## Contents

1 Introduction  
1.1 What is Orbiter .............................. 7  
1.2 Orbiter Objects .............................. 8  

2 Getting Started  
2.1 Running and Installing Orbiter .......................... 11  
2.2 The Orbiter Session .............................. 16  
2.3 Makefiles and Shell Scripts ............................ 19  
2.4 Objects and Activities ............................... 22  
2.5 Mathematical Data ................................. 26  
2.6 Set Builder .......................... 28  
2.7 Vector Builders ................................. 29  
2.8 Formula Builders ................................. 33  

3 Basic Algebra  
3.1 Basic Number Theory ............................... 35  
3.2 Prime Fields .................................. 41  
3.3 Extension Fields ................................... 47  
3.4 Linear Algebra Over Finite Fields .......................... 54  
3.5 Advanced Topics in Finite Fields .......................... 66  
3.6 Basic Ring Theory ................................. 75  

4 Geometry  
4.1 Finite Projective Spaces ............................. 77  
4.2 Indexing Points and Lines ............................ 84  
4.3 Finite Desarguesian Projective Planes .......................... 89  
4.4 The Grassmannian .................................... 93  
4.5 Algebraic Sets ...................................... 95  
4.6 The Klein Quadric and the Plücker Map .......................... 100  
4.7 Orthogonal Spaces .................................. 102  
4.8 Hermitian Varieties ................................. 109  
4.9 Advanced Topics ................................. 111  
4.10 Geometric Objects ................................. 118
18 Orbiter on Windows 485
  18.1 Using Windows Subsystem Linux 485

19 The Makefile 495
  19.1 The Makefile 495
  References 885
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].
1.2 Orbiter Objects

Orbiter objects are models of mathematical objects. Objects are created and activities are invoked for these objects. This way, most of the functionality of Orbiter is attached to an object. Tables 1.1-1.2 give an overview of Orbiter objects, with pointers to the relevant chapter or section in this user's guide. Besides these Objects and their activities, there are still functions that are global. This means that these functions do not require any objects.
<table>
<thead>
<tr>
<th>Object</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finite field</td>
<td>Chapter 3</td>
</tr>
<tr>
<td>Polynomial ring</td>
<td>Chapter 8</td>
</tr>
<tr>
<td>Projective space</td>
<td>Chapter 4</td>
</tr>
<tr>
<td>Orthogonal space</td>
<td>Chapter 4</td>
</tr>
<tr>
<td>Any group</td>
<td>Chapter 5</td>
</tr>
<tr>
<td>Linear group</td>
<td>Section 5.2</td>
</tr>
<tr>
<td>Permutation group</td>
<td>Section 5.1</td>
</tr>
<tr>
<td>Group modification</td>
<td>Section 5.5</td>
</tr>
<tr>
<td>Formula</td>
<td></td>
</tr>
<tr>
<td>Collection</td>
<td></td>
</tr>
<tr>
<td>Cubic surface</td>
<td>Chapter 7</td>
</tr>
<tr>
<td>Quartic curve</td>
<td>Chapter 7.2</td>
</tr>
<tr>
<td>Classification of c.s.</td>
<td>Chapter 7.3</td>
</tr>
<tr>
<td>Geometric object</td>
<td>Section 4.10</td>
</tr>
<tr>
<td>Graph</td>
<td>Chapter 13</td>
</tr>
<tr>
<td>Graph classification</td>
<td>Section 13.3</td>
</tr>
<tr>
<td>Spread table</td>
<td>Section 12.1</td>
</tr>
</tbody>
</table>

Table 1.1: Orbiter Objects
<table>
<thead>
<tr>
<th>Object</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packing w.a.s.</td>
<td>Section 12.3</td>
</tr>
<tr>
<td>Packing w.a.s.f.</td>
<td>Section 12.3</td>
</tr>
<tr>
<td>Packing long orbits</td>
<td>Section 12.3</td>
</tr>
<tr>
<td>Diophant</td>
<td>Section 11.2</td>
</tr>
<tr>
<td>Design</td>
<td>Section 11.5</td>
</tr>
<tr>
<td>Design table</td>
<td>Section 11.5</td>
</tr>
<tr>
<td>Large set w.a.s.</td>
<td>Section 11.6</td>
</tr>
<tr>
<td>Set</td>
<td>Section 2.6</td>
</tr>
<tr>
<td>Vector / matrix</td>
<td>Section 2.7</td>
</tr>
<tr>
<td>Combinatorial objects</td>
<td>Chapter 14</td>
</tr>
<tr>
<td>Geometry builder</td>
<td>Section 11.4</td>
</tr>
<tr>
<td>Group action</td>
<td>Section 5.5</td>
</tr>
<tr>
<td>Poset</td>
<td>Section 6.2</td>
</tr>
<tr>
<td>Poset classification</td>
<td>Section 6.2</td>
</tr>
</tbody>
</table>

Table 1.2: Orbiter Objects
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 platform, 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

\texttt{src/apps/orbiter/orbiter.out}

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

\begin{verbatim}
    |-- orbiter
    |    |-- work
\end{verbatim}

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 environ-
ment. 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 \n    /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
Figure 2.1: The commit number

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, 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
004f1eed87df: 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 (gnuc and make). Windows users may have to install cygwin (plus the extra packages git, make, gnuc). Macintosh
users may have to install the xcode development tools from the appstore (it is free). Linux users may have to install the development packages. Orbiter often produces latex reports. In order to compile these files, make sure you have latex installed

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

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

where `<github-orbiter-path>` has to be replaced by the actual address provided by github. To obtain this path, find Orbiter on github, then click on the green box that says “Code” and copy the address into the clipboard by clicking the clipboard symbol (see 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`.
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. Once started, the Orbiter session will produce a short welcome message:

```
Welcome to Orbiter! Your build number is 1081.
A user’s guide is available here:
https://www.math.colostate.edu/~betten/orbiter/users_guide.pdf
The sources are available here:
https://github.com/abetten/orbiter
An example makefile with many commands from the user’s guide is here:
https://github.com/abetten/orbiter/tree/master/examples/users_guide/makefile
Orbiter session finished.
```
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Meaning</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</td>
<td>$L M f t s$</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 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.

One tool that stands out in the unix world is called make. Make is a command that allows to execute certain processes on a need basis. The need is defined by means of time stamps on files. The rules are defined in a file called makefile. Make is very popular in software engineering, where there are dependencies between source code, object code and executable files. We note that it is not necessary to use makefiles. However, because of the convenience they offer in defining lightweight commands, this user’s guide will rely on the make / makefile tool. It would also be possible to define shell scripts for each of the commands.

Orbiter can be used through makefiles, with or without using the dependency functionality. One feature of makefiles that is very useful is that commands can be defined very quickly, and that one makefile can hold many commands. This provides an advantage over shell scripting, where separate shell scripts are needed for each command. For instance, this user’s guide is based on a makefile that contains all commands shown. The makefile is listed in full in Section 19.1. In the user’s guide, the relevant pieces of code are shown one at a time. Make also allows to use variables, which are used by means of text substitution. A variable is defined as

\[ A=\text{"I am a variable"} \]

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

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

The purpose of this command is to produce a file called

```
GF_16.tex
```

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

```
F_16:
  ⊳ $(ORBITER) -v 3 \n  ⊳ ⊳ -define F -finite_field -q 16 -end \n  ⊳ ⊳ -with F -do -finite_field_activity -cheat_sheet_GF -end \n  ⊳ pdflatex GF_16.tex
```

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

Makefiles are somewhat picky when it comes to whitespacing. The command sequence needs to be indented with tab symbols. Leading spaces will cause make to issue an error message. Also, there should be no whitespace after the trailing backslash symbol. Some editors can display whitespace characters. This may be helpful when editing the makefile.

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 **ORBITER_PATH** below.

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

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

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

\texttt{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. In this case, we should change the line

ORBITER\_PATH=\$(MY\_PATH)/src/apps/orbiter/

to

ORBITER\_PATH=

in the makefile.
2.4 Objects and Activities

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

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

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 Table 2.2. 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). Once the field is created, the orbiter session terminates. The command

```
object PG_3_2:
$(ORBITER) \n\define F -finite_field -q 2 -end \n\define P -projective_space -n 3 -field F -v 0 -end
```

creates the same finite field $F$ as well as an object $P$ representing PG(3, 2). Note how the creation of $P$ relies on the existence of $F$. The `-projective_space` option requires two parameters, the dimension of the projective space and the field over which it is defined. In the example, the field $F$ which has been created earlier is referenced by its label as the second argument.

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>Description</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>-projective_space</td>
<td>A projective space of dimension $n$ over a finite field $F$. See Section 4.1.</td>
</tr>
<tr>
<td>-orthogonal_space</td>
<td>A non-degenerate orthogonal space. See Section 4.7.</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>-formula</td>
<td>A symbolic expression. See Section 10.6.</td>
</tr>
<tr>
<td>-collection</td>
<td>A collection of objects.</td>
</tr>
<tr>
<td>-graph</td>
<td>A graph. See Section 13.1.</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>
<tr>
<td>-packing_choose_fixed_points</td>
<td>A selection of fixed orbits for packings with assumed symmetry. See Section 12.3.</td>
</tr>
<tr>
<td>-packing_long_orbits</td>
<td>A search for long orbits for packings with assumed symmetry. See Section 12.3.</td>
</tr>
<tr>
<td>-graph_classification</td>
<td>An object which allows classifying graphs and tournaments. See Section 13.3.</td>
</tr>
<tr>
<td>-diophant</td>
<td>A diophantine system, i.e., a system of positive integer equations). See Section 11.2.</td>
</tr>
<tr>
<td>-design</td>
<td>A combinatorial design. See Section 11.5.</td>
</tr>
<tr>
<td>-design_table</td>
<td>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 Section 11.5.</td>
</tr>
<tr>
<td>-large_set_with_symmetry_assumption</td>
<td>An object to create a large set of designs. See Section 11.5.</td>
</tr>
<tr>
<td>-set</td>
<td>A set. See Section 2.6.</td>
</tr>
<tr>
<td>-vector</td>
<td>A vector over a finite field. See Section 2.7.</td>
</tr>
</tbody>
</table>

Table 2.2: Orbiter Objects
command sequence is used. Here, LABEL is the name under which the object is registered in the symbol table. DESCRIPTION is the activity that should be applied. Some activities require more than one object, in which case the syntax

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

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

Table 2.3 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>Description</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 2.3: 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>
<tr>
<td>Hyperovals</td>
<td>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 $e = 3, 4, 5$.</td>
</tr>
<tr>
<td>Dual hyperovals</td>
<td>A $k$-dimensional dual hyperoval in an ambient space $\mathbb{F}_2^n$ is called a $\text{DH}(k, n)$. Orbiter knows the classification of dual hyperovals $\text{DH}(4, 7)$ and $\text{DH}(4, 8)$.</td>
</tr>
<tr>
<td>Packings</td>
<td>Orbiter knows the classification of packings of $\text{PG}(3, 3)$.</td>
</tr>
</tbody>
</table>

Table 2.4: 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 2.4. 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

```
create_BLT_5_1:
▷ $(ORBITER) -v 2 \\
▷ ▷ -define F -finite_field -q 5 -end \\
▷ ▷ ▷ -define 0 -orthogonal_space 0 5 F -end \\
```
recalls the BLT-set with Orbiter Catalogue Number 1 in $Q(4,5)$. A latex report 
catalogue_q5_iso1.tex is written. For more details about BLT-sets, see Section 12.4.

The command

```
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 \\
  -with P -do \\
  -projective_space_activity \\
  -define_surface S4_0 -q 4 -catalogue 0 -end \\
  -end \\
  -with S4_0 -do \\
  -cubic_surface_activity \\
  -report \\
  -end
```

recalls the cubic surface with Orbiter Catalogue Number 0 in $PG(3,4)$. A latex report 
surface_catalogue_q4_iso0_report.tex is written. For more details about cubic sur-
faces, 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\}:

```bash
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.

```bash
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

```c
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
```

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
```
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"``

```
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="\n1101110001000001010000\n11110101111101000001011\n0000010000010001001001\n11111011010001001110\n01010100000010011101\n0000000000010001001001\n0100000000010001001001\n0001000011000000111111\n1110100100110100001110\n0000000000000001000101\n0000000000000100010111\n0000000001000100010101\n0000000000000100010001\n0000000000000110010101\n0000000000000101010101\n0000000000000100010100\n0000000000000100010111\n0000000000000100010001"```
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:

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

The command

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

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 \( a = 2, b = 3, x = 4 \):

\[
\text{formula\_evaluate:}
\]
\[
\text{\$\{ORBITER\} \ -v \ 3 \ }
\]
\[\text{-define } F \text{-finite-field } -q 5 \text{-end} \]
\[\text{-define } f \text{-formula} \]
\[\text{"test formula" "test\_formula" ""} \]
\[\text{$(TEST\_FORMULA)$} \]
\[\text{-with } F \text{-do } \text{-finite-field-activity} \]
\[\text{-evaluate } f \text{ "a=2,b=3,x=4" -end} \]
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 -primitive_root command can be used. The algorithm is randomized. For instance,

PR29:
$\langle$ORBITER$\rangle$ -v 1 -smallest_primitive_root 29

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

PR_915839:
$\langle$ORBITER$\rangle$ -v 5 -primitive_root 915839

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

PR_915839_check:
$\langle$ORBITER$\rangle$ -v 5 -power_mod 43085 49842 915839

computes $43085^{49842} \mod 915839$

which is 487320.

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

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

For instance, the discrete log of 487320 with respect to the base 43085 modulo 915839 is 49842, based on the previous example. We can compute the discrete logarithm using the command
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-power_mod</td>
<td>a n p</td>
<td>Computes $a^n \pmod{p}$.</td>
</tr>
<tr>
<td>-discrete_log</td>
<td>b a p</td>
<td>Computes $n$ such that $a^n \equiv b \pmod{p}$.</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 $b^2 \equiv a \pmod{p}$.</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 $ab \equiv 1 \pmod{p}$.</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 $n_{\text{min}} \leq n \leq n_{\text{max}}$ for which $\gcd(n, q) = 1$. Also computes $\varphi(n)$ and $\varphi(n)/\text{ord}(q, n)$.</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 \(-\text{inverse\_mod}\) command. The command

\[
\text{IM}: \quad \text{
$(\text{ORBITER})$ -v 5 -inverse\_mod 1865025205 2147483647}
\]

computes the inverse of 1865025205 modulo 2147483647 which is 579785381.

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

\[
\gcd(a, b) = ua + vb
\]

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

\[
\text{IM.gcd}: \quad \text{
$(\text{ORBITER})$ -v 5 -extended\_gcd 1865025205 2147483647}
\]

The output is

\[
1 = -503526232 \times 2147483647 + 579785381 \times 1865025205,
\]

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

In order to compute the modular power

\[
a^e \mod n,
\]

the \(-\text{power\_mod}\) command can be used. For instance,

\[
\text{PM3a}: \quad \text{
$(\text{ORBITER})$ -v 5 -power\_mod 16807 1073741823 2147483647}
\]

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

The modular square root of \( a \) modulo \( p \) is any \( x \) in

\[
x^2 \equiv a \mod p.
\]

The command \(-\text{square\_root\_mod}\) 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>131</td>
<td>14</td>
<td>64</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 $\text{order\_of\_q\_mod\_n}$ 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,

$\text{order\_of\_2\_mod\_n}$:

▷ $\$(ORBITER) -v 3 -order\_of\_q\_mod\_n 2 3 151
▷ $\$(ORBITER) -v 1 -csv\_file\_latex 1 \$
▷ $\$ 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

38
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>
<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>
<td>101</td>
<td>100</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>
<td>102</td>
<td>32</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>
<td>103</td>
<td>102</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>
<td>104</td>
<td>48</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>
<td>105</td>
<td>48</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>
<td>106</td>
<td>52</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>
<td>107</td>
<td>106</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>
<td>108</td>
<td>36</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>
<td>109</td>
<td>108</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>
<td>110</td>
<td>40</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>
<td>111</td>
<td>72</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>
<td>112</td>
<td>48</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>
<td>113</td>
<td>112</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>
<td>114</td>
<td>36</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>
<td>115</td>
<td>88</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>
<td>116</td>
<td>56</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>
<td>117</td>
<td>72</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>
<td>118</td>
<td>58</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>
<td>119</td>
<td>96</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>
<td>120</td>
<td>32</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>
<td>121</td>
<td>110</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>
<td>122</td>
<td>60</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>
<td>123</td>
<td>80</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>
<td>124</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>
<td>125</td>
<td>100</td>
</tr>
</tbody>
</table>

Eulerfunction\_150:

- $(\text{ORBITER}) -v 1 -eulerfunction\_interval 1 150$
- $(\text{ORBITER}) -v 1 -csv\_file\_latex 1$
- 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 \( a \mapsto a^k \mod 11 \):

power\_function\_2\_mod\_11:

