdflatex poly_orbits_d3_n3_q2_F8_report.tex
8215  ➤  open poly_orbits_d3_n3_q2_F8_report.pdf
8216
8217
8218  # compute and analyze properties over F16
8219
8220  poly_orbits_d3_n3_q2_F16.csv: poly_orbits_d3_n3_q2.csv
8221  ➤  $(ORBITER) -v 4 \\
8222  ➤  ➤  -define F -finite_field -q 16 -end \\
8223  ➤  ➤  -define P -projective_space -n 3 -field F -v 0 -end \\
8224  ➤  ➤  -with P -do \\
8225  ➤  ➤  -projective_space_activity \\
8226  ➤  ➤  -table_of_cubic_surfaces.compute.properties \\
8227  ➤  ➤  ➤  poly_orbits_d3_n3_q2.csv 2 0 \\
8228  ➤  ➤  -end
8229
8230
8231  Dickson_q16.analyze: poly_orbits_d3_n3_q2_F16.csv
8232  ➤  $(ORBITER) -v 4 \\
8233  ➤  ➤  -define F -finite_field -q 16 -end \\
8234  ➤  ➤  -define P -projective_space -n 3 -field F -v 0 -end \\

686
Section 7.6: Cubic Surfaces - ATLAS and Tables

Section_CUBIC_SURFACES_ATLAS_AND_TABLES:

test_7.6:

- make cubic_surfaces.tables.17
- make cubic_surfaces_table_latex.17
- make cubic_surfaces_tables_up_to_17
- make cubic_surfaces_tables_19_37
- make cubic_surfaces_tables_41_and_up
- make cubic_surfaces_tables_latex
- make cubic_surfaces_tables_latex_big
- make surface_table
- make surface_atlas
- make surface_reports
- make quartic_curve_tables.23
- make quartic_curve_tables
- make quartic_curve_tables_latex

MAKE_TABLE_OF_CUBIC_SURFACES=-define \n- P -projective_space -n 3 -field F -v 0 -end \n- with P -do \n- -projective_space_activity \n- -table_of_cubic_surfaces \n- -end

cubic_surfaces.tables.17:

$ (ORBITER) -v 3 \n$ (MAKE_TABLE_OF_CUBIC_SURFACES)

cubic_surfaces_table_latex.17:
cubic_surfaces_tables_up_to_17:

```bash
$(ORBITER) -v 3 -define F -finite_field -q 4 -end $(MAKE_TABLE_OF_CUBIC_SURFACE S)
```

```bash
$(ORBITER) -v 3 -define F -finite_field -q 7 -end $(MAKE_TABLE_OF_CUBIC_SURFACE S)
```

```bash
$(ORBITER) -v 3 -define F -finite_field -q 8 -end $(MAKE_TABLE_OF_CUBIC_SURFACE S)
```

```bash
$(ORBITER) -v 3 -define F -finite_field -q 9 -end $(MAKE_TABLE_OF_CUBIC_SURFACE S)
```

```bash
$(ORBITER) -v 3 -define F -finite_field -q 11 -end $(MAKE_TABLE_OF_CUBIC_SURFACE S)
```

```bash
$(ORBITER) -v 3 -define F -finite_field -q 13 -end $(MAKE_TABLE_OF_CUBIC_SURFACE S)
```

```bash
$(ORBITER) -v 3 -define F -finite_field -q 16 -end $(MAKE_TABLE_OF_CUBIC_SURFACE S)
```

```bash
$(ORBITER) -v 3 -define F -finite_field -q 17 -end $(MAKE_TABLE_OF_CUBIC_SURFACE S)
```

cubic_surfaces_tables_19_37:

```bash
$(ORBITER) -v 3 -define F -finite_field -q 19 -end $(MAKE_TABLE_OF_CUBIC_SURFACE S)
```

```bash
$(ORBITER) -v 3 -define F -finite_field -q 23 -end $(MAKE_TABLE_OF_CUBIC_SURFACE S)
```

```bash
$(ORBITER) -v 3 -define F -finite_field -q 25 -end $(MAKE_TABLE_OF_CUBIC_SURFACE S)
```

```bash
$(ORBITER) -v 3 -define F -finite_field -q 27 -end $(MAKE_TABLE_OF_CUBIC_SURFACE S)
```

```bash
$(ORBITER) -v 3 -define F -finite_field -q 29 -end $(MAKE_TABLE_OF_CUBIC_SURFACE S)
```

```bash
$(ORBITER) -v 3 -define F -finite_field -q 31 -end $(MAKE_TABLE_OF_CUBIC_SURFACE S)
```

```bash
$(ORBITER) -v 3 -define F -finite_field -q 32 -end $(MAKE_TABLE_OF_CUBIC_SURFACE S)
```

```bash
$(ORBITER) -v 3 -define F -finite_field -q 37 -end $(MAKE_TABLE_OF_CUBIC_SURFACE S)
```

cubic_surfaces_tables_41_and_up:

```bash
$(ORBITER) -v 3 -define F -finite_field -q 41 -end $(MAKE_TABLE_OF_CUBIC_SURFACE S)
```

```bash
$(ORBITER) -v 3 -define F -finite_field -q 43 -end $(MAKE_TABLE_OF_CUBIC_SURFACE S)
```
\begin{verbatim}
8311 \$(ORBITER) -v 3 -define F -finite_field -q 47 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
8312 \$(ORBITER) -v 3 -define F -finite_field -q 49 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
8313 \$(ORBITER) -v 3 -define F -finite_field -q 53 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
8314 \$(ORBITER) -v 3 -define F -finite_field -q 59 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
8315 \$(ORBITER) -v 3 -define F -finite_field -q 61 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
8316 \$(ORBITER) -v 3 -define F -finite_field -q 64 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
8317 \$(ORBITER) -v 3 -define F -finite_field -q 67 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
8318 \$(ORBITER) -v 3 -define F -finite_field -q 71 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
8319 \$(ORBITER) -v 3 -define F -finite_field -q 73 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
8320 \$(ORBITER) -v 3 -define F -finite_field -q 79 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
8321 \$(ORBITER) -v 3 -define F -finite_field -q 81 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
8322 \$(ORBITER) -v 3 -define F -finite_field -q 83 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
8323 \$(ORBITER) -v 3 -define F -finite_field -q 89 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
8324 \$(ORBITER) -v 3 -define F -finite_field -q 97 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
8325 \$(ORBITER) -v 3 -define F -finite_field -q 101 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
8326 \$(ORBITER) -v 3 -define F -finite_field -q 103 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
8327 \$(ORBITER) -v 3 -define F -finite_field -q 107 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
8328 \$(ORBITER) -v 3 -define F -finite_field -q 109 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
8329 \$(ORBITER) -v 3 -define F -finite_field -q 113 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
8330 \$(ORBITER) -v 3 -define F -finite_field -q 121 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
8331 \$(ORBITER) -v 3 -define F -finite_field -q 127 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
8332 \$(ORBITER) -v 3 -define F -finite_field -q 128 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
8333 8334
8335 cubic_surfaces_tables_latex:
\end{verbatim}
cubic_surfaces_tables latex_big:

```
$ORBITER$ -v 3 -make_table_of_surfaces
```

```
pdflatex surfaces_report.tex
open surfaces_report.pdf
```

surface_atlas:

```
$ORBITER$ -v 3 -create_surface_atlas 97
```

```
~/bin/tth surface_atlas.tex
```

surface_reports:

```
$ORBITER$ -v 3 \\\n- orbiter_path $ORBITER\_PATH$ -create_surface_reports 4,7,8,9,11
```

```
quartic_curve_tables_19:
  $(ORBITER) -v 3 \ 
  -define F -finite_field -q 19 -end \ 
  -define P -projective_space -n 2 -field F -v 0 -end \ 
  -with P -do \ 
  -projective_space_activity \ 
  -table_of_quartic_curves \ 
  -end

quartic_curve_tables_23:
  $(ORBITER) -v 3 \ 
  -define F -finite_field -q 23 -end \ 
  -define P -projective_space -n 2 -field F -v 0 -end \ 
  -with P -do \ 
  -projective_space_activity \ 
  -table_of_quartic_curves \ 
  -end

quartic_curve_tables:
  $(ORBITER) -v 3 \ 
  -define F -finite_field -q 9 -end \ 
  -define P -projective_space -n 2 -field F -v 0 -end \ 
  -with P -do \ 
  -projective_space_activity \ 
  -table_of_quartic_curves \ 
  -end

  $(ORBITER) -v 3 \ 
  -define F -finite_field -q 13 -end \ 
  -define P -projective_space -n 2 -field F -v 0 -end \ 
  -with P -do \ 
  -projective_space_activity \ 
  -table_of_quartic_curves \ 
  -end

  $(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
$(ORBITER) -v 3 \
$define F - finite_field -q 19 -end \
$define P - projective_space -n 2 - field F - v 0 - end \
-with P -do \
-projective_space_activity \
-table_of_quartic_curves \
-end

$(ORBITER) -v 3 \
$define F - finite_field -q 23 -end \
$define P - projective_space -n 2 - field F - v 0 - end \
-with P -do \
-projective_space_activity \
-table_of_quartic_curves \
-end

$(ORBITER) -v 3 \
$define F - finite_field -q 25 -end \
$define P - projective_space -n 2 - field F - v 0 - end \
-with P -do \
-projective_space_activity \
-table_of_quartic_curves \
-end

$(ORBITER) -v 3 \
$define F - finite_field -q 27 -end \
$define P - projective_space -n 2 - field F - v 0 - end \
-with P -do \
-projective_space_activity \
-table_of_quartic_curves \
-end

$(ORBITER) -v 3 \
$define F - finite_field -q 29 -end \
$define P - projective_space -n 2 - field F - v 0 - end \
-with P -do \
-projective_space_activity \
-table_of_quartic_curves \
-end

$(ORBITER) -v 3 \
$define F - finite_field -q 31 -end \
$define P - projective_space -n 2 - field F - v 0 - end \
-with P -do \
-projective_space_activity \
-table_of_quartic_curves \
-end

#quartic_curves_q9.info.csv
#quartic_curves_q13.info.csv
#quartic_curves_q17.info.csv
quartic_curve_tables latex:

# $(ORBITER) -v 3 -csv_file_latex 1 test.csv
# $(ORBITER) -v 3 -csv_file_latex 0 quartic_curves_q13_info.csv
# $(ORBITER) -v 3 -csv_file_latex 0 quartic_curves_q17_info.csv
# $(ORBITER) -v 3 -csv_file_latex 0 quartic_curves_q23_info.csv
# $(ORBITER) -v 3 -csv_file_latex 0 quartic_curves_q25_info.csv
# $(ORBITER) -v 3 -csv_file_latex 0 quartic_curves_q27_info.csv
# $(ORBITER) -v 3 -csv_file_latex 0 quartic_curves_q29_info.csv
# $(ORBITER) -v 3 -csv_file_latex 0 quartic_curves_q31_info.csv
# $(ORBITER) -v 3 -csv_file_latex 1 quartic_curves_q9_info.csv

# pdflatex quartic_curves_q13_info.tex
# open quartic_curves_q13_info.pdf
#/bin/tth quartic_curves_q13_info.tex
# open quartic_curves_q13_info.html

# 9#0 has K=63 and ago=12096
# 23#40 has K=21 and ago=168
# 25#44 has K=21 and ago=336
# 29#121 has K=21 and ago=168

# Chapter 8 - Ring Theory
# Section 8.1: Polynomials over Finite Fields
SECTION_POLYNOMIALS:
test_8.1:
make sift_polynomials_deg3_q2
make sift_polynomials_deg4_q2
8524 ▷ make_d_poly_division
8525 ▷ make_d_poly_division2
8526 ▷ make_d_poly_gcd
8527 ▷ make_d_poly_multiplication_mod1
8528 ▷ make_d_poly_multiplication_mod2
8529 ▷ make_d_poly_multiplication_mod_F4
8530 ▷ make_d_mult_polynomials_2_5_7
8531 ▷ make_d_polynomial_division_ranked_2_27_13
8532 ▷ make_d_mult_polynomials_2_8_15
8533 ▷ make_d_polynomial_division_ranked_2_120_25
8534 ▷ make_d_mult_polynomials_2_7_7
8535 ▷ make_d_mult_polynomials_2_4_6
8536 ▷ make_d_polynomial_division_ranked_2_24_13
8537 ▷ make_d_mult_polynomials_1024_999_997
8538 ▷ make_d_polynomial_division_ranked_2_349147_1033
8539 ▷ make_d_mult_polynomials_1024_999_997_check
8540 ▷ make_d_mult_polynomials_17_12
8541 ▷ make_d_polynomial_division_ranked_2_204_37
8542 ▷ make_d_test_crc32
8543 ▷ make_d_Berlekamp_matrix_crc32
8544 ▷ make_d_power_mod_inverse
8545 ▷ make_d_mult_mod_to_get_one
8546 ▷ make_d_Berlekamp_matrix_2_3
8547 ▷ make_d_Berlekamp_matrix_2_4
8548 ▷ make_d_Berlekamp_matrix_4_3a
8549 ▷ make_d_Berlekamp_matrix_4_3b
8550 ▷ make_d_find_roots_a
8551 ▷ make_d_find_roots_b
8552 ▷ make_d_find_roots_c
8553 ▷ make_d_find_roots_d
8554 ▷ make_d_find_roots_e
8555 ▷ make_d_roots_over_F2
8556 ▷ make_d_roots_over_F8
8557 ▷ make_d_irred_3_2
8558 ▷ make_d_irred_4_2
8559 ▷ make_d_irred_5_2
8560 ▷ make_d_irred_6_2
8561 ▷ make_d_irred_7_2
8562 ▷ make_d_irred_8_2
8563 ▷ make_d_irred_9_2
8564 ▷ make_d_irred_10_2
8565 ▷ make_d_irred_2_4
8566 ▷ make_d_irred_3_4
8567 ▷ make_d_search_primitive_poly_2
8568 ▷ make_d_search_primitive_poly_3
8569 ▷ make_d_search_primitive_poly_4
8570 ▷ make_d_search_primitive_poly_5
8571  ▷ make search_primitive_poly_7
8572  ▷ make search_primitive_poly_8
8573  ▷ make search_primitive_poly_9
8574  ▷ make search_primitive_poly_11
8575  ▷ make search_primitive_poly_13
8576  ▷ make search_primitive_poly_degree_16
8577  ▷ make search_primitive_poly_32
8578
8579
8580  # check which polynomials are irreducible and which are primitive:
8581
8582  sift_polynomials_deg3_q2:
8583  ▷ $(ORBITER) -v 2 \
8584  ▷ ▷ -define F -finite_field -q 2 -end \
8585  ▷ ▷ -with F -do \
8586  ▷ ▷ -finite_field_activity -sift_polynomials 8 16 -end
8587
8588
8589
8590  sift_polynomials_deg4_q2:
8591  ▷ $(ORBITER) -v 2 \
8592  ▷ ▷ -define F -finite_field -q 2 -end \
8593  ▷ ▷ -with F -do \
8594  ▷ ▷ -finite_field_activity -sift_polynomials 16 32 -end
8595
8596
8597
8598
8599  poly_division:
8600  ▷ $(ORBITER) -v 2 \
8601  ▷ ▷ -define F -finite_field -q 2 -end \
8602  ▷ ▷ -with F -do \
8603  ▷ ▷ -finite_field_activity \
8604  ▷ ▷ -polynomial_division "1,0,0,0,0,0,0,0,0,0,1" "1,0,1,1" -end
8605
8606  poly_division2:
8607  ▷ $(ORBITER) -v 2 \
8608  ▷ ▷ -define F -finite_field -q 2 -end \
8609  ▷ ▷ -define A -vector -field F -sparse 11 "1,0,1,10" -end \
8610  ▷ ▷ -define B -vector -field F -dense "1,0,1,1" -end \
8611  ▷ ▷ -with F -do \
8612  ▷ ▷ -finite_field_activity \
8613  ▷ ▷ -polynomial_division A B -end
8614
8615
8616  poly_gcd:
8617  ▷ $(ORBITER) -v 2 \

695
-define F -finite_field -q 2 -end \n-with F -do \n-finite_field_activity \n-extended_gcd_for_polynomials "1,0,0,0,0,0,0,0,0,0,1" "1,0,1,1" -end

poly_mult_mod1:
$\text{(ORBITER)} -v 2 \n-def F -finite_field -q 7 -end \n-with F -do \n-finite_field_activity \n-polynomial_mult_mod "1,2,3" "3,4,5" "6,0,0,1" -end

poly_mult_mod2:
$\text{(ORBITER)} -v 2 \n-def F -finite_field -q 7 -end \n-with F -do \n-finite_field_activity \n-polynomial_mult_mod "3,1,2" "5,3,4" "6,0,0,1" -end

poly_mult_mod_F4:
$\text{(ORBITER)} -v 2 \n-def F -finite_field -q 2 -end \n-with F -do \n-finite_field_activity \n-polynomial_mult_mod "1,1" "1,1" "1,1,1" -end

mult_polynomials_2_5_7:
$\text{(ORBITER)} -v 2 \n-def F -finite_field -q 2 -end \n-with F -do \n-finite_field_activity -mult_polynomials 5 7 -end
8665  pdflatex polynomial_mult_5_7.tex
8666  open polynomial_mult_5_7.pdf
8667
8668  polynomial_division_ranked_2_27_13:
8669  $(ORBITER) -v 2 \ 
8670  -define F -finite_field -q 2 -end \ 
8671  -with F -do \ 
8672  -finite_field_activity \ 
8673  - with polynomial_division_ranked 27 13 \ 
8674  -end
8675  pdflatex polynomial_division_27_13.tex
8676  open polynomial_division_27_13.pdf
8677
8678
8679
8680
8681  mult_polynomials_2_8_15:
8682  $(ORBITER) -v 2 \ 
8683  -define F -finite_field -q 2 -end \ 
8684  -with F -do \ 
8685  -finite_field_activity -mult_polynomials 8 15 -end
8686  pdflatex polynomial_mult_8_15.tex
8687  open polynomial_mult_8_15.pdf
8688
8689  polynomial_division_ranked_2_120_25:
8690  $(ORBITER) -v 2 \ 
8691  -define F -finite_field -q 2 -end \ 
8692  -with F -do \ 
8693  -finite_field_activity \ 
8694  - with polynomial_division_ranked 120 25 \ 
8695  -end
8696  pdflatex polynomial_division_120_25.tex
8697  open polynomial_division_120_25.pdf
8698
8699  # the answer is 5
8700
8701
8702  mult_polynomials_2_7_7:
8703  $(ORBITER) -v 2 \ 
8704  -define F -finite_field -q 2 -end \ 
8705  -with F -do \ 
8706  -finite_field_activity \ 
8707  - with mult_polynomials 7 7 -end
8708  pdflatex polynomial_mult_7_7.tex
8709  open polynomial_mult_7_7.pdf
8710
8711
8712 8713  mult_polynomials_2.4.6:
8714  8715  > $(ORBITER) -v 2 \
8716  > -define F -finite_field -q 2 -end \
8717  > -with F -do \
8718  > -finite_field_activity \
8719  > -mult_polynomials 4 6 -end
8720  > pdflatex polynomial_mult_4_6.tex
8721  > open polynomial_mult_4_6.pdf
8722  polynomial_division_ranked_2.24.13:
8723  > $(ORBITER) -v 2 \
8724  > -define F -finite_field -q 2 -end \
8725  > -with F -do \
8726  > -finite_field_activity \
8727  > -polynomial_division_ranked 24 13 \
8728  > -end
8729  > pdflatex polynomial_division_24.13.tex
8730  > open polynomial_division_24.13.pdf
8731
8732
8733  mult_polynomials_1024.999.997:
8734  > $(ORBITER) -v 2 \
8735  > -define F -finite_field -q 2 -end \
8736  > -with F -do \
8737  > -finite_field_activity \
8738  > -mult_polynomials 999 997 \
8739  > -end
8740  > pdflatex polynomial_mult_999.997.tex
8741  > open polynomial_mult_999.997.pdf
8742
8743
8744  polynomial_division_ranked_2.349147.1033:
8745  > $(ORBITER) -v 2 \
8746  > -define F -finite_field -q 2 -end \
8747  > -with F -do \
8748  > -finite_field_activity \
8749  > -polynomial_division_ranked 349147 1033 \
8750  > -end
8751  > pdflatex polynomial_division_349147.1033.tex
8752  > open polynomial_division_349147.1033.pdf
8753
8754
8755
8756
8757  mult_polynomials_1024.999.997_check:
8758  > $(ORBITER) -v 3 \

8759
8759 ▷ ▷ -define F -finite_field -q 1024 -end \
8760 ▷ ▷ -with F -do \
8761 ▷ ▷ -finite_field_activity -parse_and_evaluate \
8762 ▷ ▷ "test" "a*b" "a=999,b=997" -end \
8763 8764 # evaluates to 61 
8765 8766 8767 mult.polynomials_17_12: 
8768 ▷ $(ORBITER) -v 2 \ 
8769 ▷ ▷ -define F -finite_field -q 2 -end \
8770 ▷ ▷ -with F -do \ 
8771 ▷ ▷ -finite_field_activity \ 
8772 ▷ ▷ -mult.polynomials 17 12 -end \
8773 ▷ pdflatex polynomial_mult_17_12.tex 
8774 ▷ open polynomial_mult_17_12.pdf 
8775 ▷ 
8776 # gives 204 
8777 8778 polynomial_divisionRanked_2_204_37: 
8779 ▷ $(ORBITER) -v 2 \ 
8780 ▷ ▷ -define F -finite_field -q 2 -end \
8781 ▷ ▷ -with F -do \ 
8782 ▷ ▷ -finite_field_activity \ 
8783 ▷ ▷ ▷ -polynomial_divisionRanked 204 37 \ 
8784 ▷ ▷ ▷ -end \
8785 ▷ pdflatex polynomial_division_204_37.tex 
8786 ▷ open polynomial_division_204_37.pdf 
8787 ▷ 
8788 # answer is 18 
8789 ▷ 
8790 ▷ 
8791 8792 8793 8794 8795 CRC32 SPARSE="1,32,1,26,1,23,1,22,1,16,1,12,1,11,\ 
8796 1,10,1,8,1,7,1,5,1,4,1,2,1,1,1,0" 
8797 8798 8799 #ToDo: 
8800 8801 test.crc32: 
8802 ▷ $(ORBITER) -v 3 \ 
8803 ▷ ▷ -crc32 "123456789"
Berlekamp_matrix_crc32:

```
> $(ORBITER) -v 2 \
>   -define F -finite_field -q 2 -end \n>   -define v -vector -field F -sparse 33 $(CRC32_SPARSE) -end \n>   -with F -do \n>   -finite_field_activity \n>   -Berlekamp_matrix v -end
```

# N = 2^32-1 = 3 * 5 * 17 * 257 * 65537
# N / 3 = 1431655765
# N / 5 = 858993459
# N / 17 = 252645135
# N / 257 = 16711935
# N / 65537 = 65535

TWO_TO_THE_32_MINUS_2=4294967294

```
power_mod_inverse:
```

```
> $(ORBITER) -v 2 \
>   -define F -finite_field -q 2 -end \n>   -define M -vector -field F -sparse 33 $(CRC32_SPARSE) -end \n>   -define A -vector -field F -sparse 2 "1,1" -end \n>   -with F -do \n>   -finite_field_activity \n>   -polynomial_power_mod A $(TWO_TO_THE_32_MINUS_2) M \n>   -end
```

```
INVERSE_SPARSE="1,31,1,25,1,22,1,21,1,15,\n1,11,1,10,1,9,1,7,1,6,1,4,1,3,1,1,1,0"
```

```
#A(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
```

```
mult_mod_to_get_one:
```

```
> $(ORBITER) -v 2 \
>   -define F -finite_field -q 2 -end \n>   -define M -vector -field F -sparse 33 $(CRC32_SPARSE) -end \n>   -define A -vector -field F -sparse 2 "1,1" -end \n>   -define B -vector -field F -sparse 33 $(INVERSE_SPARSE) -end \n>   -with F -do \n>   -finite_field_activity \n>   -polynomial_mult_mod A B M \n>   -end
```

```
C(X)=1
```

700
Berlekamp matrix 2:

```bash
$ (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
```

The polynomial $X^3 + X + 1$ is irreducible over $GF(2)$ because the rank of the Berlekamp matrix is 2.

Berlekamp matrix 2.4:

```bash
$ (ORBITER) -v 2 \
  -define F -finite_field -q 2 -end \
  -define v -vector -field F -dense "1,1,0,0,1" -end \
  -with F -do \
  -finite_field_activity \
  -Berlekamp_matrix v -end
```

The polynomial $X^4 + X + 1$ is irreducible over $GF(2)$ because the rank of the Berlekamp matrix is 3.

Berlekamp matrix 4.3a:

```bash
$ (ORBITER) -v 2 \
  -define F -finite_field -q 4 -end \
  -define v -vector -field F -dense "1,3,0,1" -end \
  -with F -do \
  -finite_field_activity \
  -Berlekamp_matrix v -end
```

Berlekamp matrix 4.3b:

```bash
$ (ORBITER) -v 2 \
  -define F -finite_field -q 4 -end \
  -define v -vector -field F -dense "1,3,1,1" -end \
  -with F -do \
  -finite_field_activity \
  -Berlekamp_matrix v -end
```

701
8898
8899  find_root_a:
8900  ▶ $(ORBITER) -v 2 \\
8901  ▶ ▶ -define F -finite_field -q 19 -end \\
8902  ▶ ▶ -define v -vector -field F -dense "18,1,1" -end \\
8903  ▶ ▶ -with F -do \\
8904  ▶ ▶ -finite_field_activity \\
8905  ▶ ▶ -polynomial_find_roots v -end \\
8906
8907  find_root_b:
8908  ▶ $(ORBITER) -v 2 \\
8909  ▶ ▶ -define F -finite_field -q 19 -end \\
8910  ▶ ▶ -define v -vector -field F -dense "1,3,1" -end \\
8911  ▶ ▶ -with F -do \\
8912  ▶ ▶ -finite_field_activity \\
8913  ▶ ▶ -polynomial_find_roots v -end \\
8914
8915  find_root_c:
8916  ▶ $(ORBITER) -v 2 \\
8917  ▶ ▶ -define F -finite_field -q 19 -end \\
8918  ▶ ▶ -define v -vector -field F -dense "1,16,1" -end \\
8919  ▶ ▶ -with F -do \\
8920  ▶ ▶ -finite_field_activity \\
8921  ▶ ▶ -polynomial_find_roots v -end \\
8922
8923  find_root_d:
8924  ▶ $(ORBITER) -v 2 \\
8925  ▶ ▶ -define F -finite_field -q 19 -end \\
8926  ▶ ▶ -define v -vector -field F -dense "1,18,1" -end \\
8927  ▶ ▶ -with F -do \\
8928  ▶ ▶ -finite_field_activity \\
8929  ▶ ▶ -polynomial_find_roots v -end \\
8930
8931  find_root_e:
8932  ▶ $(ORBITER) -v 2 \\
8933  ▶ ▶ -define F -finite_field -q 19 -end \\
8934  ▶ ▶ -define v -vector -field F -dense "1,16,3" -end \\
8935  ▶ ▶ -with F -do \\
8936  ▶ ▶ -finite_field_activity \\
8937  ▶ ▶ -polynomial_find_roots v -end \\
8938
8939
8940  roots_over_F2:
8941  ▶ $(ORBITER) -v 2 \\
8942  ▶ ▶ -define F -finite_field -q 2 -end \\
8943  ▶ ▶ -define v -vector -field F -dense "0,1,0,1,1,1" -end \\
8944  ▶ ▶ -with F -do \\

702
roots_over_F8:

```bash
$ (ORBITER) -v 2 \
define F -finite_field -q 8 -override_polynomial 11 -end \ndefine v -vector -field F -dense "0,1,0,1,1,1" -end \nwith F -do \n-finite_field_activity \npolynomial_find_roots v -end
```

# degree and then order of the field of coefficients:

irred_3_2:

```bash
$ (ORBITER) -v 3 \
define F -finite_field -q 2 -end \nwith F -do \n-finite_field_activity \nmake_table_of_irreducible_polynomials 3 -end
```

```
pdflatex Irred_q2_d3.tex
open Irred_q2_d3.pdf
```

irred_4_2:

```bash
$ (ORBITER) -v 3 \
define F -finite_field -q 2 -end \nwith F -do \n-finite_field_activity \nmake_table_of_irreducible_polynomials 4 -end
```

```
pdflatex Irred_q2_d4.tex
open Irred_q2_d4.pdf
```

# 3 polys

irred_5_2:

```bash
$ (ORBITER) -v 3 \
define F -finite_field -q 2 -end \nwith F -do \n-finite_field_activity \nmake_table_of_irreducible_polynomials 5 -end
```

```
pdflatex Irred_q2_d5.tex
open Irred_q2_d5.pdf
```
irred_6.2:
- $(ORBTER) -v 3 \\
- define F -finite_field -q 2 -end \\
- with F -do \\
- finite_field_activity \\
- make_table_of_irreducible_polynomials 6 -end

irred_7.2:
- $(ORBTER) -v 3 \\
- define F -finite_field -q 2 -end \\
- with F -do \\
- finite_field_activity \\
- make_table_of_irreducible_polynomials 7 -end

irred_8.2:
- $(ORBTER) -v 3 \\
- define F -finite_field -q 2 -end \\
- with F -do \\
- finite_field_activity \\
- make_table_of_irreducible_polynomials 8 -end

irred_9.2:
- $(ORBTER) -v 3 \\
- define F -finite_field -q 2 -end \\
- with F -do \\
- finite_field_activity \\
- make_table_of_irreducible_polynomials 9 -end

# 6 polys

# 9 polys

# 18 polys

# 30 polys

# 56 polys
$\text{irred\_10\_2}$:

```
$\text{ORBITER}$ -v 3 \
$\text{-define F -finite\_field -q 2 -end}$ \
$\text{-with F -do}$ \
$\text{-finite\_field\_activity}$ \
$\text{-make\_table\_of\_irreducible\_polynomials 10 -end}$ \
\text{pdflatex Irred\_q2\_d10.tex} \
\text{open Irred\_q2\_d10.pdf}$

# 99 polys

$\text{irred\_2\_4}$:

```
$\text{ORBITER}$ -v 3 \
$\text{-define F -finite\_field -q 4 -end}$ \
$\text{-with F -do}$ \
$\text{-finite\_field\_activity}$ \
$\text{-make\_table\_of\_irreducible\_polynomials 2 -end}$ \
\text{pdflatex Irred\_q4\_d2.tex} \
\text{open Irred\_q4\_d2.pdf} 
```

# 6 polys

$\text{irred\_3\_4}$:

```
$\text{ORBITER}$ -v 6 \
$\text{-define F -finite\_field -q 4 -end}$ \
$\text{-with F -do}$ \
$\text{-finite\_field\_activity}$ \
$\text{-make\_table\_of\_irreducible\_polynomials 3 -end}$ \
\text{pdflatex Irred\_q4\_d3.tex} \
\text{open Irred\_q4\_d3.pdf}$

# 20 polys

$\text{search\_primitive\_poly\_2}$:

```
$\text{ORBITER}$ -v 3 \
$\text{-search\_for\_primitive\_polynomial\_in\_range 2 2 2 10}$ | grep // 
```

# stuck in factoring $2^{61}-1$ (which is prime)

$\text{search\_primitive\_poly\_3}$:

```
$\text{ORBITER}$ -v 6 \
```

705
search for primitive polynomial in range 3 3 2 60

search for primitive polynomial in range 4 4 2 30

search for primitive polynomial in range 5 5 2 30

search for primitive polynomial in range 7 7 2 20

search for primitive polynomial in range 8 8 2 20

search for primitive polynomial in range 9 9 2 15

search for primitive polynomial in range 11 11 2 15

search for primitive polynomial in range 13 13 2 15

search for primitive polynomial in range 2 2 16 16

search for primitive polynomial in range 32 32 2 10

###############################################################################
Section 8.2: Multivariate Polynomials

SECTION_MULTIVARIATE_POLYNOMIALS:

test_8_2:

declare
make Cremona_map
make arcs_5_2_q11
make arcs_5_2_q11_ideal
make surface_9lines_4E_ideal
make F_9_q7
make random_k_subsets_PG_2_11
make line_type_in_PG_2_11
make random_arc_5_2_q11_ideal
make Endrass_F7.txt
make octic.prepare

Cremona_map:

```plaintext
$\text{CREMONA\_MAP\_Y0}="3*y0*y0*y0*y0*y0*y2+4*y0*y0*y1*y1*y1*y2\"
$\text{CREMONA\_MAP\_Y1}="y0*y0*y0*y0*y0*y1+5*y0*y0*y0*y1*y1*y1\"
$\text{CREMONA\_MAP\_Y2}="10*y0*y0*y0*y0*y0*y0+11*y0*y0*y0*y0*y1*y1\"
$\text{CREMONA\_MAP\_Y3}="0"
```

Cremona_map:

```plaintext
$(\text{ORBITER}) -v 3 \
define F finite_field -q 13 -end \
define P projective_space -n 2 -field F -v 0 -end 
define R polynomial_ring 
field F 
number_of_variables 3 
homogeneous_degree 6 
monomial_ordering lex 
variables "$y0,y1,y2" "$y_0,y_1,y_2" 
end 
```
-define Y0 -formula "y0" "y_0" "y0,y1,y2"

-define Y1 -formula "y1" "y_1" "y0,y1,y2"

-define Y2 -formula "y2" "y_2" "y0,y1,y2"

$(CREMONA_MAP_Y0)

$(CREMONA_MAP_Y1)

$(CREMONA_MAP_Y2)

-define Cremona -collection "Y0,Y1,Y2"

-with P -do

-projective_space_activity

-map R Cremona ""

-end

-end

-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

# 2 orbits:

-arcs_5_2_q11_ideal:

$(ORBITER) -v 2 

-define F -finite_field -q 11 -end

-define R -polynomial_ring 

708
define C -combinatorial_objects  
-file_of_points arcs_5_2 q11 lvl_5  
-end  
-with C -do  
-combinatorial_object_activity  
-ideal R  
-end  

#( 0, 1, 2, 3, 37 )  
generator 0 / 1 is 7*x0*x1 + 5*x0*x2 + 10*x1*x2  
We found 12 points on the generator of the ideal  
They are : ( 0, 1, 2, 3, 37, 54, 74, 80, 93, 105, 121, 128 )  

#( 0, 1, 2, 3, 49 )  
generator 0 / 1 is 4*x0*x1 + 8*x0*x2 + 10*x1*x2  
looping over all generators of the ideal:  
generator 0 / 1 is ( 0, 4, 8, 0, 10, 0 ) :  
We found 12 points on the generator of the ideal  
They are : ( 0, 1, 2, 3, 41, 49, 58, 77, 83, 95, 109, 130 )  

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"  

surface_9lines_4E_ideal:  
$(ORBITER) -v 2 \  
-def Pts -vector -dense \  
$(PTS_OF_SURFACE_ORBIT211_Q3_L9_E4) \  
-end \  
-def F -finite_field -q 3 -end \  
-def 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 \  

709
# The ideal has dimension 2
# generators for the ideal:
0 1 0 0 2 0 2 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 2 0#
x0*x0*x1 + 2*x0*x1*x1 + 2*x0*x1*x3
#2*x2*x2*x3 + 2*x2*x3*x3

SURFACE_F.9 = \textquoteleft x0*x0*x1 - x0*x1*x1 - x0*x1*x3 - x2*x2*x3 - x2*x3*x3 \textquoteright

F.9_q7:
\$(ORBITER) -v 3 \$
$\langle ORBITER \rangle -v 3$
-define F -finite_field -q 7 -end \$
-define F -finite_field -q 7 -end$
-define P -projective_space -n 3 -field F -v 0 -end \$
-define P -projective_space -n 3 -field F -v 0 -end$
-define F.9 -cubic_surface -space P \$
-define F.9 -cubic_surface -space P$
-by_equation "F.9" \$
-by_equation "F.9$
"DF.9\D" "x0,x1,x2,x3" \$
"DF.9\D" "x0,x1,x2,x3$
$(SURFACE_F.9) \$
$(SURFACE_F.9$
"\Dno parameters\D" \$
"\Dno parameters\D$
-end \$
-end$
-with F.9 -do \$
-with F.9 -do$
cubic_surface_activity \$
cubic_surface_activity$
-report \$
-report$
end \$
end$
pdflatex surface_equation_F.9_q7_report.tex
open surface_equation_F.9_q7_report.pdf

# we create 20 5-subsets of PG(2,11) at random. Note that PG(2,11) has 133 points.
random_k_subsets_PG.2.11:
\$(ORBITER) -v 4 \$
\$(ORBITER) -v 4$
-create_random_k_subsets 133 5 20

#random_k_subsets_n133_k5_nb20.csv
# We compute the line intersections:

```plaintext
line_type_in_PG_2.11:
  $(ORBITER) -v 3 \n  -orbiter_path $(ORBITER_PATH) \n  -define F -finite_field -q 11 -end \n  -define P -projective_space -n 2 -field F -v 0 -end \n  -define C -combinatorial_objects \n  -file_of_points random_k_subsets_n133_k5_nb20.csv \n  -end \n  -with C -do \n  -combinatorial_object_activity \n  -line_type P random_sets \n  -end

# the second one is an arc: 3,33,40,83,102

# we compute the ideal:

random_arc_5_2_q11Ideal:
  $(ORBITER) -v 2 \n  -define F -finite_field -q 11 -end \n  -define R -polynomial_ring \n  -number_of_variables 3 \n  -homogeneous_of_degree 2 \n  -monomial_ordering lex \n  -variables "x0,x1,x2" "x_0,x_1,x_2" \n  -end \n  -define C -combinatorial_objects \n  -set_of_points "3,33,40,83,102" \n  -end \n  -with C -do \n  -combinatorial_object_activity \n  -ideal R \n  -end

#generator 0 / 1 is 10*x0*x0 + 3*x0*x1 + 8*x0*x2 + 2*x1*x1 + 10*x2*x2
#We found 12 points on the generator of the ideal
#They are: ( 3, 15, 19, 33, 40, 42, 46, 50, 83, 88, 102, 108 )
```
\texttt{-define F -finite_field -q 7 -end \ }
\texttt{-define R -polynomial_ring -field F \ }
\texttt{-number_of_variables 4 \ }
\texttt{-homogeneous_of_degree 8 \ }
\texttt{-end \ }
\texttt{-define eqn -vector -field F -sparse 165 \ }
\texttt{-($ENDRASS\_SPARSE$) -end \ }
\texttt{-define P -projective_space -n 3 -field F -v 0 -end \ }
\texttt{-define Endrass$_F^7$ -geometric_object P \ }
\texttt{-define Endrass$_F^7$ -projective_variety R \ }
\texttt{-define P -projective_space -n 3 -field F -v 0 -end \ }
\texttt{-define Endrass$_F^7$ -geometric_object P \ }
\texttt{-define Endrass$_F^7$ -projective_variety R \ }
\texttt{-end \ eqn \ }
\texttt{-end \ }
\texttt{-with Endrass$_F^7$ -do \ }
\texttt{-combinatorial_object_activity -save \ }
\texttt{-end \ }

\texttt{# we created a set of 33 points, called Endrass$_F^7$.txt}
\texttt{octic_prepare:}
\texttt{-define A -vector -format 1 -dense "1,1,1,1" -end \ }
\texttt{-define D -diophant \ }
\texttt{-label octic_monomials \ }
\texttt{-coefficient_matrix A \ }
\texttt{-RHS "8,8,1" \ }
\texttt{-x_min_global 0 -x_max_global 8 \ }
\texttt{-end \ }
\texttt{-with D -do \ }
\texttt{-diophant_activity -solve_mckay \ }
\texttt{-end \ }
\texttt{sort -r octic_monomials.sol >octic_monomials_sorted.txt}
\texttt{#Found 165 solutions with 210 backtrack steps}
\texttt{# 165=binomial(11,3)}

\texttt{# Chapter 9 - Applications}
# Section 9.1: Number Theory

section 9.1:

```
test 9.1:

  make inverse_mod a
  make jacobi_35_41
  make jacobi_33_41
  make jacobi_a
  make jacobi_5_19
  make sqrt_mod_7817

inverse_mod_26_99:

  $(ORBITER) -v 2 -inverse_mod 26 99

inverse_mod_a:

  $(ORBITER) -v 2 -inverse_mod 18059241 58014043

jacobi_35_41:

  $(ORBITER) -v 5 -jacobi 35 41
  pdflatex jacobi_35_41.tex
  open jacobi_35_41.pdf

jacobi_33_41:

  $(ORBITER) -v 5 -jacobi 33 41
  pdflatex jacobi_33_41.tex
  open jacobi_33_41.pdf

jacobi_a:

  $(ORBITER) -v 5 -jacobi 2221 7817

jacobi_5_19:

  $(ORBITER) -v 5 -jacobi 5 19

sqrt_mod_7817:

  $(ORBITER) -v 2 -sqrt_mod_2221 7817
```

713
# Section 9.2: Representation Theory

SECTION REPRESENTATION THEORY:

test_9.2:

- make representation_on_polynomials_of_degree_3
- make representation_tetrahedral_group_on_polynomials_of_degree_3

representation_on_polynomials_of_degree_3:

```bash
$(ORBITER) -v 4 \\
  -define G -linear_group -PGL 4 3 -end \\
  -with G -do \\
  -group_theoretic_activity \\
  -representation_on_polynomials 3 \\
  -end
```

representation_tetrahedral_group_on_polynomials_of_degree_3:

```bash
$(ORBITER) -v 4 \\
  -define G -linear_group -GL 3 3 \\
  -subgroup_by_generators "tetra" "12" 2 \\
  -representation_on_polynomials 3 \\
  -end
```
SECTION_9.3: Cryptography:

- make EC_add
- make EC_cyclic_subgroup
- make EC_points_13
- make EC_points_199
- make EC_Koblitz_encoding
- make EC_bsgs
- make EC_bsgs_decode
- make NTRU_Alice1
- make NTRU_Alice2
- make NTRU_Alice_public_key
- make NTRU_encrypt
- make NTRU_decrypt1
- make NTRU_decrypt2
- make NTRU_decrypt3
- make NTRU_decrypt4
- make NTRU_decrypt5
- make inv_59_mod
- make RSA_e
- make RSA_d
- make im1
- make RSA_e1
- make RSA_d1
- make im1061
- make RSA_e2
- make RSA_d2
- make im3
- make RSA_e3
- make RSA_d3
- make im4
- make RSA_e4
- make RSA_d4
9555 ▷ make im5
9556 ▷ make RSA_e5
9557 ▷ make RSA_d5
9558 ▷ make RSA_d6
9559 ▷ make smooth
9560 ▷ make im7
9561 ▷ make RSA_e7
9562 ▷ make im8
9563 ▷ make RSA_e8
9564 ▷ make sqrt_big
9565 ▷ make sqrt_mod_33_41
9566 ▷ make quadratic_sieve
9567 ▷ make pseudoprime3
9568 ▷ make pseudoprime10
9569 ▷ make PR10_test1
9570 ▷ make pseudoprime11
9571 ▷ make pseudoprime20
9572 ▷ make PR10
9573 ▷ make pseudoprime50
9574 ▷ make pseudoprime61
9575 ▷ make pseudoprime30
9576 ▷ make pseudoprime31
9577 ▷ make pseudoprime33
9578 ▷ make pseudoprime34
9579 ▷ make pseudoprime35
9580 ▷ make pseudoprime36
9581 ▷ make MATH360_hw2
9582 ▷ make F_256_Rijndahl
9583 ▷ make all_square_roots_mod_n_1549411
9584 ▷ make power_mod_211
9585 ▷ make power_mod_2_31
9586 ▷ make power_mod_3_31
9587
9588
9589
9590 EC_add:
9591 ▷ $(ORBITER) -v 2 \n9592 ▷ ▷ -define F -finite_field -q 11 -end \n9593 ▷ ▷ -with F -do \n9594 ▷ ▷ -finite_field_activity \n9595 ▷ ▷ -EC.add 1 3 "1,4" "1,4" -end
9596
9597 EC_cyclic_subgroup:
9598 ▷ $(ORBITER) -v 2 \n9599 ▷ ▷ -define F -finite_field -q 11 -end \n9600 ▷ ▷ -with F -do \n9601 ▷ ▷ -finite_field_activity \n
716
9602  ⧿ ⧿ \texttt{-EC_cyclic_subgroup 1 3 "1,4" -end}
9603
9604
EC_points.13:
9605  ⧿ ⧿ \texttt{$(ORBITER) -v 2 \ \backslash}
9606  ⧿ ⧿ \texttt{-define F -finite_field -q 13 -end \}
9607  ⧿ ⧿ \texttt{-with F -do \}
9608  ⧿ ⧿ \texttt{-finite_field_activity \}
9609  ⧿ ⧿ \texttt{-EC_points "EC.2_5.q13" 2 5 -end}
9610  ⧿ ⧿ \texttt{$(ORBITER) -v 2 -draw_matrix \}
9611  ⧿ ⧿ \texttt{-input_csv_file EC.2_5.q13.points.xy.csv \}
9612  ⧿ ⧿ \texttt{-box_width 20 -bit_depth 24 \}
9613  ⧿ ⧿ \texttt{-partition 2 "1,1,1,1,1,1,1,1,1,1,1,1,1,1,1" "1,1,1,1,1,1,1,1,1,1,1,1,1,1,1" -end}
9614
9615
9616
9617
9618
EC_points.199:
9619  ⧿ ⧿ \texttt{$(ORBITER) -v 2 \ \backslash}
9620  ⧿ ⧿ \texttt{-define F -finite_field -q 199 -end \}
9621  ⧿ ⧿ \texttt{-with F -do \}
9622  ⧿ ⧿ \texttt{-finite_field_activity \}
9623  ⧿ ⧿ \texttt{-EC_points "EC.5_7.q199" 5 7 -end}
9624  ⧿ ⧿ \texttt{$(ORBITER) -v 2 \ \backslash}
9625  ⧿ ⧿ \texttt{-draw_matrix -input_csv_file EC.5_7.q199.points.xy.csv \}
9626  ⧿ ⧿ \texttt{-box_width 10 -bit_depth 24 \}
9627  ⧿ ⧿ \texttt{-partition 2 199 199 -end}
9628
9629
9630  EC_Koblitz_encoding:
9631  ⧿ ⧿ \texttt{$(ORBITER) -v 6 -seed 17 \ \backslash}
9632  ⧿ ⧿ \texttt{-define F -finite_field -q 199 -end \}
9633  ⧿ ⧿ \texttt{-with F -do \}
9634  ⧿ ⧿ \texttt{-finite_field_activity \}
9635  ⧿ ⧿ \texttt{-EC_Koblitz_encoding 5 7 67 "147,164" "DEADBEEF" \}
9636  ⧿ ⧿ \texttt{-end}
9637
9638  EC_bsgs:
9639  ⧿ ⧿ \texttt{$(ORBITER) -v 2 \ \backslash}
9640  ⧿ ⧿ \texttt{-define F -finite_field -q 199 -end \}
9641  ⧿ ⧿ \texttt{-with F -do \}
9642  ⧿ ⧿ \texttt{-finite_field_activity \}
9643  ⧿ ⧿ \texttt{-EC_bsgs 5 7 "147,164" 212 \}
9644  ⧿ ⧿ \texttt{"172,158,45,195,50,22,10,103,55,33,50,22,145,105,31,74,73,155,67,60,25,6" \}
9645  ⧿ ⧿ \texttt{-end}
9646
9647  EC_bsgs_decode:
9648  ⧿ ⧿ \texttt{$(ORBITER) -v 2 \ \backslash
-define F -finite_field -q 199 -end \
-with F -do \n-finite_field_activity \n-EC.bgs_decode 5 7 "129,176" 212 \n"127,188,51,141,85,29,106,90,41,105,179,71,171,2,16,197,183,72,27,129,37,10" \n"50,179,169,13,153,169,115,116,188,110,176" \n-end

NTRU N=7
NTRU P=3
NTRU Q=41
NTRU D=2
NTRU_XN1="-1,0,0,0,0,0,0,1,"

# D + 1 plus ones and D minus ones
ALICE_PRIVATE_F="-1,0,1,1,-1,0,1"

# D plus ones and D minus ones
ALICE_PRIVATE_G="0,-1,-1,0,1,0,1"

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

#F.q(x) = 8X^6 + 26X^5 + 31X^4 + 21X^3 + 40X^2 + 2X + 37
ALICE_PRIVATE_FQ="37,2,40,21,31,26,8"

ALICE2:
$(ORBITER) -v 2 \
-define F -finite_field -q $(NTRU P) -end \

9695 ▶ ▶ -with F -do \\
9696 ▶ ▶ -finite_field_activity \\
9697 ▶ ▶ -extended_gcd_for_polynomials \\
9698 ▶ ▶ ▶ $(NTRUE_XN1) $(ALICE_PRIVATE_F) \\
9699 ▶ ▶ -end \\
9700 9701 #F.p(x) = X^6 + 2X^5 + X^3 + X^2 + X + 1 \\
9702 ALICE_PRIVATE_FP="1,1,1,0,2,1" \\
9703 9704 NTRU_Alice_public_key: \\
9705 ▶ $(ORBITER) -v 2 \\
9706 ▶ ▶ -define F -finite_field -q $(NTRU_Q) -end \\
9707 ▶ ▶ -with F -do \\
9708 ▶ ▶ -finite_field.activity \\
9709 ▶ ▶ -polynomial_mult_mod $(ALICE_PRIVATE_F) \\
9710 ▶ ▶ ▶ $(ALICE_PRIVATE_G) $(NTRUE_XN1) \\
9711 ▶ ▶ -end \\
9712 9713 #C(X)=20X^6 + 40X^5 + 2X^4 + 38X^3 + 8X^2 + 26X + 30 \\
9714 ALICE_PUBLIC_KEY="30,26,8,38,2,40,20" \\
9715 9716 BOB_MESSAGE="1,-1,1,1,0,-1" \\
9717 9718 BOB_ONE_TIME_KEY="-1,1,0,0,0,-1,1" \\
9719 9720 9721 NTRU_encrypt: \\
9722 ▶ $(ORBITER) -v 2 \\
9723 ▶ ▶ -define F -finite_field -q $(NTRU_Q) -end \\
9724 ▶ ▶ -with F -do \\
9725 ▶ ▶ -finite_field_activity \\
9726 ▶ ▶ -NTRU_encrypt $(NTRUE_N) $(NTRUE_P) $(ALICE_PUBLIC_KEY) \\
9727 ▶ ▶ ▶ $(BOB_ONE_TIME_KEY) $(BOB_MESSAGE) -end \\
9728 9729 # E(X) = 31X^6 + 19X^5 + 4X^4 + 2X^3 + 40X^2 + 3X + 25 \\
9730 BOB_ENCRYPT= "25,3,40,2,4,19,31" \\
9731 9732 9733 NTRU_decrypt1: \\
9734 ▶ $(ORBITER) -v 2 \\
9735 ▶ ▶ -define F -finite_field -q $(NTRU_Q) -end \\
9736 ▶ ▶ -with F -do \\
9737 ▶ ▶ -finite_field_activity \\
9738 ▶ ▶ -polynomial_mult_mod $(ALICE_PRIVATE_F) \\
9739 ▶ ▶ ▶ $(BOB_ENCRYPT) $(NTRUE_XN1) \\
9740 ▶ ▶ -end \\
9741 719
#C(X)=X^6 + 10X^5 + 33X^4 + 40X^3 + 40X^2 + X + 40

ALICE

C1="40,1,40,40,33,10,1"

NTRU decrypt2:
$\langle ORBITER \rangle -v 2$
-define F -finite_field -q $(NTRU_Q) -end
-with F -do
-finite_field_activity
-polynomial_center_lift $(ALICE_C1) -end

#A(X)=X^6 + 10X^5 - 8X^4 - X^3 - X^2 + X - 1

ALICE_C2="-1,1,-1,-1,-8,10,1"

NTRU decrypt3:
$\langle ORBITER \rangle -v 2$
-define F -finite_field -q $(NTRU_Q) -end
-with F -do
-finite_field_activity
-polynomial_reduce_mod $('(ALICE_C2) -end

#A(X)=X^6 + X^5 + X^4 + 2X^3 + 2X^2 + X + 2

ALICE_C3="2,1,2,2,1,1,1"

NTRU decrypt4:
$\langle ORBITER \rangle -v 2$
-define F -finite_field -q $(NTRU_P) -end
-with F -do
-finite_field_activity
-polynomial_mult_mod $(ALICE_PRIVATE_FP) $(ALICE_C3) $(NTRU_XN1)
-end

#C(X)=2X^5 + X^3 + X^2 + 2X + 1

ALICE_C4="1,2,1,1,0,2"

NTRU decrypt5:
$\langle ORBITER \rangle -v 2$
-define F -finite_field -q $(NTRU_P) -end
-with F -do
-finite_field_activity
-polynomial_center_lift $(ALICE_C4) -end

#A(X)= - X^5 + X^3 + X^2 - X + 1

plaintext BOB_MESSAGE
inv_59_mod:
$ (ORBITER) -v 2 -inverse_mod 59 10200
# the inverse of 59 mod 10200 is 2939

RSA_e:
$ (ORBITER) -v 2 \\
-RSA 59 10403 2 "1921,1605,1804,2116,0518"

RSA_d:
$ (ORBITER) -v 2 \\
-RSA 2939 10403 2 "902,3509,9833,3548,5181"

im1:
$ (ORBITER) -v 2 -inverse_mod 869 1843488
# the inverse of 869 mod 1843488 is 386093

# FUNFACTOR:
RSA_e1:
$ (ORBITER) -v 2 \\
-RSA 386093 1846303 3 "62114,60103,201518"

RSA_d1:
$ (ORBITER) -v 2 \\
-RSA 869 1846303 3 "1248407,345776,317846"

# 5503*4603 = 25330309
# 5502*4602 = 25320204

im1061:
$ (ORBITER) -v 2 \\
-inverse_mod 1061 25320204
9836  ▶
9837  # the inverse of 1061 mod 25320204 is 2076209
9838
9839
9840
9841 RSA_e2:
9842  ▶ $(ORBITER) -v 2 \n9843  ▶ ▶ -RSA_encrypt_text 2076209 25330309 3 creamcheese
9844
9845  #-RSA_encrypt_text 386093 1846303 creamcheese
9846  #408918,1735142,239809,654636
9847
9848
9849 RSA_d2:
9850  ▶ $(ORBITER) -v 2 \n9851  ▶ ▶ -RSA 1061 25330309 3 "19019931,1619805,740498,2671344"
9852
9853
9854  # 7253*8171 = 59264263
9855  # 7252*8170 = 59248840
9856
9857
9858
9859 im3:
9860  ▶ $(ORBITER) -v 2 \n9861  ▶ ▶ -inverse_mod 2909 59248840
9862  ▶
9863  #the inverse of 2909 mod 59248840 is 4358629
9864
9865 RSA_e3:
9866  ▶ $(ORBITER) -v 2 \n9867  ▶ ▶ -RSA_encrypt_text 2909 59264263 3 encrypted
9868
9869 RSA_d3:
9870  ▶ $(ORBITER) -v 2 \n9871  ▶ ▶ -RSA 4358629 59264263 3 "35270141,9642524,49091707"
9872
9873  #51403,182516,200504 = encrypted
9874
9875
9876 ####
9877  # 7879 * 7901 = 62251979
9878  # 7878 * 7900 = 62236200
9879
9880  # e =
9881
9882 im4:
9883  ➤ $(ORBITER) -v 2 -inverse_mod 583 62236200
9884
9885  # the inverse of 583 mod 62236200 is 32559247
9886
9887  RSA_e4:
9888  ➤ $(ORBITER) -v 2 \ 
9889  ➤ ➤ -RSA_encrypt_text 583 62251979 3 venividivici
9890
9891  #-RSA_encrypt_text 583 62251979 venividivici
9892  #40513610,53979973,56449676,35068535
9893
9894  RSA_d4:
9895  ➤ $(ORBITER) -v 2 \ 
9896  ➤ ➤ -RSA 32559247 62251979 "40513610,53979973,56449676,35068535"
9897
9898
9899
9900  ######
9901  # 7369 * 7127 = 52518863
9902  # 7368 * 7126 = 52504368
9903
9904
9905  im5:
9906  ➤ $(ORBITER) -v 2 -inverse_mod 173 52504368
9907
9908  #the inverse of 173 mod 52504368 is 38543669
9909  ➤
9910  RSA_e5:
9911  ➤ $(ORBITER) -v 2 \ 
9912  ➤ ➤ -RSA_encrypt_text 38543669 52518863 3 fascinating
9913
9914  #-RSA_encrypt_text 38543669 52518863 fascinating
9915  #31526751,8962078,51045732,51894467
9916  ➤ ➤
9917
9918  RSA_d5:
9919  ➤ $(ORBITER) -v 2 \ 
9920  ➤ ➤ -RSA 173 52518863 "31526751,8962078,51045732,51894467"
9921
9922
9923  RSA_d6:
9924  ➤ $(ORBITER) -v 2 \ 
9925  ➤ ➤ -RSA 47177497 55040413 "28702119,48926559"
9926
9927
9928  smooth:
9929  ➤ $(ORBITER) -v 2 \ 

723
-sift_smooth 100000 100 "2,3,5,7,11,13,17,19"

# 1999 * 7907 = 15806093
# 1998 * 7906 = 15796188

im7:
$(ORBITER) -v 2 -inverse_mod 3221 15796188

#the inverse of 3221 mod 15796188 is 10048553

RSA_e7:
$(ORBITER) -v 2
-RSA_encrypt_text 10048553 15806093 3 beachandfun

# 7853 * 7673 = 60256069
# 7852 * 7672 = 60240544

im8:
$(ORBITER) -v 2 -inverse_mod 9017 60240544

#the inverse of 9017 mod 60240544 is 14430473

RSA_e8:
$(ORBITER) -v 2
-RSA_encrypt_text 9017 60256069 3 strawberry

sqrt_big:
$(ORBITER) -v 2 -square_root 1002001

sqrt_mod_33_41:
$(ORBITER) -v 2 -square_root_mod 33 41

quadratic_sieve:
$(ORBITER) -v 5 -quadratic_sieve 31 500 1

pseudoprime3:
9977  $(\text{ORBITER}) -v 5 \\
9978  \> -seed 2531011 -find_pseudoprime 3 5 0 0 \\
9979  \> pdflatex pseudoprime_3.tex \\
9980  \> open pseudoprime_3.pdf \\
9981 \\
9982  \> pseudoprime10: \\
9983  \> $(\text{ORBITER}) -v 5 \\
9984  \> -seed 2531011 -find_pseudoprime 10 5 5 5 \\
9985  \> pdflatex pseudoprime_10.tex \\
9986  \> open pseudoprime_10.pdf \\
9987 \\
9988  \> # 4460190157 \\
9989 \\
9990 \\
9991  \> PR10_test1: \\
9992  \> $(\text{ORBITER}) -v 5 -power_mod 1293 2230095078 4460190157 \\
9993  \> $(\text{ORBITER}) -v 5 -power_mod 9865 2230095078 4460190157 \\
9994  \> $(\text{ORBITER}) -v 5 -power_mod 19645 2230095078 4460190157 \\
9995  \> $(\text{ORBITER}) -v 5 -power_mod 974586571 2230095078 4460190157 \\
9996  \> $(\text{ORBITER}) -v 5 -power_mod 974586571 4460190157 \\
9997  \> $(\text{ORBITER}) -v 5 -power_mod 974586571 15222492 4460190157 \\
9998  \> $(\text{ORBITER}) -v 5 -power_mod 974586571 284796 4460190157 \\
9999 \\
10000 \\
10001  \> pseudoprime11: \\
10002  \> $(\text{ORBITER}) -v 5 \\
10003  \> -seed 2531011 -find_pseudoprime 11 5 5 5 \\
10004  \> pdflatex pseudoprime_11.tex \\
10005  \> open pseudoprime_11.pdf \\
10006 \\
10007 \\
10008  \> # 63814633367 \\
10009 \\
10010  \> # product is 284625399616057168619 \\
10011 \\
10012  \> pseudoprime20: \\
10013  \> $(\text{ORBITER}) -v 5 \\
10014  \> -seed 2531011 -find_pseudoprime 20 5 5 5 \\
10015  \> pdflatex pseudoprime_20.tex \\
10016  \> open pseudoprime_20.pdf \\
10017 \\
10018 \\
10019 \\
10020  \> PR10: \\
10021  \> $(\text{ORBITER}) -v 5 -primitive_root 4460190157 \\
10022 \\
10023
# mistake! long integer overflow
# a primitive root modulo 165222861 is 1293

```
pseudoprime50:
  $(ORBITER) -v 5 \
  -seed 2531011 -find_pseudoprime 50 5 0 0
  pdflatex pseudoprime_50.tex
  open pseudoprime_50.pdf

#91322792878581218181431392170986926262336688354473

pseudoprime51:
  $(ORBITER) -v 5 \
  -seed 2531011 -find_pseudoprime 51 5 5 5
  pdflatex pseudoprime_51.tex
  open pseudoprime_51.pdf

#754600727746834470214089702490004944659715367045417

# product 68912245966050819606199994423264315732335295324400658436661744403244049
572914094379904326661586100241

pseudoprime30:
  $(ORBITER) -v 5 \
  -seed 2531011 -find_pseudoprime 30 5 5 5
  pdflatex pseudoprime_30.tex
  open pseudoprime_30.pdf

# 286525565474504516914595596387

pseudoprime31:
  $(ORBITER) -v 5 \
  -seed 2531011 -find_pseudoprime 31 5 5 5
  pdflatex pseudoprime_31.tex
  open pseudoprime_31.pdf

#877726676542264552372412985331

# maybe 2 seconds
```

pseudoprime33:
(ORBITER) -v 5 \n-seed 2531011 -find_pseudoprime 33 5 5 5 \npdflatex pseudoprime_33.tex \nopen pseudoprime_33.pdf

#371674199498295345543363004459891

pseudoprime34:
(ORBITER) -v 5 \n-seed 2531011 -find_pseudoprime 34 5 5 5 \npdflatex pseudoprime_34.tex \nopen pseudoprime_34.pdf

#9309708224110488378214945245346817

# 3460178351758962531912872979731874528849142238619677890786061016947
18 sec

pseudoprime35:
(ORBITER) -v 5 \n-seed 2531011 -find_pseudoprime 35 5 5 5 \npdflatex pseudoprime_35.tex \nopen pseudoprime_35.pdf

#81329557792505271120435930267680203

MATH360_hw2:
(ORBITER) -v 3 \n-define F -finite_field -q 16 -end \n-with F -do -finite_field_activity \nparse_and_evaluate "test" "a+b" "a=8,b=14" -end
10117 ➤ $(ORBITER) -v 3 \\
10118 ➤  -define F -finite_field -q 16 -end \\
10119 ➤  -with F -do -finite_field_activity \\
10120 ➤  -parse_and_evaluate "test" "" "a*b" "a=9,b=13" -end \\
10121 ➤ $(ORBITER) -v 3 \\
10122 ➤  -define F -finite_field -q 16 -end \\
10123 ➤  -with F -do -finite_field_activity \\
10124 ➤  -parse_and_evaluate "test" "" "a*a*a*a*a" "a=9" -end \\
10125 ➤ $(ORBITER) -v 3 \\
10126 ➤  -define F -finite_field -q 16 -end \\
10127 ➤  -with F -do -finite_field_activity \\
10128 ➤  -parse_and_evaluate "test" "" "(a+b)*(a+b)" "a=5,b=7" -end \\
10129 ➤ $(ORBITER) -v 3 \\
10130 ➤  -define F -finite_field -q 16 -end \\
10131 ➤  -with F -do -finite_field_activity \\
10132 ➤  -parse_and_evaluate "test" "" "a*a+b*b" "a=5,b=7" -end \\
10133 ➤ \\
10134 ➤ F_256.Rijndahl: \\
10135 ➤ $(ORBITER) -v 3 \\
10136 ➤  -define F -finite_field -q 256 -override_polynomial 283 -end \\
10137 ➤  -with F -do -finite_field_activity -cheat_sheet_GF -end \\
10138 ➤ \\
10139 ➤ 10140 ➤ 10141 ➤ 10142 ➤ 10143 ➤ 10144 ➤ 10145 ➤ all_square_roots_mod_n_1549411: \\
10146 ➤ $(ORBITER) -v 3 -all_square_roots_mod_n 1075922 1549411 \\
10147 ➤ \\
10148 ➤ 10149 ➤ 10150 ➤ power_mod_211: \\
10151 ➤ $(ORBITER) -v 3 -power_mod_n 2 211 \\
10152 ➤ $(ORBITER) -v 3 \\
10153 ➤  -plot_function power_mod_n_a2_n211.csv \\
10154 ➤ $(ORBITER) -v 2 -draw_matrix \\
10155 ➤  -input_csv_file power_mod_n_a2_n211_graph.csv \\
10156 ➤  -box_width 10 -bit_depth 8 -partition 3 211 211 -end \\
10157 ➤ \\
10158 ➤ power_mod_2_31: \\
10159 ➤ $(ORBITER) -v 3 -power_mod_n 2 31 \\
10160 ➤ $(ORBITER) -v 3 \\
10161 ➤  -plot_function power_mod_n_a2_n31.csv \\
10162 ➤ \\
10163 ➤ $(ORBITER) -v 2 -draw_matrix \\
10164 ➤ 728
10164  ▶  ▶  -input_csv_file power_mod_n_a2_n31_graph.csv \\
10165  ▶  ▶  -box_width 10 -bit_depth 8 -partition 3 31 31 -end \\
10166  ▶  ▶  power_mod_3_31: \\
10167  ▶  ▶  $(ORBITER) -v 3 -power_mod_n 3 31 \\
10168  ▶  ▶  $(ORBITER) -v 3 \\
10169  ▶  ▶  -plot_function power_mod_n_a3_n31.csv \\
10170  ▶  ▶  $(ORBITER) -v 2 -draw_matrix \\
10171  ▶  ▶  -input_csv_file power_mod_n_a3_n31_graph.csv \\
10172  ▶  ▶  -box_width 10 -bit_depth 8 -partition 3 31 31 -end \\
10173  ▶  ▶  
10174  ▶  ▶  
10175  ▶  ▶  
10176  ▶  ▶  
10177  # Chapter 10 - Coding Theory 
10178  # Section 10.1: Coding Theory 
10179  # Section CODING THEORY INTRODUCTION: 
10180  
10181  
10182  
10183  
10184  test_10_1: 
10185  ▶  make Allen_Gates_noise_1_percent 
10186  ▶  make Hamming_space_4_2_distance_matrix 
10187  ▶  make Hamming_space_4_2_distance_matrix_draw 
10188  ▶  make Hamming_code_macwilliams 
10189  ▶  make code_5_2_3_diagram 
10190  ▶  make Hamming_5_2_graph 
10191  ▶  make Hamming_5_2_with_5_2_3_code 
10192  ▶  make code_6 
10193  ▶  
10194  Allen_Gates_noise_1_percent: 
10195  ▶  $(ORBITER) -v 3 \\
10196  ▶  ▶  -random_noise_in_bitmap_file \\
10197  ▶  ▶  allen_Gates.bmp \\
10198  ▶  ▶  allen_Gates_1.bmp \\
10199  ▶  ▶  1 100 \\
10200  ▶  ▶  open allen_Gates_1.bmp \\
10201  ▶  ▶  
10202  ▶  ▶  Hamming_space_4_2_distance_matrix: 
10203  ▶  ▶  $(ORBITER) -Hamming_space_distance_matrix 4 2 
10204  ▶  ▶  
10205  ▶  ▶  
10206  ▶  ▶  
10207  ▶  ▶  
10208  ▶  ▶  
10209  ▶  ▶  
10210  ▶  ▶  

729
Hamming space 4 2 distance matrix draw:

```bash
$ (ORBITER) -v 2 -draw_matrix \
```

```bash
> -input_csv_file Hamming_n4_q2.csv \
```

```bash
> -box_width 20 -bit_depth 24 \
```

```bash
> -partition 4 16 16 \
```

```bash
> -end 
```

```bash
open Hamming_n4_q2_draw.bmp
```

Hamming code macwilliams:

```bash
$ (ORBITER) -v 2 \
```

```bash
> -make_macwilliams_system 7 4 2 \
```

```bash
pdflatex MacWilliams_n7_k4_q2.tex
```

```bash
open MacWilliams_n7_k4_q2.pdf
```

```bash

```

Hamming 5 2 graph:

```bash
$ (ORBITER) -v 2 \
```

```bash
> -define G -graph -Hamming 5 2 -end \
```

```bash
> -with G -do \
```

```bash
> -graph_theoretic_activity -export_csv -end \
```

```bash
> -with G -do \
```

```bash
> -graph_theoretic_activity -export_graphviz -end \
```

```bash
> -with G -do \
```

```bash
> -graph_theoretic_activity -save -end 
```
Hamming 5_2 with 5_2_3 code:

```
$(ORBITER) -v 2 -define G -graph -Hamming 5 2
-subset "\code5_2_3" "\code5_2 2_3"
$(CODE5_2_3_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
```

```
Hamming 5_2, code 5_2_3:
```

```
pdflatex Hamming_5_2.code_5_2_3.report.tex
```

```
open Hamming_5_2.code_5_2_3.report.pdf
```

```
# group has order 32
```

```
```

code 6:

```
$(ORBITER) -v 2
-define F -finite_field -q 2 -end
-with F -do -coding_theoretic_activity
-general_code_binary 6 "0,60,50,41,14,21,27,39"
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```
```
# Section 10.2: Linear codes

SECTION CODING THEORY LINEAR CODES:

test_10_2:

- make RM_3_1
- make Hamming_generator
- make simplex_code
- make Hamming_generator
- make Hamming_code
- make RM_3_1_and_codewords
- make RM_3_1_from_generator_matrix
- make RM_4_1_and_codewords
- make RM_5_1_and_codewords
- make Hamming_code_by_rows
- make Hamming_weight Enumerator
- make Hamming_minimum_distance
- make Golay23_minimum_distance
- make Hamming_code_diagram
- make code_Hamming_systematic
- make Hamming_RREF
- make Hamming_nullspace
- make Hamming_long
- make Hamming_singer
- make Hamming_cyclic_generator
- make Hamming_cyclic_long
- make Hamming_cyclic
- make Hamming_cyclic_clean ns
- make Hamming_cyclic_clean
- make Hamming_cyclic_clean_long

RM_3_1:

```
$(ORBITER) -v 2 \n-define F -finite_field -q 2 -end \n-define C -code -field F \n-first_order_Reed_Muller 3 \n-end \n```
10352 \> \> -with C -and F -do -coding_theoretic_activity \ 
10353 \> \> \> -export_magma RM_3_1.magma \ 
10354 \> \> -end \ 
10355 \ 
10356 \ 
10357 \ 
10358 \ 
10359 \ 
10360 simplex_code: \ 
10361 \> \> $(ORBITER) -v 2 \ 
10362 \> \> \> -define F -finite_field -q 2 -end \ 
10363 \> \> \> -define v -vector -field F -format 3 \ 
10364 \> \> \> \> -dense $(SIMPLEX_CODE_GENERATOR) \ 
10365 \> \> \> -end \ 
10366 \> \> \> -define C -code -field F \ 
10367 \> \> \> \> -linear_code_through_generator_matrix v \ 
10368 \> \> \> -end \ 
10369 \ 
10370 \ 
10371 \ 
10372 Hamming_generator: \ 
10373 \> \> $(ORBITER) -v 2 \ 
10374 \> \> \> -define F -finite_field -q 2 -end \ 
10375 \> \> \> -define v -vector -field F -format 3 \ 
10376 \> \> \> \> -dense $(SIMPLEX_CODE_GENERATOR) \ 
10377 \> \> \> -end \ 
10378 \> \> \> -with F -do \ 
10379 \> \> \> \> -finite_field_activity \ 
10380 \> \> \> \> \> -nullspace v \ 
10381 \> \> \> \> -end \ 
10382 \> \> \> \> pdflatex nullspace_3_7.tex \ 
10383 \> \> \> \> open nullspace_3_7.pdf \ 
10384 \ 
10385 \# basis in binary: \ 
10386 \# 67,37,22,15 \ 
10387 \#-normalize_from_the_right \ 
10388 \ 
10389 \ 
10390 \ 
10391 \ 
10392 Hamming_code: \ 
10393 \> \> $(ORBITER) -v 2 \ 
10394 \> \> \> -define F -finite_field -q 2 -end \ 
10395 \> \> \> -define v -vector -field F -format 3 \ 
10396 \> \> \> \> -dense $(SIMPLEX_CODE_GENERATOR) \ 
10397 \> \> \> \> -end \ 
10398 \> \> \> \> -define C -code -field F \ 

733
-linear_code_through_generator_matrix v \n-dual \n-end \n-with C -do -coding_theoretic_activity \n-export_magma Hamming.magma \n-end

# writes Hamming.magma

RM_3_1_and_codewords:

-define F -finite_field -q 2 -end \n-define C -code -field F -first_order_Reed_Muller 3 -end \n-with C -and F -do -coding_theoretic_activity \n-export_magma RM_3_1.magma \n-end \n-with C -and F -do -coding_theoretic_activity \n-export_codewords RM_3_1_codewords.csv \n-end \n-with C -and F -do -coding_theoretic_activity \n-export_genma RM_3_1_genma.csv \n-end

#Codewords: (0,255,170,85,102,153,240,15,90,165,60,195,150,105)

RM_3_1_from_generator_matrix:

-define F -finite_field -q 2 -end \n-define genma -vector -format 8 -field F \n-compact $(CODE_RM_3_1_GENMA) \n-end \n-define C -code -field F \n-linear_code_through_generator_matrix genma \n-end

#pdflatex code_n8_k4_q2.tex
#open code_n8_k4_q2.pdf

10437 #Codewords: (0,255,170,85,204,51,102,153,240,15,90,165,60,195,150,105)

RM_4_1_and_codewords:

-define F -finite_field -q 2 -end \n-define C -code -field F -first_order_Reed_Muller 4 -end \n-with C -and F -do -coding_theoretic_activity \n-export_magma RM_4_1.magma \n
Hamming code by rows:

```
$\text{"ORBITER"} -v 2 \n$\text{"define F -finite_field -q 2 -end \n}$\text{"define v -vector -dense \"HAMMING_CODE_ROWS_IN_BINARY_RANKS\" -end \n}$\text{"define C -code -field F \n}$\text{"linear_code_through_basis 7 v \n}$\text{"end \n}$\text{"pflatex code_n7_k4_q2.tex \n}$\text{"open code_n7_k4_q2.pdf \n```

Hamming weight enumerator:

```
$\text{"ORBITER"} -v 2 \n$\text{"define F -finite_field -q 2 -end \n}$\text{"define v -vector -field F -format 4 \n}$\text{"dense \"HAMMING_CODE_GENERATOR\" \n}$\text{"end \n}$\text{"define C -code -field F \n}$\text{"linear_code_through_generator_matrix v \n}$\text{"end \n}$\text{"with C -do \n```
Hamming minimum distance:

```bash
$ (ORBITER) -v 2 \n-define F -finite_field -q 2 -end \n-define v -vector -field F -format 4 \n-dense $(HAMMING_CODE_GENERATOR) \n-end \n-with F -do \n-coding_theoretic_activity \n-minimum_distance v \n-end
```

Hamming code diagram:

```bash
$ (ORBITER) -v 2 \n-define F -finite_field -q 2 -end \n-with F -do -coding_theoretic_activity \n-code_diagram "Hamming_7.4" \n-metric_balls 1 \n-end
```

Golay23 minimum distance:

```bash
$ (ORBITER) -v 2 \n-define F -finite_field -q 2 -end \n-define v -vector -field F -format 12 \n-dense $(GOLAY23_CODE_GENERATOR) \n-end \n-with F -do \n-coding_theoretic_activity \n-minimum_distance v \n-end
```

Golay23 code diagram:

```bash
$ (ORBITER) -v 2 \n-define F -finite_field -q 2 -end \n-with F -do -coding_theoretic_activity \n-code_diagram "Golay23_23.12" \n-metric_balls 1 \n-end
```

#d=7 in 0 sec

Hamming code diagram:

```bash
$ (ORBITER) -v 2 \n-define F -finite_field -q 2 -end \n-with F -do -coding_theoretic_activity \n-code_diagram "Hamming_7.4" \n-metric_balls 1 \n-end
```

Golay23 code diagram:

```bash
$ (ORBITER) -v 2 \n-define F -finite_field -q 2 -end \n-with F -do -coding_theoretic_activity \n-code_diagram "Golay23_23.12" \n-metric_balls 1 \n-end
```
$(ORBITER) -v 2 -define F -finite_field -q 2 -end -with F -do -coding_theoretic_activity -code_diagram "Hamming_7.4_word_0" "00" 7 -metric_balls 1 -end
$(ORBITER) -v 2 -define F -finite_field -q 2 -end -with F -do -coding_theoretic_activity -code_diagram "Hamming_7.4_word_1" "67" 7 -metric_balls 1 -end
$(ORBITER) -v 2 -define F -finite_field -q 2 -end -with F -do -coding_theoretic_activity -code_diagram "Hamming_7.4_word_2" "37" 7 -metric_balls 1 -end
$(ORBITER) -v 2 -define F -finite_field -q 2 -end -with F -do -coding_theoretic_activity -code_diagram "Hamming_7.4_word_3" "102" 7 -metric_balls 1 -end
$(ORBITER) -v 2 -define F -finite_field -q 2 -end -with F -do -coding_theoretic_activity -code_diagram "Hamming_7.4_word_4" "22" 7 -metric_balls 1 -end
$(ORBITER) -v 2 -define F -finite_field -q 2 -end -with F -do -coding_theoretic_activity -code_diagram "Hamming_7.4_word_5" "85" 7 -metric_balls 1 -end
$(ORBITER) -v 2 -define F -finite_field -q 2 -end -with F -do -coding_theoretic_activity -code_diagram "Hamming_7.4_word_6" "51" 7 -metric_balls 1 -end
$(ORBITER) -v 2 -define F -finite_field -q 2 -end -with F -do -coding_theoretic_activity -code_diagram "Hamming_7.4_word_7" "112" 7 -metric_balls 1 -end
$(ORBITER) -v 2 -define F -finite_field -q 2 -end -with F -do -coding_theoretic_activity -code_diagram "Hamming_7.4_word_8" "15" 7 -metric_balls 1 -end
$(ORBITER) -v 2 -define F -finite_field -q 2 -end -with F -do -coding_theoretic_activity -code_diagram "Hamming_7.4_word_9" "76" 7 -metric_balls 1 -end
$(ORBITER) -v 2 -define F -finite_field -q 2 -end -with F -do -coding_theoretic_activity -code_diagram "Hamming_7.4_word_10" "42" 7 -metric_balls 1 -end
$(ORBITER) -v 2 -define F -finite_field -q 2 -end -with F -do -coding_theoretic_activity -code_diagram "Hamming_7.4_word_11" "105" 7 -metric_balls 1 -end
$(ORBITER) -v 2 -define F -finite_field -q 2 -end -with F -do -coding_theoretic_activity -code_diagram "Hamming_7.4_word_12" "25" 7 -metric_balls 1 -end
$(ORBITER) -v 2 -define F -finite_field -q 2 -end -with F -do -coding_theoretic_activity -code_diagram "Hamming_7.4_word_13" "90" 7 -metric_balls 1 -end
$(ORBITER) -v 2 -define F -finite_field -q 2 -end -with F -do -coding_theoretic_activity -code_diagram "Hamming_7.4_word_14" "60" 7 -metric_balls 1 -end
$(ORBITER) -v 2 -define F -finite_field -q 2 -end -with F -do -coding_theoretic_activity -code_diagram "Hamming_7.4_word_15" "127" 7 -metric_balls 1 -end
$(ORBITER) -v 2 -loop L 0 16 1 -draw_matrix \
$(ORBITER) -v 2 -input_csv_file Hamming_7.4_word_\%L_diagram_7_1.csv \
$(ORBITER) -v 2 -box_width 25 -bit_depth 8 -partition 4 16 8 -end \
$(ORBITER) -v 2 -end_loop
$ORBITER code_Hamming_systematic:
$(ORBITER) -v 2 \
$(ORBITER) -define F -finite_field -q 2 -end \

737
-define v -vector -dense $(HAMMING_CODE_ROWS_IN_BINARY_RANKS) -end \\
-define C -code -field F \\
-define F -finite_field -q 2 -end \\
-define v -vector -format 4 -field F \\
-define F2 -finite_field -q 2 -end \\
-define v -vector -format 4 -field F \\
-define F -finite_field -q 2 -end \\
-define v -vector -format 4 -field F \\
-with F -do \\
-finite_field_activity \\
-RREF v \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end \\
-end
#check equations of the Hamming code:
#  a4+a5+a6+a7 =1+0+1+0=0 \text{ mod } 2 \text{ OK.}
#  a2+a3+a6+a7 =0+1+1+0=0 \text{ mod } 2 \text{ OK.}
#  a1+a3+a5+a7 =1+1+0+0=0 \text{ mod } 2 \text{ OK.}

Hamming_long:
$\text{(ORBITER)} -v 2$
$\text{-define C -code -field F -long code 7 4 -box width 25}$
$\text{"0,5,6" -partition 3 4 2} \text{-end}$
$\text{-input csv file long_code_genma_n7_k4_codeword_7.csv}\parallel$
$\text{-loop L 0 16 1 -draw matrix}$
$\text{-bit_depth 8 -partition 3 4 2 -end}$

#long_code_genma_n7_k4_codeword_0.csv
#long_code_genma_n7_k4_codeword_15.csv

#Weight distribution: (0, 3^7, 4^7, 7)

Hamming_singer:
$\text{(ORBITER)} -v 3$
$\text{-define G -linear_group -PGL 3 2 -singer 1} \text{-end}$
define Orb -orbits -group G 
define on_points 
end

#pdflatex PGL_3_2_Singer_3_2_1_report.tex
#open PGL_3_2_Singer_3_2_1_report.pdf

cycle is 0,1,2,5,3,4,6

Hamming_cyclic_generator:

Hamming_cyclic_long:

# ToDo:

Hamming_cyclic:
Hamming cyclic clean:
$\text{(ORBITER)} -v 2 \$
$\text{-define F -finite_field -q 2 -end} \$
$\text{-define v -vector -dense "88,44,22,11" -end} \$
$\text{-define C -code -field F} \$
$\text{-linear_code_through_basis 7 v} \$
$\text{-end} \$
$\text{nullspace v} \$
$\text{-normalize_from_the_right} \$
$\text{-end} \$
$\text{pdflatex nullspace_4_7.tex} \$
$\text{open nullspace_4_7.pdf} \$

Hamming cyclic clean long:
$\text{(ORBITER)} -v 2 -long_code 7 4 \$
$\text{"0,2,3"} \$
$\text{"1,3,4"} \$
$\text{"2,4,5"} \$

# ToDo:
SECTION CODING THEORY GOLAY CODES:

test.10.3:

make Golay23_code_words
make Golay23_code_diagram
make Golay23_code_diagram_draw

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 \
-linear_code_from_from_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 \

Golay23 code diagram:

```
$\text{ORBITER} -v 2 \
$define F -finite_field -q 2 -end 
$with F -do 
$coding_theoretic_activity 
$linear_code_through_columns_of_parity_check_projectively 
$12 v 
$end 
```

```
$#metric_balls 3
```

Golay23 code diagram draw:

```
$\text{ORBITER} -v 2 \
$draw_matrix 
$input_csv_file \text{Golay}_23\_diagram\_01\_23\_4096.csv 
$box_width 4 -bit_depth 8 
$partition 20 4096 2048 
$end 
```

# Section 10.4: Coding Theory - CRC codes

```
$\text{Section}\_\text{Coding}\_\text{Theory}\_\text{CRC}\_\text{Codes}$
```

```
\text{test}\_\text{10}\_\text{4}$:
\text{make encode_text}\_5\text{bits}$
\text{make encode_text}\_5\text{bits}\_\text{check}$
\text{make encode_text}\_5\text{bits}\_\text{error}$
10853 ▶ make CRC_3_128_10
10854 ▶ make crc32_test
10855 ▶ make crc32_test_hexdata
10856 ▶ make crc32_Berlekamp_matrix
10857 ▶ make CRC_F256_roots_771
10858 ▶ make CRC_F256_BCH_code_d2
10859 ▶ make CRC_F256_BCH_write_code_for_division_d2
10860 ▶ make CRC_F256_BCH_code_d3
10861 ▶ make CRC_F256_BCH_write_code_for_division_Bravo
10862 ▶ make CRC_F256_BCH_code_d7
10863 ▶ make F256_BCH_code_d16
10864 ▶ make F256_BCH_write_code_for_division_d16
10865 ▶ make F256_BCH_code_d16_division
10866 ▶ make F256_BCH_code_d16_error
10867 ▶ make crc_encode_16
10868 ▶ make crc_encode_32
10869 ▶ make introduce_errors_16_500000
10870 ▶ make introduce_errors_32_100000
10871 ▶ make check_errors_16
10872 ▶ make check_errors_32
10873 ▶ make extract_block
10874
10875
10876 # ToDo
10877
10878 encode_text_5bits:
10879 ▶ $(ORBITER) -encode_text_5bits \n10880 ▶ "Hithere" "text.csv"
10881 ▶ $(ORBITER) -v 2 \n10882 ▶ -define F -finite_field -q 2 -end \n10883 ▶ -with F -do \n10884 ▶ -coding_theoretic_activity \n10885 ▶ -polynomial_division_from_file \n10886 ▶ text.csv 13 -end
10887 ▶ pdflatex polynomial_division_file_13.tex
10888 ▶ open polynomial_division_file_13.pdf
10889
10890
10891 encode_text_5bits_check:
10892 ▶ $(ORBITER) -v 2 \n10893 ▶ -define F -finite_field -q 2 -end \n10894 ▶ -with F -do \n10895 ▶ -coding_theoretic_activity \n10896 ▶ -polynomial_division_from_file \n10897 ▶ text_with_1error.csv 13 \n10898 ▶ -end
10899 ▶ pdflatex polynomial_division_file_13.tex

744
encode_text_5bits_1error:
  $\text{(ORBITER)}$ -encode_text_5bits \\
  $\text{"Hithere" "text.csv"}$
  $\text{$\text{(ORBITER)}$ -v 2 \}}$
  $\text{\text{-define F -finite_field -q 2 -end \}}$
  $\text{\text{-with F -do \}}$
  $\text{\text{-coding_theoretic_activity \}}$
  $\text{\text{-polynomial_division_from_file_all_k_bit_error_patterns \}}$
  $\text{\text{-text.csv 13 1 \}}$
  $\text{\text{-end \}}$
  $\text{\text{pdflatex polynomial_division_file_all_1_error_patterns_13.pdf \}}$

CRC_3_128_10:
  $\text{$\text{(ORBITER)}$ -v 1 \}}$
  $\text{\text{-define F -finite_field -q 2 -end \}}$
  $\text{\text{-with F -do -coding_theoretic_activity \}}$
  $\text{\text{-find_CRC_polynomials 3 128 10 \}}$
  $\text{\text{-end \}}$

$\text{crc32_test:}$
  $\text{$\text{(ORBITER)}$ -v 3 \}}$
  $\text{\text{-define F -finite_field -q 2 -end \}}$
  $\text{\text{-with F -do -coding_theoretic_activity \}}$
  $\text{\text{-crc32 "123456789" \}}$
  $\text{\text{-end \}}$

$\text{crc32_test_hexdata:}$
  $\text{$\text{(ORBITER)}$ -v 3 \}}$
  $\text{\text{-define F -finite_field -q 2 -end \}}$
  $\text{\text{-with F -do -coding_theoretic_activity \}}$
  $\text{\text{-crc32_hexdata "7BD11C4010" \}}$
  $\text{\text{-end \}}$

$\text{crc32_Berlekamp_matrix:}$
  $\text{$\text{(ORBITER)}$ -v 2 \}}$
  $\text{\text{-define F -finite_field -q 2 -end \}}$
  $\text{\text{-define v -vector -field F -sparse 33 $(CRC32\_ETHERNET)$ -end \}}$

745
10947 >> >> -with F -do \
10948 >> >> -finite_field_activity \
10949 >> >> >> -Berlekamp_matrix v \
10950 >> >> -end 
10951 
10952 
10953 CRC_F256_roots_771: 
10954 >> $(ORBITER) -v 3 \ 
10955 >> >> -define F -finite_field -q 256 -end \ 
10956 >> >> -with F -do -coding_theoretic_activity \ 
10957 >> >> >> -nth_roots 771 \ 
10958 >> >> -end 
10959 
10960 
10961 # alfa:
10962 
10963 CRC_F256_BCH_code_d2: 
10964 >> $(ORBITER) -v 2 \ 
10965 >> >> -define F -finite_field -q 256 -end \ 
10966 >> >> -define C -code -field F \ 
10967 >> >> >> -BCH 771 2 \ 
10968 >> >> -end \ 
10969 >> >> >> -with C -and F -do -coding_theoretic_activity \ 
10970 >> >> >> >> -export_magma BCH_lq8_n771_d2.magma \ 
10971 >> >> >> -end 
10972 >> pdflatex BCH_codes_q256_n771_d2.tex 
10973 >> open BCH_codes_q256_n771_d2.pdf 
10974 
10975 # degree of polynomial = 3 
10976 #-dense "214,167,1" 
10977 #-sparse "214,0,167,1,1,2" 
10978 
10979 
10980 
10981 CRC_POLY_Q256_DEG2_DENSE="214,167,1" 
10982 
10983 
10984 CRC_F256_BCH_write_code_for_division_d2: 
10985 >> $(ORBITER) -v 2 \ 
10986 >> >> -define F -finite_field -q 256 -end \ 
10987 >> >> -define A -vector -field F -sparse 772 "1,771,1,0" -end \ 
10988 >> >> -define B -vector -field F -dense $(CRC_POLY_Q256_DEG2_DENSE) -end \ 
10989 >> >> -with F -do \ 
10990 >> >> -coding_theoretic_activity \ 
10991 >> >> >> -write_code_for_division \ 
10992 >> >> >> alfa A B \ 
10993 >> >> -end
g++ crcalfa.cpp -o crcalfa.out
./crcalfa.out

# bravo:

# degree of polynomial = 4
#-dense "1,23,27,213,1"
#-sparse "1,0,23,1,27,2,213,3,1,4"

CRC_F256_BCH_code_d3:

$(ORBITER) -v 2 \
-define F -finite_field -q 256 -end \
-define C -code -field F \
-define BCH 771 3 \
-with C -and F -do -coding_theoretic_activity \
-define BCH 771 3 \
-export_magma BCH_lq8_n771_d3.magma \
-end

pdflatex BCH_codes_q256_n771_d3.tex
open BCH_codes_q256_n771_d3.pdf

CRC_POLY_BRAVO_DENSE="1,23,27,213,1"

CRC_F256_BCH_write_code_for_division_Bravo:

$(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_BRAVO_DENSE) -end \
-with F -do \n-coding_theoretic_activity \n-write_code_for_division \n-braavo A B \n-end

g++ crcbravo.cpp -o crcbravo.out
./crcbravo.out

# Charlie

CRC_F256_BCH_code_d7:

$(ORBITER) -v 2 \
-define F -finite_field -q 256 -end \n
---

11041  \>  \>  \> -define C -code -field F \\
11042  \>  \>  \> -BCH 771 7 \\
11043  \>  \>  \> -end \\
11044  \>  \>  \> -with C -and F -do -coding.theoretic_activity \\
11045  \>  \>  \> -export_magma BCH_lq8_n771_d7.magma \\
11046  \>  \>  \> -end \\
11047  \>  \>  \> pdflatex BCH_codes.q256_n771_d7.tex \\
11048  \>  \>  \> open BCH_codes_q256_n771_d7.pdf \\
11049 \\
11050 \\
11051 # polynomial of degree 12:
11052 #-dense "1,126,25,1,196,209,244,3,121,126,35,65,1"
11053 #-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"
11054 
11055 
11056 CRC_POLY_CHARLIE_DENSE="1,126,25,1,196,209,244,3,121,126,35,65,1"
11057 
11058 CRC_F256_BCH_write_code_for_division_Charlie:
11059  \> \$(ORBITER) -v 2 \\
11060  \>  \> -define F -finite_field -q 256 -end \\
11061  \>  \> -define A -vector -field F -sparse 772 "1,771,1,0" -end \\
11062  \>  \> -define B -vector -field F -dense $(CRC_POLY_CHARLIE_DENSE) -end \\
11063  \>  \> -with F -do \\
11064  \>  \> -coding.theoretic_activity \\
11065  \>  \>  \> -write_code_for_division \\
11066  \>  \>  \>  \> charlie A B \\
11067  \>  \>  \> -end \\
11068  \>  g++ crc_charlie.cpp -o crc_charlie.out \\
11069  \>  ./crc_charlie.out \\
11070 
11071 
11072 
11073 
11074 
11075 
11076 F256_BCH_code_d16:
11077  \> \$(ORBITER) -v 3 \\
11078  \>  \> -define F -finite_field -q 256 -end \\
11079  \>  \> -define C -code -field F \\
11080  \>  \>  \> -BCH 771 16 \\
11081  \>  \>  \> -end \\
11082  \> pdflatex BCH_codes_q256_n771_d16.tex \\
11083  \> open BCH_codes_q256_n771_d16.pdf \\
11084 
11085 #generator polynomial is $X^{-30} + 253X^{-29} + 174X^{-28} + 109X^{-27} + 97X^{-26} + 144X^{-25} + 112X^{-24} + 212X^{-23} + 192X^{-22} + 169X^{-21} + 24X^{-20} + 150X^{-19} + 110X^{-18} + 248X^{-17} + 3X^{-16} + 193X^{-15} + 194X^{-14} + 205X^{-13} + 9X^{-12} + 748$
56X^{11} + 95X^{10} + 199X^{9} + 108X^{8} + 58X^{7} + 160X^{6} + 148X^{5} + 138X^{4} + 24X^{3} + 210X^{2} + 26X + 1