- $(\text{ORBITER}) -v 5 -power\_function\_mod\_n 2 11$
- $(\text{ORBITER}) -v 1 -csv\_file\_latex 1 power\_function\_k2\_n11.csv$
- 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>$a^2 \mod 11$</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:
  $(ORBITER) -v 2 \$
  $-draw\_options -embedded -end \$
  $-draw\_mod\_n -n 13 -file mod_13 -power\_cycle 2 -end$
  pdflatex mod_13\_draw.tex
  open mod_13\_draw.pdf
```

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

Let \( \mathbb{F}_q \) denote the finite field with \( q \) elements. Up to isomorphism, there is only one field of order \( q \). Finite fields of prime order can be created as integer factor ring.

Important comment: Orbiter implements finite fields using tables for addition and multiplication. This imposes a limitation on the size of the field that can be created.

See Section 17.2 for a list of limitations of Orbiter.

If \( p \) is a prime number, the integer factor ring \( \mathbb{Z}/I(p) \) is a finite field. Here, 
\[
I(p) = p\mathbb{Z} = \{ pk \mid k \in \mathbb{Z} \} = \{ 0, \pm k, \pm 2k, \pm 3k, \ldots \}
\]
is the ideal of all integer multiples of \( p \). The elements of \( \mathbb{F}_p \) are the residue classes of the ideal given by the integer multiples of \( p \). Each residue class has the form 
\[
\{ a + kp \mid k \in \mathbb{Z} \}.
\]

Standard representatives of the equivalence classes can be chosen as the smallest non-negative member in each class. This means that the standard representatives are the integers from 0 to \( p - 1 \). This canonical representative is the remainder after division by \( p \). Two integers belong to the same residue class if they have the same remainder after division by \( p \). For instance, 11 and 46 are in the same residue class modulo 5 because both have a remainder of 1 after division by five. It is convenient to identify the residue classes mod \( p \) with the integers from 0 to \( p - 1 \). In Orbiter, this convention is used automatically. The addition table and the multiplication table can be used to add and multiply in \( \mathbb{F}_p \).

For instance, in Figure 3.2 the addition and multiplication tables of \( \mathbb{F}_7 \) are shown, both numerically and using colors. The natural ordering of the integers in the interval \([0, 6]\) is used. Different integers are represented by different colors. It is customary to restrict the multiplication table to the non-zero elements of the field.

A finite field \( \mathbb{F}_q \) can be created using the \(-finite_field\) command. Table 3.5 lists Orbiter commands for creating a finite field that can come after \(-finite_field\). For instance,

\( \mathbb{F}_2 \):

\[
\begin{align*}
\text{\texttt{F.2:}} & \quad \text{\texttt{$(ORBITER) -v 3 -list\_arguments \ \}} \\
\text{\texttt{\quad \quad -define F -finite_field -q 2 -end \ \}} \\
\text{\texttt{\quad \quad -with F -do -finite_field\_activity -cheat\_sheet\_GF -end \ \}} \\
\text{\texttt{\quad \quad pdflatex GF.2.tex \ \}} \\
\text{\texttt{\quad \quad open GF.2.pdf \ \}}
\end{align*}
\]

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

Table 3.6 lists basic Orbiter activities for finite fields. More activities will follow in Section 3.3. Here is the cheat sheet for \( \mathbb{F}_7 \). The element \( \alpha \) is a primitive element.
Figure 3.2: Addition and multiplication tables of $\mathbb{F}_7$

<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-q</code></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><code>-override_polynomial</code></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><code>-without_tables</code></td>
<td></td>
<td>Create the field without precomputing the tables.</td>
</tr>
</tbody>
</table>

Table 3.5: Options for Creating Finite Fields
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Description</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</td>
<td>$v$</td>
<td>Compute the product of all field elements in the vector $v$.</td>
</tr>
<tr>
<td>-sum_of</td>
<td>$v$</td>
<td>Compute the sum of all field elements in the vector $v$.</td>
</tr>
<tr>
<td>-negate</td>
<td>$v$</td>
<td>Negate each field element in the vector $v$.</td>
</tr>
<tr>
<td>-inverse</td>
<td>$v$</td>
<td>Compute the multiplicative inverse of each field element in the vector $v$.</td>
</tr>
<tr>
<td>-power_map</td>
<td>$k, v$</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>

$$+$ | 0 1 2 3 4 5 6 |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 1 2 3 4 5 6</td>
</tr>
<tr>
<td>1</td>
<td>1 2 3 4 5 6 0</td>
</tr>
<tr>
<td>2</td>
<td>2 3 4 5 6 0 1</td>
</tr>
<tr>
<td>3</td>
<td>3 4 5 6 0 1 2</td>
</tr>
<tr>
<td>4</td>
<td>4 5 6 0 1 2 3</td>
</tr>
<tr>
<td>5</td>
<td>5 6 0 1 2 3 4</td>
</tr>
<tr>
<td>6</td>
<td>6 0 1 2 3 4 5</td>
</tr>
</tbody>
</table>

43
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 \(-1 \mod 11\): 

\[
\begin{array}{c|cccccc}
\cdot & 1 & 2 & 3 & 4 & 5 & 6 \\
\hline
1 & 1 & 2 & 3 & 4 & 5 & 6 \\
2 & 2 & 4 & 6 & 1 & 3 & 5 \\
3 & 3 & 6 & 2 & 5 & 1 & 4 \\
4 & 4 & 1 & 5 & 2 & 6 & 3 \\
5 & 5 & 3 & 1 & 6 & 4 & 2 \\
6 & 6 & 5 & 4 & 3 & 2 & 1 \\
\end{array}
\]

\[
\begin{array}{ll}
3^0 \equiv 1 & \quad 3^4 \equiv 4 \\
3^1 \equiv 3 & \quad 3^5 \equiv 5 \\
3^2 \equiv 2 & \quad 3^6 \equiv 1 \\
3^3 \equiv 6 & \\
\end{array}
\]

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.

\[
\begin{array}{c|cccccc}
\cdot & 1 & 2 & 3 & 4 & 5 & 6 \\
\hline
1 & 1 & 2 & 3 & 4 & 5 & 6 \\
2 & 2 & 4 & 6 & 1 & 3 & 5 \\
3 & 3 & 6 & 2 & 5 & 1 & 4 \\
4 & 4 & 1 & 5 & 2 & 6 & 3 \\
5 & 5 & 3 & 1 & 6 & 4 & 2 \\
6 & 6 & 5 & 4 & 3 & 2 & 1 \\
\end{array}
\]

\[
\begin{array}{ll}
3^0 \equiv 1 & \quad 3^4 \equiv 4 \\
3^1 \equiv 3 & \quad 3^5 \equiv 5 \\
3^2 \equiv 2 & \quad 3^6 \equiv 1 \\
3^3 \equiv 6 & \\
\end{array}
\]

F_11.product_of_all_nonzero.elements:
\[
\begin{array}{c|cccccc}
\cdot & 1 & 2 & 3 & 4 & 5 & 6 \\
\hline
1 & 1 & 2 & 3 & 4 & 5 & 6 \\
2 & 2 & 4 & 6 & 1 & 3 & 5 \\
3 & 3 & 6 & 2 & 5 & 1 & 4 \\
4 & 4 & 1 & 5 & 2 & 6 & 3 \\
5 & 5 & 3 & 1 & 6 & 4 & 2 \\
6 & 6 & 5 & 4 & 3 & 2 & 1 \\
\end{array}
\]

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 3.3, 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.

The class `finite_field` uses a precomputed tables for the arithmetic operations. The option `-without_tables` can be given to avoid precomputing tables. This may be helpful for large fields. Here is an example. We create the field \( \mathbb{F}_{101} \) without precomputed tables:

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

\[
0, \\
1, \\
X, \\
X + 1.
\]

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

\[
\begin{array}{c|cccc}
 & 0 & 1 & X & X + 1 \\
\hline
0 & 0 & 1 & X & X + 1 \\
1 & 1 & 0 & X + 1 & X \\
X & X & X + 1 & 0 & 1 \\
X + 1 & X + 1 & X & 1 & 0 \\
\end{array}
\]

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 $X + 1$:

\[
\begin{align*}
X \cdot X &= X^2 \\ X \cdot (X + 1) &= X^2 + X \\ (X + 1) \cdot X &= X^2 + X \\ (X + 1) \cdot (X + 1) &= X^2 + 1
\end{align*}
\]

\[
\equiv X + 1 \mod X^2 + X + 1, \quad \equiv 1 \mod X^2 + X + 1, \quad \equiv 1 \mod X^2 + X + 1, \quad \equiv X \mod X^2 + X + 1,
\]

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

\[
\begin{array}{c|cccc}
 & 0 & 1 & X & X + 1 \\
\hline
0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & X & X + 1 \\
X & 0 & X & X + 1 & 1 \\
X + 1 & 0 & X + 1 & 1 & X \\
\end{array}
\]

Figure 3.4 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>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Description</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</td>
<td>$d$</td>
<td>Computes a normal basis for $\mathbb{F}_{q^d}$.</td>
</tr>
</tbody>
</table>

Table 3.7: More Finite Field Activities

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
Polynomial | Numerical | Relation
--- | --- | ---
$X^2 + X + 1$ | 7 | $\omega^2 = \omega + 1$
$X^3 + X^2 + 1$ | 13 | $\gamma^3 = \gamma^2 + 1$
$X^2 + X + 2$ | 14 | 
$X^4 + X^3 + 1$ | 25 | $\delta^4 = \delta^3 + 1$
$X^2 + X + 2$ | 22 | 
$X^3 + 2X + 1$ | 34 | 
$X^5 + X^2 + 1$ | 37 | $\eta^5 = \eta^2 + 1$
$X^2 + X + 3$ | 59 | 
$X^6 + X^5 + 1$ | 97 | 
$X^4 + X^3 + 2$ | 110 | 
$X^2 + 4X + 2$ | 167 | 
$X^3 + X^2 + X + 2$ | 86 | 
$X^7 + X^6 + 1$ | 193 | $\zeta^7 = \zeta^6 + 1$
$X^2 + X + 2$ | 184 | 
$X^5 + 2X + 1$ | 250 | 
$X^8 + X^4 + X^3 + X^2 + 1$ | 285 | 
$X^2 + X + 3$ | 309 | 
$X^3 + 3X + 2$ | 366 | 
$X^2 + X + 2$ | 382 | 
$X^9 + X^4 + 1$ | 529 | 
$X^2 + 2X + 5$ | 580 | 
$X^4 + X^3 + X + 2$ | 326 | 
$X^6 + X^5 + 2$ | 974 | 
$X^2 + 5X + 2$ | 988 | 
$X^2 + 2X + 3$ | 1026 | 
$X^{10} + X^3 + 1$ | 1033 | 

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

$$F_4:$$

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

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

$\$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
```

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: $X^4 + X^3 + 1 = 25$

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

**Subfields:**

<table>
<thead>
<tr>
<th>Subfield</th>
<th>Polynomial</th>
<th>Numerical Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_4$</td>
<td>$X^2 + X + 1$</td>
<td>7</td>
</tr>
</tbody>
</table>

<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>
<th>$\phi(\gamma_i)$</th>
<th>$T(\gamma_i)$</th>
<th>$N(\gamma_i)$</th>
<th>$T_2(\gamma_i)$</th>
<th>$N_2(\gamma_i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>DNE</td>
<td>1</td>
<td>DNE</td>
<td>1</td>
<td>0</td>
<td>1</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>12</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>$\alpha = \delta$</td>
<td>2</td>
<td>12</td>
<td>4</td>
<td>1</td>
<td>11</td>
<td>11</td>
<td></td>
<td></td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td>$\alpha + 1 = \delta^{12}$</td>
<td>3</td>
<td>8</td>
<td>4</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>$\alpha^2 = \delta^2$</td>
<td>4</td>
<td>6</td>
<td>3</td>
<td>9</td>
<td>1</td>
<td>11</td>
<td></td>
<td></td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>$\alpha^2 + 1 = \delta^9$</td>
<td>5</td>
<td>15</td>
<td>10</td>
<td>8</td>
<td>1</td>
<td>10</td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>$\alpha^2 + \alpha = \delta^{13}$</td>
<td>6</td>
<td>13</td>
<td>13</td>
<td>8</td>
<td>1</td>
<td>11</td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>$\alpha^2 + \alpha + 1 = \delta^{7}$</td>
<td>7</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>11</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>$\alpha^3 = \delta^3$</td>
<td>8</td>
<td>3</td>
<td>14</td>
<td>6</td>
<td>1</td>
<td>11</td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>$\alpha^3 + 1 = \delta^{14}$</td>
<td>9</td>
<td>13</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>11</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>$\alpha^3 + \alpha = \delta^{10}$</td>
<td>10</td>
<td>11</td>
<td>10</td>
<td>5</td>
<td>11</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>11</td>
<td>$\alpha^3 + \alpha + 1 = \delta^5$</td>
<td>11</td>
<td>10</td>
<td>13</td>
<td>14</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>$\alpha^3 + \alpha^2 = \delta^{14}$</td>
<td>12</td>
<td>2</td>
<td>14</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>10</td>
<td>1</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>1</td>
<td>1</td>
<td>1</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>$\alpha^3 + \alpha^2 + \alpha = \delta^8$</td>
<td>14</td>
<td>7</td>
<td>12</td>
<td>11</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>10</td>
<td>10</td>
<td>1</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>10</td>
<td>1</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 3.5: 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 3.5, 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 3.6.
Figure 3.6: Addition and multiplication table of $\mathbb{F}_9$ using the cyclic ordering of elements
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Description</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 $F_q$</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_</td>
<td></td>
<td>Normalize the result of -RREF or nullspace from the right</td>
</tr>
<tr>
<td>the_right</td>
<td>-RREF</td>
<td></td>
</tr>
<tr>
<td>-normalize_from_</td>
<td></td>
<td>Normalize the result of -RREF or nullspace from the left</td>
</tr>
<tr>
<td>the_left</td>
<td>-nullspace</td>
<td></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 $F_q$</td>
</tr>
<tr>
<td>-all_rational</td>
<td></td>
<td>Produces a report of all rational normal forms of endomorphisms of $F_q^d$</td>
</tr>
<tr>
<td>_normal_forms</td>
<td>$d$</td>
<td></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 $F_2$. The output is the matrix

54
The -RREF command produces a latex log of the steps. This can be used to follow the algorithm along. For a somewhat longer example, consider the Vandermonde matrix over the field $\mathbb{F}_7$. Suppose we want to compute the inverse matrix directly. We can use the following command to do so. Notice how we first create the matrix and an identity matrix next to it. After that we apply the -RREF command:

\begin{verbatim}
RREF_V7:
$\langle$ORBITER\rangle -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
\end{verbatim}

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

\begin{verbatim}
A matrix over the field $\mathbb{F}_7$
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 2 & 4 & 1 & 2 & 4 & 1 & 0 & 0 & 1 & 0 & 0 \\
1 & 3 & 2 & 6 & 4 & 5 & 1 & 0 & 0 & 0 & 1 & 0 \\
1 & 4 & 2 & 1 & 4 & 2 & 1 & 0 & 0 & 0 & 0 & 1 \\
1 & 5 & 4 & 6 & 2 & 3 & 1 & 0 & 0 & 0 & 0 & 1 \\
1 & 6 & 1 & 6 & 1 & 6 & 1 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\end{verbatim}

Position $(i, j) = (0, 0)$, found pivot in column 0
After making pivot 1:

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

After elimination below pivot:

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

Position \((i, j) = (1, 1)\), found pivot in column 1

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

After making pivot 1:
After elimination below pivot:

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

After elimination below pivot:

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

Position \((i, j) = (2, 2)\), found pivot in column 2

After making pivot 1:

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

After elimination below pivot:

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

After elimination below pivot:
Position \((i, j) = (3, 3)\), found pivot in column 3

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

After making pivot 1:

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

After elimination below pivot:

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

Position \((i, j) = (4, 4)\), found pivot in column 4
After making pivot 1:

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

After elimination below pivot:

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

Position \((i,j) = (5,5)\), found pivot in column 5

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

After making pivot 1:

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

59
After elimination below pivot:

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

Position \((i, j) = (6, 6)\), found pivot in column 6

After making pivot 1:

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

After elimination below pivot:
Did not find pivot. The rank of the matrix is 7.

After elimination above pivot 6 in position (6,6):

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

After elimination above pivot 5 in position (5,5):

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

After elimination above pivot 4 in position (4,4):

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

After elimination above pivot 3 in position (3,3):
After elimination above pivot 2 in position (2,2):

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

After elimination above pivot 1 in position (1,1):

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

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

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

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

Another task is computing the nullspace of a matrix. The command
nullspace:
▷ $(ORBITER) -v 2 \\
▷ ▷ -define F2 -finite_field -q 2 -end \\
▷ ▷ -define v -vector -field F2 -format 2 \\
▷ ▷ ▷ -dense "1,1,1,0,1,0,1,0,1" \\
▷ ▷ -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

classes_GL_3_2:
▷ $(ORBITER) -v 7 \\

63
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].