```
POLY_Q256_DEG30_SPARSE="1,0,26,1,210,2,24,3,138,4,148,5,160,6,58,7,108,8,199,9,95,10,56,11,9,12,205,13,194,14,193,15,3,16,248,17,110,18,150,19,24,20,169,21,192,22,212,23,112,24,25,144,25,97,26,109,27,174,28,253,29,130"

POLY_Q256_DEG30_DENSE="1,26,210,24,138,148,160,58,108,199,95,56,9,205,194,193,3,248,17,110,150,24,169,212,112,144,253,1"
```

```
F256_BCH_write_code_for_division_d16:

```
F256_BCH_code_d16_division:

```
F256_BCH_code_d16_error:

```
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

#define F -finite_field -q 2 -end

with F -do
-coding_theoretic_activity
-crc_encode_file_based $(CRC_FILE).$(CRC_FILE_EXTENSION) $(CRC_FILE)_crc16.bin crc16 771 -end

#crc_encode new:

define F -finite_field -q 256 -end

-crc_encode_file_based $(CRC_FILE).$(CRC_FILE_EXTENSION) $(CRC_FILE)_crc32.bin crc32 771 -end
introduce_errors_16_500000:
$(ORBITER) -v 3
-introduce_errors
-input $(CRC_FILE)_crc16.bin
-output $(CRC_FILE)_crc16.e.bin
-block_based_error_generator
-block_length 771
-threshold 500000
-file_based_error_generator 500000
-nb_repeats 30
-end

introduce_errors_32_100000:
$(ORBITER) -v 3
-introduce_errors
-input $(CRC_FILE)_crc32.bin
-output $(CRC_FILE)_crc32.e.bin
-block_based_error_generator
-block_length 771
-threshold 100000
-file_based_error_generator 100000
-nb_repeats 30
-end

check_errors_16:
$(ORBITER) -v 3
-check_errors
-input $(CRC_FILE)_crc16.e.bin
-output $(CRC_FILE)_recovered.$(CRC_FILE_EXTENSION)
-crc_type crc16
-error_log $(CRC_FILE)_crc16.e_pattern.csv
-block_length 771
-end

check_errors_32:
$(ORBITER) -v 3
-check_errors
-input $(CRC_FILE)_crc32.e.bin
-output $(CRC_FILE)_recovered.$(CRC_FILE_EXTENSION)

extract_block:
 $(ORBITER) -v 3 \n extract_block \n -input $(CRC_FILE)_crc32_e.bin \n -output $(CRC_FILE)_recovered.$(CRC_FILE_EXTENSION) \n -error_log $(CRC_FILE)_crc32_e_pattern.csv \n -block_length 771 \n -selected.block 813731 \n -end

# Section 10.5: Coding Theory - Reed-Muller codes

SECTION CODING THEORY REED MULLER CODES:

test_10_5:
 make RM_3_1_Hamming_space_diagram
 make RM_3_1_draw
 make RM_3_1_split
 make RM_3_1_holes_draw
 make RM_3_1_hole0
 make RM_3_1_hole1
 make RM_3_1_hole2
 make RM_4_1
 make RM_4_1_diagram
 make RM_4_1_diagram_draw
 make RM_4_1_split
 make RM_4_1_diagram_holes
 make RM_4_1_diagram_metric_balls
 make RM_4_1_hole0

RM_3_1_Hamming_space_diagram:
 $(ORBITER) -v 2 \n
-define F -finite_field -q 2 -end \\
-with F -do \\
-coding_theoretic_activity \\
-code_diagram "RM_3_1" \\
$(REED_MULLER_3_1_CODEWORDS) 8 \\
-metric_balls 1 \\
-end \\

# creates 
RM_3_1_diagram_8_16.tex
RM_3_1_diagram_01_8_16.csv
RM_3_1_holes_8_16.csv
RM_3_1_split:
RM_3_1_holes_draw:

$(ORBITER) -v 2 \\
draw_matrix \\
-input_csv_file RM_3_1_holes_8_16.csv \\
-box_width 25 -bit_depth 8 \\
-partition 4 16 16 \\
-end \\

$(ORBITER) -v 2 \\
draw_matrix \\
-input_csv_file RM_3_1_diagram_01_8_16.csv \\
-box_width 25 -bit_depth 8 \\
-partition 4 16 16 \\
-end \\

$(ORBITER) -v 2 \\
draw_matrix \\
-input_csv_file RM_3_1_diagram_8_16.csv \\
-box_width 25 -bit_depth 8 \\
-partition 4 16 16 \\
-end \\

open RM_3_1_diagram_8_16_draw.bmp
RM_3_1_split:
RM_3_1_holes_draw:

$(ORBITER) -v 2 \\
-loop L 0 3 1 \\
draw_matrix \\
-input_csv_file RM_3_1_holes_8_16_value%L.csv \\
-box_width 25 -bit_depth 8 -partition 5 16 16 \\
-end \\

# creates 
RM_3_1_diagram_8_16.tex
RM_3_1_diagram_01_8_16.csv
RM_3_1_holes_8_16.csv
RM_3_1_split:
RM_3_1_holes_draw:

$(ORBITER) -v 2 \\
-loop L 0 3 1 \\
draw_matrix \\
-input_csv_file RM_3_1_holes_8_16_value%L.csv \\
-box_width 25 -bit_depth 8 -partition 5 16 16 \\
-end \\

open RM_3_1_diagram_8_16_draw.bmp
RM_3_1_split:
RM_3_1_holes_draw:

$(ORBITER) -v 2 \\
-loop L 0 3 1 \\
draw_matrix \\
-input_csv_file RM_3_1_holes_8_16_value%L.csv \\
-box_width 25 -bit_depth 8 -partition 5 16 16 \\
-end \\

open RM_3_1_diagram_8_16_draw.bmp
RM_3_1.hole0:

RM_3_1.hole1:
-define v -vector -field F -file RM_3.1_holes_8.16_value1.csv -end \n
# define F -finite_field -q 2 -end \n
# define v -vector -field F -file RM_3.1_holes_8.16_value2.csv -end \n
# with F -do -finite_field_activity \n
# algebraic_normal_form \n
# end

# define v -vector -field F -file RM_3.1_holes_8.16_value1.csv -end \n
# define F -finite_field -q 2 -end \n
# define v -vector -field F -file RM_3.1_holes_8.16_value2.csv -end \n
# with F -do -finite_field_activity \n
# algebraic_normal_form \n
# end

#X0*X8^7 + X1*X8^7 + X2*X8^7 + X3*X8^7 + X4*X8^7 + X5*X8^7 + X6*X8^7 + X7*X8^7 = E_1

#E_1 = X_0X_8^7 + X_1X_8^7 + X_2X_8^7 + X_3X_8^7 + X_4X_8^7 + X_5X_8^7 + X_6X_8^7 + X_7X_8^7

# end
\[\sum_{i\in\{1,\ldots,8\}} X_i^4 + X_2 X_4 X_5 X_7 X_8^4 + X_2 X_4 X_6 X_7 X_8^4 + X_2 X_5 X_6 X_7 X_8^4 + X_3 X_4 X_5 X_7 X_8^4 + X_3 X_4 X_6 X_7 X_8^4 + X_3 X_5 X_6 X_7 X_8^4 + X_3 X_4 X_5 X_7 X_8^4 + X_3 X_4 X_6 X_7 X_8^4 + X_3 X_5 X_6 X_7 X_8^4\]

\[E_2 + E_3 + E_4\]

RM\_4\_1:

```
$(ORBITER) -v 2 \
-define F -finite_field -q 2 -end \
-define C -code -field F -first_order_Reed_Muller 4 -end \
-with C -and F -do -coding_theoretic_activity \ 
-export_magma RM\_4\_1.magma \ 
-end \
-with C -and F -do -coding_theoretic_activity \ 
-export_codewords RM\_4\_1_codewords.csv \ 
-end
```

ToDo: codewords_n16_k5_q2.csv does not exist

RM\_4\_1_diagram:

```
$(ORBITER) -v 2 \
-define F -finite_field -q 2 -end \
-with F -do \
-coding_theoretic_activity \ 
-code_diagram_from_file "RM\_4\_1" \ 
codewords_n16_k5_q2.csv 16 \ 
-end
```

```
#-enhance 4 
#-metric_balls 3
```

RM\_4\_1_diagram_draw:

```
$(ORBITER) -v 2 \
-draw_matrix \
-input_csv_file RM\_4\_1_diagram_01_16_32.csv \ 
-box_width 25 -bit_depth 8 \
-partition 10 256 256 \
-end
open RM\_4\_1_diagram_01_16_32_draw.bmp
```

RM\_4\_1_split:

```
$(ORBITER) -split_by_values RM\_4\_1_holes_16_32.csv
```
11398 RM_4_1_diagram_draw_holes:
11399 \> $(ORBITER) -v 2 \n11400 \> \> -draw_matrix \n11401 \> \> \> -input_csv_file RM_4_1_holes_16_32.csv \n11402 \> \> \> -box_width 5 -bit_depth 8 \n11403 \> \> \> -partition 10 256 256 \n11404 \> \> -end
11405 \> $(ORBITER) -v 2 \n11406 \> \> -loop L 0 7 1 \n11407 \> \> \> -draw_matrix \n11408 \> \> \> \> -input_csv_file RM_4_1_holes_16_32.value%L.csv \n11409 \> \> \> \> -box_width 5 -bit_depth 8 \n11410 \> \> \> \> -partition 10 256 256 \n11411 \> \> \> \> -end \n11412 \> \> \> -end_loop
11413
11414
11415
11416 RM_4_1_diagram_metric_balls:
11417 \> $(ORBITER) -v 2 \n11418 \> \> -define F -finite_field -q 2 -end \n11419 \> \> -with F -do \n11420 \> \> -coding_theoretic_activity \n11421 \> \> \> -code_diagram_from_file "RM_4_1" \n11422 \> \> \> \> codewords_n16_k5_q2.csv 16 \n11423 \> \> \> \> -metric_balls 3 \n11424 \> \> \> -end
11425 \> $(ORBITER) -v 2 \n11426 \> \> -draw_matrix \n11427 \> \> \> -input_csv_file RM_4_1_diagram_16_32.csv \n11428 \> \> \> -box_width 25 -bit_depth 8 \n11429 \> \> \> -partition 10 256 256 \n11430 \> \> -end
11431
11432
11433
11434 RM_4_1_hole0:
11435 \> $(ORBITER) -v 3 \n11436 \> \> -define F -finite_field -q 2 -end \n11437 \> \> -define v -vector -field F -file RM_4_1_holes_16_32.value0.csv -end \n11438 \> \> -with F -do -finite_field.activity \n11439 \> \> \> -algebraic_normal_form \n11440 \> \> \> 16 v \n11441 \> \> \> -end
11442
11443
11444
Section 10.6: Coding Theory - BCH Codes

SECTION CODING THEORY BCH CODES:

test 10.6:

```
make draw_cyclotomic_mod_21_q8
make F_8_BCH_code_d3
make F_8_BCH_code_d4
make F_8_BCH_code_d5
make F_8_BCH_code_d5_minimum_distance
make F_8_BCH_code_d7
make F8_BCH_code_n63_d43
make F8_BCH_code_n63_d43_minimum_distance
make F2_BCH_code_n21
make F7_RS_code_n6
make F_64_again
make BCH_255_5_evaluate_elementary_symmetric_functions_1
make BCH_255_5_evaluate_elementary_symmetric_functions_2
make BCH15
make draw_mod_31
make PR127
make draw_mod_127_power
make draw_mod_251
make draw_mod_255_cyclotomic_1
make draw_mod_255_cyclotomic_3
make draw_mod_255_cyclotomic_1_and_3
make draw_mod_63_4_cyclotomic_3_6
make BCH_F_64
make BCH_elementary_symmetric_functions_3
make BCH_63_4_evaluate_elementary_symmetric_functions_1
make BCH_63_4_evaluate_elementary_symmetric_functions_2
make BCH_21_poly_mult_mod_F4
make BCH_21_poly_division_a
make BCH_21_poly_division_b
make BCH_21_poly_division_ab
make BCH_21_generator_matrix
make BCH_21_15_weight Enumerator
make BCH_21_15_dual
make BCH_21_6_weight Enumerator
make BCH_21_6_4_macwilliams
make BCH_21_15_4_field_reduction
make BCH_21_poly_division_c
```
11492 \texttt{make F16\_roots\_5}
11493 \texttt{make F64\_roots\_21}
11494 \texttt{make BCH\_F256\_roots\_771}
11495 \texttt{make BCH\_F256\_BCH\_code\_d16}
11496
11497
11498 \texttt{draw\_cyclotomic\_mod\_21\_q8:}
11500 \texttt{$(ORBITER) -v 2 \backslash}
11501 \texttt{\> -draw\_options \}
11502 \texttt{\> \> -radius 100 \}
11503 \texttt{\> \> \> -line\_width 1.0 -embedded \}
11504 \texttt{\> \> -end \}
11505 \texttt{\> \> -draw\_mod\_n -n 21 -file mod\_21\_cyclotomic \}
11506 \texttt{\> \> -cyclotomic\_sets 8 "1,2,4,5,7,10,13" -end}
11507 \texttt{pdflatex mod\_21\_cyclotomic\_draw.tex}
11508 \texttt{open mod\_21\_cyclotomic\_draw.pdf}
11509
11510
11511 \texttt{F\_8\_BCH\_code\_d3:}
11512 \texttt{$(ORBITER) -v 3 \backslash}
11513 \texttt{\> -define F -finite\_field -q 8 -override\_polynomial 11 -end \}
11514 \texttt{\> \> -define C -code -field F \}
11515 \texttt{\> \> \> -BCH 21 3 \}
11516 \texttt{\> \> -end}
11517 \texttt{pdflatex BCH\_codes\_q8\_n21\_d3.tex}
11518 \texttt{open BCH\_codes\_q8\_n21\_d3.pdf}
11519
11520 \#generator polynomial is $X^4 + 4X^3 + 4X^2 + 3X + 4$
11521
11522 \texttt{F\_8\_BCH\_code\_d4:}
11523 \texttt{$(ORBITER) -v 3 \backslash}
11524 \texttt{\> -define F -finite\_field -q 8 -override\_polynomial 11 -end \}
11525 \texttt{\> \> -define C -code -field F \}
11526 \texttt{\> \> \> -BCH 21 4 \}
11527 \texttt{\> \> -end}
11528
11529 \#generator polynomial is $X^5 + 6X^4 + 7X^3 + 2X + 3$
11530
11531
11532 \texttt{F\_8\_BCH\_code\_d5:}
11533 \texttt{$(ORBITER) -v 3 \backslash}
11534 \texttt{\> -define F -finite\_field -q 8 -override\_polynomial 11 -end \}
11535 \texttt{\> \> -define C -code -field F \}
11536 \texttt{\> \> \> -BCH 21 5 \}
11537 \texttt{\> \> -end}
11538 \texttt{pdflatex BCH\_codes\_q8\_n21\_d5.tex}
open BCH_codes_q8_n21_d5.pdf

-override_polynomial 11

#generator polynomial is \(X^7 + 3X^6 + 3X^5 + 2X^4 + X^3 + 2X^2 + X + 2\)

#CODE BCH F8 N21 D5 GENMA

CODE_BCH_F8_N21_D5_GENMA_OVERRIDE_POLYNOMIAL11="\n2,1,2,1,2,3,3,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,\n0,2,1,2,1,2,3,3,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,\n0,0,0,0,0,0,2,1,2,1,2,3,3,1,0,0,0,0,0,0,0,0,0,0,\n0,0,0,0,0,0,0,2,1,2,1,2,3,3,1,0,0,0,0,0,0,0,0,0,\n0,0,0,0,0,0,0,0,2,1,2,1,2,3,3,1,0,0,0,0,0,0,0,0,\n0,0,0,0,0,0,0,0,0,2,1,2,1,2,3,3,1,0,0,0,0,0,0,0,\n0,0,0,0,0,0,0,0,0,0,2,1,2,1,2,3,3,1,0,0,0,0,0,0,\n0,0,0,0,0,0,0,0,0,0,0,2,1,2,1,2,3,3,1,0,0,0,0,0,\n0,0,0,0,0,0,0,0,0,0,0,0,2,1,2,1,2,3,3,1,0,0,0,0,\n0,0,0,0,0,0,0,0,0,0,0,0,0,2,1,2,1,2,3,3,1,0,0,0,\n0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,1,2,1,2,3,3,1,0,0,\n0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,1,2,1,2,3,3,1,0,\n0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,1,2,1,2,3,3,1,\nF 8 BCH code d5 minimum distance:
-with F -do \\
-coding-theoretic_activity \\
\-minimum_distance v \\
\-end

# important: use the same polynomial as when creating the code.

# d=5

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

F8_BCH_code_n63_d43:

\$(ORBITER) -v 3 \\
\-define F -finite_field -q 8 -override_polynomial 11 -end \\
\-define C -code -field F \\
\-BCH 63 43 \\
\-end

pdflatex BCH_codes_q8_n63_d43.tex

open BCH_codes_q8_n63_d43.pdf
F8 BCH code n63 d43 minimum distance:

```bash
$(ORBITER) -v 2 -define F -finite_field -q 8 -override_polynomial 11 -end
-define v -vector -format 9 -field F
-compact $(CODE_BCH_F8_N63_K9_D43_GENMA)
-compact $(CODE_BCH_F8_N63_K9_D43_GENMA)
-coding_theoretic_activity
-minimum_distance v
-end
```

#coding_theory_domain::do_minimum_distance The minimum distance is d = 45, computed in 0 days, 0 hours, 1 minutes, 32 seconds

1:32

# ToDo:

F7 RS code n6:

```bash
$(ORBITER) -v 3
-define F -finite_field -q 7 -end
-define C -code -field F
-BCH 6 3
-end
```

F64 again:

```bash
```
BCH_255_5_evaluate_elementary_symmetric_functions_1:
$(ORBITER) -v 3 -define F -finite_field -q 256 -end \n$ORBITER -v 3 -define F -finite_field -q 64 -end \n$ORBITER -define F -finite_field -q 64 -end \n$ORBITER -define F -finite_field -q 256 -end \n$ORBITER -define e1 -formula "e1" "e1" "" $(ELEMENTARY_SYMMEtric_8.1) \n$ORBITER -define e2 -formula "e2" "e2" "" $(ELEMENTARY_SYMMEtric_8.2) \n$ORBITER -define e3 -formula "e3" "e3" "" $(ELEMENTARY_SYMMEtric_8.3) \n$ORBITER -define e4 -formula "e4" "e4" "" $(ELEMENTARY_SYMMEtric_8.4) \n$ORBITER -define e5 -formula "e5" "e5" "" $(ELEMENTARY_SYMMEtric_8.5) \n$ORBITER -define e6 -formula "e6" "e6" "" $(ELEMENTARY_SYMMEtric_8.6) \n$ORBITER -define e7 -formula "e7" "e7" "" $(ELEMENTARY_SYMMEtric_8.7) \n$ORBITER -define e8 -formula "e8" "e8" "" $(ELEMENTARY_SYMMEtric_8.8) \n$ORBITER -define E8 -collection "e1,e2,e3,e4,e5,e6,e7,e8" \n$ORBITER -with F -do -finite_field_activity \n$ORBITER -evaluate E8 "x0=2,x1=4,x2=16,x3=29,x4=76,x5=157,x6=95,x7=133" -end

BCH_255_5_evaluate_elementary_symmetric_functions_2:
$(ORBITER) -v 3 -define F -finite_field -q 256 -end \n$ORBITER -define e1 -formula "e1" "e1" "" $(ELEMENTARY_SYMMEtric_8.1) \n$ORBITER -define e2 -formula "e2" "e2" "" $(ELEMENTARY_SYMMEtric_8.2) \n$ORBITER -define e3 -formula "e3" "e3" "" $(ELEMENTARY_SYMMEtric_8.3) \n$ORBITER -define e4 -formula "e4" "e4" "" $(ELEMENTARY_SYMMEtric_8.4) \n$ORBITER -define e5 -formula "e5" "e5" "" $(ELEMENTARY_SYMMEtric_8.5) \n$ORBITER -define e6 -formula "e6" "e6" "" $(ELEMENTARY_SYMMEtric_8.6) \n$ORBITER -define e7 -formula "e7" "e7" "" $(ELEMENTARY_SYMMEtric_8.7) \n$ORBITER -define e8 -formula "e8" "e8" "" $(ELEMENTARY_SYMMEtric_8.8) \n$ORBITER -define E8 -collection "e1,e2,e3,e4,e5,e6,e7,e8" \n$ORBITER -with F -do -finite_field_activity \n$ORBITER -evaluate E8 "x0=8,x1=4,x2=16,x3=29,x4=76,x5=157,x6=95,x7=133" -end

BCH15:
$(ORBITER) -v 3 -define F -finite_field -q 2 -end \n$ORBITER -define F -finite_field -q 2 -end \n$ORBITER -with F -do \n$ORBITER -coding_theoretic_activity \n$ORBITER -BCH 15 2 5 \n$ORBITER -end

763
draw_mod_31:
  > $(ORBITER) -v 2 \
  > -draw_options -embedded -end \n  > -draw_mod_n 31 \n  > -draw_mod_n_power_cycle 2 \n  > -end
  > pdflatex mod_31_draw.tex
  > open mod_31_draw.pdf

PR127:
  > $(ORBITER) -v 5 -primitive_root 127

draw_mod_127_power:
  > $(ORBITER) -v 2 \
  > -draw_options -scale 0.4 -embedded -end \n  > -draw_mod_n 127 mod_127 -draw_mod_n_power_cycle 3
  > pdflatex mod_127_draw.tex
  > open mod_127_draw.pdf

draw_mod_251:
  > $(ORBITER) -v 2 \
  > -draw_options -nodes_empty -radius 10 -embedded -end \n  > -draw_mod_n 251 mod_251
  > pdflatex mod_251_draw.tex
  > open mod_251_draw.pdf

#-draw_mod_n_inverse

draw_mod_255_cyclotomic_1:
  > $(ORBITER) -v 2 \
  > -draw_options -nodes_empty -radius 10 \n  > -line_width 0.4 -embedded -end \n  > -draw_mod_n -n 255 -file mod_255_cyclotomic_1 \n  > -cyclotomic_sets 2 "1" -end
  > pdflatex mod_255_cyclotomic_1_draw.tex
  > open mod_255_cyclotomic_1_draw.pdf

draw_mod_255_cyclotomic_3:
11763 $\text{(ORBITER)} -v 2 \$
11764 $\text{draw_options -nodes_empty -radius 10} \$
11765 $\text{draw_mod_n -n 255 -file mod\_255\_cyclotomic\_3} \$
11766 $\text{draw_mod_n -n 255 -file mod\_255\_cyclotomic\_3} \$
11767 $\text{cyclotomic\_sets 2 "3" -end} \$
11768 $\text{pdflatex mod\_255\_cyclotomic\_3\_draw.tex} \$
11769 $\text{open mod\_255\_cyclotomic\_3\_draw.pdf} \$
11770
draw\_mod\_255\_cyclotomic\_1\_and\_3:\n11771 $\text{(ORBITER)} -v 2 \$
11772 $\text{draw_options -nodes_empty -radius 10} \$
11773 $\text{draw_mod_n -n 255 -file mod\_255\_cyclotomic\_1\_and\_3} \$
11774 $\text{cyclotomic\_sets 2 "1,3" -end} \$
11775 $\text{pdflatex mod\_255\_cyclotomic\_1\_and\_3\_draw.tex} \$
11776 $\text{open mod\_255\_cyclotomic\_1\_and\_3\_draw.pdf} \$
11777
draw\_mod\_63\_4\_cyclotomic\_3\_6:\n11778 $\text{(ORBITER)} -v 2 \$
11779 $\text{draw_options -radius 20} \$
11780 $\text{draw_mod_n -n 63 -file mod\_63\_4\_cyclotomic\_3\_6} \$
11781 $\text{cyclotomic\_sets 4 "3,6" -end} \$
11782 $\text{pdflatex mod\_63\_4\_cyclotomic\_3\_6\_draw.tex} \$
11783 $\text{open mod\_63\_4\_cyclotomic\_3\_6\_draw.pdf} \$
11784
draw\_mod\_63\_4\_evaluate\_elementary\_symmetric\_functions\_1:\n11785 $\text{(ORBITER)} -v 3 \$
11786 $\text{define F -finite_field -q 64 -end} \$
11787 $\text{define e1 -formula "e1" "e1" " $\text{elementary\_symmetric\_functions\_3} \$
11788 $\text{define e2 -formula "e2" "e2" " $\text{elementary\_symmetric\_functions\_3} \$
11789 $\text{define e3 -formula "e3" "e3" " $\text{elementary\_symmetric\_functions\_3} \$
11790 $\text{define E3 -collection "e1,e2,e3"}$
#The values of the formulae are:
0 : 57
1 : 0
2 : 1

# poly: 1,0,2,1

BCH_63_4_evaluate_elementary_symmetric_functions_2:

#The values of the formulae are:
0 : 56
1 : 0
2 : 1

# poly: 1,0,3,1

BCH_21_poly_mult_mod_F4:

#C(X)=X^{6} + X^{5} + X^{4} + X^{2} + 1
d # poly 1,0,1,0,1,1,1
11857 ▶ ▶ -define F -finite_field -q 4 -end \\
11858 ▶ ▶ -with F -do \\
11859 ▶ ▶ -finite_field_activity \\
11860 ▶ ▶ -polynomial_division \\
11861 ▶ ▶ ▶ "1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1" \\
11862 ▶ ▶ ▶ "1,0,2,1" \\
11863 ▶ ▶ -end \\
11864 \\
11865 BCH_21_poly_division_b:
11866 ▶ $(ORBITER) -v 2 \\
11867 ▶ ▶ -define F -finite_field -q 4 -end \\
11868 ▶ ▶ -with F -do \\
11869 ▶ ▶ -finite_field_activity \\
11870 ▶ ▶ -polynomial_division \\
11871 ▶ ▶ ▶ "1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1" \\
11872 ▶ ▶ ▶ "1,0,3,1" \\
11873 ▶ ▶ -end \\
11874 \\
11875 \\
11876 BCH_21_poly_division_ab:
11877 ▶ $(ORBITER) -v 2 \\
11878 ▶ ▶ -define F -finite_field -q 4 -end \\
11879 ▶ ▶ -with F -do \\
11880 ▶ ▶ -finite_field_activity \\
11881 ▶ ▶ -polynomial_division \\
11882 ▶ ▶ ▶ "1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1" \\
11883 ▶ ▶ ▶ "1,0,1,0,1,1,1" \\
11884 ▶ ▶ -end \\
11885 \\
11886 BCH_21_generator_matrix:
11887 ▶ $(ORBITER) -v 2 \\
11888 ▶ ▶ -define F -finite_field -q 4 -end \\
11889 ▶ ▶ -with F -do \\
11890 ▶ ▶ -coding_theoretic_activity \\
11891 ▶ ▶ ▶ -generator_matrix_cyclic_code \\
11892 ▶ ▶ ▶ ▶ 21 "1,0,1,0,1,1,1" \\
11893 ▶ ▶ ▶ -end \\
11894 \\
11895 \\
11896 \\
11897 \\
11898 BCH_21_15_weightenumerator:
11899 ▶ $(ORBITER) -v 2 \\
11900 ▶ ▶ -define F -finite_field -q 4 -end \\
11901 ▶ ▶ -define v -vector -field F -format 15 \\
11902 ▶ ▶ ▶ -dense $(BCH_21_15_GENERATOR_MATRIX) \\
11903 ▶ ▶ -end \\
767
-define C -code -field F \
-linear_code_through_generator_matrix v \
-end \n-with C -do \n-coding_theoretic_activity \n-weight Enumerator \n-end \n
# too slow!

BCH_21_15_dual:

$(ORBITER) -v 2 \n
$-define F -finite_field -q 4 -end \n-$define v -vector -field F -format 15 \n-$define $(BCH_21_15_GENERATOR_MATRIX) -end \n-with F -do -finite_field_activity \n-nullspace v \n-normalize_from_the_right \n-end \n
BCH_21_6_weight Enumerator:

$(ORBITER) -v 2 \n-$define F -finite_field -q 4 -end \n-$define v -vector -format 6 -field F \n-$define $(BCH_21_6_GENERATOR_MATRIX) \n-end \n
C -code -field F \n-linear_code_through_generator_matrix v \n-end \n-with F -do \n-coding_theoretic_activity \n-weight Enumerator \n-end \n
# 1y^{21} + 63x^8y^{13} + 294x^{12}y^9 + 756x^{14}y^7 + 1890x^{16}y^5 + 1092x^{18}y^3

#( 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 63, 0, 0, 0, 0, 0, 294, 0, 0, 0, 294, 0, 0, 0, 0, 756, 0, 0, 0, 0, 0, 0, 1890, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1092, 0, 0, 0, 0 )

BCH_21_6_4_macwilliams:

$(ORBITER) -v 2 \n-make_macwilliams_system 21 6 4

pdflatex MacWilliams_n21_k6_q4.tex
open MacWilliams_n21_k6_q4.pdf
\[
\begin{align*}
\text{ww} & := [1, 0, 0, 84, 252, 1575, 10080, 5133744, 15282792, 37951620, 79336530, 135622080, 213273081, 188911548, 125744304, 59721732, 17767512, 2580255] \\
\end{align*}
\]

BCH_21.15.4_field_reduction:

```
$(ORBITER) -v 2 -define F -finite_field -q 4 -end -with F -do -finite_field_activity -field_reduction "BCH_21.15.4" 2 15 21 $(BCH_21.15) -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 -nth_roots 5 -end
```

```
pdflatex Nth_roots_q2_n5.tex
```

```
open Nth_roots_q2_n5.pdf
```

F16_roots_5:

```
$(ORBITER) -v 3 -define F -finite_field -q 2 -end -with F -do -coding_theoretic_activity -nth_roots 5 -end
```

```
pdflatex Nth_roots_q2_n5.tex
```

```
open Nth_roots_q2_n5.pdf
```

769
11994
11995
11996
11997  F64\_roots\_21:
11998  \$ (ORBITER) -v 3 \n11999  \&\& -define F -finite\_field -q 2 -end \n12000  \&\& -with F -do -coding\_theoretic\_activity \n12001  \&\& \&\& -nth\_roots 21 \n12002  \&\& \&\& -end
12003  pdflatex Nth\_roots\_q2\_n21.tex
12004  open Nth\_roots\_q2\_n21.pdf
12005
12006
12007
12008  BCH\_F256\_roots\_771:
12009  \$ (ORBITER) -v 3 \n12010  \&\& -define F -finite\_field -q 256 -end \n12011  \&\& -with F -do -coding\_theoretic\_activity \n12012  \&\& \&\& -nth\_roots 771 \n12013  \&\& \&\& -end
12014
12015
12016  BCH\_F256\_BCH\_code\_d16:
12017  \$ (ORBITER) -v 3 \n12018  \&\& -define F -finite\_field -q 256 -end \n12019  \&\& -define C -code -field F \n12020  \&\& \&\& -BCH 771 16 \n12021  \&\& \&\& -end
12022  pdflatex BCH\_codes\_q256\_n771\_d16.tex
12023  open BCH\_codes\_q256\_n771\_d16.pdf
12024
12025
12026  #generator polynomial is X^{30} + 253X^{29} + 174X^{28} + 109X^{27} + 97X^{26} + 
12027  144X^{25} + 112X^{24} + 212X^{23} + 192X^{22} + 169X^{21} + 24X^{20} + 150X^{19} + 
12028  110X^{18} + 248X^{17} + 3X^{16} + 193X^{15} + 194X^{14} + 205X^{13} + 9X^{12} + 
12029  56X^{11} + 95X^{10} + 199X^{9} + 108X^{8} + 58X^{7} + 160X^{6} + 148X^{5} + 138X^{4} + 
12030  24X^{3} + 210X^{2} + 26X + 1
12031
12032
12033
12034
12035 ###############################################################################
12036  # Section 10.7: Coding Theory - Reed Solomon codes
SECTION CODING THEORY REED.SOLOMON.CODES:

test.10_7:
  make F.7.BCH_code_n6
  make RREF_RS_6.4.7.weight Enumerator
  make Code_RS_11
  make Code_RS_11.weight Enumerator
  make RREF_RS_8.weight Enumerator
  make RS_8.field_reduction
  make RREF_RS_8.reduced.weight Enumerator
  make CODE_21.15.4_store
  make CODE_21.15.4.weight Enumerator
  make CODE_21.15.4.minimum_distance
  make Reed.solomon_F8_work

# ToDo:

F.7.BCH_code_n6:
  $(ORBITER) -v 3 \
  -define F -finite_field -q 7 -end \
  -with F -do \
  -coding_theoretic_activity 7 3 \
  -end

RREF_RS_6.4.7.weight Enumerator:
  $(ORBITER) -v 2 \
  -define F -finite_field -q 7 -end \
  -define v -vector -format 4 -field F \
  -compact $(CODE_RS_6.4.7) \
  -end \
  -define C -code -field F \
  -linear_code_through_generator_matrix v \
  -end \
  -with C -do \
  -coding_theoretic_activity \
  -weight Enumerator \
  -end

#1y^6 + 120x^3y^3 + 360x^4y^2 + 972x^5y + 948x^6
#weight Enumerator:
  $( 1, 0, 0, 120, 360, 972, 948 )
Code_RS_11:
$(ORBITER) -v 2 \
  \-define F -finite_field -q 11 -end \
  \-define v -vector -format 8 -field F \
  \-compact $(CODE_RS_10_8_11) \
  \-end \
  \-with F -do \
  \-finite_field_activity -RREF v -end
$pdflatex RREF.example.q11_8_10.tex
$#gs -sDEVICE=png16 -dFIXEDMEDIA -dDEVICEWIDTHPOINTS=500 -dDEVICEHEIGHTPOINTS=450 \
  \-r240 -oRREF.example.q11_8_10.page%02d.png \
  \#RREF.example.q11_8_10.pdf
$open RREF.example.q11_8_10.pdf

Code_RS_11_weight Enumerator:
$(ORBITER) -v 2 \
  \-define F -finite_field -q 11 -end \
  \-define v -vector -format 8 -field F \
  \-compact $(CODE_RS_11_RREF) \
  \-end \
  \-define C -code -field F \
  \-linear_code_through_generator_matrix v \
  \-end \
  \-with C -do \
  \-coding_theoretic_activity \
  \-weight Enumerator \
  \-end

#1*y^(10) + 1200*x^3*y^7 + 16800*x^4*y^6 + 209160*x^5*y^5 + 1734600*x^6*y^4 + 991800*x^7*y^3 + 37189800*x^8*y^2 + 82644700*x^9*y + 82644620*x^(10)

RREF_RS_8.weight Enumerator:
$(ORBITER) -v 2 \
  \-define F -finite_field -q 8 -end \
  \-define v -vector -format 5 -field F \
  \-compact $(CODE_RS_8) \
  \-end \
  \-define C -code -field F \

772
-linear_code_through_generator_matrix v \ 
-end \ 
-with C -do \ 
-coding_theoretic_activity \ 
-weight Enumerator \ 
-end \ 

# the group cannot be computed

RS_8_field_reduction:

$(ORBITER) -v 2 \ 
-define F -finite_field -q 8 -end \ 
-with F -do \ 
-finite_field_activity \ 
-field_reduction "RS_8_red_2" \ 
-2 5 7 $(CODE_RS_8) \ 
-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

RREF_RS_8_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 \ 
-linear_code_through_generator_matrix v \ 
-end \ 
-with C -do \ 
-coding_theoretic_activity \ 
-weight Enumerator \ 
-end

CODE_21_15_4_store:

$(ORBITER) -v 2 \ 
-store_as_csv_file "code_21_15_4.csv" \ 
-15 21 $(CODE_21_15_4) 

$(ORBITER) -v 2 -draw_matrix \
CODE_21_15_4.weight Enumerator:

```plaintext
$\text{ORBITER} -v 2 \\
$\text{define F -finite field -q 2 -end } \\
$\text{define v -vector -format 15 -field F } \\
$\text{compact $(\text{CODE}_21_15_4) } \\
$\text{-end } \\
$\text{-define C -code -field F } \\
$\text{-linear code through generator matrix v } \\
$\text{-end } \\
$\text{-with C -do } \\
$\text{-coding theoretic activity } \\
$\text{-weight enumerator } \\
$\text{-end }
```

12195 CODE_21_15_4.minimum_distance:

```plaintext
$\text{ORBITER} -v 2 \\
$\text{define F -finite field -q 2 -end } \\
$\text{define v -vector -format 15 -field F } \\
$\text{compact $(\text{CODE}_21_15_4) } \\
$\text{-end } \\
$\text{-with F -do } \\
$\text{-coding theoretic activity } \\
$\text{-minimum distance v } \\
$\text{-end }
```

12194 #d=4

12206 Reed solomon F8 work:

```plaintext
$\text{ORBITER} -v 3 -define F -finite field -q 8 -end \\
$\text{with F -do -finite field activity } \\
$\text{parse and evaluate "test" "(t-a)*(t-a*a)" "a=2" -end }
```

12215 # Section 10.8: Coding Theory - Bounds

12218 SECTION_CODING_THEORY_BOUNDS:

12219 test_10_8:

```plaintext
$\text{make bounds for d given n6_k4_q7}$
```
make bounds_for_d_given_n15_k6_q2
make coding_theory_bounds_q2
make coding_theory_bounds_q8
make GV_n15_k6_d5
make bounds_for_d_given_n12_k4_q13
make GV_n15_k6_d5_weight_enumerator
make code_n15_k6_d6_a_we
make code_n15_k6_d6_RREF
make code_n15_k6_d6_check_RREF

bounds_for_d_given_n6_k4_q7:
$(ORBITER) -v 2 -make bounds_for_d_given_n_and_k_and_q 6 4 7

bounds_for_d_given_n15_k6_q2:
$(ORBITER) -v 2 -make bounds_for_d_given_n_and_k_and_q 15 6 2

n = 15 k=6 q=2
d_GV = 5
d_singleton = 10
d_hamming = 6
d_plotkin = 7
d_griesmer = 6

coding_theory_bounds_q2:
$(ORBITER) -v 2 -table_of_bounds 20 2
# produces table_of_bounds_n20_q2.csv

coding_theory_bounds_q8:
$(ORBITER) -v 2 -table_of_bounds 20 8

ToDo

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

# [15,6] code created
bounds_for_d_given_n12_k4_q13:

```
$(ORBITER) -v 2 \ 
-make_bounds_for_d_given_n_and_k_and_q 12 4 13
```

de_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 \ 
-linear_code_through_generator_matrix v \ 
-end \ 
-with C -do \ 
coding_theoretic_activity \ 
-weight Enumerator \ 
-end
```

```
#1y^{15} + 27x^6y^9 + 24x^8y^7 + 9x^{10}y^5 + 3x^{12}y^3
```

```
# surprise: d = 6
```

code_n15_k6_d6_a_we:

```
$(ORBITER) -v 2 \ 
-define F -finite_field -q 2 -end \ 
-define v -vector -format 6 -field F \ 
-compact $(CODE_15_6_6_A) \ 
-end \ 
-define C -code -field F \ 
-linear_code_through_generator_matrix v \ 
-end \ 
-with C -do \ 
coding_theoretic_activity \ 
-weight Enumerator \ 
-end
```

```
#1y^{15} + 27x^6y^9 + 24x^8y^7 + 9x^{10}y^5 + 3x^{12}y^3
```

```
# weight Enumerator
```

```
#1y^{15} + 28x^6y^9 + 21x^8y^7 + 12x^{10}y^5 + 2x^{12}y^3
```
12317
12318
code_n15_k6_d6_RREF:
12320 ▶ $(ORBITER) -v 2 \ 
12321 ▶ ▶ -define F -finite_field -q 2 -end \ 
12322 ▶ ▶ -define v -vector -format 6 -field F \ 
12323 ▶ ▶ ▶ -compact $(CODE_GV_N15_K6) \ 
12324 ▶ ▶ -end \ 
12325 ▶ ▶ -with F -do -finite_field_activity \ 
12326 ▶ ▶ -RREF v -normalize_from_the_right \ 
12327 ▶ ▶ -end
12328 ▶ pdflatex RREF_example_q2_6_15.tex
12329 ▶ open RREF_example_q2_6_15.pdf
12330
code_n15_k6_d6_check_RREF:
12332 ▶ $(ORBITER) -v 2 \ 
12333 ▶ ▶ -define F -finite_field -q 2 -end \ 
12334 ▶ ▶ -define v -vector -format 9 -field F \ 
12335 ▶ ▶ ▶ -compact $(CODE_GV_N15_K6_CHECK) \ 
12336 ▶ ▶ -end \ 
12337 ▶ ▶ -with F -do -finite_field_activity \ 
12338 ▶ ▶ -RREF v -normalize_from_the_right \ 
12339 ▶ ▶ -end
12340 ▶ pdflatex RREF_example_q2_9_15.tex
12341 ▶ open RREF_example_q2_9_15.pdf
12342
12343
12344
12345
12346
12347 # Section 10.9: Coding Theory - Classification
12348
12349 SECTION CODING_THEORY_CLASSIFICATION:
12351
test_10_9:
12353 ▶ make codes_8_4_4
12354 ▶ make codes_8_4_4_draw
12355 ▶ make codes_14_4_9_3
12356 ▶ make codes_15_6_6_2
12357 ▶ make codes_16_5_9_3
12358 ▶ make codes_d4
12359 ▶ make codes_24_12_8
12360 ▶ make codes_24_12_8_draw
12361 ▶ make glyn_arc
12362 ▶ make five_points_in_general
12363 ▶ make codes_q13_12_4

777
# code classification:

codes_8_4_4:

```
$ (ORBITER) -v 6 \
  -orbiter_path $(ORBITER_PATH) \
  -define G \
  -linear_group -PGL 4 2 -end \
  -with G -do \
  -group_theoretic_activity \
  -poset_classification_control \
  -problem_label codes_8_4_4 \
  -draw_options -embedded -radius 250 \
  -line_width 1.0 -spanning_tree -end \
  -end \
  -linear_codes 3 8 \
  -end
```

```
# pdflatex codes_8_4_4.poset_lvl_8.tex
# open codes_8_4_4.poset_lvl_8.pdf
```

codes_8_4_4.draw:

```
$ (ORBITER) -v 3 \
  -draw_layered_graph \
  codes_8_4_4.poset_lvl_8.layered_graph \
  -radius 250 -embedded -line_width 1.0 \
  -y_stretch 1.0 -scale 0.5 \
  -end
```

```
pdflatex codes_8_4_4.poset_lvl_8.draw.tex
```

```
open codes_8_4_4.poset_lvl_8.draw.pdf
```

codes_14_4_9_3:

```
$ (ORBITER) -v 6 \
  -define G \
  -linear_group -PGL 10 3 -end \
  -with G -do \
  -group_theoretic_activity \
  -poset_classification_control \
  -problem_label codes_14_4_9_3 \
  -draw_options \
  -embedded -radius 250 \
  -end \
  -end
```
12411 ▶ ▶  -linear_codes 9 14 \  
12412 ▶ ▶  -end  
12413 ▶  pdflatex codes_14_4_9_3_poset_lvl_13.tex  
12414 ▶  open codes_14_4_9_3_poset_lvl_13.pdf  
12415  
12416  
12417 codes_15_6_6_2:  
12418 ▶  $(ORBITER) -v 6 \  
12419 ▶ ▶  -define G \  
12420 ▶ ▶  -linear_group -PGL 9 2 -end \  
12421 ▶ ▶  -with G -do \  
12422 ▶ ▶  -group_theoretic_activity \  
12423 ▶ ▶ ▶  -poset_classification_control \  
12424 ▶ ▶ ▶ ▶  -problem_label codes_15_6_6_2 \  
12425 ▶ ▶ ▶ ▶ ▶  -draw_options \  
12426 ▶ ▶ ▶ ▶ ▶ ▶  -embedded -radius 250 \  
12427 ▶ ▶ ▶ ▶ ▶ ▶ ▶  -end \  
12428 ▶ ▶ ▶ ▶ ▶  -end \  
12429 ▶ ▶ ▶ ▶  -linear_codes 6 15 \  
12430 ▶ ▶ ▶  -end  
12431 ▶  pdflatex codes_15_6_6_2_poset_lvl_15.tex  
12432 ▶  open codes_15_6_6_2_poset_lvl_15.pdf  
12433  
12434  
12435  
12436 # ToDo  
12437  
12438 codes_16_5_9_3:  
12439 ▶  $(ORBITER) -v 6 \  
12440 ▶ ▶  -codes_classify -n 16 -k 5 -q 3 -d 9 -w -W -lex \  
12441 ▶ ▶  -draw_poset \  
12442 ▶ ▶   -end  
12443 ▶ ▶  
12444  
12445 # 5/31/2020: 28 min 22 sec on Mac  
12446  
12447 #0 : 1 orbits  
12448 #1 : 1 orbits  
12449 #2 : 1 orbits  
12450 #3 : 1 orbits  
12451 #4 : 1 orbits  
12452 #5 : 1 orbits  
12453 #6 : 1 orbits  
12454 #7 : 1 orbits  
12455 #8 : 1 orbits  
12456 #9 : 2 orbits  
12457 #10 : 3 orbits
12458 #11 : 4 orbits
12459 #12 : 5 orbits
12460 #13 : 5 orbits
12461 #14 : 4 orbits
12462 #15 : 3 orbits
12463 #16 : 1 orbits
12464 #total: 36
12465
12466
codes_d4:
12467 $(ORBITER) -v 3 \
12468 -define G -linear_group -PGL 4 2 -end \
12469 -with G -do \
12470 -group.theoretic_activity \
12471 -poset.classification.control -W \
12472 -problem.label codes_r4_d4 \
12473 -embedded -end -linear.codes 4 100 \
12474 -end \
12475 -end
12476
12477
codes_24_12_8:
12478 $(ORBITER) -v 6 \
12479 -orbiter.path $(ORBITER_PATH) \
12480 -define G \
12481 -linear_group -PGL 12 2 -end \
12482 -with G -do \
12483 -group.theoretic_activity \
12484 -poset.classification.control \
12485 -problem.label codes_24_12_8 \
12486 -draw.options -embedded -radius 250 \
12487 -line_width 1.0 -spanning_tree -end \
12488 -end \
12489 -end
12490
codes_24_12_8.poset.tex
12491
codes_24_12_8.poset.pdf
12492
codes_24_12_8.poset lvl 24.layered.graph
12493
codes_24_12_8.draw:
12494 $(ORBITER) -v 3 \
12495 -draw.layered_graph \
12496 codes_24_12_8.poset lvl 24.layered.graph \
12497 -radius 100 -spanning_tree -embedded \
12498 -line_width 0.5 -x_stretch 1.4 \
12499 -scale 0.25 -nodes_empty \
12500
glynn arc:
$(ORBITER) -v 5 \n-orbiter_path $(ORBITER_PATH) \n-define G \n-linear_group -PGGL 5 9 -end \n-with G -do \n-group_theoretic_activity \n-poset_classification_control \n-define G \n-linear_group -PGGL 5 9 -end \n-with G -do \n-group_theoretic_activity \n-poset_classification_control \n-problem_label glynn_arc \n-draw_options -embedded -radius 250 \n-line_width 1.0 -spanning_tree -end \n-end \n-linear_codes 6 10 \n-end

five_points_in_general:
$(ORBITER) -v 5 \n-orbiter_path $(ORBITER_PATH) \n-define G \n-linear_group -PGGL 5 9 -end \n-with G -do \n-group_theoretic_activity \n-poset_classification_control \n-define G \n-linear_group -PGGL 5 9 -end \n-with G -do \n-group_theoretic_activity \n-poset_classification_control \n-problem_label five_points_in_general \n-draw_options -embedded -radius 250 \n-line_width 1.0 -spanning_tree -end \n-end \n-linear_codes 4 5 \n-end

codes_q13_12_4:
$(ORBITER) -v 6 \n
test 11:
make Sym 4 Conj classes
make Sym 10 Conj classes
make Sym 15 Conj classes
make Char Sym 4
make Char Sym 5
make Char Sym 6
make all subsets 10 3
make random k subsets
make rank k subsets test
make Walsh matrix 4
make Dedekind 10 10
make Dedekind 30 2
make Dedekind 100 2
make Elementary symmetric functions 4
make Elementary symmetric functions 8
make file S
make Large set H5
12599 ▶ make Large_set_C13
12600 ▶ make Perm13_Subgroup_C13_13
12601 ▶ make Large_set_mult_C13xS
12602 ▶ make Large_set_mult_C13xSxH5
12603 ▶ make Large_set_mult_C13xSxH5_apply
12604 ▶ make domino_portrait
12605 ▶ make domino_portrait_input
12606
12607 12608
12609 Sym_4_conj_classes:
12610 ▶ $(ORBITER) -v 2 -conjugacy_classes_Sym_n 4
12611
12612 Sym_10_conj_classes:
12613 ▶ $(ORBITER) -v 2 -conjugacy_classes_Sym_n 10
12614 ▶ open classes_Sym_10.csv
12615
12616 Sym_15_conj_classes:
12617 ▶ $(ORBITER) -v 2 -conjugacy_classes_Sym_n 15
12618
12619 Char_Sym_4:
12620 ▶ $(ORBITER) -v 2 -character_table_symmetric_group 4
12621
12622 Char_Sym_5:
12623 ▶ $(ORBITER) -v 2 -character_table_symmetric_group 5
12624
12625
12626 # too slow:
12627
12628 Char_Sym_6:
12629 ▶ $(ORBITER) -v 2 -character_table_symmetric_group 6
12630
12631 all_subsets_10_3:
12632 ▶ $(ORBITER) -v 2 -tree_of_all_k_subsets 10 3
12633
12634 12635 random_k_subsets:
12636 ▶ $(ORBITER) -v 4 \n12637 ▶ ▶ -create_random_k_subsets 10 5 20
12638
12639
12640 rank_k_subsets_test:
12641 ▶ $(ORBITER) -v 2 \n12642 ▶ ▶ -rank_k_subset 10 3 0,1,2,0,3,4,1,3,5,2,4,5,3,6,7,1,6,8,0,6,9
12643
12644
12645 Walsh_matrix_4:
12646  $(ORBITER) -v 3 \ 
12647  \ > \ -define F -finite_field -q 2 -end \ 
12648  \ > \ -with F -do -finite_field_activity \ 
12649  \ > \ > \ -Walsh_matrix 4 -end 
12650  \ > $(ORBITER) -v 2 -draw_matrix \ 
12651  \ > \ -input_csv_file Walsh_01_4.csv \ 
12652  \ > \ -box_width 10 -bit_depth 24 -partition 3 16 16 -end 
12653  \ > #pdflatex GF_2.tex 
12654  \ > #open GF_2.pdf 
12655 
12656 
12657  Dedekind_10.10: 
12658  $(ORBITER) -v 3 -Dedekind_numbers 2 10 2 10 
12659 
12660 
12661  Dedekind_30.2: 
12662  $(ORBITER) -v 3 -Dedekind_numbers 2 30 2 2 
12663 
12664 
12665  Dedekind_100.2: 
12666  $(ORBITER) -v 3 -Dedekind_numbers 2 100 2 2 
12667 
12668 
12669 
12670  elementary_symmetric_functions_4: 
12671  $(ORBITER) -make_elementary_symmetric_functions 4 4 
12672 
12673  elementary_symmetric_functions_8: 
12674  $(ORBITER) -make_elementary_symmetric_functions 8 8 
12675 
12676 
12677 
12678 
12679 
12680 
12681 
12682  LARGE_SET_S0="0,1,2,3,4,5,6,7,8,9,10,11,12" 
12683  # identity 
12684 
12685  LARGE_SET_S1="6,8,9,2,7,10,1,11,0,3,5,4,12" 
12686 
12687  #(0, 6, 1, 8)(2, 9, 3)(4, 7, 11)(5, 10),\ 
12688 
12689 
12690 
12691  LARGE_SET_S2="2,0,1,6,3,4,11,5,7,8,10,9,12" 
12692  #(0, 2, 1)(3, 6, 11, 9, 8, 7, 5, 4),\ 

784
LARGE_SET_S3="12,5,6,11,3,7,10,8,9,1,4,2,0"
#(0, 12)(1, 5, 7, 8, 9)(2, 6, 10, 4, 3, 11),\nLARGE_SET_S4="5,8,10,3,11,0,2,1,12,4,6,7,9"
#(0, 5)(1, 8, 12, 9, 4, 11, 7)(2, 10, 6),\nLARGE_SET_S5="10,11,0,7,12,2,3,1,4,5,8,6,9"
#(0, 10, 8, 4, 12, 9, 5, 2)(1, 11, 6, 3, 7),\nLARGE_SET_S6="3,4,1,9,5,6,8,2,7,11,12,10,0"
#(0, 3, 9, 11, 10, 12)(1, 4, 5, 6, 8, 7, 2),\nLARGE_SET_S7="9,11,0,6,1,3,5,4,2,12,8,7,10"
#(0, 9, 12, 10, 8, 2)(1, 11, 7, 4)(3, 6, 5),\nLARGE_SET_S8="10,2,12,8,0,3,4,1,5,6,9,7,11"
#(0, 10, 9, 6, 4)(1, 2, 12, 11, 7)(3, 8, 5),\nLARGE_SET_S9="1,3,4,10,5,6,9,7,8,11,0,12,2"
#(0, 1, 3, 10)(2, 4, 5, 6, 9, 11, 12),\nLARGE_SET_S10="7,12,1,6,0,4,5,2,3,10,9,8,11"
#(0, 7, 2, 1, 12, 11, 8, 3, 6, 5, 4)(9, 10).

file_S:
> echo ROW,C0"
>0,"$(LARGE
SET_S0)"
>1,"$(LARGE
SET_S1)"
>2,"$(LARGE
SET_S2)"
>3,"$(LARGE
SET_S3)"
>4,"$(LARGE
SET_S4)"
>5,"$(LARGE
SET_S5)"
>6,"$(LARGE
SET_S6)"
>7,"$(LARGE
SET_S7)"
>8,"$(LARGE
SET_S8)"
>9,"$(LARGE
SET_S9)"
>10,"$(LARGE
SET_S10)"
>"\n"END"
" >S.csv

Large_set_H5:
$ (ORBITER) -v 10 \
-define G -permutation_group -symmetric_group 13 \
-subgroup_by_generators H5 5 1 $(GENERATORS_H5) -end \
-with G -do \
-group_theoretic_activity \
-report \
-end \

785
```
12737 ▷▷ -with G -do \
12738 ▷▷ -group_theoretic_activity \
12739 ▷▷ ▷▷ -save_elements_csv "H5_elts.csv" \
12740 ▷▷ ▷▷ -end 
12741 ▷▷ pdflatex Perm13_Subgroup_H5_5_report.tex 
12742 ▷▷ open Perm13_Subgroup_H5_5_report.pdf 
12743 
12744 Large_set_C13: 
12745 ▷ $(ORBITER) -v 10 \
12746 ▷ ▷ -define G -permutation_group -symmetric_group 13 \
12747 ▷ ▷ ▷ -subgroup_by_generators C13 13 1 $(GENERATORS_C13) -end \
12748 ▷ ▷ ▷ -with G -do \
12749 ▷ ▷ ▷ -group_theoretic_activity \
12750 ▷ ▷ ▷ ▷ -export_orbiter \
12751 ▷ ▷ ▷ ▷ -end \
12752 ▷ ▷ ▷ -with G -do \
12753 ▷ ▷ ▷ -group_theoretic_activity \
12754 ▷ ▷ ▷ ▷ -report \
12755 ▷ ▷ ▷ ▷ -end \
12756 ▷ ▷ ▷ -with G -do \
12757 ▷ ▷ ▷ -group_theoretic_activity \
12758 ▷ ▷ ▷ ▷ ▷ -save_elements_csv "C13_elts.csv" \
12759 ▷ ▷ ▷ ▷ ▷ -end 
12760 ▷ pdflatex Perm13_Subgroup_C13_13_report.tex 
12761 ▷ open Perm13_Subgroup_C13_13_report.pdf 
12762 
12763 
12764 ## the following lines were created using -export_orbiter: 
12765 
12766 GENERATORS_Perm13_Subgroup_C13_13= \
12767 ▷ "11,0,10,12,5,3,7,4,2,8,6,9,1"
12768 
12769 Perm13_Subgroup_C13_13: 
12770 ▷ $(ORBITER) -v 2 \
12771 ▷ -define G -permutation_group -symmetric_group 13 \
12772 ▷ -subgroup_by_generators Perm13_Subgroup_C13_13 13 1 \
12773 ▷ ▷ $(GENERATORS_Perm13_Subgroup_C13_13) \
12774 ▷ ▷ -end 
12775 
12776 ###
12777 
12778 
12779 Large_set_mult_C13xS: 
12780 ▷ $(ORBITER) -v 10 \
12781 ▷ ▷ -define G -permutation_group -symmetric_group 13 -end \
12782 ▷ ▷ ▷ -with G -do \
12783 ▷ ▷ ▷ -group_theoretic_activity \\```

786
Large_set_mult_C13xSxH5:
$\text{do }$ 
\begin{align*}
&\text{define } G \text{ -permutation group }-\text{symmetric group } 13 \text{ -end } \\
&\text{with } G \text{ -do } \\
&\text{-group_theoretic_activity } \\
&\text{with } G \text{ -do } \\
&\text{-group_theoretic_activity } \\
&\text{-apply_elements_csv_to_set } \\
&\text{apply_elements_csv_to_set } C13xSxH5.csv C13xSxH5.images.csv "$0,1,2,3" \\
&\text{-end } \\
$\end{align*}$
Large_set_mult_C13xSxH5.apply:
$\text{do }$ 
\begin{align*}
&\text{define } G \text{ -permutation group }-\text{symmetric group } 13 \text{ -end } \\
&\text{with } G \text{ -do } \\
&\text{-group_theoretic_activity } \\
&\text{-apply_elements_csv_to_set } \\
&\text{-apply_elements_csv_to_set } C13xSxH5.csv C13xSxH5.images.csv "$0,1,2,3" \\
&\text{-end } \\
$\end{align*}$
domino_portrait:
$\text{do }$ 
\begin{align*}
&\text{define } r \text{ -vector -repeat } 1 28 \text{ -end } \\
&\text{define } c \text{ -vector -repeat } 1 32 \text{ -end } \\
&\text{-draw matrix } \\
&\text{-grayscale } \\
&\text{-invert colors } \\
&\text{-input_csv_file anton_28x32.m.csv } \\
&\text{-box_width } 20 \text{ -bit_depth } 8 \\
&\text{-partition } 3 \\
&\text{all_one_c all_one_r } \\
&\text{-end } \\
&\text{open anton_28x32_m_draw.bmp } \\
$\end{align*}$
## Section 11.2: Diophantine Systems
12831 test_11_2:
12832 make part10
12833 make octicモノmials
12834 make solve_test_system
12835 make McKay_test
12836 make DLX_test
12837
12838 12839
12840 12841 part10:
12842 . $(ORBITER) -v 4 \\n12843 . -define A -vector -format 1 -dense "10,9,8,7,6,5,4,3,2,1" -end \n12844 . -define D -diophant \n12845 . -label part10 \n12846 . -coefficient_matrix A \n12847 . -RHS "10,10,1" \n12848 . -x_min_global 0 -x_max_global 10 \n12849 . -end \n12850 . -with D -do \n12851 . -diophant_activity -solve_mckay \n12852 . -end
12853
12854 12855
12856
12857 # Finds 42 solutions with 67 backtrack steps
12858
12859 12860 octicモノmials:
12861 . $(ORBITER) -v 4 \\n12862 . -define A -vector -format 1 -dense "1,1,1,1" -end \n12863 . -define D -diophant \n12864 . -label octicモノmials \n12865 . -coefficient_matrix A \n12866 . -RHS "8,8,1" \n12867 . -x_min_global 0 -x_max_global 8 \n12868 . -end \n12869 . -with D -do \n12870 . -diophant_activity -solve_mckay \n12871 . -end
12872 . sort -r octicモノmials.sol >octicモノmials_sorted.txt
12873
12874 # Found 165 solutions with 210 backtrack steps
12875 12876 12877

788
solve_test_system:
-define A -vector -format 7 -dense $(TEST_SYSTEM) -end
-define D -diophant
-label test_system
-coefficient_matrix A
-RHS $(TEST_RHS)
-x_min_global 0 -x_max_global 1
-end

McKay_test:
-define A -vector -format 7 -dense $(TEST_SYSTEM) -end
-define D -diophant -label test_system
-coefficient_matrix A
-RHS $(TEST_RHS)
-x_min_global 0 -x_max_global 1
-end
-with D -do
-diohant_activity -solve_mckay
-end

DLX_test:
-define A -vector -format 7 -dense $(TEST_SYSTEM) -end
-define D
-diohant -label test_system
-coefficient_matrix A
-RHS $(TEST_RHS)
-x_min_global 0 -x_max_global 1
-end
-with D -do
-diohant_activity -solve_DLX
-end

#DLX_test.sol
# 1 solution in 6 backtrack steps

SECTION COMBINATORIAL LINEAR SPACES:
test_11.3:

make linsp6
make linsp7
make linsp30.pt_types
make linsp30.pt_distribution

linsp6:

$\text{(ORBITER)} -v 4 \$
-define A -vector -format 1 -dense "15,10,6,3,1" -end 
-define D -diophant -label linsp6 
-coefficient_matrix A 
-RHS "15,15,1" 
-x_min_global 0 
-x_max_global 15 
-end 
-with D -do 
-diophant_activity -solve_mckay 
-end

# Found 15 solutions with 22 backtrack steps

linsp7:

$\text{(ORBITER)} -v 4 \$
-define A -vector -format 1 -dense "21,15,10,6,3,1" -end 
-define D -diophant -label linsp7 
-coefficient_matrix A 
-RHS "21,21,1" 
-x_min_global 0 
-x_max_global 21 
-end 
-with D -do 
-diophant_activity -solve_mckay 
-end

# 32 solutions in 45 backtrack steps
linsp30_pt_types:
$\text{(ORBITER) -v 4} \\$
-define A -vector -format 1 -dense "6,4,3" -end \$
-define D -diophant \$
-define A -vector -format 6 -dense \$
coefficients matrix A \$
\text{RHS } 29,29,1 -x \text{bounds } 0,1,0,27,0,24 \$
\text{-end} \$
\text{with D -do} \$
\text{with D -do} \$
\text{with D -do} \$
\text{diophant_activity -solve_mckay} \$
\text{end} \$
\text{with D -do} \$
\text{diophant_activity -solve_mckay} \$
\text{end} \$
\text{with D -do} \$
\text{diophant_activity -draw_as_bitmap 20 8} \$
\text{end} \$
\\
# Section 11.4: Combinatorial Linear Spaces

\text{test 11.4:}
\text{make geo_pasch} 
\text{make geo_petersen} 
\text{make geo_7.3} 
\text{make geo_7.3_no_square_test} 
\text{make geo_7.3_no_square_test_draw} 
\text{make geo_7.3_orderly} 
\text{make geo_7.3_orderly_draw} 
\text{make geo_7.3_orderly_mem_debug} 
\text{make geo_8.3}
13019 > make geo_9.3
13020 > make geo_10.3
13021 > make geo_10.3_inc_draw
13022 > make geo_10.3_orderly
13023 > make geo_10.3_orderly_mem_debug
13024 > make geo_10.3_tree
13025 > make geo_10.3_tree_path
13026 > make Desargues_path_lex_least_draw
13027 > make Desargues_path_can_anc_draw
13028 > make geo_11.3
13029 > make geo_12.3
13030 > make geo_12.3_orderly
13031 > make geo_13.3
13032 > make geo_13.3_orderly
13033 > make geo_14.3
13034 > make geo_14.3_orderly
13035 > make 15.3.inc
13036 > make geo_15.3_g4
13037 > make geo_17.3_g4_orderly
13038 > make geo_18.3_g4
13039 > make geo_19.3_g4
13040 > make geo_20.3_g4
13041 > make geo_21.3_g4
13042 > make geo_15.4
13043 > make geo_16.4_g4
13044 > make geo_LSQ6
13045 > make geo_16
13046 > make 40.4_g4.inc
13047 > make geo_63.3_g6
13048
13049
13050 geo_pasch:
13051 > $(ORBITER) -v 8 \n13052 > -define Test_lines -set -loop 1 7 1 -end \n13053 > -define Geo -geometry_builder \n13054 > -V 6 -B 4 -TDO 2 -fuse 1 \n13055 > -fname_GE0 pasch \n13056 > -output_to_inc_file \n13057 > -test Test_lines \n13058 > -end
13059
13060
13061 geo_petersen:
13062 > $(ORBITER) -v 8 \n13063 > -define Test_lines -set -loop 3 11 1 -end \n13064 > -define Geo -geometry_builder \n13065 > -V 10 -B 15 -TDO 3 -fuse 1 \n
792
geo_7_3:
$(ORBITER) -v 2 \\
(define Test_lines -set -loop 3 8 1 -end \\
(define Geo -geometry builder \\
(V 7 -B 7 -TDO 3 \\
(fuse 1 \\
(fname GEO 7_3 \\
(output_to_inc_file \\
(test Test_lines \\
(end \\
geo_7_3_no_square_test:
$(ORBITER) -v 2 \\
(define Test_lines -set -loop 3 8 1 -end \\
(define Geo -geometry builder \\
(V 7 -B 7 -TDO 3 \\
(fuse 1 \\
(fname GEO 7_3_nst \\
(output_to_inc_file \\
(test Test_lines \\
(no_square_test \\
(end \\
geo_7_3_no_square_test_draw:
$(ORBITER) -v 10 \\
(draw.incidence_structure_description \\
(width 60 -with_10 6 -end \\
(define C -combinatorial_objects \\
(file_of_incidence_geometries 7_3_nst.inc 7 7 21 \\
(end \\
(with C -do \\
(combinatorial_object_activity \\
(draw.incidence_matrices \\
(7_3_nst \\
(end \\
pdflatex 7_3_nst.incma.tex \\
open 7_3_nst.incma.pdf \\
geo_7_3_orderly:
13113 $\text{(ORBITER)} -v 200 \$
13114 $\text{-define Test_lines -set -loop 3 8 1 -end}
13115 $\text{-define Geo -geometry_builder}
13116 $\text{-V 7 -B 7 -TDO 3}
13117 $\text{-fuse 1 -fname GEO 7_3}
13118 $\text{-output_to_inc_file}
13119 $\text{-test Test_lines}
13120 $\text{-search_tree}
13121 $\text{-orderly}
13122 $\text{-end}
13123
13124 $\text{geo_7_3_orderly_draw:}$
13125 $\text{$(ORBITER) -v 20 \}$}
13126 $\text{-draw_options -embedded -radius 50 \}$
13127 $\text{-xin 10000 -yin 10000 \}$
13128 $\text{-xout 1000000 -yout 1000000 \}$
13129 $\text{-nodes_empty}
13130 $\text{-scale 0.5 -line_width 0.3 \}$
13131 $\text{-end \}$
13132 $\text{-tree_draw -file 7_3_tree.txt -end \}$
13133 $\text{pdflatex 7_3_tree_draw.tex}$
13134 $\text{open 7_3_tree_draw.pdf}$
13135
13136 $\text{geo_7_3_orderly_mem_debug:}$
13137 $\text{$(ORBITER) -v 20 \}$}
13138 $\text{-memory_debug 2 \}$
13139 $\text{-define Test_lines -set -loop 3 8 1 -end \}$
13140 $\text{-define Geo -geometry_builder \}$
13141 $\text{-V 7 -B 7 -TDO 3 \}$
13142 $\text{-fuse 1 -fname GEO 7_3 \}$
13143 $\text{-output_to_inc_file \}$
13144 $\text{-test Test_lines \}$
13145 $\text{-search_tree \}$
13146 $\text{-orderly \}$
13147 $\text{-end \}$
13148
13149 $\text{geo_8_3:}$
13150 $\text{$(ORBITER) -v 2 \}$}
13151 $\text{-define Test_lines -set -loop 3 9 1 -end \}$
13152 $\text{-define Geo -geometry_builder \}$
13153 $\text{-V 8 -B 8 -TDO 3 \}$
13154 $\text{-fuse 1 -fname GEO 8_3 \}$
13155 $\text{-output_to_inc_file \}$
13156 $\text{-test Test_lines \}$
13157 $\text{-end \}$
13158
13159 $\text{#-print_at_line 4}$
# 1 geo: 0 11 18 29 30 38 44 54

ago=48

# 10 geos

# 8/26/2021: 0 sec on Mac

geo_9.3:

$ (ORBITER) -v 2 \
  -define Test_lines -set -loop 3 10 1 -end \
  -define Geo -geometry_builder \
  -V 9 -B 9 -TDO 3 \
  -fuse 1 -fname GEO 9.3 \
  -output_to_inc_file \
  -test Test_lines \
  -end

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 \
  -test Test_lines \
  -end

geo_10.3_inc_draw:

$ (ORBITER) -v 10 \
  -draw_incidence_structure_description \
  -width 60 -with_10 6 -end \
  -define C -combinatorial_objects \
  -file_of_incidence_geometries \
  10.3.inc 10 10 30 \
  -end \
  -with C -do \
  -combinatorial_object_activity \
  -draw_incidence_matrices \
  10.3_inc \

795
geo_10_3.orderly:

geo_10_3.orderly_mem_debug:

geo_10_3.tree:

$(ORBITER) -v 20 \
-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 \
-test Test_lines \
-orderly \
-end
13254 \> \> \texttt{-end} \\
13255 \> \> \texttt{-tree\_draw} \\
13256 \> \> \texttt{-file 10\_3\_tree\_txt} \\
13257 \> \> \texttt{-end} \\
13258 \> pdflatex 10\_3\_tree\_draw.tex \\
13259 \> open 10\_3\_tree\_draw.pdf \\
13260 \\
13261 \\
13262 \\
13263 \\
13264 geo\_10\_3\_tree\_path: \\
13265 \> \$(\texttt{ORBITER}) -v 20 \\
13266 \> \> \texttt{-define Test\_lines -set -loop 0 11 1 -end} \\
13267 \> \> \texttt{-define GEO -geometry\_builder} \\
13268 \> \> \> \texttt{-V 10 -B 10 -TDO 3 -fuse 1} \\
13269 \> \> \> \texttt{-fname\_GEO 10\_3} \\
13270 \> \> \> \texttt{-output\_to\_inc\_file} \\
13271 \> \> \> \texttt{-search\_tree} \\
13272 \> \> \> \texttt{-test Test\_lines} \\
13273 \> \> \> \texttt{-end} \\
13274 \> \$(\texttt{ORBITER}) -v 20 \\
13275 \> \> \texttt{-draw\_options -embedded -radius 20} \\
13276 \> \> \> \texttt{-paperheight 220} \\
13277 \> \> \> \texttt{-paperwidth 330} \\
13278 \> \> \> \texttt{-xin 10000 -yin 10000} \\
13279 \> \> \> \texttt{-xout 1000000 -yout 500000} \\
13280 \> \> \> \texttt{-scale 2 -line\_width 0.3} \\
13281 \> \> \> \texttt{-end} \\
13282 \> \> \> \texttt{-tree\_draw} \\
13283 \> \> \> \texttt{-restrict 2} \\
13284 \> \> \> \texttt{-file 10\_3\_tree\_txt} \\
13285 \> \> \> \texttt{-select\_path "0,0,15,26,46,56,72,80,93,106,119"} \\
13286 \> \> \> \texttt{-end} \\
13287 \> pdflatex 10\_3\_tree\_draw.tex \\
13288 \> open 10\_3\_tree\_draw.pdf \\
13289 \\
13290 \#\> \> \> \texttt{-nodes\_empty} \\
13291 \#-sideways \\
13292 \\
13293 \\
13294 \# ToDo: \\
13295 \#Desargues\_path\_lex\_least\_inc is missing \\
13296 \\
13297 Desargues\_path\_lex\_least\_draw: \\
13298 \> echo \$(\texttt{DESARGUES\_PATH\_LEX\_LEAST}) \>Desargues\_path\_lex\_least\_inc \\
13299 \> \$(\texttt{ORBITER}) -v 10 \\
13300 \> \> \texttt{-draw\_incidence\_structure\_description} \\

797
13301 \> \> \> -width 60 -with_10_6 -end \n
13302 \> \> \> -define C -combinatorial_objects \n
13303 \> \> \> -file_of_incidence_geometries_by_row_ranks \n
13304 \> \> \> Desargues_path_lex_least.inc 10 10 3 \n
13305 \> \> \> -end \n
13306 \> \> \> -with C -do \n
13307 \> \> \> -combinatorial_object_activity \n
13308 \> \> \> -draw_incidence_matrices \n
13309 \> \> \> Desargues_path_lex_least \n
13310 \> \> \> -end \n
13311 \> pdflatex Desargues_path_lex_least_incma.tex \n
13312 \> open Desargues_path_lex_least_incma.pdf \n
13313 \n
13314 DESARGUES_PATH_CANONICAL_ANCESTOR="10 10 3\n0\n0 112 119\n3 89 112 119\n4 118 89 82\n5 106 114 69 107 111\n6 85 105 112 99 83 61\n7 94 105 113 85 35 83 6\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\n-1\n" \n
13315 \n
13316 Desargues_path_can_anc.draw: \n
13317 \> echo $(DESARGUES_PATH_CANONICAL_ANCESTOR) >Desargues_path_can_anc.inc \n
13318 \> $(ORBITER) -v 10 \n
13319 \> \> -draw_incidence_structure_description \n
13320 \> \> \> -width 60 -with_10_6 -end \n
13321 \> \> \> -define C -combinatorial_objects \n
13322 \> \> \> -file_of_incidence_geometries_by_row_ranks Desargues_path_can_anc.inc 10 10 3 \n
13323 \> \> \> -end \n
13324 \> \> \> -with C -do \n
13325 \> \> \> -combinatorial_object_activity \n
13326 \> \> \> -draw_incidence_matrices \n
13327 \> \> \> Desargues_path_can_anc \n
13328 \> \> \> -end \n
13329 \> pdflatex Desargues_path_can_anc_incma.tex \n
13330 \> open Desargues_path_can_anc_incma.pdf \n
13331 \n
13332 \n
13333 \n
13334 geo_11_3: \n
13335 \> $(ORBITER) -v 2 \n
13336 \> \> -define Test_lines -set -loop 4 12 1 -end \n
13337 \> \> -define Geo -geometry_builder \n
13338 \> \> \> -V 11 -B 11 -TDO 3 \n
13339 \> \> \> -fuse 1 -fname_GEO 11_3 \n
13340 \> \> \> -output_to_inc_file \n
13341 \> \> \> -test Test_lines \n
13342 \> \> \> -end \n
13343
geo_12.3:

$(ORBITER) -v 2 \
$define Test_lines -set -loop 4 13 1 -end 
$define Geo -geometry_builder 
-V 12 -B 12 -TDO 3 
-fuse 1 -fname_GEO 12_3 
-output_to_inc_file 
-test Test_lines 
-output_to_sage_file 
-end

geo_12.3_orderly:

$(ORBITER) -v 2 \
$define Test_lines -set -loop 4 13 1 -end 
$define Geo -geometry_builder 
-V 12 -B 12 -TDO 3 
-fuse 1 -fname_GEO 12_3 
-output_to_inc_file 
-test Test_lines 
-f_orderly 
-end

geo_13.3:

$(ORBITER) -v 2 \
$define Test_lines -set -loop 4 14 1 -end 
$define Geo -geometry_builder 
-V 13 -B 13 -TDO 3 
-fuse 1 -fname_GEO 13_3 
-output_to_inc_file 
-test Test_lines 
-end

geo_13.3_orderly:
13391 $\$(ORBITER) -v 2 \\
13392  -define Test_lines -set -loop 4 14 1 -end \\
13393  -define Geo -geometry_builder \\
13394  -V 13 -B 13 -TDO 3 \\
13395  -fuse 1 -fname_GEO 13_3 \\
13396  -output_to_inc_file \\
13397  -test Test_lines \\
13398  -f_orderly \\
13399  -end
13400
13401
13402
13403 geo_14_3:
13404 $\$(ORBITER) -v 2 \\
13405  -define Test_lines -set -loop 4 15 1 -end \\
13406  -define Geo -geometry_builder \\
13407  -V 14 -B 14 -TDO 3 \\
13408  -fuse 1 -fname_GEO 14_3 \\
13409  -output_to_inc_file \\
13410  -test Test_lines \\
13411  -end
13412
13413 # 21399 geos, 56448, 128, 24^2, 16^3, 14^3, 12^7, 8^15, 7, 6^12, 4^91, 3^19, 2^91
13414 6, 1^20328
13415 #User time: 0:55
13416
13417
13418 geo_14_3_orderly:
13419 $\$(ORBITER) -v 2 \\
13420  -define Test_lines -set -loop 4 15 1 -end \\
13421  -define Geo -geometry_builder \\
13422  -V 14 -B 14 -TDO 3 \\
13423  -fuse 1 -fname_GEO 14_3 \\
13424  -output_to_inc_file \\
13425  -test Test_lines \\
13426  -f_orderly \\
13427  -end
13428
13429 #User time: 0:50
13430
13431
13432 15_3.inc:
13433 $\$(ORBITER) -v 2 \\
13434  -define Test_lines -set -loop 4 16 1 -end \\
13435  -define Geo -geometry_builder \\
13436  -V 15 -B 15 -TDO 3 \

800
# 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

# 8 min 11 sec on Mac

# running out of memory

The unique Cremona Richmond configuration with group of order 720

User time: 0 of a second, dt=0 tps = 100

#calls_to_densenauty=23
13483 \-girth 4 \
13484 \-test Test_lines \
13485 \-orderly \
13486 \-end
13487
13488 \# 1 sol
13489
13490 geo_18_3_g4:
13491 \$(ORBITER) -v 2 \
13492 \-define Test_lines -set -loop 4 19 1 -end \
13493 \-define Geo -geometry_builder \
13494 \-V 18 -B 18 -TDO 3 \
13495 \-fuse 1 -fname_GEO 18_3_g4 \
13496 \-output_to_inc_file \
13497 \-girth 4 \
13498 \-search_tree \
13499 \-test Test_lines \
13500 \-end
13501
13502 \# 4 sol, 1 sec
13503
13504
13505 geo_19_3_g4:
13506 \$(ORBITER) -v 2 \
13507 \-define Test_lines -set -loop 4 20 1 -end \
13508 \-define Geo -geometry_builder \
13509 \-V 19 -B 19 -TDO 3 \
13510 \-fuse 1 -fname_GEO 19_3_g4 \
13511 \-output_to_inc_file \
13512 \-girth 4 \
13513 \-test Test_lines \
13514 \-end
13515
13516 \# 14 sol, 10 sec on Mac
13517
13518 geo_20_3_g4:
13519 \$(ORBITER) -v 2 \
13520 \-define Test_lines -set -loop 4 21 1 -end \
13521 \-define Geo -geometry_builder \
13522 \-V 20 -B 20 -TDO 3 \
13523 \-fuse 1 -fname_GEO 20_3_g4 \
13524 \-output_to_inc_file \
13525 \-girth 4 \
13526 \-test Test_lines \
13527 \-end
13528
13529 \# 162 sol, User time: 2:5 on Mac
geo_21_3_g4:
$(ORBITER) -v 2 \n-define Test_lines -set -loop 4 22 1 -end \n-define Geo -geometry_builder \n-V 21 -B 21 -TDO 3 \n-fuse 1 -fname_GEO 21_3_g4 \n-output_to_inc_file \ngirth 4 \n-test Test_lines \n-end

geo_15_4:
$(ORBITER) -v 2 \n-define Test_lines -set -loop 4 16 1 -end \n-define Geo -geometry_builder \n-V 15 -B 15 -TDO 4 \n-fuse 1 -fname_GEO 15_4 \n-output_to_inc_file \n-search_tree \n-test Test_lines \n-end
$(ORBITER) -v 2 \n-draw_options -embedded -radius 50 \n-nodes_empty \n-scale 0.5 -line_width 0.3 -end \ntree_draw -file 15_4_tree.txt -end
pdflatex 15_4_tree_draw.tex
open 15_4_tree_draw.pdf

# 4 objects
# ago=360, 30, 24, 15
#User time: 0.15 of a second, dt=15 tps = 100
#nb_calls_to_densenauty=6767

geo_16_4_g4:
$(ORBITER) -v 2 \n-define Test_lines -set -loop 4 17 1 -end \n-define Geo -geometry Builder \n-V 16 -B 16 -TDO 4 \n-fuse 1 -fname_GEO 16_4_g4 \n-output_to_inc_file \ngirth 4 \n-test Test_lines \n
geo_15_6:
\$(ORBITER) -v 2 \\
> -define Test_lines -set -loop 4 16 1 -end \\
> -define Geo -geometry_builder \\
> -V 15 -B 15 -TDO 6 -fuse 1 \\
> -fname_GEO 15_6 \\
> -output_to_inc_file \\
> -test Test_lines \\
> -output_to_sage_file \\
> -end \\
geo_LSQ6:
\$(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 \\
geo_16:
\$(ORBITER) -v 2 \\
> -define Test_lines -set -loop 3 17 1 -end \\
> -define Geo -geometry_builder \\
> -V 16 -B 20 -TDO 5 \\
> -fuse 1 -fname_GEO geo_16 \\
> -output_to_inc_file \\
> -test Test_lines \\
> -end \\
geo_16:
\$(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
#-special_test_not_orderly is important for speed purposes

# 2 geos, ago=51840^2
# 40_g4.inc

geo_63_3_g6:

SECTION DESIGN THEORY:
test_11_5:

design_PG_2.3:
 $(ORBITER) -v 8 \
   -define F -finite_field -q 3 -end \
   -define D -design -field F -family PG_2.3 -end \
   -with D -do \
   -design_activity \
   -export_inc \
   -end

# writes PG_2.3_inc.txt

design_PG_2.4:
 $(ORBITER) -v 8 \
   -define F -finite_field -q 4 -end \
   -define D -design -field F -family PG_2.4 -end \
   -with D -do \
   -design_activity \
   -export_inc \
   -end

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

wreath_product_designs_n8_k6_inc.txt:
 $(ORBITER) -v 8 \

# wreath_product_designs_n8_k6_inc.txt

The design has 16 points and 3920 blocks of size 6.

KM_cyclic_7:

```
$\text{(ORBITER)} \text{-v} 3 \$

-define orbs -vector -dense "1,2,3,4,5,6,0" -end
-define G -permutation_group -symmetric_group 7
-subgroup_by_generators "C7" 7 1 gens
-end

-define Control -poset_classification_control
-problem_label C7 -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

-define A -vector -file C7_KM_2_3.csv -end
-define D -diophant
-label "C7_KM_2_3_system"
-coefficient_matrix A
-RHS_constant "1,1,1"
-x_min_global 0 -x_max_global 1
-end
-end
```
# to create simple 7-designs on 33 points with block size 8 and lambda = 10 invariant under PGGL(2,32):

```
KM_PGGL_2_32:
$(ORBITER) -v 3 \n-define_orbiter=$(ORBITER) \n-problem_label KM_PGGL_2_32 \n-W -depth 8 \n-draw_options -embedded -sideways -radius 50 \n-scale 0.5 -line_width 0.3 -end \n
-define G -linear_group -PGGL 2 32 -end \n-define Orb -orbits -group G \n-on_subsets 8 Control \n-end \n
-define A -vector -file KM_PGGL_2_32_KM_7_8.csv -end \n-define D -diophant \n-label "KM_PGGL_2_32_KM_7_8_system" \n-coefficient_matrix A \n-RHS_constant "10,10,1" \n-x_min_global 0 -x_max_global 1 \n-end \n
-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
```

13750 KM_PSL_3_5:
```
13811 \$\text{(ORBITER)} -v 3 \$
13812 \$\text{orbit_path }\$\text{(ORBITER\_PATH)} \$
13813 \$\text{define Control -poset\_classification\_control }\$
13814 \$\text{-problem\_label KM\_PSL\_3\_5 }\$
13815 \$\text{-W -depth 10 }\$
13816 \$\text{-draw\_options -embedded -sideways }\$
13817 \$\text{-radius 50 -scale 0.5 -line\_width 0.3 -end }\$
13818 \$\text{-end }\$
13819 \$\text{-define G \_linear\_group \_PSL 3 5 -end }\$
13820 \$\text{-define Orb \_orbits \_group G }\$
13821 \$\text{-on\_subsets 10 Control }\$
13822 \$\text{-end }\$
13823 \$\text{-with Orb -do \_orbits\_activity }\$
13824 \$\text{-Kramer\_Mesner\_matrix 8 10 }\$
13825 \$\text{-end }\$
13826 \$\text{-with Orb -do \_orbits\_activity }\$
13827 \$\text{-report }\$
13828 \$\text{-report\_options -draw\_poset -end }\$
13829 \$\text{-end }\$
13830 \$\text{$\文字{(ORBITER)} -v 2 -draw\_matrix }\$
13831 \$\text{-input\_csv\_file KM\_PSL\_3\_5\_KM\_8\_10.csv }\$
13832 \$\text{-box\_width 10 -bit\_depth 8 -partition 3 42 174 -end }\$
13833 \$\text{$\文字{(ORBITER)} -v 4 }\$
13834 \$\text{-define A \_vector \_file KM\_PSL\_3\_5\_KM\_8\_10.csv -end }\$
13835 \$\text{-define D \_diophant }\$
13836 \$\text{-label "KM\_PSL\_3\_5\_KM\_8\_10\_system" }\$
13837 \$\text{-coefficient\_matrix A }\$
13838 \$\text{-RHS\_constant "93,93,1" }\$
13839 \$\text{-x\_min\_global 0 -x\_max\_global 1 }\$
13840 \$\text{-end }\$
13841 \$\text{-with D -do }\$
13842 \$\text{-diophant\_activity -solve\_mckay }\$
13843 \$\text{-end }\$
13844
13845
13846
13847
13848 # Section 11.6: Design Theory - Large Sets
13849
13850
13851 SECTION\_DESIGN\_THEORY\_LARGE\_SETS:
13852
13853 test_11_6:
13854 \$\text{make AG\_2\_3.inc }\$
13855 \$\text{make LS\_AG\_2\_3\_design\_table\_create }\$
13856 \$\text{make AG\_2\_3\_on\_designs }\$
13857 \$\text{make AG\_2\_3\_stab\_orb\_0 }\$
```

809
make AG_2.3_stab_orb_0_Perm840_res192
make LS_AG_2.3_disjoint_sets_graph_and_cliques
make LS_AG_2.3_disjoint_sets_split
make LS_AG_2.3_export_solutions
make design_PG_2.3_table_create
make design_PG_2.3_group_5
make design_PG_2.3_group_5_sol_0

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 \
  -end

#9 12 3
#0 13 22 27 35 41 47 53 55 59 71 76
#-1 1
#432

# ToDo

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 9 3 B -end \
  -define Sym9 -permutation_group -symmetric_group 9 -end \
  -define T -design_table D "AG_2.3" Sym9 -end

# creates AG_2.3_design_table.csv
# and AG_2.3.makefile

#0,0,13,22,27,35,41,47,53,55,59,71,76
# is the first design in AG_2.3_design_table.csv

#poset.orbit_node::init_root_node storing strong generators for a group of order 362880
# stabilizer order 432
# 840 designs
$\text{AG}_2.3\text{.on\_designs:}$