### Conjugacy Classes of $GL(3, 2)$

The number of conjugacy classes of $GL(3, 2)$ is 6:

- Class 0 / 6
  
  $3, 1, 0$
  
  | $\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ |
  | $\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}$ |
  | $\begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ |
  | $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ |

  centralizer order 7
  class size 24

- Class 1 / 6
  
  $2, 1, 0$
  
  | $\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}$ |

  centralizer order 7
  class size 24

- Class 2 / 6

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

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

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 \ 
▷ \ -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

$$
\begin{align*}
b_1 &= 1 + X + X^2, \\
b_2 &= X, \\
b_3 &= X^2.
\end{align*}
$$

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

$$
\begin{align*}
b_1^\Phi &= (1 + X + X^2)^2 = 1 + X^2 + X^4 = 1 + X^2 + X^3 + X = 1 + X + X^2 + X^2 + 1 = X = b_2, \\
b_2^\Phi &= X^2 = b_3, \\
b_3^\Phi &= X^4 = X^3 + X = X^2 + X + 1 = b_1.
\end{align*}
$$

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 field chosen basis.
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-write_code_for_division</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>-polynomial_division</td>
<td>A B</td>
<td>Divides polynomial B by polynomial A.</td>
</tr>
<tr>
<td>-extended_gcd_for_polynomials</td>
<td>A B</td>
<td>Computes the extended gcd of polynomials A and B.</td>
</tr>
<tr>
<td>-polynomial_mult_mod</td>
<td>A B M</td>
<td>Computes the product of polynomials A and B modulo the polynomial M.</td>
</tr>
<tr>
<td>-polynomial_power_mod</td>
<td>A N M</td>
<td>Computes the n-th power of the polynomial A modulo the polynomial M.</td>
</tr>
<tr>
<td>-Berlekamp_matrix</td>
<td>A</td>
<td>Compute the Berlekamp matrix associated with the polynomial A.</td>
</tr>
<tr>
<td>-normal_basis</td>
<td>d</td>
<td>Computes a normal basis for $\mathbb{F}_{q^d}$ over $\mathbb{F}_q$.</td>
</tr>
<tr>
<td>-polynomial_find_roots</td>
<td>A</td>
<td>Computes the roots of the polynomial A.</td>
</tr>
<tr>
<td>-nullspace</td>
<td>A</td>
<td>Computes the right nullspace of the matrix A.</td>
</tr>
<tr>
<td>-RREF</td>
<td>A</td>
<td>Computes the RREF of the matrix A.</td>
</tr>
<tr>
<td>-weight Enumerator</td>
<td>A</td>
<td>Computes the weight enumerator of the code whose generator matrix is A.</td>
</tr>
<tr>
<td>-Walsh_Hadamard_transform</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>-algebraic_normal_form</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>-apply_trace_function</td>
<td>fname</td>
<td>Applies the absolute trace function to the function in the given file.</td>
</tr>
<tr>
<td>-apply_power_function</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>-identity_function</td>
<td>fname_csv</td>
<td>Creates the identity function and stores it in the given csv file.</td>
</tr>
<tr>
<td>-Walsh_matrix</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>Description</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>$L1$ $L2$ $P$</td>
<td>Computes the unique transversal to the lines $L1$ and $L2$ through the point $P$ in 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>$L1$ $L2$</td>
<td>Computes the intersection of two lines in 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 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 PG($n, q$) from the Orbiter rank value $r$.</td>
</tr>
<tr>
<td>-inverse_isomorphism_klein_quadric</td>
<td>$L36$</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 3.7: The field reduction from \( \mathbb{F}_{64} \) to \( \mathbb{F}_8 \)

The output is shown in Figure 3.7. Note that the dimension of the vector space is 2, so the block matrices are 2 × 2. Observe that \( \mathbb{F}_{64} \) has many subfields. Figure 3.8 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 × 3 and 6 × 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.
Also, let γ be the generator of the subgroup of \( q - 1 \) th roots, which are the elements of the multiplicative group of \( \mathbb{F}_q \). The output lists the \( n \)-th roots first, generated by β. After that, the \( q - 1 \)th roots are shown, generated by γ. Finally, a table is produced which shows the irreducible polynomials over \( \mathbb{F}_q \) associated with the \( n \)-th roots of unity. For instance, the following command computes the minimum polynomials of all 21st roots of unity over \( \mathbb{F}_8 \):

\[
\text{F}_8\_\text{Nth\_roots\_21:}
\]

\[
\begin{array}{l}
\text{\texttt{\$(ORBITER) -v 3 \}} \\
\text{\texttt{\begin{verbatim}
\define F -finite_field -q 8 -override_polynomial 11 -end \\
\define with F -do -coding_theoretic_activity \\
\define \begin{verbatim}
\define -ninth Roots 21 \\
\define \end
\define \end
\define pdflatex Nth\_roots\_q8\_n21.tex \\
\define open Nth\_roots\_q8\_n21.pdf
\end{verbatim}
\end{verbatim}}
\end{array}
\]

The output is:

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

\[
\begin{align*}
\beta^0 &= 100000 = 1 \\
\beta^1 &= 000100 = \alpha^3 \\
\beta^2 &= 100001 = \alpha^5 + 1 \\
\beta^3 &= 111101 = \alpha^5 + \alpha^3 + \alpha^2 + \alpha + 1
\end{align*}
\]
\( \beta^4 = 011111 = \alpha^5 + \alpha^4 + \alpha^3 + \alpha^2 + \alpha \)
\( \beta^5 = 101010 = \alpha^4 + \alpha^2 + 1 \)
\( \beta^6 = 110100 = \alpha^3 + \alpha + 1 \)
\( \beta^7 = 100111 = \alpha^5 + \alpha^4 + \alpha^3 + 1 \)
\( \beta^8 = 101101 = \alpha^5 + \alpha^3 + \alpha^2 + 1 \)
\( \beta^9 = 011101 = \alpha^5 + \alpha^3 + \alpha^2 + \alpha \)
\( \beta^{10} = 011011 = \alpha^5 + \alpha^4 + \alpha^2 + \alpha \)
\( \beta^{11} = 001011 = \alpha^5 + \alpha^3 + \alpha^2 + 1 \)
\( \beta^{12} = 010100 = \alpha^3 + \alpha + 1 \)
\( \beta^{13} = 111000 = \alpha^2 + \alpha + 1 \)
\( \beta^{14} = 001111 = \alpha^5 + \alpha^4 + \alpha^3 \)
\( \beta^{15} = 101000 = \alpha^5 + \alpha^2 + 1 \)
\( \beta^{16} = 111010 = \alpha^3 + \alpha^2 + \alpha + 1 \)
\( \beta^{17} = 100110 = \alpha^4 + \alpha^3 + 1 \)
\( \beta^{18} = 010100 = \alpha^3 + \alpha \)
\( \beta^{19} = 100011 = \alpha^5 + \alpha^4 + 1 \)
\( \beta^{20} = 001100 = \alpha^3 + \alpha^2 \)

Let \( \gamma \) be a primitive 7-th root in GF(64), so \( \gamma = \alpha^9 \).
\( \gamma^0 = 100000 = 1 \)
\( \gamma^1 = 111101 = \alpha^5 + \alpha^3 + \alpha^2 + \alpha + 1 \)
\( \gamma^2 = 110100 = \alpha^3 + \alpha + 1 \)
\( \gamma^3 = 011101 = \alpha^5 + \alpha^3 + \alpha^2 + \alpha \)
\( \gamma^4 = 001001 = \alpha^5 + \alpha^2 \)
\( \gamma^5 = 101001 = \alpha^5 + \alpha^2 + 1 \)
\( \gamma^6 = 010100 = \alpha^3 + \alpha \)

The \( q \)-cyclotomic set for \( q = 8 \) are:
\{ 0 \}
\{ 1, 8 \}
\{ 2, 16 \}
\{ 3 \}
\{ 4, 11 \}
\{ 5, 19 \}
\{ 6 \}
\{ 7, 14 \}
\{ 9 \}
\{ 10, 17 \}
\{ 12 \}
\{ 13, 20 \}
\{ 15 \}
\{ 18 \}
Subfield basis, a basis for GF(8) inside GF(64):

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

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

<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>$(110100)X^0 + (100000)X^1$</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 $\mathbb{F}_7$. Let us do the same for the field $\mathbb{F}_8$ instead. We use the following command:

`F_8.vandermonde:
$\text{(ORBITER) -v 3 \
-define F -finite_field -q 8 -end \n72`
The output is shown below. Again, the first matrix is $V = (x_i^j)$. The second matrix is $V^{-1}$:

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

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

```bash
F_1024.vandermonde:
 $\$(ORBITER) -v 3 \\
 $\$ -define F -finite_field -q 1024 -end \\
 $\$ -with F -do -finite_field_activity \\
 $\$ -Vandermonde_matrix \\
 $\$ -end
```

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. Because compiling code is a bit more complicated, additional makefile options are necessary. Suppose we want to create the number theoretic transform for the 16th roots of unity inside the field $\mathbb{F}_{17}$. Here is the command to generate the Orbiter source code:

```bash
NTT_k4_q17.cpp:
 $\$(ORBITER) -v 3 \\
 $\$ -define F -finite_field -q 17 -end \\
```
This produces a C++ file `NTT_k4_q17.cpp`. This file should be compiled and linked against the Orbiter library. Because of this, we define the following makefile variables at the top of the makefile.

The command

```make
F_17_NTT_compile: NTT_k4_q17.cpp
  $(MY_CPP) NTT_k4_q17.cpp $(CPPFLAGS) \n  $(LIB) $(LFLAGS) -o NTT_k4_q17.out
  ./NTT_k4_q17.out
```

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

Orbiter can deal with multivariate polynomial rings with coefficients over finite fields. Orbiter creates the 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:

\begin{verbatim}
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
\end{verbatim}

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 $F_2$. 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:

\[
\text{PG}_3.2_{\text{easy}}:
\]
\[
\begin{array}{c}
\text{\texttt{ORBITER)} -v 2 \\
\text{\texttt{-define F -finite_field -q 2 -end}} \\
\text{\texttt{-define P -projective_space -n 3 -field F -end}}
\end{array}
\]

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:

\[
\text{PG}_2.4:
\]
\[
\begin{array}{c}
\text{\texttt{ORBITER)} -v 2 \\
\text{\texttt{-define F -finite_field -q 4 -end}} \\
\text{\texttt{-define P -projective_space -n 2 -field F -v 0 -end}}
\end{array}
\]

77
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.

The plane 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:

\( P_0 = (1, 0, 0) \)
\( P_2 = (0, 0, 1) \)
\( P_4 = (1, 1, 0) \)
\( P_{10} = (0, 1, 1) \)
\( P_1 = (0, 1, 0) \)
\( P_3 = (1, 1, 1) \)
\( P_7 = (1, 0, 1) \)
\( P_{12} = (1, 0, 1) \)

There are 7 elements in the Baer subgeometry.

Normalized from the left:

\( P_0 = (1, 0, 0) \)
\( P_2 = (0, 0, 1) \)
\( P_4 = (1, 1, 0) \)
\( P_{10} = (0, 1, 1) \)
\( P_1 = (0, 1, 0) \)
\( P_3 = (1, 1, 1) \)
\( P_7 = (1, 0, 1) \)
\( P_{12} = (1, 0, 1) \)

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 PG(3, 2).

```
PG_3.2:
▷ $(\text{ORBITER}) -v 2 \$
▷ ▷ -define F -finite_field -q 2 -end \$
▷ ▷ -define P -projective_space -n 3 -field F -v 0 -end \$
▷ ▷ -with P -do -projective_space_activity \$
▷ ▷ ▷ -cheat_sheet \$
▷ ▷ -end
▷ pdflatex PG_3.2.tex
▷ open PG_3.2.pdf
```

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

**The projective space PG(3, 2)**

\[
q = 2
\]
\[
p = 2
\]
\[
e = 1
\]
\[
n = 3
\]
Number of points = 15
The points of PG(3, 2)

PG(3, 2) has 15 points:

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

Normalized from the left:

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

The lines of PG(3, 2)

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

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

Lines sorted by Pluecker coordinates

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

\[ 1 = \text{Pl}(0, 1, 0, 0, 0, 0) = L_{34} = \begin{bmatrix} 0010 \\ 0001 \end{bmatrix} \]

\[ 2 = \text{Pl}(0, 0, 1, 0, 0, 0) = L_4 = \begin{bmatrix} 1000 \\ 0010 \end{bmatrix} \]

\[ 3 = \text{Pl}(0, 0, 0, 1, 0, 0) = L_{30} = \begin{bmatrix} 0100 \\ 0001 \end{bmatrix} \]

\[ 4 = \text{Pl}(0, 0, 0, 0, 1, 0) = L_6 = \begin{bmatrix} 1000 \\ 0001 \end{bmatrix} \]

\[ 5 = \text{Pl}(0, 0, 0, 0, 0, 1) = L_{28} = \begin{bmatrix} 0100 \\ 0010 \end{bmatrix} \]

\[ \vdots \]

\[ 34 = \text{Pl}(0, 1, 1, 1, 1, 1) = L_{26} = \begin{bmatrix} 1101 \\ 0011 \end{bmatrix} \]

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

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

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

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

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

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

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

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

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

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

$\vdots$

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

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

<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 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 \(x \in \mathbb{F}_q\), \textsc{Rank}(\mathbb{F}_q, x) is a number \(b\) in \(\mathbb{Z}_q\). Also, for \(b \in \mathbb{Z}_q\), we assume that \textsc{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 \(\text{PG}(2,4)\):

```
P_2_4_rank_point:
> $(ORBITER) -v 2 \n> -define F -finite_field -q 4 -end \n> -with F -do -finite_field_activity \n> -rank_point_in_PG 2 "3,3,1" -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 \leftarrow \max\{j \in \mathbb{Z}_n \mid x_j \neq 0\}
8: x \leftarrow \frac{1}{x_i} x
9: a := 0
10: for j = i − 1, ..., 1, 0 do
11: a \leftarrow a + \text{Rank}(\mathbb{F}_q, x_j)
12: if j > 0 then
13: a \leftarrow a \cdot q
14: if i = n − 1 and a ≥ \sum_{j=0}^{i-1} q^j then
15: a \leftarrow a − 1
16: a \leftarrow a + n − i + \sum_{j=0}^{i-1} q^j
17: return a

\[
\begin{array}{|c|c|}
\hline
a = \text{Rank}(x) & x = \text{Unrank}(a) \\
\hline
0 & (1, 0, 0) \\
1 & (0, 1, 0) \\
2 & (0, 0, 1) \\
3 & (1, 1, 1) \\
4 & (1, 1, 0) \\
5 & (1, 0, 1) \\
6 & (0, 1, 1) \\
\hline
\end{array}
\]

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 ← a − n
6:   if $a = 0$ then
7:     return 1
8:   a ← a − 1
9:   x ← 0
10:   for $i = 1, \ldots, n-1$ do
11:       if $a \geq \sum_{j=1}^{i-1} q^j$ then
12:           a ← a − $\sum_{j=1}^{i-1} q^j$
13:       else
14:           $x_i$ ← 1
15:           break
16:   for $k = i + 1, \ldots, n-1$ do
17:       $x_k$ ← 0
18:   a ← a + 1
19:   if $i = n-1$ and $a \geq \sum_{j=0}^{i-1} q^j$ then
20:       a ← a + 1
21:   j ← 0
22:   while $a > 0$ do
23:       r ← a mod q
24:       $x_j$ ← Unrank($\mathbb{F}_q, r$)
25:       j ← j + 1
26:       a ← $(a - r)/q$
27:   for $h = j, \ldots, i - 1$ do
28:       $x_h$ ← 0
29:   return x
The rank turns out to be 20.

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.

PG_2_8.incidence_matrix:
```
$(ORBITER) -v 2 \
  -define F -finite_field -q 8 -end \n  -define P -projective_space -n 2 -field F -v 0 -end \n  -with P -do -projective_space_activity \n  -export_point_line_incidence_matrix \n  -end 
$(ORBITER) -v 2 \
  -define all_one -vector -repeat 1 73 -end \n  -draw_matrix \n  -input_csv_file PG_n2_q8.incidence_matrix.csv \n  -box_width 20 -bit_depth 8 \n  -partition 3 \n  -all_one all_one \n  -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_216:  
  $\$(ORBITER) -v 2 \n  \( -\text{draw options} -\text{xin 20000} -\text{yin 20000} \n  \( -\text{radius 200} -\text{line width 0.3} -\text{nodes empty} -\text{end} \n  \( -\text{define F -finite field -q 16 -end} \n  \( -\text{define P -projective space -n 2 -field F -v 0 -end} \n  \( -\text{with P -do -projective space activity} \n  \( -\text{-cheat sheet} \n  \( -\text{-end} \n  pdflatex PG_216.tex  
  open PG_216.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.