```bash
$(ORBITER) -v 2 \n  -define gens -vector -file AG_2.3.gens.csv -end \n  -define G -permutation_group \n  -bsgs AG_2.3 "AG_2.3" 840 362880 "0,1,2,3,4,5,6,7" 8 gens -end \n  -define Orb -orbits -group G \n  -on_points \n  -end \n  -with Orb -do -orbits_activity \n  -stabilizer 0 \n  -end

#Written file AG_2.3_stab_orb_0.makefile of size 239

# the stabilizer of the first design:

AG_2.3_stab_orb_0:

$(ORBITER) -v 2 \n  -define gens -vector -file AG_2.3_stab_orb_0.gens.csv -end \n  -define G -permutation_group \n  -bsgs AG_2.3_stab_orb_0 "AG_2.3_stab_orb_0" \n  -end \n  -define Gr -modified_group -from G \n  -restricted_action $(LARGE_SET_AG_2.3_NEIGHBOR_SET) \n  -end \n  -with Gr -do \n  -group_theoretic_activity \n  -export_orbiter \n  -end

AG_2.3_stab_orb_0.Perms_840_res192:

$(ORBITER) -v 2 \n  -define gens -vector -file Perm840_res192.gens.csv -end \n  -define G -permutation_group \n  -bsgs Perm840_res192 "Perm840 \{\text{rm res192}\}" \n  -end \n```

811
13951 \[
\text{\textbackslash r} \quad \text{\textbackslash > \text{-with G -do \textbackslash r}} \\
\text{\textbackslash r} \quad \text{\textbackslash > \text{-group_theoretic_activity \textbackslash r}} \\
\text{\textbackslash r} \quad \text{\textbackslash > \text{-report \textbackslash r}} \\
\text{\textbackslash r} \quad \text{\textbackslash > \text{-end \textbackslash r}} \\
\text{\textbackslash r} \quad \text{pdflatex Perm840_res192_report.tex} \\
\text{\textbackslash r} \quad \text{open Perm840_res192_report.pdf} \\
\text{\textbackslash r} \quad \text{13957} \\
\text{\textbackslash r} \quad \text{13958} \\
\text{\textbackslash r} \quad \text{13959} \\
\text{\textbackslash r} \quad \text{13960} \\
\text{\textbackslash r} \quad \text{\textbackslash > \text{LS_AG_2_3.disjoint_sets_graph_and_cliques: \textbackslash r}} \\
\text{\textbackslash r} \quad \text{\textbackslash > \text{$(ORBITER) \text{-v 2 \textbackslash r}} \\
\text{\textbackslash r} \quad \text{\textbackslash > \text{-define Gamma -graph \textbackslash r}} \\
\text{\textbackslash r} \quad \text{\textbackslash > \text{-disjoint_sets_graph \textbackslash r}} \\
\text{\textbackslash r} \quad \text{\textbackslash > \text{AG.2.3.design_table.csv \textbackslash r}} \\
\text{\textbackslash r} \quad \text{\textbackslash > \text{-end \textbackslash r}} \\
\text{\textbackslash r} \quad \text{\textbackslash > \text{-with Gamma -do \textbackslash r}} \\
\text{\textbackslash r} \quad \text{\textbackslash > \text{-graph_theoretic_activity \textbackslash r}} \\
\text{\textbackslash r} \quad \text{\textbackslash > \text{-save \textbackslash r}} \\
\text{\textbackslash r} \quad \text{\textbackslash > \text{-end \textbackslash r}} \\
\text{\textbackslash r} \quad \text{\textbackslash > \text{-with Gamma -do \textbackslash r}} \\
\text{\textbackslash r} \quad \text{\textbackslash > \text{-graph_theoretic_activity \textbackslash r}} \\
\text{\textbackslash r} \quad \text{\textbackslash > \text{-find_cliques -target_size 7 -end \textbackslash r}} \\
\text{\textbackslash r} \quad \text{\textbackslash > \text{-end \textbackslash r}} \\
\text{\textbackslash r} \quad \text{\textbackslash > \text{-print_symbols \textbackslash r}} \\
\text{\textbackslash r} \quad \text{13975} \\
\text{\textbackslash r} \quad \text{13976} \\
\text{\textbackslash r} \quad \text{13977} \\
\text{\textbackslash r} \quad \text{#AG_2_3.design_table_disjoint_sets.colored_graph} \\
\text{\textbackslash r} \quad \text{13978} \\
\text{\textbackslash r} \quad \text{#User time: 0.66 of a second, dt=66 tps = 100} \\
\text{\textbackslash r} \quad \text{13979} \\
\text{\textbackslash r} \quad \text{#AG_2_3.design_table_disjoint_sets.sol.txt} \\
\text{\textbackslash r} \quad \text{13980} \\
\text{\textbackslash r} \quad \text{#AG_2_3.design_table_disjoint_sets.sol.csv} \\
\text{\textbackslash r} \quad \text{13981} \\
\text{\textbackslash r} \quad \text{13982} \\
\text{\textbackslash r} \quad \text{#15360 solutions} \\
\text{\textbackslash r} \quad \text{13983} \\
\text{\textbackslash r} \quad \text{13984} \\
\text{\textbackslash r} \quad \text{LS_AG_2_3.disjoint_sets_split:} \\
\text{\textbackslash r} \quad \text{\textbackslash > $(ORBITER) \text{-v 4 \textbackslash r}} \\
\text{\textbackslash r} \quad \text{\textbackslash > -define Gamma -graph -load \textbackslash r}} \\
\text{\textbackslash r} \quad \text{\textbackslash > AG.2.3.design_table_disjoint_sets.colored_graph \textbackslash r}} \\
\text{\textbackslash r} \quad \text{\textbackslash > -end \textbackslash r}} \\
\text{\textbackslash r} \quad \text{\textbackslash > -with Gamma -do \textbackslash r}} \\
\text{\textbackslash r} \quad \text{\textbackslash > -graph_theoretic_activity \textbackslash r}} \\
\text{\textbackslash r} \quad \text{\textbackslash > -find_clique "0" "0" \textbackslash r}} \\
\text{\textbackslash r} \quad \text{\textbackslash > -split_by_clique "0" "0" \textbackslash r}} \\
\text{\textbackslash r} \quad \text{\textbackslash > -end \textbackslash r}} \\
\text{\textbackslash r} \quad \text{13993} \\
\text{\textbackslash r} \quad \text{13994} \\
\text{\textbackslash r} \quad \text{13995} \\
\text{\textbackslash r} \quad \text{#AG_2_3.design_table_disjoint_sets.0.graph} \\
\text{\textbackslash r} \quad \text{13996} \\
\text{\textbackslash r} \quad \text{#AG_2_3.design_table_disjoint_sets.0_subset.txt} \\
\text{\textbackslash r} \quad \text{13997}
LS_AG_2_3_export_solutions:

```bash
$(ORBITER) -v 20 \
-define B -vector -dense $(AG_2_3_BLOCKS) -end \n-define D -design -list_of_blocks 9 3 B -end \n-define Sym9 -permutation_group -symmetric_group 9 -end \n-define T -design_table D "AG_2_3" Sym9 -end \n-with D -do \n-design_activity \n-define Sym9 -permutation_group -symmetric_group 9 -end \n-define T -design_table D "AG_2_3" Sym9 -end \n-design_activity
```

```
#User time: 0.39 of a second, dt=39 tps = 100
# solutions.csv
# written file PG_2_13_design_table.csv
# 1108800 designs
#User time: 7:30
```

```
design_PG_2_3_table_create:

$(ORBITER) -v 2 \
-define F -finite_field -q 3 -end \n-define D -design -field F -family PG_2.q -end \n-define Sym13 -permutation_group -symmetric_group 13 -end \n-define T -design_table D "PG_2.13" Sym13 -end
```

```
# written file PG_2.13_design_table.csv
```

```
design_PG_2_3_group_5:

$(ORBITER) -v 2 \
-define F -finite_field -q 3 -end \n-define D -design -field F -family PG_2.q -end \n-define T -design_table D "PG_2.13" Sym13 -end \n-define LSW -large_set_with_symmetry_assumption T \n-H "5" $(GENERATORS_H5) \n-N "1200" $(GENERATORS_N5) \n-prefix "H5" \n-selected_orbit_length 5 \n-end \n-with LSW -do \n-large_set_with_symmetry_assumption_activity \n```
\begin{verbatim}
14045  \$ (ORBITER) -v 2 \$
14046  \$ -define F -finite_field -q 3 -end \$
14047  \$ -define D -design -field F -family PG.2.q -end \$
14048  \$ -define T -design_table D "PG.2.13" Sym13 -end \$
14049  \$ -define LSW -large_set_with_symmetry_assumption T \$
14050  \$ -define H "5" $(GENERATORS_H5) \$
14051  \$ -prefix "H5" \$
14052  \$ -selected_orbit_length 5 \$
14053  \$ -end \$
14054  \$ -with LSW -do \$
14055  \$ -large_set_with_symmetry_assumption_activity \$
14056  \$ -read_solution_file 5 case_0_sol.txt \$
14057  \$ -end  
814
\end{verbatim}
DD_CC:
  $(ORBITER) -v 6 \
  $(DELANDTSHEER_DOYEN_PROBLEM_COLBOURN_COLBOURN_7_13) \
  $(DELANDTSHEER_DOYEN_PROBLEM_COLBOURN_COLBOURN_7_13_GROUP1) \
  $(DELANDTSHEER_DOYEN_PROBLEM_COLBOURN_COLBOURN_7_13_MASK1) \
  -end

# target level: 6
# k2: 15
# number of k-orbits at target level: 1774964
# creates DD_CC_7_13_pair_covering.csv

DD_CC_system:
  $(ORBITER) -v 4 \
  -Delandtsheer_Doyen \
  -define D -diophant -label DD_CC_7_13 \
  -problem_of_Steiner_type 45 DD_CC_7_13_pair_covering.csv \
  -has_sum 3 \
  -end \
  -with D -do \
  -diophant_activity -solve_mckay \
  -end

# no solution

DD_M1_G1:
  $(ORBITER) -v 4 \
  -Delandtsheer_Doyen \
  $(DELANDTSHEER_DOYEN_PROBLEM_27_53) \
  $(DELANDTSHEER_DOYEN_PROBLEM_27_53_GROUP1) \
  $(DELANDTSHEER_DOYEN_PROBLEM_27_53_MASK1) \
  -end

DD_M1_G1_S:
\begin{verbatim}
\$\text{ORBITER} -v 4 \"Delandtsheer Doyen \"
\$\text{DELANDTSHEER DOYEN PROBLEM 27} \"53 \"
\$\text{DELANDTSHEER DOYEN PROBLEM 27 GROUP1} \"
\$\text{DELANDTSHEER DOYEN PROBLEM 27 MASK1} \"
\$\text{DELANDTSHEER DOYEN PROBLEM 3} \"7 \"
\$\text{DELANDTSHEER DOYEN PROBLEM 3 GROUP1} \"
\$\text{DELANDTSHEER DOYEN PROBLEM 3 MASK1} \"
$\text{MAXIMAL ARC PARAMETERS} 16 4$
$\text{CONVERT STACK TO TDO MAX ARC q16 r4.stack}$
\end{verbatim}

# Section 11.8: Tactical Decompositions

SECTION_TACTICAL_DECOMPOSITIONS:

\texttt{max.arc.16.4.start}:

\texttt{$(\text{ORBITER}) -v 4$ -maximal.arc.parameters 16 4}

\texttt{max.arc.16.4.convert_stack.tdo}:

\texttt{$(\text{ORBITER}) -v 4$ -convert_stack_to_tdo max.arc.q16_r4.stack}

\texttt{max.arc.16.4.refine}:
# Chapter 12 - Finite Geometry

## Section 12.1: Spreads

SECTION_SPREADS:

```
make create_spread_9a
make create_spread_9b
make create_spread_25_7
make create_spread_Rao_Rao_27
make desarguesian_spread_in_PG_3_2
make desarguesian_spread_in_PG_5_2
make desarguesian_spread_in_PG_3_4
make desarguesian_spread_in_PG_3_5
make classify_spreads_4
make classify_spreads_16_4
make classify_spreads_25_starter_lift_case_0
make spreads_25_starter_0_cliques
make classify_spreads_25_starter_lift_all_cases
make spreads_25_starter_cliques
make classify_spreads_25_isomorph
make classify_spreads_27_3_starter
make classify_spreads_27_starter_lift_all_cases
make spreads_27_starter_cliques
make classify_spreads_27_isomorph_and_recognize
make classify_spreads_27_0
make create_spread_27_1
make create_spread_27_2
make create_spread_27_3
make create_spread_27_4
make create_spread_27_5
make create_spread_27_6
```
14233 \> make classify_spreads_32_starter
14234 \> make classify_spreads_49_starter_lift_all_cases
14235 \> make spreads_49_starter_cliques_loop
14236 \> make spreads_49_starter_cliques_0
14237
14238
14239
14240 create_spread_9a:
14241 \> $(ORBITER) -v 3 \\
14242 \> \> -define F -finite_field -q 3 -end \\
14243 \> \> -define G -linear_group -PGL 4 F -end \\
14244 \> \> -define S -spread -kernel_field F \\
14245 \> \> \> -group G -k 2 -catalogue 0 \\
14246 \> \> \> -end
14247
14248 # desarguesian spread, ago = 5760
14249
14250 create_spread_9b:
14251 \> $(ORBITER) -v 3 \\
14252 \> \> -define F -finite_field -q 3 -end \\
14253 \> \> -define G -linear_group -PGL 4 F -end \\
14254 \> \> -define S -spread -kernel_field F \\
14255 \> \> \> -group G -k 2 -catalogue 1 \\
14256 \> \> \> -end
14257
14258
14259 # Hall spread, ago = 1920
14260
14261
14262 create_spread_25.7:
14263 \> $(ORBITER) -v 3 \\
14264 \> \> -define F -finite_field -q 5 -end \\
14265 \> \> -define G -linear_group -PGL 4 F -end \\
14266 \> \> -define S -spread -kernel_field F \\
14267 \> \> \> -group G -k 2 -catalogue 7 \\
14268 \> \> \> -end
14269
14270
14271 SPREAD_SET_27_RAO_RAO="\n14272 0,0,0,0,0,0,0,0,0, \\
14273 1,1,0,2,1,1,0,0,2, \\
14274 1,0,1,1,2,2,0,1,0, \\
14275 1,2,2,1,2,0,2,2,2, \\
14276 0,0,2,2,2,0,1,2,0, \\
14277 1,1,2,0,2,1,2,1,0, \\
14278 0,1,0,1,0,1,0,2,1, \\
14279 2,0,2,0,0,2,1,1,0, \\
14280 818
create_spread_Rao_Rao_27:

\$\text{(ORBITER)} \ -v \ 3 \ \\
\$ -define \ F \ -finite_field \ -q \ 3 \ -end \ \\
\$ -define \ SS \ -vector \ -dense \ $(\text{SPREAD}\_\text{SET}\_27\_\text{RAO}\_\text{RAO}) \ -end \ \\
\$ -define \ G \ -linear\_\text{group} \ -PGL \ 6 \ F \ -end \ \\
\$ -define \ S \ -spread \ -kernel\_\text{field} \ F \ \\
\$ -group \ G \ -k \ 3 \ -spread\_set \ SS \ \\
\$ -end

\text{SPREAD}\_S27\_\text{RAO}\_\text{RAO}="

0, 33879, 5418, 13103, 30556, 22107, 27225, 4045, 24924, 31961, \\
0, 3196, 30100, 28081, 25862, 1339, 6696, 8242, 11747, 14000, 14705, \\
9784, 17843, 20772, 9271, 19413, 18678, 16109, 23924"

\text{desarguesian}\_\text{spread}\_\text{in}\_\text{PG}\_3\_2:

\$\text{(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

\text{pdflatex} \ Desarguesian\_Spread\_3\_2.tex
\text{open} \ Desarguesian\_Spread\_3\_2.pdf

\text{desarguesian}\_\text{spread}\_\text{in}\_\text{PG}\_5\_2:

\$\text{(ORBITER)} \ -v \ 3 \ \\

819
$\texttt{define FQ -finite\_field -q 8 -end}$

$\texttt{define Fq -finite\_field -q 2 -end}$

$\texttt{-with FQ -and Fq -do -finite\_field\_activity}$

$\texttt{cheat\_sheet\_desarguesian\_spread 2 -end}$

$\texttt{pdflatex Desarguesian\_Spread\_5.2.tex}$

$\texttt{open Desarguesian\_Spread\_5.2.pdf}$

$\texttt{desarguesian\_spread\_in\_PG\_3.4:}$

$\texttt{$(\text{ORBITER}) -v 3 \}$

$\texttt{-define FQ -finite\_field -q 16 -end}$

$\texttt{-define Fq -finite\_field -q 4 -end}$

$\texttt{-with FQ -and Fq -do -finite\_field\_activity}$

$\texttt{cheat\_sheet\_desarguesian\_spread 2 -end}$

$\texttt{pdflatex Desarguesian\_Spread\_3.4.tex}$

$\texttt{open Desarguesian\_Spread\_3.4.pdf}$

$\texttt{desarguesian\_spread\_in\_PG\_3.5:}$

$\texttt{$(\text{ORBITER}) -v 3 \}$

$\texttt{-define FQ -finite\_field -q 25 -end}$

$\texttt{-define Fq -finite\_field -q 5 -end}$

$\texttt{-with FQ -and Fq -do -finite\_field\_activity}$

$\texttt{cheat\_sheet\_desarguesian\_spread 2 -end}$

$\texttt{pdflatex Desarguesian\_Spread\_3.5.tex}$

$\texttt{open Desarguesian\_Spread\_3.5.pdf}$

$\texttt{classify\_spreads\_4:}$

$\texttt{$(\text{ORBITER}) -v 3 \}$

$\texttt{-define F -finite\_field -q 2 -end}$

$\texttt{-define P -projective\_space -n 3 -field F -v 0 -end}$

$\texttt{-define C -spread\_classifier}$

$\texttt{-projective\_space P}$

$\texttt{-k 2}$

$\texttt{-starter\_size 5}$

$\texttt{-poset\_classification\_control}$

$\texttt{-draw\_options}$

$\texttt{-embedded}$

$\texttt{-end}$

$\texttt{-problem\_label spreads\_2.2}$

$\texttt{-end}$

$\texttt{-output\_prefix "."}$

$\texttt{-end}$

$\texttt{-with C -do -spread\_classify\_activity}$

$\texttt{-compute\_starter}$

$\texttt{-problem\_label spreads\_2.2}$
classify_spreads_16_4:
\$(ORBITER) -v 4 \n-define F -finite_field -q 4 -end \n-define P -projective_space -n 3 -field F -v 0 -end \n-define C -spread_classifier \n-projective space P \n-k 2 \n-starter_size 17 \n-postet_classification_control \n-draw_options \n-radius 20 \n-nodes_empty \n-line_width 0.2 \n-embedded \n-end \n-problem_label spreads_16_4 \n-end \n-output_prefix "." \n-with C -do -spread_classify_activity \n-compute_starter \n-problem_label spreads_16_4 \n-W -depth 17 \n-report -end \n-end
\pdflatex spreads_16_4.poset_lvl_17.tex
\open spreads_16_4.poset_lvl_17.pdf

classify_spreads_25_starter_lift_case_0:
\$(ORBITER) -v 3 \n-define F -finite_field -q 5 -end \n-define P -projective_space -n 3 -field F -v 0 -end \n-define C -spread_classifier \n-projective space P \n
-k 2
-starter_size 5
-recoordinatize
-poset_classification_control
-draw_options
-radius 20
-nodes_empty
-line_width 0.2
-embedded
-end
-parm_problem_label spreads_25
-end
-output_prefix ""
-end
-with C -do -spread_classify_activity
-compute_starter
-parm_problem_label spreads_25
-W -depth 5
-report -end
-end
-
-parm_problem_label spreads_25
-W -depth 5
-report -end

#save colored graph fname=spreads_25_graph_0.bin
#save colored graph nb_vertices=225
#save colored graph nb_colors=21
#save colored graph nb_colors_per_vertex=1
#save colored_graph done
#colored_graph::save done
#Written file spreads_25_graph_0.bin of size 5914

spread_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

#graph_theoretic_activity::perform_activity Gr->label=spreads_25_graph_0 nb_sol = 7680
classify_spreads_25_starter_lifting_all_cases:

-define F -finite_field -q 5 -end
-define P -projective_space -n 3 -field F -v 0 -end
-define C -spread_classifier
-define projective_space P
-k 2
-starter_size 5
-recoordinatize
-poset_classification_control
-draw_options
-radius 20
-nodes_empty
-line_width 0.2
-embedded
-end
-W
-problem_label spreads_25
-end
-output_prefix ""
-end
-with C -do -spread_classify_activity
-compute_starter
-problem_label spreads_25
-W -depth 5
-report -end
-end

spreads_25_starter_cliques:

-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

spreads_25_isomorph:

-define F -finite_field -q 5 -end
-define P -projective_space -n 3 -field F -v 0 -end \
-define C -spread_classifier \
-define P -projective_space P \ 
-k 2 \ 
-starter_size 5 \ 
-recoordinatize \ 
-poset_classification_control \ 
-draw_options \ 
-radius 20 \ 
-nodes_empty \ 
-line_width 0.2 \ 
-embedded \ 
-W \ 
-problem_label spreads_25 \ 
-end \ 
-output_prefix "" \ 
-end \ 
-with C -do -spread_classify_activity \ 
-compute_starter \ 
-problem_label spreads_25 \ 
-W -depth 5 \ 
-report -end \ 
-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

-with C -do -spread_classify_activity

-isomorph

-prefix_iso "./spreads_25"

-use_database_for_starter

-isomorph_testing

-solution_prefix ""

-prefix_iso "./spreads_25"

-use_database_for_starter

-isomorph_report

-solution_prefix ""

-prefix_iso "./spreads_25_graph"

-end

-end

pdflatex spreads_25_isomorphism_types.tex

open spreads_25_isomorphism_types.pdf

# We found 21 isomorphism types

#1:33

classify_spreads_27_3_starter:

$ (ORBITER) -v 10 

-define F -finite_field -q 3 -end 

-define P -projective_space -n 5 -field F -v 0 -end 

-define C -spread_classifier 

-projective_space P 

-k 3 

-starter_size 5 

-recoordinatize 

-poset_classification_control 

-draw_options 

-radius 20 

-nodes_empty 

-line_width 0.2 

-embedded 

-end 

825
classify_spreads_27_starter_lift_all_cases:
$(ORBITER) -v 3 \
-define F -finite_field -q 3 -end \n-define P -projective_space -n 5 -field F -v 0 -end \n-define C -spread_classifier \n-projective_space P \n-k 3 \n-starter_size 5 \n-recoordinatize \n-poset_classification_control \n-draw_options \n-radius 20 \n-nodes_empty \n-line_width 0.2 \n-nodes_empty \n-embedded \n-end \n-W \n-problem_label spreads_27 \n-end \n-output_prefix "" \n-end \n
-with C -do -spread_classify_activity \
-compute_starter \
-problem_label spreads_27 \
-W -depth 5 \
-report -end \
-end \
-with C -do -spread_classify_activity \
-prepare_lifting_all_cases \
-end \

# 50 graphs 

spreads_27_starter_cliques: 

define G -graph -load spreads_27_graph_\%L.bin -end \
-with G -do \
-graph_theoretic_activity \
-find_cliques -rainbow -target_size 23 -end \
-end_loop

classify_spreads_27_isomorph_and_recognize: 

define F -finite_field -q 3 -end \
-projective_space \-v 0 -end \
-projective_space P \
-k 3 \ 
-starter_size 5 \ 
-recoordinatize \ 
-poset_classification_control \
-draw_options \
-radius 20 \
-nodes_empty \
-line_width 0.2 \
-embedded \
-end \
-W \

14749 \[ \text{with C -do -spread\_classify\_activity} \]
14750 \[ \text{-isomorph} \]
14751 \[ \text{-prefix\_iso "/spreads\_27"} \]
14752 \[ \text{-use\_database\_for\_starter} \]
14753 \[ \text{-isomorph\_report} \]
14754 \[ \text{-solution\_prefix ""} \]
14755 \[ \text{-base\_fname "spreads\_27\_graph"} \]
14756 \[ \text{-end} \]
14757 \[ \text{-end} \]
14758 \[ \text{with C -do -spread\_classify\_activity} \]
14759 \[ \text{-isomorph} \]
14760 \[ \text{-prefix\_iso "/spreads\_27"} \]
14761 \[ \text{-use\_database\_for\_starter} \]
14762 \[ \text{-recognize $(SPREAD\_S27\_RAO\_RAO)"} \]
14763 \[ \text{-solution\_prefix ""} \]
14764 \[ \text{-base\_fname "spreads\_27\_graph"} \]
14765 \[ \text{-end} \]
14766 \[ \text{-end} \]
14767 \[ \text{#pdf\_latex spreads\_27\_isomorphism\_types.tex} \]
14768 \[ \text{#open spreads\_27\_isomorphism\_types.pdf} \]
14769 \[ \text{#pdf\_latex spreads\_27\_aut\_group.tex} \]
14770 \[ \text{#open spreads\_27\_aut\_group.pdf} \]
14771
14772
14773 \[ \text{# SPREAD\_S27\_RAO\_RAO is isomorphic to spread 0 in the list} \]
14774 \[ \text{(which is different from the ordering of the Orbiter catalogue)} \]
14775 \[ \text{# the stabilizer of the spread has order 84.} \]
14776
14777
14778 \[ \text{#substructure\_lifting\_data::write\_hash\_and\_datref\_file id\_to\_hash\_tallied:} \]
14779 \[ \text{#( 1^6076, 2^289, 3^35, 4^5 )} \]
14780 \[ \text{# using 64 bit hash values, based on a modified version of Paul Hsieh´s SuperFast Hash} \]
14781
14782 \[ \text{#We found 7 isomorphism types} \]
14783 \[ \text{#0:36} \]
14784
14785 \[ \text{#generators for the stabilizer of the Rao/Rao spread:} \]
14786 \[ \text{#1,2,2,0,2,1,1,2,2,1,2,1,1,0,2,2,0,2,0,0,0,1,1,2,2,0,0,1,1,2,0,1} \]
14787
14788
14789
14790 \[ \text{create\_spread\_27\_0:} \]
14791 \[ \text{$(ORBITER) -v 3 \} \]
14792 \[ \text{-define F -finite\_field -q 3 -end \} \]
14793 \[ \text{-define G -linear\_group -PGL 6 F -end \} \]
define S -spread -kernel_field F \n
define G -k 3 -catalogue 0 \n
define G -k 3 -catalogue 1 \n
define G -k 3 -catalogue 2 \n
define G -k 3 -catalogue 3 \n
define F -finite_field -q 3 -end \n
define G -linear_group -PGL 6 F -end \n
define S -spread -kernel_field F \n
define G -k 3 -catalogue 1 \n
define G -k 3 -catalogue 2 \n
define G -k 3 -catalogue 3 \n
define F -finite_field -q 3 -end \n
define G -linear_group -PGL 6 F -end \n
define S -spread -kernel_field F \n
define G -k 3 -catalogue 1 \n
define G -k 3 -catalogue 2 \n
define G -k 3 -catalogue 3 \n
define F -finite_field -q 3 -end \n
define G -linear_group -PGL 6 F -end \n
define S -spread -kernel_field F \n
define G -k 3 -catalogue 1 \n
define G -k 3 -catalogue 2 \n
define G -k 3 -catalogue 3 \n
create spread 27.1:

create spread 27.2:

create spread 27.3:
create_spread_27.4:
$\text{ORBITER} -v 3 \$
\begin{verbatim}
> define F -finite_field -q 3 -end \\
> define G -linear_group -PGL 6 F -end \\
> define S -spread -kernel_field F \\
> group G -k 3 -catalogue 4 \\
> -end \\
> with S -do -spread_activity \\
> -report \\
> -end \\
pdflatex catalogue_q3_k3_4_report.tex \\
open catalogue_q3_k3_4_report.pdf \\
\end{verbatim}
create_spread_27.5:
$\text{ORBITER} -v 3 \$
\begin{verbatim}
> define F -finite_field -q 3 -end \\
> define G -linear_group -PGL 6 F -end \\
> define S -spread -kernel_field F \\
> group G -k 3 -catalogue 5 \\
> -end \\
> with S -do -spread_activity \\
> -report \\
> -end \\
pdflatex catalogue_q3_k3_5_report.tex \\
open catalogue_q3_k3_5_report.pdf \\
\end{verbatim}
create_spread_27.6:
$\text{ORBITER} -v 3 \$
\begin{verbatim}
> define F -finite_field -q 3 -end \\
> define G -linear_group -PGL 6 F -end \\
> define S -spread -kernel_field F \\
> group G -k 3 -catalogue 6 \\
> -end \\
> with S -do -spread_activity \\
> -report \\
> -end \\
pdflatex catalogue_q3_k3_6_report.tex \\
open catalogue_q3_k3_6_report.pdf \\
\end{verbatim}
classify_spreads_32_starter:

```
$ (ORBITER) -v 5 \n
define F -finite_field -q 2 -end \ndefine P -projective_space -n 9 -field F -v 0 -end \ndefine C -spread_classifier \ndefine P -projective_space P \nk 5 \n-starter_size 5 \n-recoordinatize \nposet_classification_control \ndraw_options \n-radius 20 \nnodes_empty \n-line_width 0.2 \n-embedded \n-end \n-depth 5 \n-problem_label spreads_32 \n-end \n-output_prefix "" \n-end \n-with C -do -spread_classify_activity \n-compute_starter \n-problem_label spreads_32 \n-depth 5 \n-report -end \n-end \n-with C -do -spread_classify_activity \n-prepare_lifting_single_case 0 \n-end
```

#we found 2887680 live points

#spread_classify::init The stabilizer of the first three components has order 599 96160

#poset_with_group_action::init Subset lattice degree of action = 109221651

# orbits at level in spreads_32_reps_lvl_4.csv

```
#ROW,REP,AGO,OL
#0,"0,109221650,3515245,6933315",252,1454127528968048128661913600
#1,"0,109221650,3515245,7030463",124,2955162397580226842119372800
#END
```

832
classify_spreads_49_starter_lift_all_cases:

$(ORBITER) -v 3

-define F -finite_field -q 7 -end

-define P -projective_space -n 3 -field F -v 0 -end

-define C -spread_classifier

-projective_space P

-starter_size 5

-recoordinatize

-poset_classification_control

-draw_options

-radius 20

-nodes_empty

-line_width 0.2

-embedded

-end

-W

-problem_label spreads_49

-end

-output_prefix ""

-end

-with C -do -spread_classify_activity

-compute_starter

-problem_label spreads_49

-W -depth 5

-report -end

-end

-end

-with C -do -spread_classify_activity

-prepare_lifting_all_cases

-end

#save_colored_graph fname=spreads_49_graph_125.bin

# 126 cases

spreads_49_starter_cliques_loop:

$(ORBITER) -v 2

-loop L 0 126 1

-define G -graph -load spreads_49_graph_%L.bin -end

-with G -do

-graph_theoretic_activity

-find_cliques -rainbow -target_size 45 -end

-end

-end_loop
spreads_49_starter_cliques_0:
$\$(ORBITER) -v 2 \
$\$(ORBITER) -v 2 -define G -graph -load spreads_49_graph_0.bin -end \
$\$(ORBITER) -with G -do \
$\$(ORBITER) -graph_theoretic_activity \
$\$(ORBITER) -define G -graph -load spreads_49_graph_0.bin -end \
$\$(ORBITER) -with G -do \
$\$(ORBITER) -graph_theoretic_activity \
$\$(ORBITER) -find_cliques -rainbow -target_size 45 -end \
$\$(ORBITER) -end 

# Section 12.2: Translation planes

SECTION_TRANSLATION_PLANES:

test_12_2:
make create_translation_plane_9b
make create_translation_plane_16_4_0
make create_translation_plane_16_2_0
make RREF_plane_16_2_0_rank_of_incma
make create_translation_plane_25_14_rank
make create_translation_plane_27_Rao_Rao
make RREF_Rao_Rao_rank_of_incma
make create_translation_plane_27_p_rank_of_incidence_matrix
make create_translation_plane_27_5_Rao_Rao
make create_translation_plane_27_4_block_stab
make create_translation_plane_27_5_block_stab
make create_translation_plane_27_6_block_stab

create_translation_plane_9b:
$\$(ORBITER) -v 3 \
$\$(ORBITER) -define F -finite_field -q 3 -end \
$\$(ORBITER) -define G -linear_group -PGL 4 F -end \
$\$(ORBITER) -define G1 -linear_group -PGL 5 F -end \
$\$(ORBITER) -define S -spread -kernel_field F \
$\$(ORBITER) -define T -translation_plane S G G1 -end \
$\$(ORBITER) -with T -do -translation_plane_activity \
$\$(ORBITER) -export_incma \
$\$(ORBITER) -end \
$\$(ORBITER) -with T -do -translation_plane_activity \n
834
define A -linear_group -import_group_of_plane T -end

-define Orb -orbits -group A

-on_points

-define Orb -orbits_activity

-report

-end

-define Orb -orbits_activity

-stabilizer 92

-end

-define Orb -orbits_activity

-export trees

-end

$(ORBITER) -v 2

-draw_matrix

-draw_matrix -input_csv_file plane_catalogue_q3_k2_1_incma.csv

-box_width 6 -b dialogs -b-empty sub

-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

deploy latex orbit_PGL_5_3_on_andre_3.draw.tex

deploy latex group_of_plane_PGL_5_3_on_andre_3.draw.pdf

deploy latex group_of_plane_PGL_5_3_on_andre_3.orbits_report.tex

deploy latex group_of_plane_PGL_5_3_on_andre_3.orbits_report.pdf

deploy latex group_of_plane_PGL_5_3_on_andre_3_stab_pt_92.report.tex

deploy latex group_of_plane_PGL_5_3_on_andre_3_stab_pt_92.report.pdf

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


15075 ▶ ▶ ▶ -export_incma \n15076 ▶ ▶ -end
15077 ▶ $(ORBITER) -v 2 \n15078 ▶ -draw_matrix \n15079 ▶ ▶ -input_csv_file plane_catalogue_q4_k2_0_incma.csv \n15080 ▶ ▶ -box_width 6 -bit_depth 8 \n15081 ▶ ▶ ▶ -partition 4 273 273 \n15082 ▶ ▶ -end
15083 ▶ open plane_catalogue_q4_k2_0_incma_draw.bmp
15084
15085
15086 #0 : "1200", // Hall spread
15087 #1 : "81600", // Desarguesian spread
15088 #2 : "576", // Semifield spread
15089
15090 create_translation_plane_16_2_0:
15091 ▶ $(ORBITER) -v 3 \n15092 ▶ ▶ -define F -finite_field -q 2 -end \n15093 ▶ ▶ -define G -linear_group -PGL 8 F -end \n15094 ▶ ▶ -define G1 -linear_group -PGL 9 F -end \n15095 ▶ ▶ -define S -spread -kernel_field F \n15096 ▶ ▶ ▶ -group G -k 4 -catalogue 0 \n15097 ▶ ▶ ▶ -end \n15098 ▶ ▶ ▶ -end \n15099 ▶ ▶ ▶ -define T -translation plane S G G1 -end \n15100 ▶ ▶ ▶ -with T -do -translation_plane_activity \n15101 ▶ ▶ ▶ -export_incma \n15102 ▶ ▶ -end
15103 ▶ $(ORBITER) -v 2 \n15104 ▶ ▶ -draw_matrix \n15105 ▶ ▶ ▶ -input_csv_file plane_catalogue_q2_k4_0_incma.csv \n15106 ▶ ▶ ▶ -box_width 6 -bit_depth 8 \n15107 ▶ ▶ ▶ ▶ -partition 4 273 273 \n15108 ▶ ▶ ▶ ▶ -end
15109 ▶ open plane_catalogue_q2_k4_0_incma_draw.bmp
15110
15111 #0 : "1008",
15112 #1 : "1008",
15113 #2 : "1728",
15114 #3 : "216",
15115 #4 : "360",
15116 #5 : "288",
15117 #6 : "3600",
15118 #7 : "244800",
15119
15120 RREF_plane_16_2_0_rank_of_incma:
15121 ▶ $(ORBITER) -v 2 \n
836
```plaintext
15122 \> \> -define F -finite_field -q 2 -end \\
15123 \> \> -define v -vector -field F \\
15124 \> \> \> -file plane_catalogue_q2_k4_0_incma.csv \\
15125 \> \> -end \\
15126 \> \> -with F -do -finite_field_activity \\
15127 \> \> -RREF v -normalize_from_the_right \\
15128 \> \> -end \\
15129
15130 # 2-rank is 106, so the plane is Lorimer-Rahilly 
15131
15132
15133 create_translation_plane_25_14_rank:
15134 \> $(ORBITER) -v 3 \\
15135 \> \> -define F -finite_field -q 5 -end \\
15136 \> \> -define G -linear_group -PGL 4 F -end \\
15137 \> \> -define G1 -linear_group -PGL 5 F -end \\
15138 \> \> -define S -spread -kernel_field F \\
15139 \> \> \> -group G -k 2 -catalogue 14 \\
15140 \> \> \> -end \\
15141 \> \> -define T -translation_plane S G G1 -end \\
15142 \> \> -with T -do -translation_plane_activity \\
15143 \> \> \> -export_incma \\
15144 \> \> -end \\
15145 \> $(ORBITER) -v 2 \\
15146 \> \> -define F -finite_field -q 2 -end \\
15147 \> \> -define v -vector -field F \\
15148 \> \> \> -file plane_catalogue_q5_k2_14_incma.csv \\
15149 \> \> -end \\
15150 \> \> -with F -do -finite_field_activity \\
15151 \> \> \> -RREF v -normalize_from_the_right \\
15152 \> \> -end \\
15153
15154
15155 # ToDo
15156
15157 create_translation_plane_27_Rao_Rao:
15158 \> $(ORBITER) -v 3 \\
15159 \> \> -define F -finite_field -q 3 -end \\
15160 \> \> -define SS -vector -dense $(SPREAD_SET_27_RAO_RAO) -end \\
15161 \> \> -define G -linear_group -PGL 6 F -end \\
15162 \> \> -define G1 -linear_group -PGL 7 F -end \\
15163 \> \> -define S -spread -kernel_field F \\
15164 \> \> \> -group G -k 3 -spread_set SS \\
15165 \> \> \> -end \\
15166 \> \> -define T -translation_plane S G G1 -end \\
15167 \> \> -with T -do -translation_plane_activity \\
15168 \> \> \> -export_incma \\
```
# error message because the group of the spread is not available

# creates plane_incma.csv

RREF_Rao_Rao_plane_incma_rank:

$\text{(ORBITER) -v 2} \backslash$

$\text{-define } F \text{-finite_field -q 3 -end} \backslash$

$\text{-define } v \text{-vector -field F} \backslash$

$\text{-file plane_incma.csv} \backslash$

$\text{-end} \backslash$

$\text{-with } F \text{-do -finite_field_activity} \backslash$

$\text{-RREF } v \text{-normalize_from_the_right} \backslash$

$\text{-end}$

# 3-rank is 271, so the Rao / Rao plane is Moorhouse IV.

create_translation_plane_27_p_rank_of_incidence_matrix:

$(\text{ORBITER) -v 3} \backslash$

$\text{-define } F \text{-finite_field -q 3 -end} \backslash$

$\text{-define } G \text{-linear_group -PGL 6 F -end} \backslash$

$\text{-define } G1 \text{-linear_group -PGL 7 F -end} \backslash$

$\text{-define } S \text{-spread -kernel_field F} \backslash$

$\text{-group } G \text{-k 3 -catalogue 6} \backslash$

$\text{-end} \backslash$

$\text{-define } T \text{-translation_plane } S G G1 \text{-end} \backslash$

$\text{-with } T \text{-do -translation_plane_activity} \backslash$

$\text{-p_rank 3} \backslash$

$\text{-end}$

# OCN : 3-rank : spread stab : pt-orb : line-orb : Moorhouse list


# 1 : 262 : 2106 : 1,27,729 : 1,27,729 : generalized twisted field = Moorhouse II

# 2 : 268 : 1014 : 2,26,729 : 1,54,702 : Andre = Moorhouse VII


# 4 : 274 : 274 : 28,729 : 1,756 : Hering = Moorhouse III


# so, Rao / Rao is OCN=5
create_translation_plane_27_5_Rao_Rao:

$\text{$(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$

-define T -translation_plane S G G1 -end$

-define A -linear_group -import_group_of_plane T -end$

-define Orb -orbits -group A$

create_translation_plane_27_4_block_stab:

$\text{$(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$

-define T -translation_plane S G G1 -end$

-define A -linear_group -import_group_of_plane T -end$

-define Orb -orbits -group A$

839
create_translation_plane.27.5_block_stab:

\begin{verbatim}
$\text{(ORBITER)} -v 3 \$
\end{verbatim}

\begin{verbatim}
$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 \$
$define T -translation_plane S G G1 -end \$
$with T -do -translation_plane_activity \$
$export_incma \$
$end \$
$define A -linear_group -import_group_of_plane T -end \$
$define Orb -orbits -group A \$
$on_points \$
$end

create_translation_plane.27.6_block_stab:

\begin{verbatim}
$\text{(ORBITER)} -v 3 \$
\end{verbatim}

\begin{verbatim}
$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 \$
$define T -translation_plane S G G1 -end \$
$with T -do -translation_plane_activity \$
$export_incma \$
$end \$
$define A -linear_group -import_group_of_plane T -end \$
$define Orb -orbits -group A \$
$on_points \$
$end
\end{verbatim}
Section 12.3: Packings

SECTION PACKINGS:

test 12 3:

∆ make spread_table_PG_3_4
∆ make spread_table_PG_3_5.regular
∆ make PG_3_5.element_of_order_31_GL_normalizer
∆ make PG_3_5.element_of_order_31_ME_normalizer
∆ make PG_3_5.assume_31_graph

# ToDo

spread_table_PG_3_4:
∆ mkdir SPREAD_TABLES_4
∆ $(ORBITER) -v 6 \\
∆ ▷ 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/"

5096448 spreads
1020 self dual spreads
User time: 56:38 on Mac

spread_table_PG_3_5.regular:
∆ mkdir SPREAD_TABLES_5_REG
∆ $(ORBITER) -v 6 \\
∆ ▷ 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/" \\
∆ ▷ -print_symbols

# error.

21 isomorphism types of spreads in PG(3,5)
12 is the index of the regular spread in the classification of spreads
PG_{3.5}.element_of_order_{31}.GL_normalizer:

$\text{(ORBITER)} -v 6 -\text{define G}\$

$\text{-linear_group -GL 4 5 -end}$

$\text{-with G -do }$

$\text{-group_theoretic_activity }$

$\text{-normalizer_of_cyclic_subgroup "124" }$

$\text{-with G -do }$

"2,0,0,0, 0,0,1,0, 0,0,0,1, 0,3,0,4"$

$\text{-end}$

$\text{mv normalizer_of_{31}.in_PGL_{4.5}.tex normalizer_of_{31}.AB.in_PGL_{4.5}.tex}$

$\text{pdflatex normalizer.of_{124}.in,GL_{4.5}.tex}$

$\text{open normalizer.of_{124}.in,GL_{4.5}.pdf}$

$\text{# needs magma}$

$\text{the group has order 124.}$

$\text{the normalizer has order 1488}$

$\text{normalizer has order 1488=4*372=4*4*3*31}$

PG_{3.5}.element_of_order_{31}.ME_normalizer:

$\text{(ORBITER)} -v 6 -\text{define G} \$

$\text{-linear_group -PGL 4 5 -end}\$

$\text{-with G -do}\$

$\text{-group_theoretic_activity }$

$\text{-normalizer_of_cyclic_subgroup "31" }$

"1,0,0,0, 0,3,4,3, 0,3,3,4, 0,3,2,3"$

$\text{-end}$

$\text{mv normalizer.of_{31}.in,PGL_{4.5}.tex normalizer.of_{31}.ME.in,PGL_{4.5}.tex}$

$\text{pdflatex normalizer.of_{31}.ME.in,PGL_{4.5}.tex}$

$\text{open normalizer.of_{31}.ME.in,PGL_{4.5}.pdf}$

$\text{# group has order 31}$

$\text{normalizer has order 372}$

PG_{3.5}.assume_{31}.graph:

$\text{(ORBITER)} -v 5 \$

$\text{-define F -finite_field -q 5 -end}\$

$\text{-define P -projective_space -n 3 -field F -end}\$

$\text{-define T -spread_table P 2 "12" "SPREAD_TABLES_5_REG/" }$

$\text{-define PW -packing_with_symmetry_assumption T}\$

$\text{-H "H31" $(PGL_{4.5}.SUBGROUP_{31}.ME) -end}\$

$\text{-N "N31" $(PGL_{4.5}.SUBGROUP_{31}.ME.NORMALIZER) -end}\$
-define PWF -packing_choose_fixed_points PW 0 -end \n-define L -packing_long_orbits PWF \n-orbit_length 31 -clique_size 1 -create_graphs -end \n-print_symbols
pdflatex H31_reduced_spread_orbits_orbits_report.tex
open H31_reduced_spread_orbits_orbits_report.pdf
pdflatex H31_line_orbits_orbits_report.tex
open H31_line_orbits_orbits_report.pdf
pdflatex H31_point_orbits_orbits_report.tex
open H31_point_orbits_orbits_report.pdf
pdflatex N31_line_orbits_orbits_report.tex
open N31_line_orbits_orbits_report.pdf
pdflatex H31_point_orbits_orbits_report.tex
open H31_point_orbits_orbits_report.pdf
pdflatex N31_point_orbits_orbits_report.tex
open N31_point_orbits_orbits_report.pdf

#pdflatex H31_spread_orbits_orbits_report.tex
#open H31_spread_orbits_orbits_report.pdf
#H31_line_orbits_orbits.bin
#H31_line_orbits_orbits_report.tex
#H_spread_orbits_orbit_types_report.tex
#H31_spread_orbits_orbits.bin
#H31_good_orbits
#H31_spread_types_reduced_orbit_types_report.tex
#H31_reduced_spread_orbits_orbits.bin
#H31_fpc0_lo.graph
H31:

# Section 12.4: BLT-sets
SECTION_BLT_SETS:

# make BLT_5_1
# make BLT_5_Linear
# make BLT_9_Kantor1
# make BLT_11_0
# make BLT_11_Fisher
# make BLT_11_Mondello
# make BLT_13_FTWKB
# make BLT_13_Kantor2
# make BLT_11_deep_search
# make BLT_13_deep_search
make BLT_13.classify_starter
make BLT_13.clique
make BLT_13.isomorph_read_DB
make BLT_13.isomorph_read_solutions
make BLT_13.isomorph_stabilizer_orbits
make BLT_13.isomorph_testing

:class: $(ORBITER) -v 2 \\
> -define F -finite_field -q 5 -end \\
> -define 0 -orthogonal_space 0 5 F -end \\
> -with 0 -do -orthogonal_space_activity \\
> -create_BLT_set -catalogue 1 -end \\
> -end

pdflatex BLT_catalogue_q5_iso1.tex
open BLT_catalogue_q5_iso1.pdf

:class: $(ORBITER) -v 2 \\
> -define F -finite_field -q 5 -end \\
> -define 0 -orthogonal_space 0 5 F -end \\
> -with 0 -do -orthogonal_space_activity \\
> -create_BLT_set -family "Linear" -end \\
> -end

pdflatex BLT_Linear_q5.tex
open BLT_Linear_q5.pdf

:class: $(ORBITER) -v 2 \\
> -define F -finite_field -q 9 -end \\
> -define 0 -orthogonal_space 0 5 F -end \\
> -with 0 -do -orthogonal_space_activity \\
> -create_BLT_set -family "Kantor1" -end \\
> -end

pdflatex BLT_Kantor1_q9.tex
open BLT_Kantor1_q9.pdf

:class: $(ORBITER) -v 2 \\
> -define F -finite_field -q 11 -end \\
> -define 0 -orthogonal_space 0 5 F -end \\
> -with 0 -do -orthogonal_space_activity \\
> -create_BLT_set -family "Kantor1" -end \\
> -end

pdflatex BLT_Kantor1_q11.tex
open BLT_Kantor1_q11.pdf

\textbf{Linear:}

:class: $(ORBITER) -v 2 \\
> -define F -finite_field -q 5 -end \\
> -define 0 -orthogonal_space 0 5 F -end \\
> -with 0 -do -orthogonal_space_activity \\
> -create_BLT_set -catalogue 1 -end \\
> -end

pdflatex BLT_catalogue_q5_iso1.tex
open BLT_catalogue_q5_iso1.pdf

\textbf{Kantor1:}

:class: $(ORBITER) -v 2 \\
> -define F -finite_field -q 9 -end \\
> -define 0 -orthogonal_space 0 5 F -end \\
> -with 0 -do -orthogonal_space_activity \\
> -create_BLT_set -family "Kantor1" -end \\
> -end

pdflatex BLT_Kantor1_q9.tex
open BLT_Kantor1_q9.pdf

\textbf{Kantor1:}

:class: $(ORBITER) -v 2 \\
> -define F -finite_field -q 11 -end \\
> -define 0 -orthogonal_space 0 5 F -end \\
> -with 0 -do -orthogonal_space_activity \\
> -create_BLT_set -family "Kantor1" -end \\
> -end

pdflatex BLT_Kantor1_q11.tex
open BLT_Kantor1_q11.pdf

\textbf{Linear:}

:class: $(ORBITER) -v 2 \\
> -define F -finite_field -q 5 -end \\
> -define 0 -orthogonal_space 0 5 F -end \\
> -with 0 -do -orthogonal_space_activity \\
> -create_BLT_set -catalogue 1 -end \\
> -end

pdflatex BLT_catalogue_q5_iso1.tex
open BLT_catalogue_q5_iso1.pdf

\textbf{Linear:}

:class: $(ORBITER) -v 2 \\
> -define F -finite_field -q 9 -end \\
> -define 0 -orthogonal_space 0 5 F -end \\
> -with 0 -do -orthogonal_space_activity \\
> -create_BLT_set -family "Linear" -end \\
> -end

pdflatex BLT_Linear_q5.tex
open BLT_Linear_q5.pdf

\textbf{Linear:}

:class: $(ORBITER) -v 2 \\
> -define F -finite_field -q 11 -end \\
> -define 0 -orthogonal_space 0 5 F -end \\
> -with 0 -do -orthogonal_space_activity \\
> -create_BLT_set -family "Linear" -end \\
> -end

pdflatex BLT_Linear_q11.tex
open BLT_Linear_q11.pdf
```
15497  ▶  ▶  -define 0 -orthogonal_space 0 5 F -end \n15498  ▶  ▶  -with 0 -do -orthogonal_space_activity \n15499  ▶  ▶  ▶  -create_BLT_set -catalogue 0 -end \n15500  ▶  ▶  -end
15501  ▶  #pdflatex 0_1_6_2_report.tex
15502  ▶  #open 0_1_6_2_report.pdf
15503
15504
15505  BLT_11_Fisher:
15506  ▶  $(ORBITER) -v 2 \n15507  ▶  ▶  -define F -finite_field -q 11 -end \n15508  ▶  ▶  -define 0 -orthogonal_space 0 5 F -end \n15509  ▶  ▶  -with 0 -do -orthogonal_space_activity \n15510  ▶  ▶  ▶  -create_BLT_set -family "Fisher" -end \n15511  ▶  ▶  -end
15512  ▶  pdflatex BLT_Fisher_q11.tex
15513  ▶  open BLT_Fisher_q11.pdf
15514
15515  BLT_11_Mondello:
15516  ▶  $(ORBITER) -v 2 \n15517  ▶  ▶  -define F -finite_field -q 11 -end \n15518  ▶  ▶  -define 0 -orthogonal_space 0 5 F -end \n15519  ▶  ▶  -with 0 -do -orthogonal_space_activity \n15520  ▶  ▶  ▶  -create_BLT_set -family "Mondello" -end \n15521  ▶  ▶  -end
15522  ▶  pdflatex BLT_Mondello_q11.tex
15523  ▶  open BLT_Mondello_q11.pdf
15524
15525
15526  BLT_13_FTWKB:
15527  ▶  $(ORBITER) -v 2 \n15528  ▶  ▶  -define F -finite_field -q 11 -end \n15529  ▶  ▶  -define 0 -orthogonal_space 0 5 F -end \n15530  ▶  ▶  -with 0 -do -orthogonal_space_activity \n15531  ▶  ▶  ▶  -create_BLT_set -family "FTWKB" -end \n15532  ▶  ▶  -end
15533  ▶  pdflatex BLT_FTWKB_q11.tex
15534  ▶  open BLT_FTWKB_q11.pdf
15535
15536
15537  # for K2, q must be congruent to 2 or 3 mod 5
15538  BLT_13_Kantor2:
15539  ▶  $(ORBITER) -v 2 \n15540  ▶  ▶  -define F -finite_field -q 13 -end \n15541  ▶  ▶  -define 0 -orthogonal_space 0 5 F -end \n15542  ▶  ▶  -with 0 -do -orthogonal_space_activity \n15543  ▶  ▶  ▶  -create_BLT_set -family "Kantor2" -end \n```
15544 ‡ ‡ -end
15545 ‡ pdflatex BLT_Kantor2_q13.tex
15546 ‡ open BLT_Kantor2_q13.pdf
15547
15548
15549
15550
15551
15552
15553
15554 BLT_11_deep_search:
15555 ‡ $(ORBITER) -v 2 \
15556 ‡ ‡ -define F -finite_field -q 11 -end \
15557 ‡ ‡ -define O -orthogonal_space 0 5 F -end \
15558 ‡ ‡ -define C -BLT_set_classifier 0 -starter_size 12 -end \
15559 ‡ ‡ -with C -do -BLT_set_classify_activity \n15560 ‡ ‡ ‡ -compute_starter \n15561 ‡ ‡ ‡ ‡ -problem_label BLT_q11 \n15562 ‡ ‡ ‡ ‡ -W -depth 12 \n15563 ‡ ‡ ‡ -end \n15564 ‡ ‡ -end
15565 ‡ #pdflatex BLT_q11_poset.tex
15566 ‡ #open BLT_q11_poset.pdf
15567
15568
15569
15570
15571 BLT_13_deep_search:
15572 ‡ $(ORBITER) -v 2 \
15573 ‡ ‡ -define F -finite_field -q 13 -end \
15574 ‡ ‡ -define O -orthogonal_space 0 5 F -end \
15575 ‡ ‡ -define C -BLT_set_classifier 0 -starter_size 14 -end \
15576 ‡ ‡ -with C -do -BLT_set_classify_activity \n15577 ‡ ‡ ‡ -compute_starter \n15578 ‡ ‡ ‡ ‡ -problem_label BLT_q13 \n15579 ‡ ‡ ‡ ‡ -W -depth 14 \n15580 ‡ ‡ ‡ -end \n15581 ‡ ‡ -end
15582 ‡ #pdflatex BLT_q13_poset.tex
15583 ‡ #open BLT_q13_poset.pdf
15584
15585
15586
15587
15588 BLT_13_classify_starter:
15589 ‡ $(ORBITER) -v 2 \
15590 ‡ ‡ -define F -finite_field -q 13 -end \n
846
-define O -orthogonal_space 0 5 F -end \
-define C -BLT_set_classifier 0 -starter_size 5 -end \
-with C -do -BLT_set_classify_activity \ 
-compute_starter \ 
-problem_label BLT_q13 \ 
-W -depth 5 \ 
-end \
-with C -do -BLT_set_classify_activity \ 
-create_graphs \ 
-end 

BLT_13.clique:

$\text{(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_loop

# 3 solutions:
BLT_q13_graph_5.0.sol.txt
BLT_q13_graph_5.0.sol.csv

BLT_13.isomorph_read_DB:

$\text{(ORBITER)} -v 2 \ 
-define F -finite_field -q 13 -end \ 
-define O -orthogonal_space 0 5 F -end \ 
-define C -BLT_set_classifier 0 -starter_size 5 -end \ 
-with C -do -BLT_set_classify_activity \ 
-compute_starter \ 
-problem_label BLT_q13 \ 
-W -depth 5 \ 
-end \
-with C -do -BLT_set_classify_activity \ 
-isomorph \ 
-prefix_iso "./BLT_q13" \ 
-use_database_for_starter \ 

847
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 0 -starter_size 5 -end \
  -with C -do -BLT_set_classify_activity \
  -compute_starter \
  -problem_label BLT_q13 \
  -W -depth 5 \
  -end \
  -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
```

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 0 -starter_size 5 -end \
  -with C -do -BLT_set_classify_activity \
  -compute_starter \
  -problem_label BLT_q13 \
  -W -depth 5 \
  -end \
  -end \
  -end \
  -with C -do -BLT_set_classify_activity \
  -isomorph \
  -prefix_iso './BLT_q13' \
  -use_database_for_starter \
  -compute_orbits \
```
BLT_13_isomorph_testing:

\$\text{\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbackslash d\textbacklash
# Section 13.1: Creating Graphs

SECTION_CREATING_GRAPHS:

test_13_1:

```
make Cycle_graph_13
make make_triangle_graph
make Chain_232
make Paley_13_graph
make trihedral_pair_graph
make small_graph
make petersen
make Johnson_6_2_0
make Hamming_graph_3
make Hamming_graph_7
make HJ_graph
make Group_Perm315_Orbital_3.colored_graph
make HJ_d2_graph
make Cayley_211_1mod3
make Cayley_Sym4_coxeter
make Cayley_Sym4_star

Cycle_graph_13:
```

```
$(ORBITER) -v 2 \
define Gamma -graph \
cycle 13 \
end
```

```
make_triangle_graph:
```

```
echo $(TRIANGLE_GRAPH) >triangle_graph.csv
$(ORBITER) -v 6 \
define G -graph \
load_csv_no_border \
triangle_graph.csv \
end
```

850
15779
15780 Chain_232:
15781 ▶ $(ORBITER) -v 2 \\
15782 ▶ ▶ -define P1 -vector -dense 2,3,2 -end \\
15783 ▶ ▶ -define P2 -vector -dense 2,3,2 -end \\
15784 ▶ ▶ -define Gamma -graph \\
15785 ▶ ▶ ▶ -chain_graph P1 P2 \\
15786 ▶ ▶ -end
15787
15788
15789
15790
15791 Paley_13_graph:
15792 ▶ $(ORBITER) -v 2 \\
15793 ▶ ▶ -define Gamma -graph -Paley 13 -end
15794
15795
15796
15797
15798
15799 trihedral_pair_graph:
15800 ▶ $(ORBITER) -v 2 \\
15801 ▶ ▶ -define Gamma \\
15802 ▶ ▶ ▶ -graph -trihedral_pair_disjointness_graph \\
15803 ▶ ▶ -end
15804
15805
15806 small_graph:
15807 ▶ $(ORBITER) -v 2 \\
15808 ▶ ▶ -define G -graph -edges_as_pairs \\
15809 ▶ ▶ ▶ 5 "0,1,0,2,0,3,0,4,1,3,1,4,2,4" \\
15810 ▶ ▶ -end
15811
15812
15813
15814
15815 petersen:
15816 ▶ $(ORBITER) -v 2 \\
15817 ▶ ▶ -define G -graph -Johnson 5 2 0 -end
15818
15819
15820
15821
15822 Johnson_6_2_0:
15823 ▶ $(ORBITER) -v 2 \\
15824 ▶ ▶ -define G -graph -Johnson 6 2 0 -end
15825

851
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 intersection array) was proved in Cohen & Tits (1985).

# HJ_graph:

```bash
$ (ORBITER) -v 2 \
  define G -graph -Hamming 3 2 -end
```

# Group Perm315_Orbital_3 colored_graph: halljanko315 gens.csv

```bash
$ (ORBITER) -v 2 \
  define G -graph \
  load_csv_no_border \
  halljanko315.csv \
  end
```

# HJ_d2_graph:

```bash
$ (ORBITER) -v 6 \
  define G -graph \
  load_csv_no_border \
  halljanko315.csv \
  distance 2 \
  end
```
Cayley $\mathbb{Z}_{11}$

\[ (\text{ORBITER}) -v 2 \]
\[ -\text{define F -finite \_field -q 11 -end} \]
\[ -\text{define S -vector -dense} \]
\[ "1,1, 1,4, 1,7, 1,10" -\text{-end} \]
\[ -\text{define G -linear \_group -AGL 1 F} \]
\[ -\text{subgroup \_by \_generators "Z11" 11 1 "1,1"} \]
\[ -\text{-end} \]
\[ -\text{define Gamma -graph} \]
\[ -\text{-Cayley \_graph G S} \]
\[ -\text{-end} \]

Cayley $\text{Sym} 4$ coxeter:

\[ (\text{ORBITER}) -v 2 \]
\[ -\text{define S -vector -dense "1,0,2,3, 0,2,1,3, 0,1,3,2" -end} \]
\[ -\text{define G -permutation \_group -symmetric \_group 4} \]
\[ -\text{-end} \]
\[ -\text{define Gamma -graph} \]
\[ -\text{-Cayley \_graph G S} \]
\[ -\text{-end} \]

Cayley $\text{Sym} 4$ star:

\[ (\text{ORBITER}) -v 2 \]
\[ -\text{define S -vector -dense "1,0,2,3, 2,1,0,3, 3,1,2,0" -end} \]
\[ -\text{define G -permutation \_group -symmetric \_group 4} \]
\[ -\text{-end} \]
\[ -\text{define Gamma -graph} \]
\[ -\text{-Cayley \_graph G S} \]
\[ -\text{-end} \]

\[ \text{graph \_of \_15 \_lines:} \]
\[ (\text{ORBITER}) -v 2 \]
\[ -\text{define G -graph} \]
\[ -\text{-load \_csv \_no \_border} \]
\[ -\text{\_15 \_lines.csv} \]
\[ -\text{-end} \]

# Section 13.2: Graphs Theoretic Activities
15918 SECTION_GRAPH_THEORETIC_ACTIVITIES:
15919
15920
15921 test_13.2:
15922 ▶ make triangle_graph_properties
15923 ▶ make Cycle_13 draw
15924 ▶ make Cycle_9 eigenvalues
15925 ▶ make Paley_13 draw
15926 ▶ make Paley_13 eigenvalues
15927 ▶ make Cayley_Z11“Imod3_eigenvalues_and_draw
15928 ▶ make Cayley_Sym4_coxeter_draw
15929 ▶ make Cayley_Sym5_coxeter_draw
15930 ▶ make Cayley_Sym4_star_eigenvalues_and_draw
15931 ▶ make graph.v5.e7.colored_graph
15932 ▶ make small_graph_draw
15933 ▶ make petersen_draw
15934 ▶ make Johnson_6_2_0_draw
15935 ▶ make Hamming_graph_3.draw
15936 ▶ make Hamming_graph_7.draw
15937 ▶ make Chain_232_properties
15938 ▶ make Chain_232_eigen
15939 ▶ make HJ_properties
15940 ▶ make HJ_d2_properties
15941 ▶ make PGO.5_2_collinearity_graph
15942 ▶ make trihedral_pair_graph_draw
15943
15944
15945 triangle_graph_properties:
15946 ▶ echo $(TRIANGLE GRAPH) >triangle_graph.csv
15947 ▶ $(ORBITER) -v 6 \n15948 ▶ ▶ -define G -graph \n15949 ▶ ▶ ▶ -load_csv_no_border \n15950 ▶ ▶ ▶ triangle_graph.csv \n15951 ▶ ▶ -end \n15952 ▶ ▶ -with G -do \n15953 ▶ ▶ ▶ -graph_theoretic_activity -properties \n15954 ▶ ▶ -end
15955
15956
15957 Cycle_13.draw:
15958 ▶ $(ORBITER) -v 2 \n15959 ▶ ▶ -define Gamma -graph -cycle 13 -end \n15960 ▶ ▶ -with Gamma -do \n15961 ▶ ▶ -graph_theoretic_activity -export_csv -end \n15962 ▶ ▶ -with Gamma -do \n15963 ▶ ▶ -graph_theoretic_activity -export_graphviz -end
15964 ▶ $(ORBITER) -v 2 -draw_matrix \n
854
Cycle_9_eigenvalues:

$\text{(ORBITER)} -v 2$

$-\text{define Gamma} -\text{graph}$

$-\text{cycle 9}$

$-\text{end}$

$-\text{with Gamma} -\text{do}$

$-\text{graph_theoretic_activity} -\text{eigenvalues} -\text{end}$

pdflatex Cycle_9.eigenvalues.tex

open Cycle_9.eigenvalues.pdf

Paley_13.draw:

$\text{(ORBITER)} -v 2$

$-\text{define Gamma} -\text{graph} -\text{Paley 13} -\text{end}$

$-\text{with Gamma} -\text{do}$

$-\text{graph_theoretic_activity} -\text{export.csv} -\text{end}$

$-\text{with Gamma} -\text{do}$

$-\text{graph_theoretic_activity} -\text{export.graphviz} -\text{end}$

$\text{(ORBITER)} -v 2 -\text{draw_matrix}$

$-\text{input.csv.file} \text{Paley}_13.csv$

$-\text{box_width 20} -\text{bit_depth 8} -\text{partition 4 13 13} -\text{end}$

$\text{dot -Tpng} \text{Paley}_13.gv >\text{Paley}_13.png$

open Paley_13.draw.bmp

Paley_13.eigenvalues:

$\text{(ORBITER)} -v 2$

$-\text{define Gamma} -\text{graph}$

$-\text{Paley 13}$

$-\text{end}$

$-\text{with Gamma} -\text{do}$

$-\text{graph_theoretic_activity} -\text{eigenvalues} -\text{end}$

pdflatex Paley_13.eigenvalues.tex

open Paley_13.eigenvalues.pdf

Cayley\_Z11\_1mod3.eigenvalues\_and\_draw:

$\text{(ORBITER)} -v 2$
\begin{verbatim}
16012 \> \> -draw_options -xin 2000000 \\
16013 \> \> \> -yin 2000000 -embedded -radius 20000 -end \\
16014 \> \> \> -define F -finite_field -q 11 -end \\
16015 \> \> \> -define S -vector -dense \\
16016 \> \> \> "1,1, 1,4, 1,7, 1,10" -end \\
16017 \> \> \> -define G -linear_group -AGL 1 F \\
16018 \> \> \> -subgroup_by_generators "Z11" 11 1 "1,1" \\
16019 \> \> \> -end \\
16020 \> \> \> -define Gamma -graph \\
16021 \> \> \> \> -Cayley_graph G S \\
16022 \> \> \> \> -end \\
16023 \> \> \> \> -with Gamma -do \\
16024 \> \> \> \> -graph_theoretic_activity -eigenvalues -end \\
16025 \> \> \> \> -with Gamma -do \\
16026 \> \> \> \> -graph_theoretic_activity -draw -end \\
16027 \> \> \> pdflatex Cayley_graph_AGL.1_11_draw.tex \\
16028 \> \> \> open Cayley_graph_AGL.1_11_draw.pdf \\
16029 \\
16030 Cayley_Sym4_coxeter_draw: \\
16031 \> $\$(ORBITER) \> \> -v 2 \ \\
16032 \> \> \> -draw_options -xin 2000000 -yin 2000000 \\
16033 \> \> \> \> -radius 20000 -embedded -nodes_empty -end \\
16034 \> \> \> \> -define S -vector -dense \\
16035 \> \> \> \> "1,0,2,3, 0,2,1,3, 0,1,3,2" -end \\
16036 \> \> \> \> -define G -permutation_group -symmetric_group 4 \\
16037 \> \> \> \> -end \\
16038 \> \> \> \> -define Gamma -graph \\
16039 \> \> \> \> \> -Cayley_graph G S \\
16040 \> \> \> \> \> -end \\
16041 \> \> \> \> \> -with Gamma -do \\
16042 \> \> \> \> \> -graph_theoretic_activity -draw -end \\
16043 \> \> \> \> pdflatex Cayley_graph_Sym.4_draw.tex \\
16044 \> \> \> \> open Cayley_graph_Sym.4_draw.pdf \\
16045 \\
16046 Cayley_Sym5_coxeter_draw: \\
16047 \> $\$(ORBITER) \> \> -v 2 \ \\
16048 \> \> \> -draw_options -xin 1000000 -yin 1000000 \\
16049 \> \> \> \> -embedded -radius 10000 -nodes_empty -end \\
16050 \> \> \> \> -define S -vector -dense \\
16051 \> \> \> \> "1,0,2,3,4, 0,2,1,3,4, 0,1,3,2,4, 0,1,2,4,3" -end \\
16052 \> \> \> \> -define G -permutation_group -symmetric_group 5 \\
16053 \> \> \> \> -end \\
16054 \> \> \> \> -define Gamma -graph \\
16055 \> \> \> \> \> -Cayley_graph G S \\
16056 \> \> \> \> \> -end \\
16057 \> \> \> \> \> -with Gamma -do \\
16058 \> \> \> \> \> -end \\
\end{verbatim}

16059 -graph_theoretic_activity -draw -end
16060 pdflatex Cayley_graph_Sym_5_draw.tex
16061 open Cayley_graph_Sym_5_draw.pdf
16062
16063
16064 Cayley_Sym4_star_eigenvalues_and_draw:
16065 $(ORBITER) -v 2 \$
16066 -draw_options -xin 1000000 -yin 1000000 -embedded -end \$
16067 -define S -vector -dense "1,0,2,3, 2,1,0,3, 3,1,2,0" -end \$
16068 -define G -permutation_group -symmetric_group 4 \$
16069 -end \$
16070 -define Gamma -graph \$
16071 -draw -end \$
16072 -end \$
16073 -with Gamma -do \$
16074 -graph_theoretic_activity -eigenvalues -end \$
16075 -with Gamma -do \$
16076 -graph_theoretic_activity -draw -end
16077 pdflatex Cayley_graph_Sym_4_draw.tex
16078 open Cayley_graph_Sym_4_draw.pdf
16079 pdflatex Cayley_graph_Sym_4_eigenvalues.tex
16080 open Cayley_graph_Sym_4_eigenvalues.pdf
16081
16082
16083
16084
16085 graph_v5_e7.colored_graph:
16086 $(ORBITER) -v 2 \$
16087 -define G -graph -edges_as_pairs 5 \$
16088 -define "0,1,0,2,0,3,0,4,1,3,1,4,2,4" \$
16089 -end \$
16090 -with G -do \$
16091 -graph_theoretic_activity -save -end
16092
16093
16094
16095 small_graph_draw:
16096 $(ORBITER) -v 2 \$
16097 -define G -graph -edges_as_pairs 5 \$
16098 -define "0,1,0,2,0,3,0,4,1,3,1,4,2,4" \$
16099 -end \$
16100 -with G -do \$
16101 -graph_theoretic_activity -export_csv -end \$
16102 -with G -do \$
16103 -graph_theoretic_activity -export_graphviz -end \$
16104 -with G -do \$
16105 -graph_theoretic_activity -save -end
16106  $(ORBITER) -v 2 -draw_matrix \
16107  -input_csv_file graph_v5_e7.csv \
16108  -box_width 40 -bit_depth 24 \
16109  -partition 4 "1,1,1,1" "1,1,1,1" -end 
16110  dot -Tpng graph_v5_e7.gv >graph_v5_e7.png
16111
16112  # creates graph_v5_e7.csv
16113  # creates graph_v5_e7.colored_graph
16114
16115  petersen_draw:
16116  $(ORBITER) -v 2 \n16117  -define G -graph -Johnson 5 2 0 -end \n16118  -with G -do \n16119  -graph_theoretic_activity -export.csv -end \n16120  -with G -do \n16121  -graph_theoretic_activity -export_graphviz -end \n16122  -with G -do \n16123  -graph_theoretic_activity -save -end 
16124  $(ORBITER) -v 2 -draw_matrix \
16125  -input_csv_file Johnson_5_2_0.csv \
16126  -box_width 40 -bit_depth 24 -partition 4 "10" "10" -end 
16127  dot -Tpng Johnson_5_2_0.gv >Johnson_5_2_0.png
16129
16130  Johnson_6_2_0_draw:
16131  $(ORBITER) -v 2 \n16132  -define G -graph -Johnson 6 2 0 -end \n16133  -with G -do \n16134  -graph_theoretic_activity -export.csv -end \n16135  -with G -do \n16136  -graph_theoretic_activity -export_graphviz -end \n16137  -with G -do \n16138  -graph_theoretic_activity -save -end 
16139  $(ORBITER) -v 2 -draw_matrix \
16140  -input_csv_file Johnson_6_2_0.csv \
16141  -box_width 40 -bit_depth 24 -partition 4 "10" "10" -end 
16142  dot -Tpng Johnson_6_2_0.gv >Johnson_6_2_0.png
16144
16145
16146  Hamming_graph_3_draw:
16147  $(ORBITER) -v 2 \n16148  -define G -graph -Hamming 3 2 -end \n16149  -with G -do \n16150  -graph_theoretic_activity -export.csv -end \n16151  -with G -do \n16152
Chain_232.properties:
$\texttt{(ORBITER) \ -v \ 2} \ \texttt{
-define P1 \ -vector \ -dense \ 2,3,2 \ -end} \ \texttt{
-define P2 \ -vector \ -dense \ 2,3,2 \ -end} \ \texttt{
-define Gamma \ -graph} \ \texttt{
-chain_graph P1 P2} \ \texttt{-end} \ \texttt{-with Gamma \ -do} \ \texttt{
-graph_theoretic_activity \ -export_csv} \ \texttt{-end} \ \texttt{-with Gamma \ -do} \ \texttt{
-graph_theoretic_activity \ -export_csv} \ \texttt{-end} \ \texttt{-with Gamma \ -do} \ \texttt{
-graph_theoretic_activity \ -properties} \ \texttt{-end} \ \texttt{-end}
Chain_232.eigen:
$\texttt{(ORBITER) \ -v \ 2} \ \texttt{
-define P1 \ -vector \ -dense \ 2,3,2 \ -end} \ \texttt{
-define P2 \ -vector \ -dense \ 2,3,2 \ -end} \ \texttt{859}
16200 ▶ ▶ -define Gamma -graph \ 
16201 ▶ ▶ ▶ -chain_graph P1 P2 \ 
16202 ▶ ▶ -end \ 
16203 ▶ ▶ ▶ -with Gamma -do \ 
16204 ▶ ▶ ▶ ▶ -graph_theoretic_activity \ 
16205 ▶ ▶ ▶ ▶ -eigenvalues \ 
16206 ▶ ▶ ▶ -end \ 
16207 ▶ ▶ pdflatex chain_graph_eigenvalues.tex \ 
16208 ▶ open chain_graph_eigenvalues.pdf \ 
16209 \ 
16210 \ 
16211 \ 
16212 # need the file halljanko315.csv \ 
16213 \ 
16214 HJ_properties: \ 
16215 ▶ $(ORBITER) -v 6 \ 
16216 ▶ ▶ -define G -graph \ 
16217 ▶ ▶ ▶ -load_csv_no_border \ 
16218 ▶ ▶ ▶ halljanko315.csv \ 
16219 ▶ ▶ ▶ -end \ 
16220 ▶ ▶ ▶ -with G -do \ 
16221 ▶ ▶ ▶ ▶ -graph_theoretic_activity -properties \ 
16222 ▶ ▶ ▶ ▶ -end \ 
16223 \ 
16224 #Degree type: \(10^{315}\) \ 
16225 \ 
16226 \ 
16227 \ 
16228 HJ_d2_properties: \ 
16229 ▶ $(ORBITER) -v 6 \ 
16230 ▶ ▶ -define G -graph \ 
16231 ▶ ▶ ▶ -load_csv_no_border \ 
16232 ▶ ▶ ▶ halljanko315.csv \ 
16233 ▶ ▶ ▶ -distance_2 \ 
16234 ▶ ▶ ▶ -end \ 
16235 ▶ ▶ ▶ -with G -do \ 
16236 ▶ ▶ ▶ ▶ -graph_theoretic_activity \ 
16237 ▶ ▶ ▶ ▶ -properties \ 
16238 ▶ ▶ ▶ ▶ -end \ 
16239 \ 
16240 \ 
16241 #Degree type: \(80^{315}\) \ 
16242 \ 
16243 \ 
16244 \ 
16245 \ 
16246 PGO_5_2_collinearity_graph: 0_5_2_incidence_matrix.csv \ 

860
trihtedral_pair_graph_draw:
$($ORBITER$) -v 2 -define Gamma \\n$($ORBITER$) -v 2 -draw_matrix \\nopen trihtedral_pair_disjointness_draw.bmp \\

graph_of_15_lines_properties:
$($ORBITER$) -v 2 \\n$($ORBITER$) -v 2 \\n
# Section 13.3: Graph Theory: Classification

SECTION_GRAPH_THEORY_CLASSIFICATION:
16294
16295 graph_classify_5:
16296 \$ (ORBITER) -v 2 \
16297 \$ -orбитер_path \$ (ORBITER_PATH) \
16298 -define GC -graph_classification \
16299 -n 5 \
16300 -poset_classification_control \
16301 -problem_label graphs_v5 \
16302 -depth 10 \
16303 -draw_options -radius 250 \
16304 -embedded \
16305 -end \
16306 -end \
16307 -end \
16308 -with GC -do \
16309 -graph_classification_activity \
16310 -list_graphs_at_level 5 5 \
16311 -end \
16312 -with GC -do \
16313 -graph_classification_activity \
16314 -draw_options \
16315 -radius 300 -nodes_empty \
16316 -line_width 1.5 \
16317 -scale 0.1 \
16318 -end \
16319 -draw_graphs_at_level 5 \
16320 -end \
16321 -print_symbols
16322 pdflatex graphs_v5_level_5_reps.tex
16323 open_graphs_v5_level_5_reps.pdf
16324 pdflatex graphs_v5_poset.tex
16325 open_graphs_v5_poset.pdf
16326
16327 tournament_classify_4:
16328 \$ (ORBITER) -v 2 \
16330 -define GC -graph_classification \
16331 -n 4 -tournament \
16332 -poset_classification_control \
16333 -problem_label tournament_4 \
16334 -depth 6 \
16335 -draw_options \
16336 -radius 250 -embedded \
16337 -end \
16338 -end \
16339 -end \
16340 -with GC -do \

862
16341 \> \> -graph_classification_activity \  
16342 \> \> \> -draw_options \  
16343 \> \> \> \> -radius 400 \  
16344 \> \> \> \> -line_width 2 -scale 0.10 \  
16345 \> \> \> -end \  
16346 \> \> \> -draw_graphs_at_level 6 \  
16347 \> \> -end \  
16348 \> \> -print_symbols \  
16349 \> pdflatex tournament_4_level_6_reps.tex \  
16350 \> open tournament_4_level_6_reps.pdf \  
16351 \> \  
16352 \  
16353 \  
16354 \  
16355 \  
16356 graph_classify_8_r3: \  
16357 \> $(ORBITER) -v 3 \  
16358 \> \> -define GC -graph_classification \  
16359 \> \> \> -n 8 -regular 3 \  
16360 \> \> \> -poset_classification_control \  
16361 \> \> \> \> -problem_label graphs_v8_r3 \  
16362 \> \> \> \> -depth 12 \  
16363 \> \> \> \> \> -draw_options -radius 250 \  
16364 \> \> \> \> \> \> -line_width 0.2 -embedded \  
16365 \> \> \> \> \> -end \  
16366 \> \> \> \> -end \  
16367 \> \> \> -end \  
16368 \> \> \> -with GC -do \  
16369 \> \> \> -group_theoretic_activity \  
16370 \> \> \> \> -draw_options \  
16371 \> \> \> \> \> -radius 400 \  
16372 \> \> \> \> \> \> -line_width 2 -scale 0.10 \  
16373 \> \> \> \> \> -end \  
16374 \> \> \> \> \> \> -draw_graphs_at_level 12 \  
16375 \> \> \> \> \> -end \  
16376 \> \> \> \> -print_symbols \  
16377 \> #pdflatex graphs_v8_r3_poset_lvl_12.tex \  
16378 \> #open graphs_v8_r3_poset_lvl_12.pdf \  
16379 \  
16380 \  
16381 \  
16382 \  
16383 Symmetric_4_inversion_graph_recognize: \  
16384 \> $(ORBITER) -v 10 \  
16385 \> \> -define G -permutation_group -symmetric_group 4 -end \  
16386 \> \> -with G -do \  
16387 \> \> -group_theoretic_activity \  

863
# pdflatex graphs_v5_poset.tex
# open graphs_v5_poset.pdf
# Section 13.4: Graph Theory: Clique finding

SECTION_GRAPH_THEORY_CLIQUE_FINDING:

```
test_13_4:
  make small_graph_cliques
  make BLT_q13_graph_5.0_cliques_bw
  make BLT_q13_graph_5.0_cliques_rainbow
  make small_graph_cliques_Sajeeb
  make Paley_13_aut
  make Paley_13
  make Paley_13_cliques_classify
  make Paley_13_cliques_all
  make PG0_5_2_cliques
  make HJ_d2_c5
  make HJ64_cliques5
  make HJ64_cliques5_classify

  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

  # nb_sol = 3

BLT_q13_graph_5.0_cliques_bw:
  $(ORBITER) -v 2 \
  -define G -graph -load BLT_q13_graph_5.0.bin -end \
  -with G -do \ 
  -graph_theoretic_activity \ 
  -find_cliques -target_size 9 \ 
  -end

  # all_cliques_black_and_white
```

865
16481 BLT_q13_graph_5_0_cliques_rainbow:
16482 \$ (ORBITER) -v 2 \n16483 \$ (ORBITER) -v 2 \n16484 \$ (ORBITER) -v 2 \n16485 \$ (ORBITER) -v 2 \n16486 \$ (ORBITER) -v 2 \n16487 \$ (ORBITER) -v 2 \n16488 \$ (ORBITER) -v 2 \n16489 \$ (ORBITER) -v 2 \n16490 \$ (ORBITER) -v 2 \n16491 \$ (ORBITER) -v 2 \n16492 \$ (ORBITER) -v 2 \n16493 \$ (ORBITER) -v 2 \n16494 \$ (ORBITER) -v 2 \n16495 \$ (ORBITER) -v 2 \n16496 \$ (ORBITER) -v 2 \n16497 \$ (ORBITER) -v 2 \n16498 \$ (ORBITER) -v 2 \n16499 \$ (ORBITER) -v 2 \n16500 \$ (ORBITER) -v 2 \n16501 \$ (ORBITER) -v 2 \n16502 \$ (ORBITER) -v 2 \n16503 \$ (ORBITER) -v 2 \n16504 \$ (ORBITER) -v 2 \n16505 \$ (ORBITER) -v 2 \n16506 \$ (ORBITER) -v 2 \n16507 \$ (ORBITER) -v 2 \n16508 \$ (ORBITER) -v 2 \n16509 \$ (ORBITER) -v 2 \n16510 \$ (ORBITER) -v 2 \n16511 \$ (ORBITER) -v 2 \n16512 \$ (ORBITER) -v 2 \n16513 \$ (ORBITER) -v 2 \n16514 \$ (ORBITER) -v 2 \n16515 \$ (ORBITER) -v 2 \n16516 \$ (ORBITER) -v 2 \n16517 \$ (ORBITER) -v 2 \n16518 \$ (ORBITER) -v 2 \n16519 \$ (ORBITER) -v 2 \n16520 \$ (ORBITER) -v 2 \n16521 \$ (ORBITER) -v 2 \n16522 \$ (ORBITER) -v 2 \n16523 \$ (ORBITER) -v 2 \n16524 \$ (ORBITER) -v 2 \n16525 \$ (ORBITER) -v 2 \n16526 \$ (ORBITER) -v 2 \n16527 \$ (ORBITER) -v 2

16488 # all_rainbow_cliques
16490
16491
16492 small_graph_cliques_Sajeeb:
16493 \$ (ORBITER) -v 2 \n16494 \$ (ORBITER) -v 2 \n16495 \$ (ORBITER) -v 2 \n16496 \$ (ORBITER) -v 2 \n16497 \$ (ORBITER) -v 2 \n16498 \$ (ORBITER) -v 2

16499 # nb_sol = 3
16500
16501
16502 Paley_13_aut:
16503 \$ (ORBITER) -v 2 \n16504 \$ (ORBITER) -v 2 \n16505 \$ (ORBITER) -v 2 \n16506 \$ (ORBITER) -v 2 \n16507 \$ (ORBITER) -v 2 \n16508 \$ (ORBITER) -v 2

16509 # writes Paley_13_group.makefile
16510 #User time: 0 of a second, dt=0 tps = 100
16511 #nb_calls_to_densenauty=1
16512
16513
16514
16515
16516 Paley_13:
16517 \$ (ORBITER) -v 2 \n16518 \$ (ORBITER) -v 2 \n16519 \$ (ORBITER) -v 2 \n16520 \$ (ORBITER) -v 2 \n16521

16522 Paley_13_cliques_classify:
16523 \$ (ORBITER) -v 4 \n16524 \$ (ORBITER) -v 4 \n16525 \$ (ORBITER) -v 4 \n16526 \$ (ORBITER) -v 4 \n16527 \$ (ORBITER) -v 4

866
-clique_test Gamma \
-depth 5 \n-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 13 -end 
-define Orb -orbits -group G 
-on_subsets 5 Control 
-end 

#User time: 0.01 of a second, dt=1 tps = 100

Paley_13.clique_all: 

$\text{clique_test Gamma} \\n-depth 5 \n-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 13 -end 
-define Orb -orbits -group G 
-on_subsets 5 Control 
-end 

PG0_5_2.clique: O_5_2.incidence_matrix.csv 

$\text{clique_test Gamma} \\n-depth 5 \n-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 13 -end 
-define Orb -orbits -group G 
-on_subsets 5 Control 
-end 

HJ_d2_c5: 

$\text{clique_test Gamma} \\n-depth 5 \n-end 
-define G -graph 
-load_csv_no_border \n-halljanko315.csv \n-distance 2 
-end 
-with G -do 
-graph_theoretic_activity \n-find_cliques -target_size 3 -end 
-end 

PG0_5_2.clique: O_5_2.incidence_matrix.csv 

$\text{clique_test Gamma} \\n-depth 5 \n-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 13 -end 
-define Orb -orbits -group G 
-on_subsets 5 Control 
-end 

HJ_d2_c5: 

$\text{clique_test Gamma} \\n-depth 5 \n-end 
-define G -graph 
-load_csv_no_border \n-halljanko315.csv \n-distance 2 
-end 
-with G -do 
-graph_theoretic_activity \n-find_cliques -target_size 5 -end 

16575 \> \> -end
16576
16577
16578
16579 #graph.theoretic.activity::perform_activity Gr->label=halljanko315 nb_sol = 26208 0
16580
16581
16582 HJ64.clique5:
16583 \> \> $(ORBITER) -v 6 \\n16584 \> \> \> -define Gamma -graph \\n16585 \> \> \> \> -load \\n16586 \> \> \> \> \> Group_Perms315_Orbital_3.colored_graph \\n16587 \> \> \> \> -end \\n16588 \> \> \> \> \> \> with Gamma -do \\n16589 \> \> \> \> -graph.theoretic.activity \\n16590 \> \> \> \> \> \> -find_cliques -target_size 5 -end \\n16591 \> \> \> \> \> \> \> -end
16592
16593 #graph.theoretic.activity::perform_activity Gr->label=Group_Perms315_Orbital_3 nb_sol = 1008
16594 #Group_Perms315_Orbital_3.sol.csv
16595
16596
16597
16598 HJ64.clique5.classify:
16599 \> \> $(ORBITER) -v 6 \\n16600 \> \> \> -define Control -poset_classification_control \\n16601 \> \> \> \> -W \\n16602 \> \> \> \> \> -problem_label HJ64.clique5 \\n16603 \> \> \> \> \> \> -clique_test Gamma \\n16604 \> \> \> \> \> \> \> -depth 5 \\n16605 \> \> \> \> \> \> \> \> -end \\n16606 \> \> \> \> \> \> \> \> \> -define Gamma -graph \\n16607 \> \> \> \> \> \> \> \> \> \> -load \\n16608 \> \> \> \> \> \> \> \> \> \> \> Group_Perms315_Orbital_3.colored_graph \\n16609 \> \> \> \> \> \> \> \> \> \> \> \> -end \\n16610 \> \> \> \> \> \> \> \> \> \> \> \> \> -define gens -vector \\n16611 \> \> \> \> \> \> \> \> \> \> \> \> \> \> -file halljanko315_gens.csv \\n16612 \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> -end \\n16613 \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> -define G -permutation_group \\n16614 \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> -bsgs halljanko315 "File\halljanko315" \\n16615 \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> 315 1209600 "0,1,42,95" 5 gens -end \\n16616 \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> -define Orb -orbits -group G \\n16617 \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> -on_subsets 5 Control \\n16618 \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> -end
16619