Projective spaces has 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_24_with_decomposition:  
  $\$(ORBITER) -v 2 \n  \( -\text{define F -finite field -q 4 -end} \n  \( -\text{define P -projective space -n 2 -field F -v 0 -end} \n  \( -\text{with P -do -projective space activity} \n  \( -\text{-cheat sheet for decomposition by element PG} \n  \( -\text{-1 "0,1,0, 0,0,1, 2,1,1, 0"} \n  \( -\text{-"PG_24_singer"} \n  \( -\text{-end} \n  pdflatex PG_24_singer.tex  
  open PG_24_singer.pdf
```
Considering the cyclic group generated by

\[
\begin{bmatrix}
0 & 1 & 0 \\
0 & 0 & 1 \\
\omega & 1 & 1
\end{bmatrix}_0 \cdot \begin{bmatrix}
010 \\
001 \\
211
\end{bmatrix}_0
\]

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:
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 \n  -list_arguments \n  -define R -vector -repeat 1 21 -end \n  -define C -vector -repeat 1 21 -end \n  -draw_matrix \n  -input_csv_file PG_2_4_singer_incma_cyclic.csv \n  -box_width 40 -bit_depth 24 \n  -partition 3 R C \n  -end
open PG_2_4_singer_incma_cyclic_draw.bmp
```

The cyclic incidence matrix is shown in Figure 4.4.
Figure 4.5: Tactical decomposition of the incidence matrix of PG(2, 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 \
  -define R -vector -repeat 3 7 -end \
  -define C -vector -repeat 3 7 -end \
  -draw_matrix \
  -input_csv_file PG_2_4_singer_incma_subgroup_index_3.csv \
  -box_width 40 -bit_depth 24 \
  -partition 3 R C \
  -end \
open PG_2_4_singer_incma_subgroup_index_3_draw.bmp
```

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

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

$$\left[ \begin{array}{c} n \\ k \end{array} \right]_q = \frac{(q^n - 1)(q^{n-1} - 1)\cdots(q^{n-k+1} - 1)}{(q^k - 1)(q^{k-1} - 1)\cdots(q - 1)},$$

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:
▷ $(\text{ORBITER})$ -v 2 \
▷ ▷ -define F -finite_field -q 2 -end \
▷ ▷ -with F -do -finite_field_activity \
▷ ▷ ▷ -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$(2,2)$:

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

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

The following produces a list of planes through a line. In the example, the line is 0.

planes_in_pencil:
  ▶ $(ORBITER) -v 2 \\
  ▶ ▶ -define F -finite_field -q 8 -end \\
  ▶ ▶ -define P -projective_space -n 3 -field F -v 0 -end \\
  ▶ ▶ -with P -do \\
  ▶ ▶ -projective_space_activity \\
  ▶ ▶ ▶ -planes_through_line 0 \\
  ▶ ▶ -end
Table 4.2: The partition ordering of monomials of degree 1, 2, 3 and 4 in a plane

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 $(n+1)(q-1) \cdot 30$. The associated polynomial is unique and known as the algebraic normal form of the set.

In order to work with algebraic sets, polynomial rings are required. Orbiter offers homogeneous polynomials in a finite number of variables. There are two orderings of the monomials which can be chosen. 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. The lexicographic ordering orders the monomials lexicographically. Table 4.2 shows the monomials in the partition ordering for degrees 1, 2, 3 and 4 in a plane. 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 4.2, 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:

\[
\begin{align*}
\texttt{\$(ORBITER) -v 2 \}\"n
\texttt{-define F -finite_field -q 11 -end}\"n
\texttt{-define R -polynomial_ring -field F}\"n
\texttt{-number_of_variables 3}\"n
\texttt{-homogeneous_of_degree 3}\"n
\texttt{-end}\"n
\texttt{-define P -projective_space -n 2 -field F -v 0 -end}\"n
\texttt{-define EC -geometric_object P}\"n
\texttt{-projective_variety R}\"n
\texttt{-define "EC_11" "EC\_11"}\"n
\texttt{-define $(EC_11_EQUATION)}\"n
\texttt{-end}\"n
\texttt{-with EC -do -combinatorial_object_activity -save}\"n
\texttt{-end}\n\end{align*}
\]

creates the algebraic set associated to the cubic curve \( y^2 = x^3 + x + 3 \) in 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

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

Table 4.3 shows the Orbiter monomial orderings for degrees 2 and 3 in PG(3,q). Based on the partition ordering, the equation is coded as coefficient vector

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

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

HIRSCHFELD_SURFACE_EQUATION="0,0,0,0,0,0,1,0,1,0,0,1,0,1,0,0,0,0,0,0,0,0"
Table 4.3: The Orbiter ordering of monomials of degree 1, 2 and 3 in PG(3, q)

<table>
<thead>
<tr>
<th>h</th>
<th>mon</th>
<th>vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$X_0^3$</td>
<td>(3, 0, 0, 0)</td>
</tr>
<tr>
<td>1</td>
<td>$X_1^3$</td>
<td>(0, 3, 0, 0)</td>
</tr>
<tr>
<td>2</td>
<td>$X_2^3$</td>
<td>(0, 0, 3, 0)</td>
</tr>
<tr>
<td>3</td>
<td>$X_3^3$</td>
<td>(0, 0, 0, 3)</td>
</tr>
<tr>
<td>4</td>
<td>$X_0^2 X_1$</td>
<td>(2, 1, 0, 0)</td>
</tr>
<tr>
<td>5</td>
<td>$X_0^2 X_2$</td>
<td>(2, 0, 1, 0)</td>
</tr>
<tr>
<td>6</td>
<td>$X_0^2 X_3$</td>
<td>(2, 0, 0, 1)</td>
</tr>
<tr>
<td>7</td>
<td>$X_0 X_1^2$</td>
<td>(1, 2, 0, 0)</td>
</tr>
<tr>
<td>8</td>
<td>$X_1^2 X_2$</td>
<td>(0, 2, 1, 0)</td>
</tr>
<tr>
<td>9</td>
<td>$X_1^2 X_3$</td>
<td>(0, 2, 0, 1)</td>
</tr>
<tr>
<td>10</td>
<td>$X_1 X_2^2$</td>
<td>(1, 0, 2, 0)</td>
</tr>
<tr>
<td>11</td>
<td>$X_1 X_3^2$</td>
<td>(0, 1, 2, 0)</td>
</tr>
<tr>
<td>12</td>
<td>$X_2 X_3^2$</td>
<td>(0, 0, 2, 1)</td>
</tr>
<tr>
<td>13</td>
<td>$X_2 X_3$</td>
<td>(1, 0, 0, 2)</td>
</tr>
<tr>
<td>14</td>
<td>$X_1 X_3$</td>
<td>(0, 1, 0, 2)</td>
</tr>
<tr>
<td>15</td>
<td>$X_2 X_3$</td>
<td>(0, 0, 1, 2)</td>
</tr>
<tr>
<td>16</td>
<td>$X_0 X_1 X_2$</td>
<td>(1, 1, 1, 0)</td>
</tr>
<tr>
<td>17</td>
<td>$X_0 X_1 X_3$</td>
<td>(1, 1, 0, 1)</td>
</tr>
<tr>
<td>18</td>
<td>$X_0 X_2 X_3$</td>
<td>(1, 0, 1, 1)</td>
</tr>
<tr>
<td>19</td>
<td>$X_1 X_2 X_3$</td>
<td>(0, 1, 1, 1)</td>
</tr>
</tbody>
</table>
Figure 4.6: Elliptic curve $y^2 \equiv x^3 + x + 3 \mod 11$

```plaintext
- homogeneous_of_degree 3 \\
- end \\
- define P -projective_space -n 3 -field F -v 0 -end \\
- define H4 -geometric_object P \\
- projective_variety R \\
- define Hirschfeld_surface_q4" \\
- "Hirschfeld\_surface\_q4" \\
- $(HIRSCHFELD\_SURFACE\_EQUATION) \\
- -end \\
- with H4 -do -combinatorial_object_activity -save \\
- end
```

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

The next command creates the Endrass surface over $\mathbb{F}_7$. The surface is defined as a makefile variable in sparse form.

```plaintext
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:
Suppose we want to create the monomials of degree 8 in 4 variables. We use an diophantine system to do so. The following command creates the system and solves it. After that, it applies the unix sort command to sort the monomials:

\[
\begin{align*}
\text{octic\_prepare:} \\
&\text{\$ (ORBITER) -v 4 \ } \\
&\quad \text{-define A -vector -format 1 -dense "1,1,1,1" -end \ } \\
&\quad \text{-define D -diophant \ } \\
&\quad \quad \text{-label octic\_monomials \ } \\
&\quad \quad \text{-coefficient\_matrix A \ } \\
&\quad \quad \text{-RHS "8,8,1" \ } \\
&\quad \quad \text{-x\_min\_global 0 -x\_max\_global 8 \ } \\
&\quad \quad \text{-end \ } \\
&\quad \quad \text{-with D -do \ } \\
&\quad \quad \text{-diophant\_activity -solve\_mckay \ } \\
&\quad \quad \text{-end} \\
&\text{sort -r octic\_monomials.sol >octic\_monomials\_sorted.txt }
\end{align*}
\]

There are 165 monomials. They are listed in the file octic\_monomials\_sorted.txt.
4.6 The Klein Quadric and the Plücker Map

Orbiter can work with Grassmannians over finite field. In particular, Orbiter offers indexing for these sets. For the Grassmannian $\mathcal{G}r_{4,2}(V)$, Plücker coordinates can be used to identify $\mathcal{G}r_{4,2}(V)$ with the $Q^+(5, q)$ (Klein) quadric.

The command

```bash
GR_4_2_2:
  $\text{ORBITER} -v 2 \$
  $\text{-define F -finite_field -q 2 -end}$
  $\text{-with F -do -finite_field_activity}$
  $\text{-cheat_sheet Gr 4 2 -end}$
  `pdflatex Gr_4_2_2.tex`
  `open Gr_4_2_2.pdf`
```

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

```
There are 35 lines:

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

The Plücker coordinates satisfy

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

and hence belong to the Klein quadric $Q^+(5, q)$. Orthogonal spaces and quadrics will be discussed in Section 4.7. Orbiter has a labeling of points of quadrics that can be used to enumerate the points of $Q^+(5, q)$. Using the inverse Plücker map, this gives a second way to label the lines of $PG(3, q)$. In the example of $PG(3, 2)$ this yields the following list (output shortened):

```
```

100
0 = \Pi(1, 0, 0, 0, 0) = L_0 = \begin{bmatrix} 1000 \\ 0100 \end{bmatrix} \\
1 = \Pi(0, 1, 0, 0, 0) = L_{34} = \begin{bmatrix} 0010 \\ 0001 \end{bmatrix} \\
2 = \Pi(0, 0, 1, 0, 0) = L_4 = \begin{bmatrix} 1000 \\ 0010 \end{bmatrix} \\
\vdots \\
34 = \Pi(0, 1, 1, 1, 1) = L_{26} = \begin{bmatrix} 1101 \\ 0011 \end{bmatrix}
Table 4.4: 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>$Q^+(n, q)$</td>
<td>$\sum_{i=0}^{\frac{n-1}{2}} X_{2i}X_{2i+1}$</td>
<td>$(q^{(n+1)/2} - 1)(q^{(n-1)/2} + 1)$</td>
</tr>
<tr>
<td>Hyperbolic (n is odd)</td>
<td></td>
<td>$q - 1$</td>
</tr>
<tr>
<td>$Q^-(n, q)$</td>
<td>$p(X_{n-1}, X_n) + \sum_{i=0}^{\frac{n-1}{2}-1} X_{2i}X_{2i+1}$</td>
<td>$(q^{(n+1)/2} + 1)(q^{(n-1)/2} - 1)$</td>
</tr>
<tr>
<td>Elliptic (n is odd)</td>
<td></td>
<td>$q - 1$</td>
</tr>
<tr>
<td>$Q(n, q)$</td>
<td>$X_n^2 + \sum_{i=0}^{\frac{n}{2}-1} X_{2i}X_{2i+1}$</td>
<td>$q^n - 1$</td>
</tr>
<tr>
<td>Parabolic (n is even)</td>
<td></td>
<td>$q - 1$</td>
</tr>
</tbody>
</table>

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 4.4. Here, $p(X, Y) = c_1 X^2 + c_2 XY + c_3 Y^2 \in \mathbb{F}_q[X, Y]$ 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

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

In order to create an object of type orthogonal space, the -orthogonal_space command is used inside a -definition .. -end command sequence. In Table 4.5, Orbiter command options for creating orthogonal spaces are shown.

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

Op.4.2:

```
-define F -finite_field -q 2 -end 
-define O -orthogonal_space 1 4 F -without_group -end 
-with O -do -orthogonal_space_activity 
-cheat_sheet_orthogonal -end
```

```
pdflatex 0.1_4_2_report.tex
open 0.1_4_2_report.pdf
```
The next command creates $Q(4, 2)$ together with its group $PGO(5, 2)$. There are 15 points and 15 lines. The geometry is a configuration $15_3$ which is also known as the Cremona-Richmond configuration.

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 O -do -orthogonal_space_activity 
   -export_point_line_incidence_matrix 
   -end
$ORBITER -v 2 
   -define all_one_r -vector -repeat 1 15 -end 
   -define all_one_c -vector -repeat 1 15 -end 
   -draw_matrix 
   -input_csv_file 0.5.2_incidence_matrix.csv 
   -box_width 20 -bit_depth 8 
   -partition 2 
   -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 0 -orthogonal_space 1 6 2 -end
```
creates an object $O$ of type $Q^+(5,2)$. In Table 4.6, Orbiter activities for orthogonal spaces are shown.

The command

\[
\text{Op}_6.2:
\]
\[
\begin{array}{l}
\text{\texttt{\$(ORBITER) \ -v \ 2 \ \}}}\\
\text{\texttt{\ -define F \ -finite_field \ -q \ 2 \ -end}}\\n\text{\texttt{\ -define O \ -orthogonal_space \ 1 \ 6 \ F \ -without_group \ -end}}\\n\text{\texttt{\ -with O \ -do \ -orthogonal_space_activity}}\\n\text{\texttt{\ -cheat_sheet_orthogonal \ -end}}\\n\text{\texttt{\ pdflatex 0_1_6_2_report.tex}}\\n\text{\texttt{\ open 0_1_6_2_report.pdf}}
\end{array}
\]

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):
Table 4.6: Activities related to orthogonal spaces

<table>
<thead>
<tr>
<th>Modifier</th>
<th>Arguments</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-cheat_sheet_orthogonal</td>
<td></td>
<td>Create a latex report of the orthogonal space. If the group has been</td>
</tr>
<tr>
<td></td>
<td></td>
<td>created, the report will contain information about the group also.</td>
</tr>
<tr>
<td>-unrank_line_through_two_points</td>
<td>p1 p2</td>
<td>Determine the rank of the line through p1 and p2.</td>
</tr>
<tr>
<td>-perp</td>
<td>L</td>
<td>Determine the common perp of a set of points. The point ranks are given</td>
</tr>
<tr>
<td></td>
<td></td>
<td>in the list L.</td>
</tr>
<tr>
<td>-create_BLT_set descr</td>
<td></td>
<td>Creates a BLT-set of Q(4, q). See Section 12.4.</td>
</tr>
</tbody>
</table>

The number of points is 35 points:

- \( P_0 = (1,0,0,0,0,0) \)
- \( P_1 = (0,1,0,0,0,0) \)
- \( P_2 = (0,0,1,0,0,0) \)
- \( P_3 = (1,0,1,0,0,0) \)
- \( P_4 = (0,1,1,0,0,0) \)
- \( P_5 = (0,0,0,1,0,0) \)
- \( P_6 = (1,0,0,1,0,0) \)
\[P_7 = (0, 1, 0, 1, 0, 0)\]
\[P_8 = (1, 1, 1, 0, 1, 0)\]
\[P_9 = (0, 0, 0, 0, 1, 0)\]
\[P_{10} = (1, 0, 0, 0, 1, 0)\]
\[P_{11} = (0, 1, 0, 0, 1, 0)\]
\[P_{12} = (0, 0, 1, 0, 1, 0)\]
\[P_{13} = (1, 0, 1, 0, 1, 0)\]
\[P_{14} = (0, 1, 1, 0, 1, 0)\]
\[P_{15} = (0, 0, 0, 1, 1, 0)\]
\[P_{16} = (1, 0, 0, 1, 1, 0)\]
\[P_{17} = (0, 1, 0, 1, 1, 0)\]
\[P_{18} = (1, 1, 1, 1, 1, 0)\]
\[P_{19} = (0, 0, 0, 0, 0, 1)\]
\[P_{20} = (1, 0, 0, 0, 0, 1)\]
\[P_{21} = (0, 1, 0, 0, 0, 1)\]
\[P_{22} = (0, 0, 1, 0, 0, 1)\]
\[P_{23} = (1, 0, 1, 0, 0, 1)\]
\[P_{24} = (0, 1, 1, 0, 0, 1)\]
\[P_{25} = (0, 0, 0, 1, 0, 1)\]
\[P_{26} = (1, 0, 0, 1, 0, 1)\]
\[P_{27} = (0, 1, 0, 1, 0, 1)\]
\[P_{28} = (1, 1, 1, 1, 0, 1)\]
\[P_{29} = (1, 1, 0, 0, 1, 1)\]
\[P_{30} = (1, 1, 1, 0, 1, 1)\]
\[P_{31} = (1, 1, 0, 1, 1, 1)\]
\[P_{32} = (0, 0, 1, 1, 1, 1)\]
\[P_{33} = (1, 0, 1, 1, 1, 1)\]
\[P_{34} = (0, 1, 1, 1, 1, 1)\]

The number of lines is 105

\[L_0 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ \end{bmatrix} \{P_0, P_{32}, P_{33}\}\]
\[L_1 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ \end{bmatrix} \{P_1, P_{32}, P_{34}\}\]
\[L_{104} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ \end{bmatrix} \{P_8, P_9, P_{18}\}\]

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

The command

```
Op_6_64_line_rank:
```

```
$(ORBITER) -v 4 \
```

106
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 \n  > -define F -finite_field -q 64 -end \n  > -define O -orthogonal_space 1 6 F -without_group -end \n  > -with O -do -orthogonal_space_activity \n  > -unrank_line_through_two_points 15447347 15225451 \n  > -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>0</td>
<td>4225</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4225</td>
</tr>
</tbody>
</table>