868
16620  #HJ64_cliques_reps_lvl_5.csv
16621
16622  # 1 orbit
16623  ROW,REP,AGO,OL
16624  #0,"0,8,31,110,283",1200,1008
16625  #END
16626
16627
16628
16629
16630
16631
16632  #############################################################################
16633  # Chapter 14 - Combinatorial Objects
16634  #############################################################################
16635
16636
16637  #############################################################################
16638  # Section 14.1: Finite Projective Spaces
16639
16640  SECTION_COMBINATORIAL_OBJECTS:
16641
16642  test.14.1:
16643  ▷ make Hirschfeld_q4_from_set
16644  ▷ make hyperoval_16_create
16645  ▷ make EC_read
16646  ▷ make PG_3.5_assume_31_read
16647  ▷ make LS_AG_2.3_read
16648  ▷ make geo_7.3_read
16649  ▷ make Desargues_path_lex_least_read
16650
16651
16652
16653  Hirschfeld_q4_from_set:
16654  ▷ $(ORBITER) -v 4 \n16655  ▷ ▷ -define H -set -here \n16656  ▷ ▷ ▷ $(HIRSCHFELD_SURFACE_Q4_SET_OF_POINTS) \n16657  ▷ ▷ ▷ -end \n16658  ▷ ▷ -define C -combinatorial_objects \n16659  ▷ ▷ ▷ -set_of_points H \n16660  ▷ ▷ ▷ -end
16661
16662
16663
16664  hyperoval_16_create:
16665  ▷ $(ORBITER) -v 2 \n16666
```
\$ORBITER -v 4
\$ORBITER -v 2
\$ORBITER -v 2
\$ORBITER -v 10
```

```
EC_read: elliptic_curve_b1_c3_q11.txt
PG_3_5_assume_31_read:
LS_AG_2_3_read:
geo_7_3_read:
Desargues_path_lex_least_read:
```
16714 $\$(ORBITER) -v 10 \n16715 \> \> -draw_incidense_structure_description \n16716 \> \> \> -width 60 -with_10 6 -end \n16717 \> \> \> -define C -combinatorial_objects \n16718 \> \> \> -file_of_incidense_geometries_by_row_ranks \n16719 \> \> \> \> Desargues_path_lex_least.inc 10 10 3 \n16720 \> \> \> -end
16721
16722
16723
16724 # Section 14.2: File Formats
16725 SECTION_FILE_FORMATS:
16726
16727 test_14_2:
16728 $\$(ORBITER) -v 10 \n16729 \> \> -define C -combinatorial_objects \n16730 \> \> \> -file_of_incidense_geometries \n16731 \> \> \> \> pasch.inc 6 4 12 \n16732 \> \> \> -end
16733
16734
16735
geo_pasch_read:
16736 $\$(ORBITER) -v 10 \n16737 \> \> -define C -combinatorial_objects \n16738 \> \> \> -file_of_incidense_geometries \n16739 \> \> \> \> pasch.inc 6 4 12 \n16740 \> \> \> -end
16741
16742
geo_pasch_given:
16743 $\$(ORBITER) -v 10 \n16744 \> \> -define C -combinatorial_objects \n16745 \> \> \> -incidence_geometry \n16746 \> \> \> \> "0,1,4,6,8,11,13,14,17,19,22,23" \n16747 \> \> \> \> 6 4 12 \n16748 \> \> \> -end
16749
16750
16751
16752
16753
16754
16755 # Chapter 15 - Canonical Forms with Nauty
16756
16757
16758
16759
16760

871
16761 # Section 15.1: Overview of Canonical Forms
16762
16763 16764 SECTION OVERVIEW_CANNONICAL_FORMS:
16765 16766 16767 16768 16769
16770 #################################################################
16771 # Section 15.2: Objects in projective Space
16772 16773 16774 SECTION_OBJECTS_IN_PROJECTIVE_SPACE:
16775 16776 16777 test_15_2:
16778 ▶ make EC_canon
16779 ▶ make Hirschfeld_q4_c
16780 ▶ make Hirschfeld_q4_set_c
16781 ▶ make Dickson_sets_stabilizer
16782 ▶ make Endrass_7c
16783 ▶ make hyperoval_16_canonical_form
16784 ▶ make cubic_curves_PG_2_8_cannon
16785 ▶ make F_alpha_beta_gamma_delta_classify_q7_nauty
16786 ▶ make ovoid_q8_cannon
16787 ▶ make ovoid_ST_q8_cannon
16788 ▶ make Suzuki_8
16789
16790 16791 # ToDo:
16792 16793 EC_canon: elliptic_curve_b1_c3_q11.txt
16794 ▶ $(ORBITER) -v 3 \n16795 ▶ ▶ -define C -combinatorial_objects \n16796 ▶ ▶ ▶ -file_of_points elliptic_curve_b1_c3_q11.txt \n16797 ▶ ▶ -end \n16798 ▶ ▶ -define F -finite_field -q 11 -end \n16799 ▶ ▶ -define P -projective_space -n 2 -field F -v 0 -end \n16800 ▶ ▶ -with C -do \n16801 ▶ ▶ ▶ -combinatorial_object_activity \n16802 ▶ ▶ ▶ -canonical_form_PG P \n16803 ▶ ▶ ▶ ▶ -classification_prefix EC \n16804 ▶ ▶ ▶ ▶ -label EC \n16805 ▶ ▶ ▶ ▶ -save_ago \n16806 ▶ ▶ ▶ ▶ -max_TDO_depth 4 \n16807 ▶ ▶ ▶ -end \n
872
pdflatex EC\_classification.tex
open EC\_classification.pdf

$(\text{ORBITER})$ -v 2 -draw matrix \n
-\input\_csv\_file \text{EC\_object0\_TDA\_flag\_orbits.csv}
-secondary\_input\_csv\_file \text{EC\_object0\_TDA.csv}
-box\_width 20 -bit\_depth 24
-end

open \text{EC\_object0\_TDA\_flag\_orbits\_draw.bmp}

pdflatex Hirschfeld\_surface\_q4\_classification.tex
open Hirschfeld\_surface\_q4\_classification.pdf
# group order is 51840

HIRSCHFELD_STAB_GENERATORS="1,0,0,0,0,1,0,0,0,1,0,0,0,0,1,1, 1,0,0,0,0,2,0,0,0,0,2,0,0,0,1,0,0,0,3,0,0,0,0,3,0,1,0,0,1,0, 1,0,0,0,0,1,0,0,1,1,1,0,0,1,0,0,1,1,1,0,0,1,0,0,1,1,1,0,0,1,0,0,1,0,0,0,1,0,0,1,0,0,1,0,0,1,0,1,0,0,1,1,1,0,0,1,0,0,1,0,0,0,1,0,0,1,0,0,1,0,0,0,1,0,0,1,0,0"

Hirschfeld_stab

subgroup:

"$(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 -define Gsp -modified_group -from G -create_special_subgroup -end -with Gsp -do -group_theoretic_activity -report -end -define Orb -orbits -group Gsp -on_points -end

Hirschfeld_stab_orbits:

"$(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 -report -end -define Orb -orbits -group G -on_points -end

Hirschfeld_stab_subgroup:

#orbits_PGGL_4_4_Subgroup_Hirschfeld_Stab_51840_orbit_0.csv
Hirschfeld_q4_set_c:

```plaintext
$(ORBITER) -v 4 \
-define H -set -here \n$(HIRSCHFELD_SURFACE_Q4_SET_OF_POINTS) \n-define C -combinatorial_objects \n-set_of_points H \n-define F -finite_field -q 4 -end \n-define P -projective_space -n 3 -field F -v 0 -end \n-with C -do \n-combinatorial_object_activity \n-canoncial_form PG P \n-classification_prefix Hirschfeld_surface_q4 \n-label Hirschfeld_surface_q4 \n-save_ago \n-max_TDO_depth 4 \n-end \n-report \n-prefix Hirschfeld_surface_q4 \n-show_TDO \n-show_TDA \n-end \n-end
```

Dickson_sets_stabilizer:

```plaintext
$(ORBITER) -v 3 \
-define C -combinatorial_objects \n-set_of_points "0,1,2,5,6" \n-set_of_points "0,1,2,3,6" \n-set_of_points "0,1,2,3,4" \n-set_of_points "0,1,2,3,8" \n-set_of_points "0,1,2,5,6,7,8" \n-set_of_points "0,1,2,3,5,6,7" \n-set_of_points "0,1,2,3,5,6,9" \n-set_of_points "0,1,2,3,5,6,10" \n-set_of_points "0,1,2,3,5,6,4" \n-set_of_points "0,1,2,3,5,6,8,11,13" \n-set_of_points "3,6,9,7,10,12,8,11,13,14,4" \n-set_of_points "3,5,6,9,7,10,12,11,13,14,4" \n-set_of_points "0,1,2,3,5,6,9,7,10,12,4" \n-define F -finite_field -q 2 -end \n```
-define P -projective_space -n 3 -field F -v 0 -end \
-with C -do \
-combinatorial_object_activity \
-define P -projective_space -n 3 -field F -v 0 -end \
-define C -combinatorial_objects \
-file_of_points Endrass_F7.txt \
-define F -finite_field -q 7 -end \
-define P -projective_space -n 3 -field F -v 0 -end \
-with C -do \
-combinatorial_object_activity \
-cannotonical_form_PG P \
-classification_prefix Endrass_F7 \
-label Endrass_F7 \
-save Ago \
-max_TDO_depth 4 \
-end \
-report \
-prefix Endrass_F7 \
-show_TDO \
-show_TDA \
-end \
-pdflatex Endrass_F7_classification.tex \
-open Endrass_F7_classification.pdf 

Endrass_7c: Endrass_F7.txt 

pdflatex Endrass_F7_classification.tex 

open Endrass_F7_classification.pdf 

# ToDo group order is computed wrong 
# should be: group order is 32 

876
hyperoval
canonical form:

\[ \text{ORBITER} -v 2 \]
\[ \text{-define C -combinatorial_objects} \]
\[ \text{-set_of_points $(HYPEROVAL, 16, 16320)$} \]
\[ \text{-set_of_points $(HYPEROVAL, 16, 164)$} \]
\[ \text{-define F -finite_field -q 16 -end} \]
\[ \text{-define P -projective_space -n 2 -field F -v 0 -end} \]
\[ \text{-with C -do} \]
\[ \text{-combinatorial_object_activity} \]
\[ \text{-canonical_form PG P} \]
\[ \text{-classification_prefix hyperoval_q16} \]
\[ \text{-label hyperoval_q16} \]
\[ \text{-save_ago} \]
\[ \text{-save_transversal} \]
\[ \text{-max_TDO_depth 10} \]
\[ \text{-end} \]
\[ \text{-report} \]
\[ \text{-prefix hyperoval_q16} \]
\[ \text{-export_flag_orbits} \]
\[ \text{-show_TDO} \]
\[ \text{-show_TDA} \]
\[ \text{-dont_show_incidence_matrices} \]
\[ \text{-export_group_GAP} \]
\[ \text{-end} \]
\[ \text{pdflatex hyperoval_q16_classification.tex} \]
\[ \text{open hyperoval_q16_classification.pdf} \]
\[ \text{ORBITER} -v 2 -draw_matrix \]
\[ \text{-input_csv_file hyperoval_q16_object0_TDA_flag_orbits.csv} \]
\[ \text{-secondary_input_csv_file hyperoval_q16_object0_TDA.csv} \]
\[ \text{-box_width 4 -bit_depth 24} \]
\[ \text{-end} \]
\[ \text{open hyperoval_q16_object0_TDA_flag_orbits_draw.bmp} \]
\[ \text{ORBITER} -v 2 -draw_matrix \]
\[ \text{-input_csv_file hyperoval_q16_object1_TDA_flag_orbits.csv} \]
\[ \text{-secondary_input_csv_file hyperoval_q16_object1_TDA.csv} \]
\[ \text{-box_width 4 -bit_depth 24} \]
\[ \text{-end} \]
\[ \text{open hyperoval_q16_object1_TDA.flag_orbits.draw.bmp} \]

cubic_curves_PG_2_8.canon:
$(\text{ORBITER}) -v 6 \$

```
#define C -combinatorial_objects \
#define F -finite_field -q 8 -end \
#define P -projective_space -n 2 -field F -v 0 -end \
-with C -do \
-combinatorial_object_activity \
#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 \
-classification_prefix cc_8 \
-save_ago \
-max_TDO_depth 10 \
-end \
-report \
-end \
pdflatex cc_8_classification.tex 
open cc_8_classification.pdf 
```

```
F\_alpha\_beta\_gamma\_delta\_classify\_q7\_nauty: F\_alpha\_beta\_gamma\_delta\_q7\_points.txt 
```

```
pdflatex cc_8_classification.tex 
open cc_8_classification.pdf 
```

```
F\_alpha\_beta\_gamma\_delta\_classify\_q7\_nauty: F\_alpha\_beta\_gamma\_delta\_q7\_points.txt 
```

```
pdflatex surface_15_lines_q7_classification.tex 
open surface_15_lines_q7_classification.pdf 
```

```
#pdflatex surface_15_lines_q7_classification.txt 
#open surface_15_lines_q7_classification.pdf 
```

```
17082 #User time: 4:12 on Mac
17083 # 6 orbits
```

```
878
```
$\text{ORBITER}$ -v 6 \\
-define C -combinatorial_objects \\
-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 \\
-classification_prefix ovoid \\
-label ovoid \\
-save_ago \\
-max.TDO.depth 4 \\
-end \\
-report \\
-prefix ovoid \\
-show_TDO \\
-show_TDA \\
-dont_show_incidence_matrices \\
-export_group_GAP \\
-end \\
-report \\
-prefix ovoid \\
-export_flag_orbits \\
-show_TDO \\
-show_TDA \\
-dont_show_incidence_matrices \\
-export_group_GAP \\
-end \\
-report \\
-prefix ovoid \\
-export_flag_orbits \\
-show_TDO \\
-show_TDA \\
-dont_show_incidence_matrices \\
-export_group_GAP \\
-end \\
-report \\
-prefix ovoid \\
-export_flag_orbits \\
-show_TDO \\
-show_TDA \\
-dont_show_incidence_matrices \\
-export_group_GAP \\
-end
Suzuki_8:

```
# group order 87360 = 3 * 29120
SUZUKI_8_GENERATORS="
1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1, 
1,0,0,0,6,0,0,0,0,2,0,0,0,0,3,0, 
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"
```

```
# Section 15.3: Incidence Geometries
SECTION INCIDENCE GEOMETRIES:
```
17181
17182 test_15_3:
17183 ▶ make geo.7.3_c
17184 ▶ make geo.10.3_c
17185 ▶ make geo.10.3_c_lex_least
17186 ▶ make geo.14.3_c
17187 ▶ make geo.15.3_c
17188 ▶ make TFC.24.3_c
17189 ▶ make geo.40.4_g4.c
17190 ▶ make geo.17.3_g4.c
17191 ▶ make AG.2.3_c
17192 ▶ make geo.LSQ6.c
17193 ▶ make quartic_curve.25.0.0_canonical
17194 ▶ make geo.16.c
17195
17196
17197 geo.7.3_c:
17198 ▶ $(ORBITER) -v 10 \ 
17199 ▶ ▶ -draw_incidence_structure_description \ 
17200 ▶ ▶ ▶ -width 60 -with_10 6 -end \ 
17201 ▶ ▶ -define C -combinatorial_objects \ 
17202 ▶ ▶ ▶ -file_of_incidence_geometries 7.3.inc 7 7 21 \ 
17203 ▶ ▶ -end \ 
17204 ▶ ▶ -with C -do \ 
17205 ▶ ▶ -combinatorial_object_activity \ 
17206 ▶ ▶ ▶ -canonical_form \ 
17207 ▶ ▶ ▶ ▶ -classification_prefix 7.3 \ 
17208 ▶ ▶ ▶ ▶ -label 7.3 \ 
17209 ▶ ▶ ▶ ▶ -save_ago \ 
17210 ▶ ▶ ▶ ▶ -save_transversal \ 
17211 ▶ ▶ ▶ -end \ 
17212 ▶ ▶ ▶ -report \ 
17213 ▶ ▶ ▶ ▶ -prefix 7.3 \ 
17214 ▶ ▶ ▶ ▶ -export_flag_orbits \ 
17215 ▶ ▶ ▶ ▶ -show_incidence_matrices \ 
17216 ▶ ▶ ▶ ▶ -export_group_GAP \ 
17217 ▶ ▶ ▶ ▶ -end \ 
17218 ▶ ▶ -end
17219 ▶ pdflatex 7.3_classification.tex
17220 ▶ open 7.3_classification.pdf
17221 ▶ $(ORBITER) -v 2 -draw_matrix \ 
17222 ▶ ▶ -input_csv_file 7.3_object0_TDA_flag_orbits.csv \ 
17223 ▶ ▶ -secondary_input_csv_file 7.3_object0_TDA.csv \ 
17224 ▶ ▶ -box_width 32 -bit_depth 24 \ 
17225 ▶ ▶ -end
17226 ▶ $(ORBITER) -v 2 -draw_matrix \ 
17227 ▶ ▶ -input_csv_file 7.3_object0_INP_flag_orbits.csv \ 

881
geo_10_3_c:
$\text{(ORBITER) \ -v\ 10 \ }
\text{-draw\_incidence\_structure\_description \ }
\text{-width\ 60 \ -with\_10\ 6 \ -end \ }
\text{-define\ C \ -combinatorial\_objects \ }
\text{-file\_of\_incidence\_geometries\ 10_3.inc\ 10\ 10\ 30 \ }
\text{-end \ }
\text{-width\ 60 \ -with\ 10 6 \ -end \ }
\text{-define\ C \ -combinatorial\_objects \ }
\text{-file\_of\_incidence\_geometries\ 10_3.inc\ 10\ 10\ 30 \ }
\text{-end \ }
\text{-report \ }
\text{-prefix\ 10_3 \ }
\text{-export\_flag\_orbits \ }
\text{-export\_group\_GAP \ }
\text{-end \ }
pdflatex\ 10_3\_classification.tex
open\ 10_3\_classification.pdf
$\text{(ORBITER) \ -v\ 2 \ -draw\_matrix \ }
\text{-input\\_csv\\_file\ 10_3\_object7\_TDA\_flag\_orbits.csv \ }
\text{-secondary\input\\_csv\\_file\ 10_3\_object7\_TDA.csv \ }
\text{-box\_width\ 16 \ -bit\_depth\ 24 \ }
\text{-end \ }
$\text{(ORBITER) \ -v\ 2 \ -draw\_matrix \ }
\text{-input\\_csv\\_file\ 10_3\_object7\_INP\_flag\_orbits.csv \ }
\text{-secondary\input\\_csv\\_file\ 10_3\_object7\_INP.csv \ }
\text{-box\_width\ 16 \ -bit\_depth\ 24 \ }
\text{-end \ }
geo_10_3_c\_lex\_least:
$\text{(ORBITER)}$ -v 10 \\
\text{-draw.incidence_structure_description} \\
\text{-width 60 -with 10 6 -end} \\
\text{-define Test_lines -set -loop 4 11 1 -end} \\
\text{-define Geo -geometry_builder} \\
\text{-V 10 -B 10 -TDO 3 -fuse 1} \\
\text{-fname Geo 10_3} \\
\text{-test Test_lines} \\
\text{-end} \\
\text{-define C -combinatorial_objects} \\
\text{-file of incidence geometries 10_3.inc 10 10 30} \\
\text{-end} \\
\text{-with C -do} \\
\text{-combinatorial.object.activity} \\
\text{-canonical_form} \\
\text{-classification.prefix 10_3} \\
\text{-label 10_3} \\
\text{-save ago} \\
\text{-save transversal} \\
\text{-end} \\
\text{-report} \\
\text{-prefix 10_3} \\
\text{-export.flag.orbits} \\
\text{-show.incidence.matrices} \\
\text{-export.group.GAP} \\
\text{-show.TDO} \\
\text{-show.TDA} \\
\text{-lex.least Geo} \\
\text{-end} \\
\text{-end} \\
\text{pdflatex 10_3.classification.tex} \\
\text{open 10_3.classification.pdf} \\
$\text{(ORBITER)}$ -v 2 -draw.matrix \\
\text{-input.csv_file 10_3.object7.TDA.flag.orbits.csv} \\
\text{-secondary.input.csv_file 10_3.object7.TDA.csv} \\
\text{-box.width 16 -bit.depth 24} \\
\text{-end} \\
\text{pdflatex 10_3_object7_TDA_flag_orbits.csv}
echo $(FILE_24_3_TFC_INC) >24_3_TFC.inc
$(ORBITER) -v 6 \\n-define C -combinatorial_objects \\
-file_of_incidence_geometries 24_3_TFC.inc 24 24 72 \\
-end \\
-with C -do \\
-combinatorial_object_activity \\
-canoncial_form \\
-classification_prefix 24_3_TFC \\
-label 24_3_TFC \\
-save_ago \\
-end \\
-report \\
-prefix 24_3_TFC \\
-export_flag_orbits \\
-show_TDO \\
-show_TDA \\
-show_incidence_matrices \\
-end \\
-report \\
-prefix 24_3_TFC \\
-export_flag_orbits \\
-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

geo_40_4_g4.c:
$(ORBITER) -v 2 \\
-draw_incidence_structure_description \\
-width 50 -with 10 5 -end \\
-define C -combinatorial_objects \\
-file_of_incidence_geometries 40_4_g4.inc 40 40 160 \\
-end \\
-with C -do \\
-combinatorial_object_activity \\
-canoncial_form \\
-classification_prefix 40_4_g4 \\
-label 40_4_g4 \\
-save_ago \\
-end \\
-report \\
-prefix 40_4_g4 \\
-export_flag_orbits \\
-open 24_3_TFC_object2_TDA_flag_orbits_draw.bmp

885
$\text{ORBITER} -v 2$

- draw_incidence_structure_description
  - width 50 -with 10 5 -end
- define C -combinatorial_objects
- file_of_incidence_geometries AG_2_3.inc 9 12 36
- end

- with C -do
- combinatorial_object_activity
  - canonical_form
  - classification_prefix AG_2_3
  - label AG_2_3
  - save_ago
- end

- report
- prefix AG_2_3
- export_flag_orbits
- show_TDO
- show_TDA
- show_incidence_matrices
- end

- show incidence matrices
- open AG_2_3_classification.pdf

$\text{ORBITER} -v 2$

- define C -combinatorial_objects
- file_of_incidence_geometries AG_2_3.inc 9 12 36
- end

- with C -do
- combinatorial_object_activity
  - canonical_form
  - classification_prefix AG_2_3
  - label AG_2_3
  - save_ago
  - max_TDO_depth 10
pdflatex AG_2.3_classification.tex
open AG_2.3_classification.pdf

geo_LSQ6.c:
$(ORBITER) -v 10
$ORBITER \-draw_incidence_structure_description
\-width 60 \-with 10 6 \-end
\-define C \-combinatorial_objects
\-file_of_incidence_geometries
\-LSQ6.inc 18 39 126
\-end
\-with C \-do
\-combinatorial_object_activity
\-canonical_form
\-classification_prefix LSQ6
\-label LSQ6
\-save_ago
\-save_transversal
\-end
\-prefix LSQ6
\-export_flag_orbits
\-show_incidence_matrices
\-export_group_GAP
\-end

pdflatex LSQ6_classification.tex
#open LSQ6_classification.pdf
$(ORBITER) -v 2 -draw_matrix \
#open AG_2.3_classification.pdf
$(ORBITER) -v 2 -draw_matrix
-input_csv_file LSQ6_object0_TDA_flag_orbits.csv \
-secondary_input_csv_file LSQ6_object0_TDA_flag_orbits.csv \
-box_width 32 -bit_depth 24 \
-end
$(ORBITER) -v 2 -draw_matrix \
-input_csv_file LSQ6_object0_INP_flag_orbits.csv \
-secondary_input_csv_file LSQ6_object0_INP_flag_orbits.csv \
-box_width 32 -bit_depth 24 \
-end
open LSQ6_object0_INP_flag_orbits_draw.bmp

# ToDo:

quartic_curve_25_0_0_canonical:
$(ORBITER) -v 3 \
-define F -finite_field -q 25 -end \
-define P -projective_space -n 2 -field F -v 0 -end \
-with P -do \
-projective_space_activity \
-canonical_form_PG \
-input \
-set_of_points "10,11,59,63,124,135,136,170,206,257,275,284,285,367,378,393,433,619,641,644" \
-set_of_points "2,12,48,65,87,120,189,246,305,323,354,375,434,435,455,482,496,557,586,595" \
-end \
-classification_prefix quartic_25_0_0 \
-report \
-end \
-end
pdflatex quartic_25_0_0_classification.tex
SECTION OBJECTS FROM DESIGN THEORY:

# Section 15.4: Objects from Design Theory

17549 \(\rightarrow\) open quartic\_25\_0\_0\_classification.pdf
17550
17551
17552
17553
17554 geo\_16.c:
17555 \(\rightarrow\) $(\text{ORBITER}) -v 10 \;
17556 \(\rightarrow\) \(\triangleright\) -draw\_incidence\_structure\_description \;
17557 \(\rightarrow\) \(\triangleright\) \(\triangleright\) -width 60 -with 10 6 -end \;
17558 \(\rightarrow\) \(\triangleright\) -define C -combinatorial\_objects \;
17559 \(\rightarrow\) \(\triangleright\) \(\triangleright\) -file\_of\_incidence\_geometries geo\_16.inc 16 20 80 \;
17560 \(\rightarrow\) \(\triangleright\) -end \;
17561 \(\rightarrow\) \(\triangleright\) -with C -do \;
17562 \(\rightarrow\) \(\triangleright\) -combinatorial\_object\_activity \;
17563 \(\rightarrow\) \(\triangleright\) \(\triangleright\) -canonical\_form \;
17564 \(\rightarrow\) \(\triangleright\) \(\triangleright\) \(\triangleright\) -classification\_prefix 16 \;
17565 \(\rightarrow\) \(\triangleright\) \(\triangleright\) \(\triangleright\) -label 16 \;
17566 \(\rightarrow\) \(\triangleright\) \(\triangleright\) \(\triangleright\) -save\_ago \;
17567 \(\rightarrow\) \(\triangleright\) \(\triangleright\) \(\triangleright\) -save\_transversal \;
17568 \(\rightarrow\) \(\triangleright\) \(\triangleright\) -end \;
17569 \(\rightarrow\) \(\triangleright\) -report \;
17570 \(\rightarrow\) \(\triangleright\) \(\triangleright\) \(\triangleright\) -prefix 16 \;
17571 \(\rightarrow\) \(\triangleright\) \(\triangleright\) \(\triangleright\) -export\_flag\_orbits \;
17572 \(\rightarrow\) \(\triangleright\) \(\triangleright\) \(\triangleright\) -show\_incidence\_matrices \;
17573 \(\rightarrow\) \(\triangleright\) \(\triangleright\) \(\triangleright\) -export\_group\_GAP \;
17574 \(\rightarrow\) \(\triangleright\) \(\triangleright\) -end \;
17575 \(\rightarrow\) -end
17576 \(\rightarrow\) pdflatex 16\_classification.tex
17577 \(\rightarrow\) open 16\_classification.pdf
17578 \(\rightarrow\) $(\text{ORBITER}) -v 2 -draw\_matrix \;
17579 \(\rightarrow\) \(\triangleright\) -input\_csv\_file 16\_object0\_TDA\_flag\_orbits.csv \;
17580 \(\rightarrow\) \(\triangleright\) -secondary\_input\_csv\_file 16\_object0\_TDA.csv \;
17581 \(\rightarrow\) \(\triangleright\) -box\_width 16 -bit\_depth 24 \;
17582 \(\rightarrow\) -end
17583 \(\rightarrow\) $(\text{ORBITER}) -v 2 -draw\_matrix \;
17584 \(\rightarrow\) \(\triangleright\) -input\_csv\_file 16\_object0\_INP\_flag\_orbits.csv \;
17585 \(\rightarrow\) \(\triangleright\) -secondary\_input\_csv\_file 16\_object0\_INP.csv \;
17586 \(\rightarrow\) \(\triangleright\) -box\_width 16 -bit\_depth 24 \;
17587 \(\rightarrow\) -end
17588
17589
17590 # Section 15.4: Objects from Design Theory
17591
17592
17593
17594 SECTION\_OBJECTS\_FROM\_DESIGN\_THEORY:
17595

889
test.15.4:

make LS_AG_2_3_solutions_classify
make design_27c
make design_PG_2_3.canonical
make wreath_product_designs_n8_k6_c
make wreath_product_designs_n4_k2_c

LS_AG_2_3_solutions_classify:

$(ORBITER) -v 2 \n
-draw_incidence_structure_description \n-width 20 -width.10 2 -end \n-define C -combinatorial_objects \n-file_of_designs \n-solutions.csv 9 84 3 12 \n-end \n-with C -do \n-combinatorial_object_activity \n-canoncal_form \n-save_ago \n-save_transversal \n-classification_prefix LS_AG_2_3 \n-label LS_AG_2_3 \n-max_TDO_depth 10 \n-end \n-report \n-prefix LS_AG_2_3 \n-export_flag_orbits \n-show_TDO \n-end \n-end

pdflatex LS_AG_2_3_classification.tex

open LS_AG_2_3_classification.pdf

$(ORBITER) -v 2 -draw_matrix \n-input_csv_file LS_AG_2_3.object0_INP_flag_orbits.csv \n-secondary_input_csv_file LS_AG_2_3.object0_INP.csv \n-box_width 12 -bit_depth 24 \n-end

open LS_AG_2_3.object0_INP_flag_orbits.draw.bmp

$(ORBITER) -v 2 -draw_matrix \n-input_csv_file LS_AG_2_3.object1_INP_flag_orbits.csv \n-secondary_input_csv_file LS_AG_2_3.object1_INP.csv \n-box_width 12 -bit_depth 24 \n-end

open LS_AG_2_3.object1_INP_flag_orbits.draw.bmp
design_27c:

```latex
$(ORBITER) -v 4 \ 
$define C -combinatorial_objects \ 
$set_of_points "2,56,30,112,253,90,440,508" \ 
$end \ 
$define F -finite_field -q 27 -override_polynomial 46 -end \ 
$define P -projective_space -n 2 -field F -v 0 -end \ 
$with C -do \ 
$combinatorial_object_activity \ 
$canonical_form_PG P \ 
$classification_prefix design \ 
$end \ 
$report \ 
$end
```

```bash
pdflatex design_classification.tex
open design_classification.pdf
```

design_PG_2_3_canonical:

```bash
$(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
```

```bash
$(ORBITER) -v 3 \ 
$draw_incidence_structure_description \ 
$width 60 -with_10 6 -end \ 
$define C -combinatorial_objects \ 
$file_of_incidence_geometries PG_2_3_inc.txt 13 13 52 \ 
$end \ 
$with C -do \ 
$combinatorial_object_activity \ 
$canonical_form \ 
$classification_prefix PG_2,3 \ 
$label PG_2,3 \ 
$save_ago \ 
$save_transversal \ 
$end \ 
```
wreath_product_designs_n4_k2.c: wreath_product_designs_n4_k2_inc.txt

$(ORBITER) -v 2 -draw_matrix

$ORBITER -v 10

wreath_product_designs_n8_k6_c: wreath_product_designs_n8_k6_inc.txt
Section 15.5: Linear Codes

SECTION_CANONICAL_FORMS_OF_LINEAR_CODES:

test_15_5:

make code_3_2.aut
make code_6_3.aut
make RM_3_1.group
make RM_3_1.group_and_diagram
make RM_4_1.group
make RS_6_4_7.group
make GV_n15_k6_d5_group
make code_n15_k6_d6_a_group
make code_n15_k6_d6_b_group
code_3.2.aut:
17788 \> $(ORBITER) -v 20 \ 
17789 \> \> -define F -finite_field -q 2 -end \ 
17790 \> \> -define gemma -vector -field F -format 2 \ 
17791 \> \> \> -dense $(CODE_N3_K2_Q2_GENMA) \ 
17792 \> \> -end \ 
17793 \> \> -define P -projective_space -n 1 -field F -v 0 -end \ 
17794 \> \> \> -with P -do \ 
17795 \> \> \> -projective_space_activity \ 
17796 \> \> \> \> -canonical_form_of_code \ 
17797 \> \> \> \> \> "3.2" genma -save_ago -label "3.2" \ 
17798 \> \> \> \> \> \> -classification_prefix "3.2" \ 
17799 \> \> \> \> \> \> \> -end \ 
17800 \> \> \> -end \ 
17801 \> pdflatex 3.2.classification.tex \ 
17802 \> open 3.2.classification.pdf \ 
17803 \> $(ORBITER) -v 2 -draw_matrix \ 
17804 \> \> -input_csv_file 3.2.object0_TDA_flag_orbits.csv \ 
17805 \> \> -secondary_input_csv_file 3.2.object0_TDA.csv \ 
17806 \> \> \> -box_width 16 -bit_depth 24 \ 
17807 \> \> \> -end \ 
17808 \> open 3.2.object0_TDA_flag_orbits_draw.bmp \ 
17809 \ 
17810 \ 
17811 \ 
17812 \ 
code_6.3.aut:
17813 \> $(ORBITER) -v 20 \ 
17814 \> \> -define F -finite_field -q 2 -end \ 
17815 \> \> -define gemma -vector -field F -format 3 \ 
17816 \> \> \> -compact $(CODE_N6_K3_Q2_GENMA) \ 
17817 \> \> \> -end \ 
17818 \> \> \> -define P -projective_space -n 2 -field F -v 0 -end \ 
17819 \> \> \> \> -with P -do \ 
17820 \> \> \> \> \> \> -projective_space_activity \ 
17821 \> \> \> \> \> \> \> -canonical_form_of_code \ 
17822 \> \> \> \> \> \> \> \> "6.3" genma -save_ago -label "6.3" \ 
17823 \> \> \> \> \> \> \> \> \> -classification_prefix "6.3" \ 
17824 \> \> \> \> \> \> \> \> \> \> -end \ 
17825 \> \> \> \> \> \> \> \> \> \> -end \ 
17826 \> \> \> \> \> \> \> -input_csv_file 6.3.object0_TDA_flag_orbits.csv \ 
17827 \> pdflatex 6.3.classification.tex \ 
17828 \> open 6.3.classification.pdf \ 
17829 \> $(ORBITER) -v 2 -draw_matrix \ 
17830 \> \> -input_csv_file 6.3.object0_TDA_flag_orbits.csv \ 
17831 \ 
894
RM_3.1_group:
\$\text{(ORBITER)} -v 2 \\
\$ -define F -finite_field -q 2 -end \\
\$ -define genma -vector -field F -format 4 \\
\$ -compact $(\text{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 \\
\$ -classification_prefix "RM_3.1" \\
\$ -classification_prefix "RM_3.1" -end \\
\$ -end \\
\$ pdflatex RM_3.1_classification.tex \\
\$ open RM_3.1_classification.pdf \\
\$ # group of order 24 \\
\$ #RM_3.1_group_and_diagram:
\$ $(\text{ORBITER)} -v 2 \\
\$ -define F -finite_field -q 2 -end \\
\$ -define genma -vector -field F -format 4 \\
\$ -compact $(\text{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 \\
\$ -classification_prefix "RM_3.1" \\
\$ -classification_prefix "RM_3.1" -end \\
\$ -end \\
\$ pdflatex RM_3.1_classification.tex \\
\$ open RM_3.1_classification.pdf \\
\$ # group order 1344 \\
\$ #RM_3.1_object0_INP_flag_orbits.csv
RM_4.1_group:
$(ORBITER) -v 2
> -define F -finite_field -q 2
> -define genma -vector -field F -format 5
> -compact $(CODE_RM_4.1_GENMA)
> -define P -projective_space -n 4 -field F -v 0
> -with P -do
> -projective_space_activity
> -canonical_form_of_code
> "RM_4.1" genma -save Ago -label "RM_4.1"
> -classification_prefix "RM_4.1"
> -end
> -end
pdflatex RM_4.1_classification.tex
open RM_4.1_classification.pdf
$(ORBITER) -v 2
> -input_csv_file RM_4.1_object0_INP_flag_orbits.csv
> -secondary_input_csv_file RM_4.1_object0_TDA.csv
> -box_width 16 -bit_depth 24
> -end
> open RM_4.1_object0_INP_flag_orbits_draw.bmp
> open RM_4.1_object0_TDA_flag_orbits_draw.bmp

RM_6.4.7_group:
$(ORBITER) -v 20

# group order 322560 = 24*30*28*16
-define F -finite_field -q 7 -end \\
-define genma -vector -field F -format 4 \\
-compact $(CODE_RS_6.4.7) \\
-define P -projective_space -n 3 -field F -v 0 -end \\
-with P -do \\
-projective_space_activity \\
-cannotical_form_of_code \\
"RS_6" genma -save_ago -label "RS_6" \\
-classification_prefix "RS_6" \\
-end \\
-end 

GV_n15_k6_d5_group: 
$(ORBITER) -v 20 \\
-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 \\
-cannotical_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 

#ago=12 

code_n15_k6_d6_a_group: 
$(ORBITER) -v 20 \\
-define F -finite_field -q 2 -end \\
-define genma -vector -field F -format 6 \\
-compact $(CODE_15_6.6_A) \\
-end \\
-define P -projective_space -n 5 -field F -v 0 -end \\
-with P -do \\
-projective_space_activity \\
-cannotical_form_of_code \\
"n15_k6_d6_a" genma -save_ago -label "n15_k6_d6_a" \\
-classification_prefix "n15_k6_d6_a" \\

897
\begin{verbatim}
17972 ▶ ▶ ▶ -end \ 
17973 ▶ -end
17974 ▶ pdflatex n15_k6_d6_a_classification.tex
17975 ▶ open n15_k6_d6_a_classification.pdf
17976
17977
17978 code_n15_k6_d6_b_group:
17979 ▶ $(ORBITER) -v 20 \ 
17980 ▶ ▶ -define F -finite_field -q 2 -end \ 
17981 ▶ ▶ -define genma -vector -field F -format 6 \ 
17982 ▶ ▶ ▶ -compact $(CODE_15_6_6_B) \ 
17983 ▶ ▶ -end \ 
17984 ▶ ▶ -define P -projective_space -n 5 -field F -v 0 -end \ 
17985 ▶ ▶ -with P -do \ 
17986 ▶ ▶ -projective_space_activity \ 
17987 ▶ ▶ ▶ -canonical_form_of_code \ 
17988 ▶ ▶ ▶ ▶ "n15_k6_d6_b" genma -save_ago -label "n15_k6_d6_b" \ 
17989 ▶ ▶ ▶ ▶ -classification_prefix "n15_k6_d6_b" \ 
17990 ▶ ▶ ▶ ▶ -end \ 
17991 ▶ ▶ -end \ 
17992 ▶ pdflatex n15_k6_d6_b_classification.tex
17993 ▶ open n15_k6_d6_b_classification.pdf
17994
17995
17996
17997
17998 **************************************************************************
17999 # Section 15.6: General Codes
18000
18001
18002 SECTION_CANONICAL_FORMS_OF_GENERAL_CODES:
18003
18004 test_15_6:
18005 ▶ make Hamming_graph_7_with_Hamming_code
18006
18007
18008
18009 Hamming_graph_7_with_Hamming_code:
18010 ▶ $(ORBITER) -v 2 \ 
18011 ▶ ▶ -define G -graph -Hamming 7 2 \ 
18012 ▶ ▶ ▶ -subset "\Hamming_code" "\with\Hamming\code" \ 
18013 ▶ ▶ ▶ ▶ $(HAMMING_CODE_CODEWORDS) -end \ 
18014 ▶ ▶ -with G -do \ 
18015 ▶ ▶ -graph_theoretic_activity -export_csv -end \ 
18016 ▶ ▶ -with G -do \ 
18017 ▶ ▶ -graph_theoretic_activity -export_graphviz -end \ 
18018 ▶ ▶ -with G -do \ 
\end{verbatim}
Test 15.7: Graphs

SECTION_CANONICAL_FORMS_OF_GRAPHS:

1) Cycle 13 aut
2) Inversion graph
3) Chain 232 aut
4) JK_graph_pp16_1
5) JK_graph_pp16_2
6) JK_graph_pp16_9
7) JK_graph_grid_3_3
8) JK_graph sts_13
9) Halljanko315 gens.csv
10) HJ_group and orbits
11) HJ_orbital_graph_3
12) PGO_5_2_graph_group

Cycle 13 aut:

1) \$\text{ORBITER} -v 2 \$
2) \text{define Gamma -graph -cycle 13 -end}
3) \text{with Gamma -do}
4) \text{-graph_theoretic_activity -automorphism_group}
5) \text{-end}

Inversion graph:

1) \$\text{ORBITER} -v 6 \$
2) \text{-define G -graph}
3) \text{-inversion_graph "1,0,2,3"}
4) \text{-end}
5) \text{-with G -do}
\begin{verbatim}
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
\$\$

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 \$
\$\$
\$\$ # go=34217164800
\$\$
\$\$ #nauty_interface_with_group::create_automorphism_group_of_graph_with_partition_and_labeling: nb_backtrack1 = 6
\$\$
\$\$ #nauty_interface_with_group::create_automorphism_group_of_graph_with_partition_and_labeling: nb_backtrack2 = 134
\$\$
\$\$
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 \n
# does not finish

**JK_graph_pp16_9:**

```bash
$(ORBITER) -v 2 \n(define Gamma -graph -load_dimacs \n../JUNTTILA_KASKI/benchmarks/pp/pp16-9 \n-end \n-with Gamma -do \n-graph_theoretic_activity -save -end \n-with Gamma -do \n-graph_theoretic_activity -automorphism_group -end \n```n

**JK_graph_grid_3_3:**

```bash
$(ORBITER) -v 2 \n(define Gamma -graph -load_dimacs \n../JUNTTILA_KASKI/benchmarks/grid/grid-w-3-3 \n-end \n-with Gamma -do \n-graph_theoretic_activity -save -end \n-with Gamma -do \n-graph_theoretic_activity -automorphism_group -end \n```n

```bash
#nauty_interface_with_group::create_automorphism_group_of_graph_with_partition_an
d_labeling: nb_backtrack1 = 4
#nauty_interface_with_group::create_automorphism_group_of_graph_with_partition_an
d_labeling: nb_backtrack2 = 9
#Written file grid-w-3-3_group.makefile of size 579
#User time: 0 of a second, dt=0 tps = 100
```

**JK_graph_sts_13:**

```bash
$(ORBITER) -v 2 \n(define Gamma -graph -load_dimacs \n../JUNTTILA_KASKI/benchmarks/srg/sts-13 \n-end \n-with Gamma -do \n-graph_theoretic_activity -save -end \n-with Gamma -do \n-graph_theoretic_activity -automorphism_group -end
make ORBITER_PATH=$(ORBITER_PATH) -f sts-13_group.makefile sts-13
```n

```bash
make ORBITER_PATH=$(ORBITER_PATH) -f sts-13_group.makefile sts-13
```n

901
#nauty_interface_with_group::create_automorphism_group_of_graph_with_partition_and_labeling: nb_backtrack1 = 3

#nauty_interface_with_group::create_automorphism_group_of_graph_with_partition_and_labeling: nb_backtrack2 = 24

halljanko315_gens.csv:
$ (ORBITER) -v 6 \n $define G -graph \n $load_csv_no_border \n $halljanko315.csv \n $end \n $with G -do \n $graph_theoretic_activity -automorphism.group \n $end \n $with G -do \n $graph_theoretic_activity -properties \n $end

HJ group and orbits:
$ (ORBITER) -v 2 \n $define Control -poset_classification.control \n $W \n $problem_label HJ_orbits \n $depth 2 \n $end \n $define gens -vector -file \n $halljanko315_gens.csv -end \n $define G -permutation_group \n $bsgs halljanko315 "File\_halljanko315" \n $315 1209600 "0,1,2" 6 gens \n $end \n $define Orb -orbits -group G \n $on_subsets 2 Control \n $end

#ROW,REP,AGO,OL
#0,"0,1",96,12600
#1,"0,2",48,25200
#2,"0,4",768,1575
#3,"0,8",120,10080
#END

HJ_orbital_graph_3:
$ (ORBITER) -v 2 \n
902
define gens -vector -file \halljanko315_gens.csv -end \
define G -permutation_group \bgs halljanko315 "File\halljanko315" 
315 1209600 "0,1,2" 6 gens 
end 
define Gamma -graph 
end 
with Gamma -do 
\graph_theoretic_activity 
\properties 
end 
with Gamma -do 
\graph_theoretic_activity 
save 
end 
\aut
\graph 15 lines 
\_lines 
\graph 0_5_2_graph_group: 0_5_2_incidence_matrix.csv 
\define Inc -vector -file 0_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 
\pdf \collinearity_graph_eigenvalues.tex 
\open \collinearity_graph_eigenvalues.pdf 
\aut
\graph of 15 lines: 
\$(\text{ORBITER}) -v 2 
\define Gamma -graph 
\load_csv_no_border 
\15lines.csv
# Section 15.8: Quartic Curves

SECTION_CANONICAL_FORMS_OF_QUARTIC_CURVES:

test_15_8:

\[ (\text{ORBITER}) -v 3 \]

\[ \text{define F -finite_field -q 17 -end} \]

\[ \text{define C -combinatorial_objects} \]

\[ \text{define P -projective_space -n 2 -field F -v 0 -end} \]
-combinatorial_object_activity \n-canoncial_form_PG_P \n-classification_prefix_Edge_curve_q17 \n-label_Edge_curve_q17 \n-save_ago \n-save_transversal \n-max_TDO_depth_10 \n-end \n-report \n-prefix_Edge_curve_q17 \n-export_flag_orbits \n-show_TDO \n-show_TDA \n-dont_show_incidence_matrices \n-export_group_GAP \n-end \n-pdflatex_Edge_curve_q17_classification.tex \n-open_Edge_curve_q17_classification.pdf \n-$(ORBITER) -v 2 -draw_matrix \n-input_csv_file_Edge_curve_q17_object0_TDA_flag_orbits.csv \n-secondary_input_csv_file_Edge_curve_q17_object0_TDA.csv \n-box_width 4 -bit_depth 24 \n-end \n-open_Edge_curve_q17_object0_TDA_flag_orbits_draw.bmp \n-# 9 backtrack nodes total \n # aut = 24 \n # User time: 0.04 of a second, dt=4 tps = 100 \n # generators for a group of order 24: \n #1,0,0,0,13,0,0,0,4, **\n #1,0,0,0,0,16,0,16,0 **\n #0,1,16,2,4,4,15,4,4 **\n # generators for a group of order 24: \n 1,0,0,0,13,0,0,0,4 \n 1,0,0,0,0,16,0,16,0 \n 0,1,16,2,4,4,15,4,4 **\n -end \n
# Chapter 16 - Interfaces

## Section 16.1: Graphical Output