107
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.
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:
  $\text{ORBITER} -v 2 \\n  \text{-define F -finite_field -q 4 -end} \\n  \text{-with F -do -finite_field_activity} \\n  \text{-cheat_sheet_hermitian 2 -end} \\n  \text{pdflatex H_2.4.tex} \\n  \text{open H_2.4.pdf}
```

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

The Hermitian variety $H(2, 4)$ contains 9 points:

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

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

The command

```
H_3.4:
  $\text{ORBITER} -v 2 \\n  \text{-define F -finite_field -q 4 -end} \\n  \text{-with F -do -finite_field_activity} \\n  \text{-cheat_sheet_hermitian 3 -end} \\n  \text{pdflatex H_3.4.tex} \\n  \text{open H_3.4.pdf}
```

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

109
The Hermitian variety $H(3, 4)$ contains 45 points:

\[ P_0 = (1, 1, 0, 0) = 5 \]
\[ P_1 = (2, 1, 0, 0) = 6 \]
\[ P_2 = (3, 1, 0, 0) = 7 \]
\[ P_3 = (1, 0, 1, 0) = 8 \]
\[ P_4 = (2, 0, 1, 0) = 9 \]
\[ P_5 = (3, 0, 1, 0) = 10 \]
\[ P_6 = (0, 1, 1, 0) = 11 \]
\[ P_7 = (0, 2, 1, 0) = 15 \]
\[ P_8 = (0, 3, 1, 0) = 19 \]
\[ P_9 = (1, 0, 0, 1) = 23 \]
\[ P_{10} = (2, 0, 0, 1) = 24 \]
\[ P_{11} = (3, 0, 0, 1) = 25 \]
\[ P_{12} = (0, 1, 0, 1) = 26 \]
\[ P_{13} = (0, 2, 0, 1) = 30 \]
\[ P_{14} = (0, 3, 0, 1) = 34 \]
\[ P_{15} = (1, 1, 1, 1) = 4 \]
\[ P_{16} = (2, 1, 1, 1) = 43 \]
\[ P_{17} = (3, 1, 1, 1) = 44 \]
\[ P_{18} = (1, 2, 1, 1) = 46 \]
\[ P_{19} = (2, 2, 1, 1) = 47 \]
\[ P_{20} = (3, 2, 1, 1) = 48 \]
\[ P_{21} = (1, 3, 1, 1) = 50 \]
\[ P_{22} = (2, 3, 1, 1) = 51 \]

\[ P_{23} = (3, 3, 1, 1) = 52 \]
\[ P_{24} = (0, 0, 1, 1) = 38 \]
\[ P_{25} = (1, 1, 2, 1) = 58 \]
\[ P_{26} = (2, 1, 2, 1) = 59 \]
\[ P_{27} = (3, 1, 2, 1) = 60 \]
\[ P_{28} = (1, 2, 2, 1) = 62 \]
\[ P_{29} = (2, 2, 2, 1) = 63 \]
\[ P_{30} = (3, 2, 2, 1) = 64 \]
\[ P_{31} = (1, 3, 2, 1) = 66 \]
\[ P_{32} = (2, 3, 2, 1) = 67 \]
\[ P_{33} = (3, 3, 2, 1) = 68 \]
\[ P_{34} = (0, 0, 2, 1) = 53 \]
\[ P_{35} = (1, 1, 3, 1) = 74 \]
\[ P_{36} = (2, 1, 3, 1) = 75 \]
\[ P_{37} = (3, 1, 3, 1) = 76 \]
\[ P_{38} = (1, 2, 3, 1) = 78 \]
\[ P_{39} = (2, 2, 3, 1) = 79 \]
\[ P_{40} = (3, 2, 3, 1) = 80 \]
\[ P_{41} = (1, 3, 3, 1) = 82 \]
\[ P_{42} = (2, 3, 3, 1) = 83 \]
\[ P_{43} = (3, 3, 3, 1) = 84 \]
\[ P_{44} = (0, 0, 3, 1) = 69 \]

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

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

The Orbiter commands associated with projective space objects are summarized in Tables 4.7-4.9.

Table 4.10 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 PG(3, 2). The command

```
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
```

```latex
pdflatex fix_structure_2A.tex
```

```bash
open fix_structure_2A.pdf
```

can be used.

Suppose we are looking for a projectivity of PG(3, 16) fixing the plane \(v(X_3)\) pointwise and mapping a pair of skew lines not in that plane to another pair of skew lines not in that plane. For instance, suppose we want to map

\[
M_1 = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix} \mapsto N_1 = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix},
\]

\[
M_2 = \begin{bmatrix}
1 & 1 & 0 & \delta \\
0 & 0 & 1 & 0
\end{bmatrix} \mapsto N_2 = \begin{bmatrix}
0 & 1 & 0 & 1 \\
0 & 0 & 1 & 0
\end{bmatrix}
\]

The command
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Description</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 mₙ matrix</td>
<td>Compute the automorphism group of a linear code using Nauty.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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>-define_surface</td>
<td>label descr</td>
<td>To create a cubic surface and add it to the symbol table under the given label. See Section 7.1.</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>-define_quartic_curve</td>
<td>label descr</td>
<td>To create a quartic curve and add it to the symbol table under the given label. See Section 7.2.</td>
</tr>
<tr>
<td>-classify_surfaces_with_double_sixes</td>
<td>label control</td>
<td>Classify cubic surfaces using the double six approach. See Section 7.3.</td>
</tr>
</tbody>
</table>

Table 4.7: Projective Space Activities (Part 1)
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-classify_surfaces_through_arcs_and_two_lines</td>
<td></td>
<td>See Section 7.3.</td>
</tr>
<tr>
<td>-test_nb_Eckardt_points</td>
<td>nbE</td>
<td>See Section 7.3.</td>
</tr>
<tr>
<td>-classify_surfaces_through_arcs_and_trihedral_pairs</td>
<td></td>
<td>See Section 7.3.</td>
</tr>
<tr>
<td>-sweep</td>
<td>fname</td>
<td></td>
</tr>
<tr>
<td>-sweep_4</td>
<td>fname</td>
<td>surface-descr</td>
</tr>
<tr>
<td>-sweep_4_27</td>
<td>fname</td>
<td>surface-descr</td>
</tr>
<tr>
<td>-six_arcs_not_on_conic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-filter_by_nb_Eckardt_points</td>
<td>nbE</td>
<td></td>
</tr>
<tr>
<td>-surface_quartic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-surface_clebsch</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-surface_codes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-trihedral_control</td>
<td>poset-control</td>
<td></td>
</tr>
<tr>
<td>-trihedra2_control</td>
<td>poset-control</td>
<td></td>
</tr>
<tr>
<td>-control_six_arcs</td>
<td>poset-control</td>
<td></td>
</tr>
<tr>
<td>-make_gilbert_varshamov_code</td>
<td>n d</td>
<td>See Section 10.8.</td>
</tr>
</tbody>
</table>

Table 4.8: Projective Space Activities (Part 2)
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-spread_classify</code></td>
<td><code>k control</code></td>
<td>See Section 12.1.</td>
</tr>
<tr>
<td><code>-classify_semifields</code></td>
<td><code>descr</code></td>
<td></td>
</tr>
<tr>
<td><code>-cheat_sheet</code></td>
<td></td>
<td>Produce a cheat sheet for ( \text{PG}(n, q) )</td>
</tr>
<tr>
<td><code>-classify_quartic_curves_nauty</code></td>
<td><code>fname-mask N</code></td>
<td>Classify quartic curves using Nauty.</td>
</tr>
<tr>
<td><code>-classify_quartic_curves_with_substructure</code></td>
<td><code>fname-mask N k d frame</code></td>
<td>Classify quartic curves using substructure algorithm.</td>
</tr>
<tr>
<td><code>-set_stabilizer</code></td>
<td><code>k fname-mask N col-label</code></td>
<td>Compute canonical form of sets using the substructure algorithm.</td>
</tr>
<tr>
<td><code>-lift_skew_hexagon</code></td>
<td><code>text</code></td>
<td>Lift a skew-hexagon.</td>
</tr>
<tr>
<td><code>-lift_skew_hexagon_with_polarity</code></td>
<td><code>polarity</code></td>
<td>Lift a skew-hexagon with a given polarity.</td>
</tr>
<tr>
<td><code>-arc_with_given_set_as_s_lines_after_dualizing</code></td>
<td><code>sz d d_{\min} s</code></td>
<td>Finds arcs with the given set as s-lines.</td>
</tr>
<tr>
<td><code>-arc_with_two_given_sets_of_lines_after_dualizing</code></td>
<td><code>sz d d_{\min} s t T</code></td>
<td>Finds arcs with the two given sets as s-lines and t-lines, respectively.</td>
</tr>
<tr>
<td><code>-arc_with_three_given_sets_of_lines_after_dualizing</code></td>
<td><code>sz d d_{\min} s t T u U</code></td>
<td>Finds arcs with the three given sets as s-lines and t-lines and u-lines, respectively.</td>
</tr>
<tr>
<td><code>-dualize_hyperplanes_to_points</code></td>
<td></td>
<td>Turns ranks of hyperplanes into ranks of points.</td>
</tr>
<tr>
<td><code>-dualize_points_to_hyperplanes</code></td>
<td></td>
<td>Turns ranks of points into ranks of hyperplanes.</td>
</tr>
</tbody>
</table>

Table 4.9: Projective Space Activities (Part 3)
<table>
<thead>
<tr>
<th>Modifier</th>
<th>Args</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-classify_cubic_curves</td>
<td>$q$</td>
<td>Classifies cubic curves in PG(2, $q$). Requires -control_arcs. See Section 6.7.</td>
</tr>
<tr>
<td>-control_arcs</td>
<td>description</td>
<td>Poset classification control for arcs used during the classification of cubic curves. See Table 6.2.</td>
</tr>
<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>$\epsilon$ $N$ $b$ $t_{\min}$ $t_{\max}$ 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 $[t_{\min}, t_{\max}]$.</td>
</tr>
<tr>
<td>-create_spread</td>
<td>description</td>
<td>Creates a spread according to the description. See Section 12.1.</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>
</tbody>
</table>

Table 4.10: Orbiter commands related to projective geometries
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 \(\delta^4 = \delta^3 + 1\).

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 homogeneous 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:

for \(f_3\) and

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

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

\[
deLP_F13ab_report:
\]

\[
\begin{verbatim}
$ORBITER) -v 3 \\
\end{verbatim}
\]
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 4.11 and 4.12 can be issued. Modifier options as shown in Table 4.13 apply. The following command creates an elliptic quadric ovoid on PG(3, 8):

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

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

```
ovoideq_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 
  -ovoideq_ST 
  -with O -do -combinatorial_object_activity -save 
  -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 4.2.

```
EDGE_CURVE_Q17_EQUATION="1 16 16 0 0 0 0 0 0 0 0 1 12 0 0"

EDGE_CURVE_Q17_AS_POINTS="4 7 16 19 20 23 32 35 89 100 244 251"
```
<table>
<thead>
<tr>
<th>Command</th>
<th>Args</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 Adalaide 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 $k$th BLT-set of order $q$ from the database ($k = 0, 1, ...$)</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 $\sum_{i=0}^{n} X_i^{q^{2i+1}} = 0$</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 4.11: Orbiter Objects (Part 1)
<table>
<thead>
<tr>
<th>Command</th>
<th>Args</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>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 lab_ascii_lab_tex d coeffs</td>
<td></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>$l \ d \ n \ C_1 \ldots C_n$</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 4.12: Orbiter Objects (Part 2)

<table>
<thead>
<tr>
<th>Command</th>
<th>Args</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-embedded_in_PG_4_q</td>
<td></td>
<td></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 4.13: Orbiter Objects: Modifiers
The following command computes the line type of the Edge curve:

```
Edge_curve_17_line_type:
  ▶ echo $(FILE_Q17) >edge_q17.csv
  ▶ $(ORBITER) -v 2 \
  ▶ ▶ -define F -finite_field -q 17 -end \
  ▶ ▶ -define R -polynomial_ring -field F \
  ▶ ▶ ▶ -number_of_variables 3 \
  ▶ ▶ ▶ -homogeneous_of_degree 4 \
  ▶ ▶ ▶ -end \
  ▶ ▶ -define P -projective_space -n 2 -field F -v 0 -end \
  ▶ ▶ -define C -geometric_object P \
  ▶ ▶ ▶ -projective_variety R \
  ▶ ▶ ▶ ▶ "Edge_q17" "Edge\_q17" \
  ▶ ▶ ▶ ▶ $(EDGE_CURVE_Q17_EQUATION) \
  ▶ ▶ ▶ -end \
  ▶ ▶ -with C -do -combinatorial_object_activity -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, 61]). 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 \\
▷ ▷ ▷ -report \\
▷ ▷ -end \\
▷ pslatex Perm6_report.tex \\
▷ open Perm6_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 \\
▷ ▷ ▷ -export_group_table \\
▷ ▷ -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):

```
Symmetric_3:
$ (ORBITER) -v 3 \
define G -permutation_group -symmetric_group 3 -end \
with G -do \n-group_theoretic_activity \ndo -report \nde -end \npdflatex Perm3_report.tex \open Perm3_report.pdf
```

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**

```
0
\|--|1
\|--|2
```
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.

```
Symmetric_3_group_table:
  > $(ORBITER) -v 3 \
  >   -define G -permutation_group -symmetric_group 3 -end \
  >   -with G -do \
  >   -group_theoretic_activity \
```
The next command produces a graphical representation of the elements of the symmetric group Sym(3), shown in Figure 5.3.

```
$(ORBITER) -v 3 \n  -define G -permutation_group -symmetric_group 3 -end \n  -with G -do \n  -group_theoretic_activity \n  -save_elements_csv "Symmetric3_elts.csv" \n  -end \n$(ORBITER) -v 2 \n  -define Sym3_elts -vector -load_csv_data_column \n  Symmetric3_elts.csv 1 -end \n  -save_matrix_csv Sym3_elts
```

The output of these commands is shown in Figure 5.3.
-define all_one_c -vector -repeat 1 3 -end \n-draw_matrix \n   -input_csv_file Sym3_elts_matrix.csv \n   -box_width 50 -bit_depth 8 \n   -partition 3 \n   all_one_r all_one_c \n-end
open Sym3_elts_matrix_draw.bmp
5.2 Linear Groups

Orbiter provides support for matrix groups and their various permutation representations. For background information about the classical groups of matrices over finite fields, see cf. [65]. 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 \(i\)th 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 \(V \cong \mathbb{F}_q^n\) 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

\[\Phi_0 : \mathbb{F}_q \to \mathbb{F}_q, \ x \mapsto x^p\]

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

\[(\Psi_M, \Phi_0^k) : \text{PG}(n,q) \to \text{PG}(n,q), \ P(x) \mapsto P(y), \ y = (x \cdot M)^{\Phi_0^k}\]

129
is denoted as

\[ M_k. \]  \hspace{1cm} (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

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

\[ \left( M_k \right)^{-1} = \left( (M^{-1})^{\Phi^k} \right)_{-k}. \]  \hspace{1cm} (5.3)

The affine group \( AGL(n, q) \) is the semidirect product of \( GL(n, q) \) with \( \mathbb{F}_q^n \). The affine semi-linear group \( A\Gamma L(n, q) \) is the semidirect product of \( AGL(n, q) \) with \( \text{Aut}(\mathbb{F}_q) \). The elements of \( A\Gamma L(n, q) \) are triples

\[ M_{a,k} := (M, a, k) \in GL(n, q) \times \mathbb{F}_q^n \times \text{Aut}(\mathbb{F}_q), \]

which act on \( \mathbb{F}_q^n \):

\[ (x, (M, a, k)) \mapsto (x \cdot M + a)^{\Phi^k}. \]

The multiplication in \( A\Gamma L(n, q) \) 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

\[ \rho : \mathcal{P} \leftrightarrow \mathcal{L}, \ P(x) \leftrightarrow [x]. \]

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>$Gr_1(V)$</td>
<td>$\frac{q^n-1}{q-1}$</td>
</tr>
<tr>
<td>Wreath product $GL(d, q) \wr Sym(n)$</td>
<td>$Gr_1((\mathbb{F}_q^d)^\otimes n)$ 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 \
-do -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 $Gr_1(\mathbb{F}_2^4)$. At first, the field $\mathbb{F}_2$ is created. Secondly, the group $G = PGL(3, 2)$ 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>( \text{GL}(n, q) )</td>
</tr>
<tr>
<td>-GGL</td>
<td>( n q )</td>
<td>( \Gamma \text{L}(n, q) )</td>
</tr>
<tr>
<td>-SL</td>
<td>( n q )</td>
<td>( \text{SL}(n, q) )</td>
</tr>
<tr>
<td>-SSL</td>
<td>( n q )</td>
<td>( \Sigma \text{L}(n, q) )</td>
</tr>
<tr>
<td>-PGL</td>
<td>( n q )</td>
<td>( \text{PGL}(n, q) )</td>
</tr>
<tr>
<td>-PGGL</td>
<td>( n q )</td>
<td>( \text{PΓL}(n, q) )</td>
</tr>
<tr>
<td>-PSL</td>
<td>( n q )</td>
<td>( \text{PSL}(n, q) )</td>
</tr>
<tr>
<td>-PSSL</td>
<td>( n q )</td>
<td>( \text{PΣL}(n, q) )</td>
</tr>
<tr>
<td>-AGL</td>
<td>( n q )</td>
<td>( \text{AGL}(n, q) )</td>
</tr>
<tr>
<td>-AGGL</td>
<td>( n q )</td>
<td>( \text{AΓL}(n, q) )</td>
</tr>
<tr>
<td>-ASL</td>
<td>( n q )</td>
<td>( \text{ASL}(n, q) )</td>
</tr>
<tr>
<td>-ASSL</td>
<td>( n q )</td>
<td>( \text{AΣL}(n, q) )</td>
</tr>
<tr>
<td>-PGO</td>
<td>( n q )</td>
<td>( \text{PGO}(n, q) )</td>
</tr>
<tr>
<td>-PGOp</td>
<td>( n q )</td>
<td>( \text{PGO}^+(n, q) )</td>
</tr>
<tr>
<td>-PGOm</td>
<td>( n q )</td>
<td>( \text{PGO}^-(n, q) )</td>
</tr>
<tr>
<td>-PGGO</td>
<td>( n q )</td>
<td>( \text{PGGO}(n, q) )</td>
</tr>
<tr>
<td>-PGGOp</td>
<td>( n q )</td>
<td>( \text{PGGO}^+(n, q) )</td>
</tr>
<tr>
<td>-PGGOM</td>
<td>( n q )</td>
<td>( \text{PGGO}^-(n, q) )</td>
</tr>
<tr>
<td>-GL_d_q_wr_Sym_n</td>
<td>( d q n )</td>
<td>( \text{GL}(d, q) \wr \text{Sym}(n) )</td>
</tr>
</tbody>
</table>

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

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

1,0,0,0,1,0,0,0,0,1,0,1,0,0,1,
1,0,0,0,1,0,0,0,0,1,0,0,1,0,1,
1,0,0,0,1,0,0,0,0,1,0,0,0,1,1,
1,0,0,0,1,0,0,0,0,1,0,0,0,1,0,
1,0,0,0,0,1,0,0,0,1,0,0,0,0,1,
0,1,0,0,1,0,0,0,0,1,0,0,0,0,1,

The Action

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

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

The group is a matrix group.
The group acts on projective space PG(3,2)

\[ q = 2 \]
\[ p = 2 \]
\[ e = 1 \]
\[ n = 3 \]
Number of points = 15
Number of lines = 35
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>DNE</td>
</tr>
</tbody>
</table>

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

$\begin{array}{c|c}
\cdot & 1 \\
1 & 1 \\
\end{array}$

$1^0 \equiv 1$

$1^1 \equiv 1$

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:

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

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

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

$1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1,
1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1,
1, 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 14 5 6 7 8 3 11 10 13 12 4
0 1 2 10 13 5 6 7 8 11 3 9 14 4 12
0 1 2 12 11 5 6 7 8 13 14 4 3 9 10
0 1 3 2 4 5 9 10 11 6 7 8 12 13 14
The base has length 4
The basic orbits are:
Basic orbit 0 is orbit of 0 of length 15
Basic orbit 1 is orbit of 1 of length 14
Basic orbit 2 is orbit of 2 of length 12
Basic orbit 3 is orbit of 3 of length 8

We use the following Orbiter command creates PGL(4,2) again. The command invokes two activities. The first creates a latex report for the group in the file PGL_4_2_report.tex. The second activity exports the permutation representation in Orbiter makefile format.

PGL4_2.export:
  $(ORBITER) -v 2 \
  -define F -finite_field -q 2 -end \n  -define G -linear_group -PGL 4 F -end \n  -with G -do \n  -group_theoretic_activity \n  -report \n  -end \n  -with G -do \n  -group_theoretic_activity \n  -export_orbiter \n  -end
  pdflatex PGL4_2_report.tex
  open PGL4_2_report.pdf

The file PGL4_2.makefile is created:

PGL4_2.generated:
  $(ORBITER) -v 2 \
  -define gens -vector -file PGL4_2_gens.csv -end \n  -define G -permutation_group \n  -bsgs PGL4_2 "\{rm PGL\}(4,2)" 15 20160 "0,1,2,3" 6 gens -end \n
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 PGL4_2_gens.csv:

Row,C0,C1,C2,C3,C4,C5,C6,C7,C8,C9,C10,C11,C12,C13,C14
The command

L_5.3:
▷ $(ORBITER) -v 2 \\
▷ ▷ -define F -finite_field -q 3 -end \n▷ ▷ -define G -linear_group -PSL 5 F -end \n▷ ▷ -with G -do \\
▷ ▷ -group_theoretic_activity \\
▷ ▷ ▷ -report \\
▷ ▷ -end
▷ pdflatex PSL_5_3_report.tex
▷ open PSL_5_3_report.pdf

creates PSL(5, 3) of order 237783237120.

The command

PSP_4.4:
▷ $(ORBITER) -v 2 \\
▷ ▷ -define F -finite_field -q 4 -end \n▷ ▷ -define G -linear_group -PGL 4 F \n▷ ▷ ▷ -symplectic_group \\
▷ ▷ -end \n▷ ▷ -with G -do \\
▷ ▷ -group_theoretic_activity \\
▷ ▷ ▷ -report \\
▷ ▷ -end
▷ pdflatex PGL_4_4_Sp_4_4_report.tex
▷ open PGL_4_4_Sp_4_4_report.pdf

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

The command

PGO_5.2:
▷ $(ORBITER) -v 2 \n
139
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:

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

1,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,1,0,1,0,0,1,1,  
1,0,0,0,0,0,1,0,0,0,0,0,1,0,0,1,1,0,0,1,1,0,0,0,0,  
1,0,0,0,0,0,0,1,0,0,0,0,1,0,0,1,1,1,0,1,1,0,0,0,0,1,  
1,0,0,0,0,0,0,0,1,0,0,0,1,0,0,1,1,0,0,1,1,0,0,1,1,0,  
1,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,1,1,1,0,1,0,0,0,0,0,  
1,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,1,1,1,0,1,0,1,0,0,  
1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,1,0,0,0,0,1,0,  
1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,1,0,0,0,0,0,  
1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,1,0,0,0,0,1,
The Action

Group action PGO(5, 2) of degree 15
We act on the following set:

\[
\begin{align*}
0 &= (0, 1, 0, 0, 0) & \quad 8 &= (0, 1, 1, 1, 1) \\
1 &= (0, 0, 1, 0, 0) & \quad 9 &= (1, 1, 1, 0, 0) \\
2 &= (0, 0, 0, 1, 0) & \quad 10 &= (1, 1, 1, 1, 0) \\
3 &= (0, 1, 0, 1, 0) & \quad 11 &= (1, 1, 1, 0, 1) \\
4 &= (0, 0, 1, 1, 0) & \quad 12 &= (1, 0, 0, 1, 1) \\
5 &= (0, 0, 0, 0, 1) & \quad 13 &= (1, 1, 0, 1, 1) \\
6 &= (0, 1, 0, 0, 1) & \quad 14 &= (1, 0, 1, 1, 1)
\end{align*}
\]

The group is a matrix group.
The base action is on projective space PG(4, 2)
\( q = 2 \)
\( p = 2 \)
\( e = 1 \)
\( n = 4 \)
Number of points = 31
Number of lines = 155
Number of lines on a point = 15
Number of points on a line = 3

The finite field \( \mathbb{F}_2 \)
\( Z_i = \log_\alpha (1 + \alpha^i) \)

<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>DNE</td>
</tr>
</tbody>
</table>

\[
\begin{array}{c|c|c}
+ & 0 & 1 \\
0 & 0 & 1 \\
1 & 1 & 0 \\
\cdot & 1 & 1 \\
\end{array}
\]
$1^0 \equiv 1$

$1^1 \equiv 1$

### Base and Stabilizer Chain

Group order 720

$t_1 = 15, 8, 3, 1, 1, 2,$

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

Strong generators for a group of order 720:

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

1,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,1,0,1,0,1,0,1,1,1,0,0,0,0,1,1,0,0,0,0,1,1,0,1,0,1,0,1,1,1,1,0,0,0,0,1,1,0,0,0,0,1,1,0,0,0,0,1,1,0,0,0,0,0,1,1,0,0,0,0,0,1,1,0,0,0,0,0,1,1,0,0,0,0,0,1,1,0,0,0,0,1,1,0,0,0,0,0,1,1,0,0,0,0,0,1,1,0,0,0,0,0,1

### 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>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 \
▷ ▷ -define G -linear_group -PGL 6 F \
▷ ▷ ▷ -symplectic_group \
▷ ▷ -end \
▷ ▷ -with G -do \
▷ ▷ -group_theoretic_activity \

145
The group $\text{PGO}(7,2)$, isomorphic to $\text{PSp}(6,2)$, can be created using the following command:

\begin{verbatim}
$\text{PGO}_7 2:
$ (ORBITER) -v 2 \n$ -define F -finite_field -q 2 -end \n$ -define G -linear_group -PGO 7 F -end \n$ -with G -do \n$ -group_theoretic_activity \n$ -report \n$ -end
\end{verbatim}

$\text{pdflatex PGO}_7 2\text{report.tex}$

$\text{open PGO}_7 2\text{report.pdf}$
<table>
<thead>
<tr>
<th>Modifier</th>
<th>Arguments</th>
<th>Meaning</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>$l \ o \ n \ s_1 \ldots \ s_n$</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:

```
Level | Base pt | Orbit length | Subgroup order
-----|---------|--------------|----------------
0     | 0       | 13           | 13
```

**Basic Orbit 0**

```
0
1
2
3
4
5
6
7
8
9
10
11
12
```

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 
#pdflatex Perm13_Subgroup_C13_13_report.tex
#open Perm13_Subgroup_C13_13_report.pdf
```

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 \\
```

150
creates the first Janko group as a subgroup of $\text{PGL}(7,11)$.

The command

\begin{verbatim}
PGL_3.11_singer:
\$ (ORBITER) -v 2 \n\$ define G -linear_group -PGL 3 11 -singer 19 -end \n\$ with G -do \n\$ -group_theoretic_activity \n\$ -report \n\$ -end
\end{verbatim}

\begin{verbatim}
pdflatex PGL_3.11_Singer_3_11_19_report.tex
\end{verbatim}

\begin{verbatim}
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 $\frac{q^d - 1}{q-1}$ 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 \n\$ define G -linear_group -PGL 3 11 -singer_and_frobenius 19 -end \n\$ with G -do \n\$ -group_theoretic_activity \n\$ -report \n\$ -end
\end{verbatim}

\begin{verbatim}
pdflatex PGL_3.11_Singer_and_Frob3_11_19_report.tex
\end{verbatim}

\begin{verbatim}
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 $\mathbb{R}$:

\begin{equation}
\begin{bmatrix}
i & 1 \\
1 & -1
\end{bmatrix}, \quad
\begin{bmatrix}
-1 & 1 \\
1 & 1
\end{bmatrix}, \quad
\begin{bmatrix}
0 & -1 \\
1 & 0
\end{bmatrix}.
\end{equation}

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 \\
▷ ▷ -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
```

153
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,
   2,0,0,0,1,0,0,0,1"
   -end

-define G -linear_group -GL 3 3
-define gens
-end
-with G -do
-group_theoretic_activity
   -print_elements.tex
   -report
-end
```

```
pdflatex GL_3_3_Subgroup_cube_48_report.tex
open GL_3_3_Subgroup_cube_48_report.pdf
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
   -define gens
   -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
```
The Hesse group of order 216 extended by the automorphism group of the field can be created in $PG(3, 4)$

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

Hesse group:

```
$(ORBITER) -v 3 \\
define gens -vector -compact \\
define G -linear_group -PGGL 3 4 \\
-subgroup_by_generators "Hesse" "432" 7 gens \\
end \\
with G -do \\
group_theoretic_activity \\
print_elements_tex \\
-report \\
end
```

```
pdflatex PGGL_3_4_Subgroup_Hesse_432_report.tex
open PGGL_3_4_Subgroup_Hesse_432_report.pdf
```

The group has order 432.

The Weyl group of type $E_8$ can be generated as a subgroup of $GL(8, 3)$ using the following command:

```plaintext
GENERATORS_WEYL_GROUP_E8="\n-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, \\
1,0,1,1,0,0,0,0, \\
0,0,0,0,0,1,0,0, \\
0,0,0,0,0,0,1,0, \\
0,0,0,0,0,0,0,1, \\
-1,0,-1,-1,-1,-1,-1,-1, \\
0,1,0,1,1,1,1,1, \\
```

155
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  ▶ ▶ "Weyl_E8" "696729600" 2 \n  ▶ ▶ $(GENERATORS_WEYL_GROUP_E8) \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:

```
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 \
  -with G1 -and G2 -do \
  -group_theoretic_activity \
  -coset_reps \
  -end
```

The coset representatives are written to a csv file. The (shortened) list of coset representatives in latex is:

<table>
<thead>
<tr>
<th>Coset</th>
<th>Matrix</th>
<th>Vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 / 36</td>
<td>$\begin{bmatrix} 1 &amp; 0 &amp; 0 &amp; 0 &amp; 0 &amp; 0 \ 0 &amp; 1 &amp; 0 &amp; 0 &amp; 0 &amp; 0 \ 0 &amp; 0 &amp; 1 &amp; 0 &amp; 0 &amp; 0 \ 0 &amp; 0 &amp; 0 &amp; 1 &amp; 0 &amp; 0 \ 0 &amp; 0 &amp; 0 &amp; 0 &amp; 1 &amp; 0 \ 0 &amp; 0 &amp; 0 &amp; 0 &amp; 0 &amp; 1 \end{bmatrix}$</td>
<td>$\begin{bmatrix} 100000 \ 010000 \ 001000 \ 000100 \ 000010 \ 000001 \end{bmatrix}$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>35 / 36</td>
<td>$\begin{bmatrix} 1 &amp; 0 &amp; 1 &amp; 1 &amp; 1 &amp; 0 \ 1 &amp; 0 &amp; 1 &amp; 0 &amp; 0 &amp; 0 \ 0 &amp; 1 &amp; 1 &amp; 1 &amp; 0 &amp; 1 \ 0 &amp; 1 &amp; 1 &amp; 1 &amp; 1 &amp; 1 \ 1 &amp; 0 &amp; 0 &amp; 0 &amp; 1 &amp; 0 \ 1 &amp; 1 &amp; 0 &amp; 1 &amp; 0 &amp; 0 \end{bmatrix}$</td>
<td>$\begin{bmatrix} 101110 \ 101000 \ 011101 \ 011111 \ 100010 \ 110100 \end{bmatrix}$</td>
</tr>
</tbody>
</table>

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

```
coset_reps_read:
```

157
$\$(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 \$
$\$ -define CR -vector_ge -action G2 \$
$\$ -read_csv \$
$\$ PGOp_6.2_cosep_reps.csv Element \$
$\$ -end$
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 [68].

The electronic Atlas of finite simple groups [68] 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 [63]):

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

It results in

$.1^2 + 2*$.1 + 2

which is the Magma way of printing the polynomial $X^2 + 2X + 2$. If $\alpha$ 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
U_3.3:

```
$ (ORBITER) -v 3 \
  -define F -finite_field -q 9 -override_polynomial "17" -end \
  -define G -linear_group -PGL 3 F \
  -subgroup_by_generators "U_3.3" "6048" 2 \
  "1,6,4, 5,0,6, 8,5,1, " \
  "6,2,1, 7,8,4, 0,6,6" \
  -end \
  -with G -do \
  -group_theoretic_activity \
  -report \
  -end
```

`pdflatex PGL_3.9_Subgroup_U_3.3_6048_report.tex`

`open PGL_3.9_Subgroup_U_3.3_6048_report.pdf`

Group theoretic activities will be discussed in Section 5.6.

As an example of a large group, consider the Conway group Co3. Following [62], 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 of generators, which is passed to the `-subgroup_by_generators` command. Finally, we create a report for the group.