```bash
test_16_1:
make F_7.tables
make PG_2_4_cyclic_incma
make PGL_4_2_Wedge_4_0_graphical_output
make schreier_tree_graphical_output
make Queens_graph

F_7.tables:
$(ORBITER) -v 3 \
define F -finite_field -q 7 -end \
with F -do -finite_field_activity \
do -cheat_sheet_GF \
end

$(ORBITER) -v 2 \
draw_matrix \
\cdashline{3-4} \ninput_csv_file GF_q7_addition_table.csv \
\boxwidth 40 \
\bitdepth 24 \
\partition 3 7 7 \
end
open GF_q7_addition_table_draw.bmp
```

open PGL_3_17_Subgroup_Stab_Edge_24_report.pdf
PG_2.4_cyclic_incma:
$(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_for_decomposition_by_element_PG) 
1 "0,1,0, 0,0,1, 2,1,1, 0" "PG_2.4_singer" 
(end)
$ORBITER -v 4 \
(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
1010
1211
1312
1413
PGL_4.2_Wedge_4.0_graphical_output:
$ORBITER -v 4 \
(define G -linear_group -PGL 4 2) 
(wedge_detached) 
(end)
(with G -do) 
(group_theoretic_activity) 
(report) 
(end)
pdflatex PGL_4.2_Wedge_4.2_detached_report.tex
open PGL_4.2_Wedge_4.2_detached_report.pdf
2525
2626
schreier_tree_graphical_output:
$ORBITER -v 4 \
(draw_options) 
(yout 500000) 
(radius 15 -nodes_empty) 
(line_width 0.5 -y_stretch 0.25) 
(line_width) 
(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 \ pdflatex poly_orbits_d3_n3_q2_orbit_6_tree.tex \ open poly_orbits_d3_n3_q2_orbit_6_tree.pdf \ #pdflatex poly_orbits_d3_n3_q2.tex \ #open poly_orbits_d3_n3_q2.pdf

Queens_graph:
-define G -graph -non_attacking_queens_graph 8 -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 \ -with G -do \ -graph_theoretic_activity -find_cliques \ -target_size 8 -output_file 8queens -end \ -end

# Section 16.2: The Povray Interface

SECTION_POVRAY:

test_16_2:
make cube
make math261_test
make plane1
make plane2
make analytic_geo_1
18483 ▶ make analytic_geo_1_video
18484 ▶ make monkey
18485 ▶ make Eckardt
18486 ▶ make Eckardt_deform
18487 ▶ make Eckardt_deform_2
18488 ▶ make Clebsch
18489 ▶ make endrass8
18490
18491
18492  cube:
18493 ▶ $(ORBITER) -v 2 -povray \n18494 ▶  -round 0 -nb_frames_default 30 \n18495 ▶  -output_mask cube_%.d_%.03d.pov \n18496 ▶  -video_options -W 1024 -H 768 \n18497 ▶  -global_picture_scale 0.5 \n18498 ▶  -default_angle 75 \n18499 ▶  -clipping_radius 2.7 \n18500 ▶  -end \n18501 ▶  -scene_objects \n18502 ▶  ▶  -obj_file cube_centered.obj \n18503 ▶  ▶  ▶  -edge "0, 1" \n18504 ▶  ▶  ▶  -edge "0, 2" \n18505 ▶  ▶  ▶  -edge "0, 4" \n18506 ▶  ▶  ▶  -edge "1, 3" \n18507 ▶  ▶  ▶  -edge "1, 5" \n18508 ▶  ▶  ▶  -edge "2, 3" \n18509 ▶  ▶  ▶  -edge "2, 6" \n18510 ▶  ▶  ▶  -edge "3, 7" \n18511 ▶  ▶  ▶  -edge "4, 5" \n18512 ▶  ▶  ▶  -edge "4, 6" \n18513 ▶  ▶  ▶  -edge "5, 7" \n18514 ▶  ▶  ▶  -edge "6, 7" \n18515 ▶  ▶  ▶  -group_of_things_as_interval 0 8 \n18516 ▶  ▶  ▶  -spheres 0 0.3 $(POLISHED_CHROME_WHITE) \n18517 ▶  ▶  ▶  -group_of_things_as_interval 0 6 \n18518 ▶  ▶  ▶  -prisms 1 0.05 $(YELLOW_TRANSPARENT) \n18519 ▶  ▶  ▶  -group_of_things_as_interval 0 12 \n18520 ▶  ▶  ▶  -cylinders 2 0.15 $(COLOR_RED) \n18521 ▶  ▶  -scene_objects_end \n18522 ▶  -povray_end
18523 ▶  - rm -rf POV
18524 ▶  mkdir POV
18525 ▶  mv cube_0*.pov POV
18526 ▶  mv makefile_animation POV
18527
18528
18529  math261_test:
$(ORBITER) -v 2 -povray \\
-round 0 -nb_frames_default 30 \\
-output_mask math261.%d.%03d.pov \\
-video_options -W 1024 -H 768 \\
-global_picture_scale 0.1 \\
-default_angle 75 \\
-clipping_radius 2.7 \\
-end \\
-scene_objects \\
-point "0,0,0" \\
-point "5,0,0" \\
-point "0,5,0" \\
-point "0,0,5" \\
-point "1,2,3" \\
-point "4,5,6" \\
-point "5,7,9" \\
-edge "0,1" \\
-edge "0,2" \\
-edge "0,3" \\
-edge "0,4" \\
-edge "0,5" \\
-edge "4,6" \\
-edge "5,6" \\
-face "0,4,6,5" \\
-group_of_things_as_interval 0 7 \\
-spheres 0 0.1 $(POLISHED_CHROME_WHITE) \\
-group_of_things_as_interval 0 7 \\
cylinders 1 0.05 $(COLOR_RED) \\
-prisms 2 0.05 $(YELLOW_TRANSPARENT) \\
-group_of_things_as_interval 0 1 \\
-scene_objects_end \\
-povray_end \\
- rm -rf POV \\
mkdir POV \\
mv math261.0_* .pov POV \\
mv makefile_animation POV \\
plane1: \\
$(ORBITER) -v 2 -povray \\
-round 0 -nb_frames_default 30 \\
-output_mask plane1.%d.%03d.pov \\
-video_options -W 1024 -H 768 \\
-global_picture_scale 0.40 \\
-default_angle 75 \\
-clipping_radius 5 -omit_bottom_plane \\

-camera 0 "0,0,1" "5,5,3" "0,0,0"
-rotate about z axis
-boundary box
-end
-scene_objects
-line through two_points_recentered_from_csv_file coordinate_grid.csv
-plane_by_dual_coordinates "0,0,1,0"
-plane_by_dual_coordinates "0,1,0,0"
-plane_by_dual_coordinates "1,0,0,0"
-point "-2.25,0,0"
-point "0,-1.8,0"
-point "0,0,9"
-face "0,1,2,0"
-group_of_things "0"
-group_of_things "1"
-group_of_things "2"
-lines 0 0.15 $(COLOR_RED_SHINY)
-lines 1 0.15 $(COLOR_GREEN_SHINY)
-lines 2 0.15 $(COLOR_BLUE_SHINY)
-group_of_things_as_interval 3 39
-lines 3 0.05 $(COLOR_BLACK_SHINY)
-group_of_things "0"
-lines 0 $(COLOR_BLUESEE_THROUGH)
-group_of_things "1"
-group_of_things "2"
-group_of_things "0"
-prisms 0 0.05 $(COLOR_YELLOW_THICK)
-scene_objects_end
-povray_end
- rm -rf POV
-mkdir POV
-mv plane1_0_*.pov POV
-mv makefile_animation POV

plane2:
$(ORBITER) -v 2 -povray
-round 0 -nb_frames_default 30
-output_mask plane2_%d_%03d.pov
-video_options -W 2560 -H 1920
-global_picture_scale 0.40
-default_angle 75
-clipping_radius 5 -omit_bottom_plane
-camera 0 "0,0,1" "6,6,2" "0,0,0"
-rotate about z_axis
-boundary_box
18624 -end 
18625 -scene_objects 
18626 -line_through_two_points_recentered_from_csv_file coordinate_grid.csv 
18627 -plane_by_dual_coordinates "0,0,1,0" 
18628 -plane_by_dual_coordinates "0,1,0,0" 
18629 -plane_by_dual_coordinates "1,0,0,0" 
18630 -plane_by_dual_coordinates "4,5,-1,9" 
18631 -group_of_things "0" 
18632 -group_of_things "1" 
18633 -group_of_things "2" 
18634 -group_of_things_as_interval 3 39 
18635 -lines 0 0.15 $(COLOR_RED_SHINY) 
18636 -lines 1 0.15 $(COLOR_GREEN_SHINY) 
18637 -lines 2 0.15 $(COLOR_BLUE_SHINY) 
18638 -lines 3 0.05 $(COLOR_BLACK_SHINY) 
18639 -group_of_things "0" 
18640 -planes 4 $(COLOR_BLUESEE THROUGH) 
18641 -group_of_things "3" 
18642 -scene_objects_end 
18643 -povray_end 
18644 -rm -rf POV 
18645 mkdir POV 
18646 mv plane2_0*.pov POV 
18647 mv makefile_animation POV 
18648
18649 # analytic_geo_1: 
18650 analytic_geo_1: 
18651 analytic_geo_1: 
18652 analytic_geo_1: 
18653 analytic_geo_1: 
18654 $(ORBITER) -v 2 -povray 
18655 -round 0 -nb_frames_default 30 
18656 -output_mask analytic_geo_1_%d_%03d.pov 
18657 -video_options -W 2560 -H 1920 
18658 -global_picture_scale 0.80 
18659 -default_angle 75 
18660 -clipping_radius 5 -omit_bottom_plane 
18661 -camera 0 "0,0,1" "6,6,2" "0,0,0" 
18662 -rotate_about_z_axis 
18663 -boundary_box 
18664 -end 
18665 -scene_objects 
18666 -line_through_two_points_recentered_from_csv_file coordinate_grid.csv 
18667 -plane_by_dual_coordinates "0,0,1,0" 
18668 -plane_by_dual_coordinates "0,1,0,0" 
18669 -plane_by_dual_coordinates "1,0,0,0" 
18670
-group_of_things "0"
-group_of_things "1"
-group_of_things "2"
-group_of_things as_interval 3 39
-lines 0 0.05 $(COLOR_RED_SHINY)
-lines 1 0.05 $(COLOR_GREEN_SHINY)
-lines 2 0.05 $(COLOR_BLUE_SHINY)
-lines 3 0.04 $(COLOR_BLACK_SHINY)
-group_of_things "0"
-group_of_things "1"
-group_of_things "2"
-lines 0 0.05 $(COLOR_RED_SHINY)
-lines 1 0.05 $(COLOR_GREEN_SHINY)
-lines 2 0.05 $(COLOR_BLUE_SHINY)
-lines 3 0.04 $(COLOR_BLACK_SHINY)
-group_of_things "0"
-group_of_things "1"
-group_of_things "2"
-point "0,0,0"
-point "1,0,0"
-point "1,2,0"
-point "1,2,3"
-edge "84,85"
-edge "85,86"
-edge "86,87"
-edge "84,87"
-group_of_things "84,85,86"
-spheres 7 0.1 $(POLISHED_CHROME_WHITE)
-group_of_things "87"
-spheres 8 0.10 $(COLOR_YELLOW_SHINY)
-group_of_things "0,1,2"
-cylinders 9 0.075 $(POLISHED_CHROME_WHITE)
-group_of_things "3"
-cylinders 10 0.075 $(COLOR_YELLOW_SHINY)
-scene_objects_end
-povray_end
-rm -rf POV
mkdir POV
mv analytic_geo.1.*.pov POV
mv makefile_animation POV
analytic_geo.1_video:
- rm -r FRAMES
mkdir FRAMES
mv analytic_geo.1.0.*.png FRAMES
$(ORBITER)
-prepare_frames
-i 0 30 PNG/ANALYTIC_GEO.1/analytic_geo.1.0.%03d.png
-output_starts_at 0
-o FRAMES/frame%04d.png
ffmpeg -r 5 -f image2 -i FRAMES/frame%04d.png -f mp4 -q:v 0 -vcodec mpeg4 analytic_geo_1.mp4

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

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
```plaintext
- 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) \n- lines 2 0.02 $(COLOR_BLUE_SHINY) \n- lines 3 0.02 $(COLOR_YELLOW_SHINY) \
- label 0 "a1" \n- label 2 "a2" \n- label 4 "a3" \n- label 6 "a4" \n- label 8 "a5" \n- label 10 "a6" \n- label 12 "b1" \n- label 14 "b2" \n- label 16 "b3" \n- label 18 "b4" \n- label 20 "b5" \n- label 22 "b6" \n- label 24 "c12" \n- label 26 "c13" \n- label 30 "c15" \n- label 32 "c16" \n- label 34 "c23" \n- label 36 "c24" \n- label 40 "c26" \n- label 42 "c34" \n- label 44 "c35" \n- label 48 "c45" \n- label 50 "c46" \n- label 52 "c56" \n- 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 \n- povray_end
```

```
#"-3,333,4" * 1.5 = "-4.5,3.5,6"
#M := Matrix([[[-4.5, 3.5, 6], [1, 1, 1]])
#NullSpace(M)
```
Eckardt_deform:

$\text{ORBITER} -v 2 -povray
-\text{round} 0 -\text{nb frames default} 93
-\text{output mask} \text{Eckardt_deform}._\%d._\%3d.pov
-\text{video options} -W 1024 -H 768
-\text{global picture scale} 0.9
-\text{default angle} 75
-\text{clipping radius} 2.4
-\text{camera} 0 "1,1,1" "-3,1,3" "0.12,0.12,0.12"
-\text{end}

-\text{Hilbert_Cohn_Vossen_surface}
-\text{group of things} "0"
-\text{deformation of cubic lex} 93 1.107148718 1.570796327 0
-\text{ECKARDT_CUBIC_DEFORM1_LEX}
-\text{ECKARDT_CUBIC_DEFORM2_LEX}
-\text{group of things as interval} 0 93
-\text{group is animated} 1
-\text{cubics} 1 $(\text{SURFACE_COLOR_SEETHROUGH})
-\text{scene objects end}
-\text{povray end}

-\text{rm} -rf \text{POV}
-\text{mkdir} \text{POV}
-\text{mv} \text{Eckardt_deform}_0_*\text{.pov} \text{POV}
-\text{mv} \text{makefile_animation} \text{POV}

Eckardt_deform.2:

$\text{ORBITER} -v 2 -povray
-\text{round} 0 -\text{nb frames default} 30
-\text{output mask} \text{Eckardt_deform}._\%d._\%3d.pov
-\text{video options} -W 1024 -H 768
-\text{global picture scale} 0.9
-\text{default angle} 75
-clipping_radius 2.4 \
camera 0 "1,1,1" "-3,1,3" "0.12,0.12,0.12" 
-end 

Hilbert_Cohn_Vossen_surface 
-group_of_things "0" 
deformation_of_cubic_lex 93 1.107148718 1.570796327 0 
$(ECKARDT_CUBIC_DEFORM1_LEX) 
$(ECKARDT_CUBIC_DEFORM2_LEX) 
group_of_things_as_interval 0 93 
group_is_animated 1 
group_of_things "0" 
-cubics 1 $(SURFACE_COLOR_SEETHROUGH) 
-group_of_things "24" 
cubics 2 $(COLOR_RED) 
group_of_things "70" 
cubics 3 $(COLOR_BLUE) 
-scene_objects_end 
povray_end
rm -rf POV
mkdir POV
mv Eckardt_deform_0_* .pov POV
mv makefile_animation POV

Clebsch:
$(ORBITER) -v 2 -povray 
-round 0 -nb_frames_default 30 
-output_mask Clebsch_%d_%03d.pov 
-video_options -W 1024 -H 768 
-global_picture_scale 0.9 
-default_angle 80 
-clipping_radius 2.4 
camera 0 "1,1,1" "-4.5,3.5,6" "0,0,0" 
-end 

Clebsch_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 12 15 
-lines 1 0.02 $(COLOR_RED_SHINY) 
-lines 2 0.02 $(COLOR_BLUE_SHINY) 
-lines 3 0.02 $(COLOR_YELLOW_SHINY) 
-group_of_things_as_interval 0 12 

917
-spheres 4 0.08 $(COLOR_TURQUOISE) \
-scene_objects_end \
-povray_end \
- rm -rf POV 
mkdir POV 
mv Clebsch_0_*.pov POV 
mv makefile_animation POV 

d rdrass8:
  $(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) 
  -octic_lex 0.15 $(COLOR_BLUE_SHINY) 
  -octic_lex 3 0.05 $(COLOR_BLACK_SHINY) 
  -octic_lex 165 $(ENDRASS_OCTIC_LEX_165) 
  -octic_lex 0.15 $(COLOR_BLUE_SHINY) 
  -octic_lex 3 0.05 $(COLOR_BLACK_SHINY) 
  -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
Section 16.3: Creating Animations

SECTION_ANIMATIONS:

test_16.3:

make dode
make dode_video
make monkey_video
make Eckardt_deform_video
make Eckardt_surface
make Kummer_surface
make Kummer_video
make Beauville_surface
make Clebsch_up_create_points
make Clebsch_surface
make Clebsch_surface_defining_equation
make Clebsch_surface_defining_equation_and_curves
make F7_povray
make F7_video
make McKean_povray
make McKean_video

dode:

$(ORBITER) -v 2 \
-povray \n-round 0 -nb_frames_default 30 \n-output_mask dode_%d_%03d.pov \n-video_options -W 1024 -H 768 \n-global_picture_scale 0.50 \n-default_angle 45 \n-clipping_radius 5 \ncamera 0 "1,1,1" "-2,2,4" "0,0,0" \n-rotate_about_111 \n-end \n-scene_objects \n-dodecahedron \n-group_of_things_as_interval 0 20 \n-spheres 0 0.075 $(POLISHED_CHROME_WHITE) \ngroup_of_things_as_interval 0 30\n
-cylinders 1 0.05 $(COLOR_RED_SHINY) \
-group_of_things_as_interval 0 12\n-prisms 2 0.02 $(YELLOW_TRANSPARENT) \
-scene_objects_end \
-povray_end

rm -rf POV
mkdir POV
mv dode_0.*.pov POV
mv makefile.animation POV

```
dode_video:
  - rm -r FRAMES
dir FRAMES
d - rm dode.mp4
  $(ORBITER) \
  -prepare_frames \
  -i 0 30 DODE/dode_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 dode.mp4

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

Eckardt_deform_video:
  - rm -r FRAMES
  - mkdir FRAMES
  - rm Eckardt_deform.mp4
  $(ORBITER) \
  -prepare_frames \
  -i 0 30 Eckardt_deform_0.%03d.png \
  -output_starts_at 0 \
  -o FRAMES/frame%04d.png \
```
Eckardt

surface:

$(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 \\
cubic_Goursat "6,3,-15" \\
group_of_things "0" \\
cubics 0 $(SURFACE_COLOR_SEETHROUGH) \\
-scene_objects_end \\
povray_end \\
rm -rf POV

Kummer

surface:

$(ORBITER) -v 2 -povray \\
-round 0 -nb_frames_default 30 \\
-output_mask Kummer_%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 \\
quartic_lex 35 $(KUMMER_QUARTIC_LEX_35) \\
group_of_things "0" \\
quartics 0 $(SURFACE_COLOR_SEETHROUGH) \\
-scene_objects_end \\
povray_end \\
rm -rf POV

mkdir POV

mv Kummer_0_*.pov POV
mv makefile_animation POV

# Maple:

#Kummer := expand((x0^2 + x1^2 + x2^2 + x3^2)^2 - 3*(x0^4 + x1^4 + x2^4 + x3^4))

Kummer

video:

- rm -r FRAMES
- mkdir FRAMES
- rm Kummer.mp4
- $(ORBITER) \\
  -prepare frames \\
- $i 0 30 Kummer_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 Kummer.mp4

Beauville

Beauville_surface:

$(ORBITER) -v 2 -povray \\
- round 0 -nb_frames_default 30 \\
- -output_mask Beauville%d%03d.pov \\
- -video_options -W 1024 -H 768 \\
- -global_picture_scale 0.3 \\
- -default_angle 75 \\
- -clipping_radius 2.4 \\
- -camera 0 "1,1,1" "-3,1,3" "0.12,0.12,0.12" \\
- -end \\
- -scene_objects \\
- -quintic_lex_56 $(BEAUVILLE_QUINTIC_LEX_56) \\
- -group_of_things "0" \\
- -quintics 0 $(SURFACE_COLOR_SEETHROUGH) \\
- -scene_objects_end \\
- -povray_end

- rm -rf POV
- mkdir POV
- mv Beauville_0.*.pov POV
- mv makefile_animation POV
Clebsch map up for surface created using arc lifting
We take a circle of radius \( r \) centered at the origin in the affine real plane and map it up on the surface.
The Clebsch surface has
\[ a = d = \frac{3+\sqrt{5}}{2} \]
\[ b = c = \frac{1+\sqrt{5}}{2} \]

```
CLEBSCH_A=2.618033988
CLEBSCH_D=2.618033988
CLEBSCH_B=1.618033988
CLEBSCH_C=1.618033988
TWO_PI=6.283185308
```

to go from the arclifting surface to the defining equation:
```
#Matrix(4, 4, [[-0.44721360215312733, 1.1708204000530853, 1.1708204000530853, -0.4472135957999158], [-1.1708204000530853, 0.4472136021531272, 1.4472136021531272, 0.4472135957999158], [4.2360680044124255, -4.2360680044124255, -4.2360680044124255, -4.2360680044124255], [1.6180340022062127, -2.6180340022062127, -1.6180340022062127, 0.]])
```

```
T00=-0.44721360215312733
T01=1.1708204000530853
T02=1.1708204000530853
T03=-0.4472135957999158
T10=-1.1708204000530853
T11=0.4472136021531272
T12=1.4472136021531272
T13=0.4472135957999158
```

923
19182
19183  T20=4.2360680044124255
19184
19185  T21=-4.2360680044124255
19186
19187  T22=-4.2360680044124255
19188
19189  T23=0.
19190
19191  T30=1.6180340022062127
19192
19193  T31=-2.6180340022062127
19194
19195  T32=-1.6180340022062127
19196
19197  T33=0.
19198
19199
19200  CLEBSCH_CUBICS=\n19201  push b push b mult push d push c push m mult add mult \n19202  push b push c push d push d push m mult mult add mult \n19203  push a push d push d push m add mult mult add add \n19204  push a push c push m mult add mult \n19205  store c001 \n19206  push b push d mult \n19207  push b push 1 push m push c mult add mult \n19208  push d push a push 1 push m mult add mult add \n19209  push m push a mult add push c add \n19210  push c push m push a mult add \n19211  mult mult \n19212  store c002 \n19213  push b \n19214  push d push c push a push m mult add mult \n19215  push c push a push m push 1 mult add mult add \n19216  push a push d mult push c push 1 push m mult add mult \n19217  push m mult add \n19218  push a push c push m mult add mult \n19219  store c011 \n19220  push b push b push c mult mult \n19221  push 1 push d push m mult add mult \n19222  push a push b mult push c push d push d push m mult mult add mult \n19223  push m mult add \n19224  push a push d mult push c push d push m mult add mult add \n19225  push a push c push m mult add mult \n19226  store c012 \n19227  push m \n19228  push b push d push m mult add push c mult \n
924
push d push b push 1 push m mult add mult push m mult add push a mult \\
push b push c mult push d push 1 push m mult add mult add mult \\
push b push d push m mult add mult \\
store d001 \\
push m \\
push d push c push m mult add push a push a mult mult \\
push c c push m mult push d push m mult add push a mult add \\
push m push b push c mult mult push c push 1 push m mult add mult add mult \\
push b push d push m mult add mult add mult \\
store d011 \\
push m \\
push c push d mult push d push m mult add push a push a mult mult \\
push c c push m mult push d push m mult add push a push b push m mult mult add \\
push b push d push c push m mult add mult push c push 1 push m mult add mult add mult \\
push b push d push m mult add mult add mult \\
store d012 \\
push d push 1 push m mult add push a mult push m push b mult push 1 add push c mult add \\
push b add push m push d mult add \\
push a push c mult mult \\
push b push d push m mult add mult \\
store m112 \\
push m \\
push b push d push m mult add push c mult push d push b push 1 push m mult add mult add \\
push m mult add push a mult push b push c mult push d push 1 push m mult add mult add \\
push b push d push m mult add mult mult \\
store m002 \\
push m \\
push d push c push m mult add push a push a mult mult \\
push b push c mult push d push m mult add push a mult add \\
push b push c push m mult mult push c push 1 push m mult add mult add mult \\
push b push d push m mult add mult mult \\
store m012 \\
push m \\
push c push d mult push d push m mult add push a push a mult mult \\
push m push c push c mult push d push m mult add push a push b mult mult mult add \\
push m push b push d push c push m mult add push c mult mult mult add \\
push b push d push 1 push m mult add push a mult \\
push m push b mult push 1 add push c mult add \\
push b add push m push d mult add 

19270 ▷ ▷ ▷ ▷ push a push c mult mult \n19271 ▷ ▷ ▷ ▷ push b push d push m mult add mult \n19272 ▷ ▷ ▷ ▷ store m122 \n19273 ▷ ▷ ▷ ▷ push m push a mult push c add push d mult push c push a push 1 push m mult add mult add \n19274 ▷ ▷ ▷ ▷ push b mult \n19275 ▷ ▷ ▷ ▷ push m push a push d mult push c push 1 push m mult add mult add \n19276 ▷ ▷ ▷ ▷ push b push d push m mult add mult \n19277 ▷ ▷ ▷ ▷ store n002 \n19278 ▷ ▷ ▷ ▷ push m \n19279 ▷ ▷ ▷ ▷ push c push d push m mult add push b mult push m push d push c push 1 push m mult add mult add \n19280 ▷ ▷ ▷ ▷ push a mult \n19281 ▷ ▷ ▷ ▷ push b push c mult push d push 1 push m mult add mult add \n19282 ▷ ▷ ▷ ▷ push a push b push c push m mult push d push m mult add add add mult \n19283 ▷ ▷ ▷ ▷ store n012 \n19284 ▷ ▷ ▷ ▷ push c push d push m mult add push b mult \n19285 ▷ ▷ ▷ ▷ push m push d push c push 1 push m mult add mult add \n19286 ▷ ▷ ▷ ▷ push a mult \n19287 ▷ ▷ ▷ ▷ push b push c mult push d push 1 push m mult add mult add \n19288 ▷ ▷ ▷ ▷ push a push d mult push m push b push c mult mult add mult \n19289 ▷ ▷ ▷ ▷ store n022 \n19290 ▷ ▷ ▷ ▷ push m \n19291 ▷ ▷ ▷ ▷ push c push d push m mult add push b mult \n19292 ▷ ▷ ▷ ▷ push m push d push c push 1 push m mult add mult add \n19293 ▷ ▷ ▷ ▷ push a mult \n19294 ▷ ▷ ▷ ▷ push b push c mult push d push 1 push m mult add mult add \n19295 ▷ ▷ ▷ ▷ push m push a mult push c add mult mult \n19296 ▷ ▷ ▷ ▷ store n112 \n19297 ▷ ▷ ▷ ▷ push m \n19298 ▷ ▷ ▷ ▷ push c push d push m mult add push b mult \n19299 ▷ ▷ ▷ ▷ push m push d push c push 1 push m mult add mult add \n19300 ▷ ▷ ▷ ▷ push a mult \n19301 ▷ ▷ ▷ ▷ push b push c mult push d push 1 push m mult add mult add \n19302 ▷ ▷ ▷ ▷ push a push d mult push m push b push c mult mult add mult mult \n19303 ▷ ▷ ▷ ▷ store n122
19304
19305 Clebsch_up_create_points:
19306 ▷ $\text{(ORBITER)}$ -v 2 \n19307 ▷ ▷ -smooth_curve "Clebsch_map_of_circle_to_defining_eqn_r2" \n19308 ▷ ▷ ▷ ▷ 0.07 1000 5 0 $(\text{TWO}\_\pi) \n19309 ▷ ▷ ▷ ▷ -const a $(\text{CLEBSCH}_A) b $(\text{CLEBSCH}_B) c $(\text{CLEBSCH}_C) d $(\text{CLEBSCH}_D) \n19310 ▷ ▷ ▷ ▷ t00 $(T00) t01 $(T01) t02 $(T02) t03 $(T03) \n19311 ▷ ▷ ▷ ▷ t10 $(T10) t11 $(T11) t12 $(T12) t13 $(T13) \n19312 ▷ ▷ ▷ ▷ t20 $(T20) t21 $(T21) t22 $(T22) t23 $(T23) \n19313 ▷ ▷ ▷ ▷ t30 $(T30) t31 $(T31) t32 $(T32) t33 $(T33) \n19314 ▷ ▷ ▷ ▷ r 2 one 1 m -1 \n
926
-const_end \
-var t \
c001 c002 c011 c012 \
d001 d011 d012 d112 \
m002 m012 m022 m122 \
n002 n012 n112 n022 n122 \
y0 y1 y2 \
y001 y002 y011 y012 y022 y112 y122 \
x0 x1 x2 x3 \
-var_end \
-code \
push t cos push r mult store y0 \
push t sin push r mult store y1 \
push one store y2 \
push y0 push y0 push y1 mult mult store y001 \
push y0 push y0 push y2 mult mult store y002 \
push y0 push y1 push y1 mult mult store y011 \
push y0 push y1 push y2 mult mult store y012 \
push y0 push y2 push y2 mult mult store y022 \
push y1 push y1 push y2 mult mult store y112 \
push y1 push y2 push y2 mult mult store y122 \
$(CLEBSCH_CUBICS) \ 
push c001 push y001 mult \
push c002 push y002 mult add \
push c011 push y011 mult add \
push c012 push y012 mult add \
store x0 \
push d001 push y001 mult \
push d011 push y011 mult add \
push d012 push y012 mult add \
push d112 push y112 mult add 
store x1 \
push m002 push y002 mult \
push m012 push y012 mult add \
push m022 push y022 mult add \
push m122 push y122 mult add 
store x2 \
push n002 push y002 mult \
push n012 push y012 mult add \
push n022 push y022 mult add \
push n112 push y112 mult add 
push n122 push y122 mult add 
store x3 \
push x0 push t00 mult \
push x1 push t10 mult add \
push x2 push t20 mult add 
push x3 push t30 mult add \

927
```plaintext
19362 return
19363 push x0 push t01 mult
19364 push x1 push t11 mult add
19365 push x2 push t21 mult add
19366 push x3 push t31 mult add
19367 return
19368 push x0 push t02 mult
19369 push x1 push t12 mult add
19370 push x2 push t22 mult add
19371 push x3 push t32 mult add
19372 return
19373 push x0 push t03 mult
19374 push x1 push t13 mult add
19375 push x2 push t23 mult add
19376 push x3 push t33 mult add
19377 return
19378 -code_end

19380

19381 Clebsch_surface:
19382 $(ORBITER) -v 2 -povray
19383 -round 0 -nb_frames_default 30
19384 -output_mask Clebsch_%d_03d.pov
19385 -video_options -W 1024 -H 768
19386 -global_picture_scale 0.9
19387 -default_angle 75
19388 -clipping_radius 2.4
19389 -camera 0 "1,1,1" "-3,1,3" "0.12,0.12,0.12"
19390 -end
19391 -scene_objects
19392 -cubic_orbiter "0,0,0,0,0,-4.236067972,,
19393 0,0,4.236067972,4.236067972,17.94427188,
19394 -17.94427188,0,0,- 9.472135941,0,0,5.236067971,
19395 8.472135938,- 27.41640782" 
19396 -group_of_things "0"
19397 -cubics 0 $(SURFACE_COLOR_SEETHROUGH)
19398 -point_list_from_csv_file
19399 -function_Clebsch_map_of_circle_N1000_points.csv
19400 -group_of_things_as_interval 0 954
19401 -spheres 1 0.07 $(COLOR_RED)
19402 -scene.objects.end
19403 -povray.end
19404 -rm -rf POV
19405 mkdir POV
19406 mv Clebsch_0_*.pov POV
19407 mv makefile_animation POV
```
Clebsch_surface_defining_equation:

```bash
$(ORBITER) -v 2 -povray \
-round 0 -nb_frames_default 30 \n-output_mask Clebsch_%d_%03d.pov \n-video_options -W 1024 -H 768 \nglobal_picture_scale 0.6 \ndefault_angle 75 \n-clipping_radius 1.6 \ncamera 0 "1,1,1" "-2,0,2" "0,0,0" \n-end \
cubic_orbiter "0,0,0,1,1,1,1,1,1,1,1,1,2,2,2,2" \ngroup_of_things "0" \ncubic 0 $(SURFACE_COLOR_SEETHROUGH) \nsphere 1 0.07 $(COLOR_RED) 
point_list_from_csv_file \nfunction_Clebsch_map_of_circle_to_defining_eqn_N1000_points.csv \ngroup_of_things_as_interval 0 656 \nsphere 2 0.07 "texture{ pigment{ color Blue } finish { diffuse 0.9 phong 1}}\n```

Clebsch_surface_defining_equation_and_curves:

```bash
$(ORBITER) -v 2 -povray \
-round 0 -nb_frames_default 30 \n-output_mask Clebsch_2curves_%d_%03d.pov \n-video_options -W 1024 -H 768 \nglobal_picture_scale 0.6 \ndefault_angle 75 \n-clipping_radius 1.6 \ncamera 0 "1,1,1" "-2,0,2" "0,0,0" \n-end \
cubic_orbiter "0,0,0,1,1,1,1,1,1,1,1,1,2,2,2,2" 
group_of_things "0" 
cubic 0 $(SURFACE_COLOR_SEETHROUGH) 
point_list_from_csv_file \nfunction_Clebsch_map_of_circle_to_defining_eqn_N1000_points.csv 
group_of_things_as_interval 0 656 \nsphere 1 0.07 $(COLOR_RED) 
point_list_from_csv_file 
function_Clebsch_map_of_circle_to_defining_eqn_r2_N1000_points.csv 
group_of_things_as_interval 656 1042 
sphere 2 0.07 "texture{ pigment{ color Blue } finish { diffuse 0.9 phong 1}}"
```

Clebsch_surface_defining_equation:
19456 \# -scene_objects_end \\
19457 \# -povray_end \\
19458 - rm -rf POV \\
19459 mkdir POV \\
19460 mv Clebsch_2curves_0_*.pov POV \\
19461 mv makefile_animation POV \\
19462 # \# -point_list_from_csv_file function_Clebsch_map_of_circle_N1000_points.csv \\
19463 \# -group_of_things_as_interval 0 954 \\
19464 \# -spheres 1 0.07 "texture{ pigment{ color Red } finish { diffuse 0.9 phong 1}}" \\
19466 \\
19467 \\
19468 \\
19469 \\
19470 F7_povray: \\
19471 \$(ORBITER) -v 2 -povray \\
19472 \# -round 0 -nb_frames_default 30 \\
19473 \# -output_mask F7_15_lines_%d_%03d.pov \\
19474 \# -video_options -W 1024 -H 768 \\
19475 \# -global_picture_scale 1.5 \\
19476 \# -default_angle 80 \\
19477 \# -clipping_radius 4.4 \\
19478 \# -omit_bottom_plane \\
19479 \# -camera 0 "1,1,1" "-4.5,3.5,6" "0,0,0" \\
19480 \# -end \\
19481 \# -scene_objects \\
19482 \# \# -cubic_lex "0, 0, 6, 0, 0, -13.39014946, -3.341901346, -6.972931640, 5.82718, 0, 0, 7.390149464, 7.390149464, 6.972931640, -1.512349728, -8.485281372, 0 , 0, 0" \\
19483 \# \# -group_of_things "0" \\
19484 \# \# -cubics 0 $(SURFACE_COLOR_SEETHROUGH) \\
19485 \# \# -line_through_point_with_direction "0, 0, 0, 1, 0, 0" \\
19486 \# \# -line_through_point_with_direction "0, 0, -1, 0, 1, 0" \\
19487 \# \# -line_through_point_with_direction "0, 0, 0, 0, -1" \\
19488 \# \# -line_through_point_with_direction "1, 0, 0, 1, 1" \\
19489 \# \# -line_through_point_with_direction "-1.414213562, 0, 0, 4.146264370, 1.732050808, 1.732050808" \\
19490 \# \# -line_through_point_with_direction "0, 1.732050808, -1, 2.414213562, -0.317837246, 2.414213562" \\
19491 \# \# -line_through_point_with_direction "-2.133352390, 0, -1, 1.674708020, 1, 0" \\
19492 \# \# -line_through_point_with_direction "-2.539058015, 0, 0, 2.211360755, 1, 0" \\
19493 \# \# -line_through_point_with_direction "0, 1.148188060, 0, 0, -0.9435440612, 1"
-line_through_point_with_direction "-0.9711971171, 0, 0, 1.162155272, 0, 1" \\
-line_through_point_with_direction "2.096037870, 2.096037870, 0, -1.065851905, -1.065851905, 1" \\
-line_through_point_with_direction "3.921555783, 2.921555781, 0, -1.722456585, -1.722456585, 1" \\
-group_of_things_as_interval 0 12 \\
-lines 1 0.04 $(COLOR_YELLOW) \\
-scene_objects_end \\
-povray_end \\
- rm -rf POV \\
-mkdir POV \\
mv F7.15_lines_0_*.pov POV \\
mv makefile_animation POV \\

F7_video: \\
- rm -r FRAMES \\
-mkdir FRAMES \\
- rm fifteen_with_lines.mp4 \\
$(ORBITER) \\
-prepare_frames \\
-i 0 30 F7b/F7.15_lines_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 fifteen_with_lines.mp4 \\

McKean_povray: \\
$(ORBITER) -v 2 -povray \\
-round 0 -nb_frames_default 30 \\
-output_mask McKean_%d_%03d.pov \\
-video_options -W 1024 -H 768 \\
-global_picture_scale 1.5 \\
-default_angle 80 \\
-clipping_radius 4.4 \\
-omit_bottom_plane \\
-camera 0 "1,1,1" "-4.5,3.5,6" "0,0,0" \\
-end \\
-scene_objects \\
-cubic_lex "0, 0, 1, 0, 0, -1, -2, 1, \\n
931
19538 2, 0, 0, 1, -1, -1, 0, 0, 0, 0" \n19539 ▶ ▶ ▶ -group_of_things "0" \n19540 ▶ ▶ ▶ -cubics 0 $(SURFACE_COLOR_SEETHROUGH) \n19541 ▶ ▶ -scene_objects_end \n19542 ▶ -povray_end
19543 ▶ - rm -rf POV
19544 ▶ mkdir POV
19545 ▶ mv McKean_0*.pov POV
19546 ▶ mv makefile_animation POV
19547
19548 McKean_video:
19549 ▶ - rm -r FRAMES
19550 ▶ - mkdir FRAMES
19551 ▶ - rm McKean.mp4
19552 ▶ $(ORBITER) \n19553 ▶ ▶ -prepare_frames \n19554 ▶ ▶ ▶ -i 0 30 MCKEAN/McKean_0%03d.png \n19555 ▶ ▶ ▶ -output_starts_at 0 \n19556 ▶ ▶ ▶ -o FRAMES/frame%04d.png \n19557 ▶ ▶ -end
19558 ▶ ffmpeg -r 5 -f image2 -i FRAMES/frame%04d.png \n19559 ▶ ▶ -f mp4 -q:v 0 -vcodec mpeg4 McKean.mp4
19560
19561
19562
19563
19564 #Section 16.4: Continuous Function Plotter
19565
19566
19567
19568
19569 SECTION_CONTINUOUS_FUNCTION_PLOTTER:
19570
19571
19572
19573 test_16_4:
19574 ▶ make lissajous
19575 ▶ make lissajous_plot
19576 ▶ make lissajous_3d
19577 ▶ make lissajous_3d_plot
19578
19579
19580
19581
19582 lissajous:
19583 ▶ $(ORBITER) -v 2 \n19584 ▶ ▶ -smooth_curve "lissajous" 0.07 2000 15 0 18.85 \n
932
-const a 3 b 2 c 1.57 r 7 -const_end \n-const a 3 b 2 c 1.57 r 7 -const_end 
-var t -var_end \n-code \n-code \n-push t push a mult push c add sin push r mult return \n-push t push b mult sin push r mult return \n-code_end 

#function_lissajous_N2000_points.csv

lissajous_plot:

$(ORBITER) -v 2 -povray \n-round 0 -nb_frames_default 1 \n-output_mask lissajous_%d_%03d.pov \n-video_options -W 1024 -H 768 \n-global_picture_scale 0.40 \n-default_angle 45 \n-clipping_radius 5 \n-omit_bottom_plane \n-camera 0 "0,-1,0" "0,0,12" "0,0,0" \n-rotate_about_z_axis \n-end \n-scene_objects \n-line_through_two_points_recentered_from_csv_file \n-coordinate_grid.csv \n-group_of_things "0" \n-group_of_things "1" \n-group_of_things "2" \n-lines 0 0.09 "texture{ pigment{ color Yellow } }" \n-lines 1 0.09 "texture{ pigment{ color Yellow } }" \n-lines 2 0.09 "texture{ pigment{ color Yellow } }" \n-group_of_things_as_interval 3 39 \n-lines 3 0.02 "texture{ pigment{ color Black } }" \n-point_list_from_csv_file \n-function_lissajous_N2000_points.csv \n-group_of_things_as_interval 0 6524\n-spheres 4 0.1 "texture{ pigment{ color Red } }" \n-finish { diffuse 0.9 phong 1})" \n-plane_by_dual_coordinates "0,0,1,0" \n-group_of_things "0" \n-planes 5 "texture{ pigment{ color Blue*0.5 \n-transmit 0.5 } }"" \n-scene_objects_end \n-povray_end

- rm -rf POV
-mkdir POV
-mv lissajous_0_* .pov POV
-mv makefile_animation POV
lissajous_3d:

```
$\text{(ORBITER)}$ -v 2 \n-smooth_curve "lissajous_3d" 0.07 2000 50 0 18.85 \n-const a 3 b 2 c 1.57 r 7 -const_end \n-var t -var_end \n-code \n-push t push a mult push c add sin push r mult return \n-push t push b mult sin push r mult return \n-push t return \n-code_end 
```

lissajous_3d_plot:

```
$\text{(ORBITER)}$ -v 2 -povray \nr-round 0 -nb_frames_default 30 \n-output_mask lissajous_3d%.d%.03d.pov \n-video_options -W 1024 -H 768 \n-global_picture_scale 0.40 \n-default_angle 45 \n-clipping_radius 5 \n-omit_bottom_plane \ncamera 0 "0,0,1" "7,7,5" "0,0,1" \n-rotate_about_z_axis \n-end \n-scene_objects 
```

```
mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation POV

mv makefile_animation PO
19726  -define all_one_r -vector -repeat 1 12 -end \\
19727  -define all_one_c -vector -repeat 1 12 -end \\
19728  -draw_matrix \\
19729  -input_object M \ 
19730  -box_width 50 -bit_depth 24 \ 
19731  -partition 3 \ 
19732  -all_one_r all_one_c \ 
19733  -end \\
19734 
19735 
19736 # Section 17.2: Limitations
19737 
19738 
19739 SECTION_LIMITATIONS:
19740 
19741 
19742 
19743 ####
19744 
19745 
19746 # unclassified:
19747 
19748 
19749 
19750 
19751 
19752 
19753 extract: 
19754  $(ORBITER) -v 3 \\
19755  -extract_from_file makefile Cremona_map make_Cremona_map.txt \\
19756  ~/bin/a2tex.out -numbers -text_width 80 <make_Cremona_map.txt >make_Cremona_map.tex \\
19757  pdflatex ev_diag23 \\
19758  open ev_diag23.pdf \\
19759 draw.eigenvalue_diag23: 
19760  $(ORBITER) -v 2 \\
19761  -draw_options \\
19762  -radius 10 \\
19763  -line_width 1.5 -embedded \\
19764  -end \\
19765  -draw_mod.n -n 20 \\
19766  -file ev_diag23 \\
19767  -eigenvalues 2 0 0 3 \\
19768  -end \\
19769  pdflatex ev_diag23_draw.tex \\
19770  open ev_diag23_draw.pdf \\
19771
Bibliography


Fatma Karaoglu and Anton Betten. The number of cubic surfaces with 27 lines over a finite field. To appear in Journal of Algebraic Combinatorics.


[61] 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.