```
CONWAY_GEN1="\n110111000100001010000\n1111010111110100001011\n00000100000010001010\n111110011010001001110\n01010000000001001101\n00000100000001001010\n00100000000001001010\n00010000000001001001\n00000000000001001001\n00000000000001001001\n011010100101001111111\n00000000000001001010\n0000000000000001001010\n0000000000000000110001\n0000000001100000100001\n0000000000000000010110
```

160
Co3:

```bash
$ (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 \
  -with G -do \
  -group_theoretic_activity \
  -report \
  -end
```

pdflatex PGL_22_2_Subgroup_Co3_495766656000_report.tex
open PGL_22_2_Subgroup_Co3_495766656000_report.pdf
The next example creates the Ree group in 7 dimensions over the field $\mathbb{F}_{27}$. Again, we use makefile variables to specify the two generators as $7 \times 7$ matrices over $\mathbb{F}_{27}$ 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:
$\langle ORBITER \rangle -v 2 \$
$\langle ORBITER \rangle -define F -finite_field -q 27 -override_polynomial "34" -end \$
$\langle ORBITER \rangle -define g1 -vector -field F -format 7 -dense $(Ree_gen1) -end \$
$\langle ORBITER \rangle -define g2 -vector -field F -format 7 -dense $(Ree_gen2) -end \$
$\langle ORBITER \rangle -define gens -vector -concatenate g1 -concatenate g2 -end \$
$\langle ORBITER \rangle -define G -linear_group -PGL 7 F \$
$\langle ORBITER \rangle -subgroup_by_generators "Ree_27" "10073444472" 2 gens \$
$\langle ORBITER \rangle -report \$
$\langle ORBITER \rangle -end \$
```

162
<table>
<thead>
<tr>
<th>Modifier</th>
<th>Arguments</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-wedge</td>
<td></td>
<td>action on the exterior square</td>
</tr>
<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_structure_action</td>
<td>s</td>
<td>action by field reduction to the subfield of index s</td>
</tr>
<tr>
<td>-on_k_subspaces</td>
<td>k</td>
<td>induced action on k dimensional subspaces</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_tensors</td>
<td></td>
<td>induced action of GL(d, q) ⋊ Sym(n) on the tensor space</td>
</tr>
<tr>
<td>-restricted_action</td>
<td>s</td>
<td>restricted action on the set s</td>
</tr>
</tbody>
</table>

Table 5.4: Commands for creating new actions from old

5.5 Induced Actions

It is possible to create new group actions from old. Table 5.4 lists Orbiter commands to do so. For instance, the command

```
T3_on_tensors:
▷ $(ORBITER) -v 2 \n▷ ▷ -define G \n▷ ▷ -linear_group -GL_d_q_wreath_Sym_n 2 2 3 \n▷ ▷ ▷ -on_tensors -end \n▷ ▷ -with G -do \n▷ ▷ -group_theoretic_activity \n▷ ▷ ▷ -report \n▷ ▷ -end \n▷ pdflatex GL_2_2_wreath_Sym3_report.tex \n▷ open GL_2_2_wreath_Sym3_report.pdf
▷
```

creates the group GL(2, 2) ⋊ Sym(3) acting on the 255 elements of PG(7, 2) which are identified with the tensors of type (2, 2, 2) over 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 \( \text{GL}(2, 2) \wr \text{Sym}(3) \)

The order of the group \( \text{GL}(2, 2) \wr \text{Sym}(3) \) is 1296
The group acts on a set of size 255

The Action

Group action \( \text{GL}(2, 2) \wr \text{Sym}(3) \text{res}_{255} \) 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.

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

0, 1, 2, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 
0, 1, 2, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 
0, 1, 2, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 
0, 1, 2, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 
0, 1, 2, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 
0, 1, 2, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 
0, 1, 2, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 
0, 1, 2, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 
1, 0, 2, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 
1, 0, 2, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1,
It is also possible to restrict the action on all rank-one tensors, as the following example shows:

\textbf{T3r1:}
\begin{verbatim}
  $(ORBITER) -v 4 \
  -define G \
  -linear_group -GL_d_q_wr_Sym_n 2 2 3 \
  -on_rank_one_tensors -end \
  -with G -do \
  -group_theoretic_activity \
  -report \
  -end
\end{verbatim}
\begin{verbatim}
  pdflatex GL_2_2_wreath_Sym3_report.tex
  open GL_2_2_wreath_Sym3_report.pdf
\end{verbatim}

This creates an action of degree 27. Because the degree is small, 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 $GL(2, 2) \wr Sym(3)_{res^27}$ of degree 27
We act on the following set:

0 = ( 1, 0, 0, 0, 0, 0, 0, 0 )
1 = ( 0, 1, 0, 0, 0, 0, 0, 0 )
2 = ( 1, 1, 0, 0, 0, 0, 0, 0 )
3 = ( 0, 0, 1, 0, 0, 0, 0, 0 )
4 = ( 0, 0, 0, 1, 0, 0, 0, 0 )
5 = ( 0, 0, 0, 0, 1, 0, 0, 0 )
6 = ( 1, 0, 1, 0, 0, 0, 0, 0 )
7 = ( 0, 1, 0, 1, 1, 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, 0, 1, 0 )
12 = ( 0, 0, 0, 0, 0, 0, 0, 1 )
13 = ( 0, 0, 0, 0, 0, 0, 0, 0 )
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, 0, 1, 0, 0 )
19 = ( 0, 1, 0, 0, 1, 0, 0, 0 )
20 = ( 1, 1, 0, 0, 1, 0, 0, 0 )
21 = ( 0, 0, 1, 1, 1, 1, 1, 1 )

Base and Stabilizer Chain

Group order 1296
tl=3, 2, 1, 3, 2, 3, 2, 3, 2
Strong generators for a group of order 1296:

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

\[
\begin{bmatrix}
1 & 0 & 1 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
\end{bmatrix}; (1, 2) \]

166
The group of a conic is isomorphic to the group of the projective line. This isomorphism from the group of the projective line to the group of the conic can be realized using the command `PGL2OnConic`. The action is changed using the induced action on the Veronese variety. The group elements are still represented as $2 \times 2$ matrices. Here is an example. We create the collineation group $\Gamma L(2, 8)$ of $\text{PG}(1, 8)$ and act on $\text{PG}(2, 8)$:

```
PGGL_2_8_on_conic:
▷ $(\text{ORBITER}) \ -v\ 4 \ \\
▷ \▷ \ -\text{define} \ G \ \\
▷ \▷ \ -\text{linear\_group} -\text{PGGL 2 8} -\text{PGL2OnConic} \ -\text{end} \ \\
▷ \▷ \ -\text{with} \ G \ -\text{do} \ \\
▷ \▷ \ -\text{group\_theoretic\_activity} \ \\
▷ \▷ \ \▷ \ -\text{report} \ \\
▷ \▷ \ \▷ \ -\text{end} \ \\
▷ \ \text{pdf\_latex} \ \text{PGGL_2_8_OnConic_2_8\_report.tex} \ \\
▷ \ \text{open} \ \text{PGGL_2_8_OnConic_2_8\_report.pdf}
```

This produces the following report. The generators are elements of $\Gamma L(2, 8)$ acting on $\text{PG}(2, 8)$. The first basic orbit is the conic itself and all other basic orbits are subsets of it.

**The Group $\Gamma L(2, 8)\text{OnConic}(2, 8)$**

The order of the group $\Gamma L(2, 8)\text{OnConic}(2, 8)$ is 1512
The group acts on a set of size 73
Strong generators for a group of order 1512:

\[
\begin{bmatrix}
1 & 0 \\
0 & 1 
\end{bmatrix}_1, \begin{bmatrix}
\gamma & 0 \\
0 & 1 
\end{bmatrix}_0, \begin{bmatrix}
1 & 0 \\
1 & 1 
\end{bmatrix}_0, \begin{bmatrix}
1 & 0 \\
\gamma & 1 
\end{bmatrix}_0, \begin{bmatrix}
1 & 0 \\
\gamma^2 & 1 
\end{bmatrix}_0, \begin{bmatrix}
0 & 1 \\
1 & 0 
\end{bmatrix}_0
\]

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:

\[0 = (1, 0, 0)\]
\[1 = (0, 1, 0)\]
\[2 = (0, 0, 1)\]
\[3 = (1, 1, 1)\]
\[4 = (1, 1, 0)\]
\[5 = (2, 1, 0)\]
\[6 = (2, 1, 0)\]
\[72 = (7, 7, 1)\]

The group is a matrix group.
The base action is on projective space \(\mathbb{P}G(1,8)\)

\[q = 8\]
\[p = 2\]
\[e = 3\]
\[n = 1\]
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: \(X^3 + X^2 + 1 = 13\)
\(Z_i = \log_\alpha(1 + \alpha^2)\)

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

\[ \begin{array}{c|cccccccc} \hline \cdot & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \hline 1 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 2 & 2 & 4 & 6 & 5 & 7 & 1 & 3 \\ 3 & 3 & 6 & 5 & 1 & 2 & 7 & 4 \\ 4 & 4 & 5 & 1 & 7 & 3 & 2 & 6 \\ 5 & 5 & 7 & 2 & 3 & 6 & 4 & 1 \\ 6 & 6 & 1 & 7 & 2 & 4 & 3 & 5 \\ 7 & 7 & 3 & 4 & 6 & 1 & 5 & 2 \\ \hline \end{array} \]

\[\begin{align*}
2^0 &= 1 \\
2^1 &= 2 \\
2^2 &= 4 \\
2^3 &= 5 \\
2^4 &= 7 \\
2^5 &= 3 \\
2^6 &= 6 \\
2^7 &= 1
\end{align*}\]

**Base and Stabilizer Chain**

Group order 1512

tl=9, 8, 7, 3,

**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>9</td>
<td>1512</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>8</td>
<td>168</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>7</td>
<td>21</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>
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
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 $PG(3,13)$ 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. Here is the fill command sequence, including a makefile variable for the generators of the stabilizer of the surface:

```
SURFACE_q13_STAB="1,0,0,0,0,12,0,0,0,0,12,0,0,0,0,1, \\
1,0,0,0,12,0,0,0,0,1,0,0,0,0,12, \\
1,0,0,0,0,12,0,0,12,0,0,0,0,0,1, \\
0,1,0,0,1,0,0,0,0,1,0,0,0,0,0,1"
```

```bash
del $((ORBITER_PATH)) -v 30 \\
  define F -finite_field -q 13 -end \\
  define P -projective_space -n 3 -field F -v 0 -end \\
  with P -do \\
  projective_space_activity \\
  define_surface S -q 13 \\
  arc_lifting "0,1,2,3,43,113" -end \\
  end \\
  with S -do \\
  cubic_surface_activity \\
  report_with_group \\
  end \\
  with S -do \\
  cubic_surface_activity \\
  export_tritangent_planes \\
  end
```

```
$((ORBITER_PATH)) -v 2 \\
  orbiter_path $(ORBITER_PATH) \\
  define TriP -set -file \\
  family_Eckardt_q13_a2_b1_tritangent_planes.csv \\
  end \\
  define G -linear_group -PGL 4 13 \\
  subgroup_by_generators "SURF_STAB" \\
  24 4 $((SURFACE_q13_STAB)) \\
  end \\
  define G_on_planes -modified_group -from G \\
  on_k_subspaces 3 \\
  end \\
  define Gr -modified_group -from G_on_planes \\
  restricted_action TriP \\
  end \\
  with Gr -do \\
```
-group_theoretic_activity \\
  -report \\
  -end \\
  -with Gr -do \\
  -group_theoretic_activity \\
  -orbits_on_points \\
  -stabilizer \\
  -end
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 5.5 and 5.6 list the possible commands that can come after it.

The command

\begin{verbatim}
PGL_3_2_elements:
  \texttt{\$(ORBITER) -v 5 \}
  \texttt{\define G -linear_group -PGL 3 2 -end \}
  \texttt{\with G -do \}
  \texttt{-group_theoretic_activity \}
  \texttt{\save_elements_csv "PGL_3_2_elements.csv" \}
  \texttt{-end \}
\end{verbatim}

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

The command

\begin{verbatim}
Sym_3_elements:
  \texttt{\$(ORBITER) -v 3 \}
  \texttt{\define G -permutation_group -symmetric_group 3 -end \}
  \texttt{\with G -do \}
  \texttt{-group_theoretic_activity \}
  \texttt{\print_elements_tex \}
  \texttt{-end \}
  \texttt{\$(ORBITER) -v 2 \}
  \texttt{-draw_options \}
  \texttt{\nodes \}
  \texttt{-embedded -radius 250 \}
  \texttt{-xin 10000 -yin 10000 \}
  \texttt{-xout 100000 -yout 600000 \}
  \texttt{-scale 0.3 -line_width 1.0 \}
  \texttt{-end \}
  \texttt{-tree_draw -file Perm3_elements_tree.txt -end \}
  \texttt{pdflatex Perm3_elements_tree_draw.tex \}
  \texttt{open Perm3_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>Modifier</th>
<th>Arguments</th>
<th>Meaning</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>$s_1 \ s_2$</td>
<td>Multiplies group elements $s_1$ and $s_2$, 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 $s^i$ for $i = 1, \ldots, n$. $s$ 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>-search_element_of_order</td>
<td>$i$</td>
<td>Finds all elements of order $i$ in the group $(i \in \mathbb{N})$.</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>-poset_classification_control</td>
<td></td>
<td>Poset classification options. The argument list must be terminated with -end</td>
</tr>
<tr>
<td>-classes_based_on_normal_form</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-group_table</td>
<td></td>
<td>Stores the group table as csv-file.</td>
</tr>
<tr>
<td>-report</td>
<td></td>
<td>Produce a latex report about the group.</td>
</tr>
<tr>
<td>-sylow</td>
<td></td>
<td>Include Sylow subgroups in the report (requires -report).</td>
</tr>
<tr>
<td>-print_elements</td>
<td></td>
<td>Produces a printout of all group elements.</td>
</tr>
<tr>
<td>-print_elements_tex</td>
<td></td>
<td>Produces a latex report of all group elements.</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>
</tbody>
</table>

Table 5.5: Group theoretic activities (Part 1)
<table>
<thead>
<tr>
<th>Modifier</th>
<th>Arguments</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-classify_arcs</td>
<td>description</td>
<td>Classify arcs in geometries. See Section 6.6.</td>
</tr>
<tr>
<td>-linear_codes</td>
<td>d n_{\text{max}}</td>
<td>Classify linear codes with prescribed minimum distance d. Assumes that the group is PGL(r, q) or PTL(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_classify</td>
<td>d</td>
<td>Classifies tensors of tensor-rank at most d.</td>
</tr>
<tr>
<td>-tensor_permutations</td>
<td></td>
<td>Computes the permutation representation of generators of wreath product.</td>
</tr>
<tr>
<td>-reverse_iso</td>
<td></td>
<td>Given a set of generators of a subgroup of PGO^+(6, q) as 6 \times 6 matrixes, compute the inverse image of the generators in PGL(4, q) (if possible).</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>
</tbody>
</table>

Table 5.6: Group theoretic activities (Part 2)

Figure 5.4: The elements of Sym(3) in lex-order
Cycle_{12}\_power:

\[
\begin{align*}
\text{pdflatex} & \text{ Perm12\_all\_powers.tex} \\
\text{open} & \text{ Perm12\_all\_powers.pdf}
\end{align*}
\]

We create the 12 powers of the cycle

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

The output is

\[
\begin{array}{|c|c|}
\hline
i & (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11)^i \\
\hline
1 & (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11) \\
2 & (0, 2, 4, 6, 8, 10)(1, 3, 5, 7, 9, 11) \\
3 & (0, 3, 6, 9)(1, 4, 7, 10)(2, 5, 8, 11) \\
4 & (0, 4, 8)(1, 5, 9)(2, 6, 10)(3, 7, 11) \\
5 & (0, 5, 10, 3, 8, 1, 6, 11, 4, 9, 2, 7) \\
6 & (0, 6)(1, 7)(2, 8)(3, 9)(4, 10)(5, 11) \\
7 & (0, 7, 2, 9, 4, 11, 6, 1, 8, 3, 10, 5) \\
8 & (0, 8, 4)(1, 9, 5)(2, 10, 6)(3, 11, 7) \\
9 & (0, 9, 6, 3)(1, 10, 7, 4)(2, 11, 8, 5) \\
10 & (0, 10, 8, 6, 4, 2)(1, 11, 9, 7, 5, 3) \\
11 & (0, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1) \\
12 & \text{id} \\
\hline
\end{array}
\]

The command

\[
\begin{align*}
\text{PGL}_{3, 4}\_singer: \\
& \text{pdflatex} \text{ Singer3_4.tex} \\
& \text{open} \text{ Singer3_4.pdf}
\end{align*}
\]
defines all Singer cycles in $\text{PGL}(3,4)$ 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 $\text{GL}(2,8)$. 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

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 $\equiv 0 \pmod{7}$ 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

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

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

We can compute the inverse of a group element:

GL_2_7_inv:
▷ $(ORBITER) -v 5 \$
▷ ▷ -define G -linear_group -GL 2 7 -end \$
▷ ▷ -with G -do \$
▷ ▷ -group_theoretic_activity \$
▷ ▷ ▷ -inverse "0,1,2,3" \$
▷ ▷ -end
▷ pdflatex GL_2_7_inv.tex
▷ open GL_2_7_inv.pdf

The output is

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

0,1,2,3,
2,4,1,0,
We can raise a group element to a power:

GL_2.7_power:

```
$ (ORBITER) -v 5 \
  -define G -linear_group -GL 2 7 -end \n  -group_theoretic_activity \n  -raise_to_the_power "0,1,2,3" 2 \n  -end
```

```
pdflatex GL_2.7_power.tex
open GL_2.7_power.pdf
```

The output is

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

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

The next example computes the action of a specific group element on the set of planes through a line. The planes have been computed in Section 4.4.

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 \n  -group_theoretic_activity \n  -apply "0,8,1,6,4,3,7,2,5" \n  "1,0,0,0, 0,1,0,0, 0,0,0,2, 0,0,1,1" \n  -end
```

```
pdflatex PGL_4.8.Gr_4.3_apply.tex
open PGL_4.8.Gr_4.3_apply.pdf
```

The output is
\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & \gamma \\
0 & 0 & 1 & 1 \\
\end{bmatrix} =
\begin{bmatrix}
1000 \\
0100 \\
0002 \\
0011 \\
\end{bmatrix}
\]

1,0,0,0,1,0,0,0,0,0,2,0,0,1,1, 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>Modifier</th>
<th>Arguments</th>
<th>Meaning</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 5.7: 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 5.7 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 \
▷ ▷ -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 \

▷ ▷ -define G \n
▷ ▷ -linear_group -PGGL 2 4 \n
▷ ▷ -end \n
▷ ▷ -with G -do \n
▷ ▷ -group_theoretic_activity \n
▷ ▷ ▷ -classes \n
▷ ▷ -end
```

182
computes the classes of elements in PΓL(2,4) using Orbiter-Magma-Orbiter. The first Orbiter command produces the file PGGL_2_4_classes.magma. The magma command reads this file and produces the file PGGL_2_4_classes_out.txt. The second Orbiter command reads the file PGGL_2_4_classes_out.txt and produces the latex report PGGL_2_4_classes_out.tex.

The report produced by Orbiter is too long to be reproduced here fully. Let us look at just one conjugacy class. Here is the output for class 1 / 7 (numbering starts from 0, so this is the second class):

<table>
<thead>
<tr>
<th>Order of element = 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class size = 10</td>
</tr>
<tr>
<td>Centralizer order = 12</td>
</tr>
<tr>
<td>Normalizer order = 12</td>
</tr>
</tbody>
</table>
| 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

```
PGGL_2_4_cent_2A:
$ (ORBITER) -v 3 \n$ -define G \n$ -linear_group -PGGL 2 4 -end \n$ -with G -do \n$ -group_theoretic_activity \n```
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 PGL(4,2).\(Z_2\). Orbiter produces a list of strong generators, shown below:

Strong generators for a group of order 40320:

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

The command

```
Normalizer_of_H5:
  $(ORBITER) -v 2 \n  -define G -permutation_group -symmetric_group 13 \n  -subgroup_by_generators H5 5 1 \n  $(GENERATORS_H5) -end \n  -with G -do \n  -group_theoretic_activity \n  -normalizer \n  -end
```

computes the normalizer of $H$ insize Sym(13). The normalizer is a group of order 1200. Because of the way in which Orbiter and Magma collaborate, the command has to be executed twice. After the first execution, a magma session is started. The magma session has to be terminated by typing

```
quit;
```

The Orbiter command has to be run one more time after that. The following report is produced:
The group \textit{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 in PGL 2 9:}

\[
\begin{align*}
\text{\$ (ORBITER) -v 2 \ \}
\text{\> -define G -linear_group -PGL 2 9 \ \}
\text{\> -subgroup_by_generators Z22 4 2 \ \}
\text{\> \> "2,0,0,1, 0,1,1,0" -end \ }
\text{\> \> -with G -do \ }
\text{\> \> -group_theoretic_activity \ }
\text{\> \> -normalizer \ }
\text{\> \> -end}
\end{align*}
\]

\text{pdflatex PGL_2_9_Subgroup_Z22_4_normalizer.tex}

\text{open PGL_2_9_Subgroup_Z22_4_normalizer.pdf}
It produces a report showing that the normalizer is a group of order 24 (it is isomorphic to Sym(4), though the report does not tell us this fact directly):

The group PGL(2, 9)SubgroupZ22order4 of order 4 is:
Strong generators for a group of order 4:
\[
\begin{bmatrix}
\alpha^4 & 0 \\
0 & 1
\end{bmatrix},
\begin{bmatrix}
0 & 1 \\
1 & 0
\end{bmatrix}
\]
1,0,0,2,
0,1,1,0,
Inside the group of order 720, the normalizer has order 24:
Strong generators for a group of order 24:
\[
\begin{bmatrix}
\alpha^4 & 0 \\
0 & 1
\end{bmatrix},
\begin{bmatrix}
\alpha^2 & 0 \\
0 & 1
\end{bmatrix},
\begin{bmatrix}
\alpha^4 & \alpha^4 \\
\alpha^4 & 1
\end{bmatrix},
\begin{bmatrix}
\alpha^4 & \alpha^6 \\
\alpha^2 & 1
\end{bmatrix}
\]
1,0,0,2,
1,0,0,5,
1,1,1,2,
1,7,5,2,
Chapter 6

Orbit Algorithms

6.1 Schreier Trees

Orbiter provides several different orbit algorithms. The most basic orbit algorithm uses Schreier trees. It is explained in [17, 36, 61]. This algorithm has memory and time complexity proportional to the size of the orbit. It therefore is limited to small problems. More elaborate algorithms exist, provided the set on which we act has additional structure. These algorithms offer sublinear complexity. Orbiter offers posets based algorithms, which exploit an underlying poset structure. They will be discussed in Section 6.2. Orderly generation using canonical forms is discussed in Section 15.2.

The commands discussed in this section are group theoretic activities, see Table 6.1.

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

```plaintext
orbits_PGL_4_2_on_points:
▷ $(ORBITER) -v 4 \n▷ ▷ -define G -linear_group -PGL 4 2 -end \n▷ ▷ -with G -do \n▷ ▷ ▷ -group_theoretic_activity \n▷ ▷ ▷ ▷ -report \n▷ ▷ ▷ ▷ -orbits_on_points \n▷ ▷ ▷ ▷ -export_trees \n▷ ▷ ▷ -end
▷ $(ORBITER) -v 3 \n▷ ▷ -draw_layered_graph \n▷ ▷ ▷ PGL_4_2_0.layered_graph \n▷ ▷ ▷ -radius 500 -spanning_tree -embedded \n▷ ▷ ▷ ▷ -line_width 1.1 -x_stretch 1.4 -scale 0.25 \n▷ ▷ ▷ -end
▷ pdflatex PGL_4_2_0_draw.tex
▷ open PGL_4_2_0_draw.pdf
```
<table>
<thead>
<tr>
<th>Modifier</th>
<th>Arguments</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-orbits_on_subsets</td>
<td>$k$</td>
<td>Compute orbits on $k$-subsets.</td>
</tr>
<tr>
<td>-orbits_on_points</td>
<td></td>
<td>Compute orbits in the action that was created.</td>
</tr>
<tr>
<td>-orbits_of</td>
<td>$i$</td>
<td>Compute orbit of point $i$ in the given action.</td>
</tr>
<tr>
<td>-stabilizer</td>
<td></td>
<td>Compute the stabilizer of the orbit representative (needs -orbits_on_points).</td>
</tr>
<tr>
<td>-orbits_on_set_system_from_file</td>
<td>$\text{fname} \ f \ l$</td>
<td>Reads the csv file “\text{fname}” and extract sets from columns $[f, ..., f + l - 1]$.</td>
</tr>
<tr>
<td>-orbit_of_set_from_file</td>
<td>$\text{fname}$</td>
<td>Reads a set from the text file “\text{fname}” and computes orbits on the elements of the set.</td>
</tr>
<tr>
<td>-orbits_on_polynomials</td>
<td>$d$</td>
<td>Computes the orbits of the matrix group on homogeneous polynomials of degree $d$. The number of variables is determined by the degree of the matrix group.</td>
</tr>
<tr>
<td>-conjugacy_class_of</td>
<td>$\text{label} \ s$</td>
<td>Compute the conjugacy class of the group element encoded as $s$ using the given label for file-names. Write a file containing the ranks for all elements in the class. Writes a second file containing the transporter elements for each element in the class. A transporter element maps the class representative to the given element under conjugation.</td>
</tr>
<tr>
<td>-orbits_on_group_elements_under_conjugation</td>
<td>$\text{fname-C} \ f\text{name-T}$</td>
<td>Under the centralizer of the class representative, construct the orbits on the class. For each non-trivial orbit, test whether the group generated by it and the class representative is Klein-four and all nontrivial elements are from the given class. If so, classify these groups and compute the normalizers. The arguments $\text{fname-C}$ and $\text{fname-T}$ are the files containing the elements of the class and the transporter, respectively.</td>
</tr>
</tbody>
</table>

Table 6.1: Basic Orbit algorithms
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▷ ▷ -define G \n▷ ▷ -linear_group -GL_d_q_wr_Sym_n 2 2 3 \n▷ ▷ ▷ -on_rank_one_tensors -end \n▷ ▷ -with G -do \n▷ ▷ -group_theoretic_activity \n▷ ▷ ▷ -report \n▷ ▷ ▷ -orbits_on_points \n▷ ▷ ▷ -export_trees \n▷ ▷ -end
▷ pdflatex GL_2_2_wreath_Sym3_orbits_report.tex
▷ open GL_2_2_wreath_Sym3_orbits_report.pdf

In the next example, we compute the orbits of the linear group PGL(4, 2) on homogeneous polynomials of degree 3 in 4 variables:
orbits_cubic_curves_q2:
- $(ORBITER) -v 4 \$
- $\-define G \-linear\_group \-PGL 3 2 \-end \$
- $\-with G \-do \$
- $\-group\_theoretic\_activity \$
- $\-orbits\_on\_polynomials 3 \$
- $\-end \$
- pdflatex poly_orbits_d3_n3_q2.tex
- open poly_orbits_d3_n3_q2.pdf

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:

PGGL_2.8_on_conic_orbits:
- $(ORBITER) -v 4 \$
- $\-define G \$
- $\-linear\_group \-PGGL 2 8 \-PGL2\_on\_Conic \-end \$
- $\-with G \-do \$

192
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.

**Group Orbits**

Orbits of the group $\text{PGL}(2,8)\text{OnConic}$:

Strong generators for a group of order 1512:

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

0,1,0,0,1,1, 1,0,6,0, 1,0,1,1,0, 1,0,2,1,0, 1,0,4,1,0, 0,1,1,0,0,

Considering the orbit length, there are 3 types of orbits:

\[(1, 9, 63)\]

\(i : \) orbit length : number of orbits
0 : 1 : 1
1 : 9 : 1
2 : 63 : 1
Orbits classified:
Set 0 has size 1 : \{1\}
Set 1 has size 1 : \{0\}
Set 2 has size 1 : \{2\}
Orbits of length 1:
Orbit 1: (1)

0 : 1 = ( 0, 1, 0 )

Orbits of length 9:
Orbit 0: ( 0, 2, 3, 29, 48, 38, 55, 60, 67 )

0 : 0 = ( 1, 0, 0 )
1 : 2 = ( 0, 0, 1 )
2 : 3 = ( 1, 1, 1 )
3 : 29 = ( 4, 2, 1 )
4 : 48 = ( 7, 4, 1 )
5 : 38 = ( 5, 3, 1 )
6 : 55 = ( 6, 5, 1 )
7 : 60 = ( 3, 6, 1 )
8 : 67 = ( 2, 7, 1 )

Orbits of length 63:
Orbit 2: ( 4, 5, 18, 7, 57, 25, 11, 37, 56, 10, 8, 33, 66, 45, 32, 41, 34, 14, 64, 9, 30, 47, 68, 52, 59, 71, 62, 6, 49, 65, 26, 21, 72, 54, 39, 13, 20, 43, 70, 50, 61, 17, 22, 44, 35, 23, 46, 40, 51, 28, ...12, 31, 16 )

0 : 4 = ( 1, 1, 0 )
1 : 5 = ( 2, 1, 0 )
2 : 18 = ( 0, 1, 1 )
3 : 7 = ( 4, 1, 0 )
4 : 62 = ( 6, 0, 1 )

Orbits of length 1:
Orbit 1: (1)

Stabilizer of orbit representative 1:
Strong generators for a group of order 1512:

\[
\begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix},
\begin{bmatrix}
2 & 1 \\
1 & 0
\end{bmatrix},
\begin{bmatrix}
3 & 1 \\
1 & 0
\end{bmatrix},
\begin{bmatrix}
4 & 1 \\
1 & 0
\end{bmatrix}
\]
Generator 0 / 4 is:
\[
\begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix}
\]

Generator 1 / 4 is:
\[
\begin{bmatrix}
\gamma & 0 \\
0 & 1
\end{bmatrix}
\]

Generator 2 / 4 is:
\[
\begin{bmatrix}
1 & 0 \\
\gamma & 1
\end{bmatrix}
\]

Generator 3 / 4 is:
\[
\begin{bmatrix}
0 & 1 \\
1 & 0
\end{bmatrix}
\]

Orbits of length 9:
Orbit 0: (0, 2, 3, 29, 48, 55, 60, 67)
Stabilizer of orbit representative 0:
Strong generators for a group of order 168:
\[
\begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix}, \begin{bmatrix}
\gamma^6 & 0 \\
0 & 1
\end{bmatrix}, \begin{bmatrix}
\gamma^4 & 0 \\
0 & 1
\end{bmatrix}
\]

1,0,0,1,1,
1,0,0,2,0,
1,0,3,5,0,
Generator 0 / 3 is:
\[
\begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix}
\]
Generator 1 / 3 is:
\[
\begin{bmatrix}
\gamma^6 & 0 \\
0 & 1
\end{bmatrix}
\]
Generator 2 / 3 is:
\[
\begin{bmatrix}
\gamma^4 & 0 \\
\gamma^2 & 1
\end{bmatrix}
\]
Orbits of length 63:
Orbit 2: ( 4, 5, 18, 7, 57, 25, 11, 37, 56, 10, 8, 33, 66, 45, 32, 41, 34, 14, 64, 9, 30, 47, 68, 52, 59, 71, 62, 6, 49, 65, 26, 21, 72, 54, 39, 13, 20, 43, 70, 50, 61, 17, 22, 44, 35, 23, 46, 40, 51, 28, ...12, 31, 16 )
Stabilizer of orbit representative 4:
Strong generators for a group of order 24:
\[
\begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix}, \begin{bmatrix}
1 & 0 \\
\gamma^5 & 1
\end{bmatrix}, \begin{bmatrix}
1 & 0 \\
\gamma^3 & 1
\end{bmatrix}, \begin{bmatrix}
1 & 0 \\
\gamma & 1
\end{bmatrix}
\]
1,0,0,1,1,
1,0,3,1,2,
1,0,5,1,0,
1,0,2,1,0,
Generator 0 / 4 is:

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


Generator 1 / 4 is:

\[
\begin{bmatrix}
1 & 0 \\
\gamma^5 & 1
\end{bmatrix}
\]


Generator 2 / 4 is:

\[
\begin{bmatrix}
1 & 0 \\
\gamma^3 & 1
\end{bmatrix}
\]


Generator 3 / 4 is:

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

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 $P$ if for all $x, y \in P$ and all $g \in G$,

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

For background on poset actions, see Plesken [56]. The orbits of $G$ on $P$ 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 one 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 [60] 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 PG(2, 2) explained in Section 4.1.
The commands shown in Tables 6.2-6.3 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>Modifier</th>
<th>Args</th>
<th>Meaning</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>-draw_options</td>
<td>options</td>
<td>Drawing options according to Table 16.2.</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>f_name</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>-write_tree</td>
<td></td>
<td>Write the poset of orbits as a tree file.</td>
</tr>
<tr>
<td>-find_node_by_stabilizer_order</td>
<td>i</td>
<td>Find all nodes whose stabilizer has order i.</td>
</tr>
<tr>
<td>-draw_poset</td>
<td></td>
<td>Produce a drawing of the poset of orbits.</td>
</tr>
<tr>
<td>-draw_full_poset</td>
<td></td>
<td>Produce a drawing of the full poset with elements grouped by orbits.</td>
</tr>
<tr>
<td>-plesken</td>
<td></td>
<td>Compute Plesken matrices Asup and Ainf.</td>
</tr>
<tr>
<td>-print_data_structure</td>
<td></td>
<td>Print the data structure.</td>
</tr>
<tr>
<td>-list</td>
<td></td>
<td>List orbits at level d.</td>
</tr>
</tbody>
</table>

Table 6.2: Options to control the poset classification algorithm (Part 1)
<table>
<thead>
<tr>
<th>Modifier</th>
<th>Args</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-list_all</code></td>
<td></td>
<td>List orbits at all levels.</td>
</tr>
<tr>
<td><code>-table_of_nodes</code></td>
<td></td>
<td>Produce a spreadsheet of all orbits.</td>
</tr>
<tr>
<td><code>-make_relations_</code></td>
<td></td>
<td>Produce a bitmap drawing of the neighboring relations in the poset with flag orbits.</td>
</tr>
<tr>
<td><code>-Kramer_Mesner_</code></td>
<td>$t \ k$</td>
<td>Compute the Kramer-Mesner matrix $M_{t,k}$.</td>
</tr>
<tr>
<td><code>-level_summary_csv</code></td>
<td></td>
<td>Write a summary of number of orbits at each level to a csv file.</td>
</tr>
<tr>
<td><code>-orbit_reps_csv</code></td>
<td></td>
<td>Write orbit representatives to a csv file.</td>
</tr>
<tr>
<td><code>-report</code></td>
<td><code>-end</code></td>
<td>Produce a latex report. Requires -orbiter_path option from Section 2.2.</td>
</tr>
<tr>
<td><code>-node_label_</code></td>
<td><code>is_group_order</code></td>
<td>When drawing the poset of orbits, display the group order in the orbit nodes.</td>
</tr>
<tr>
<td><code>-node_label_</code></td>
<td><code>is_element</code></td>
<td>When drawing the poset of orbits, display the element rank in the orbit nodes.</td>
</tr>
<tr>
<td><code>-show_orbit_</code></td>
<td><code>decomposition</code></td>
<td>Show the orbits of the stabilizers.</td>
</tr>
<tr>
<td><code>-show_stab</code></td>
<td></td>
<td>Show the stabilizers.</td>
</tr>
<tr>
<td><code>-save_stab</code></td>
<td></td>
<td>Save the stabilizer generators.</td>
</tr>
<tr>
<td><code>-show_whole_orbits</code></td>
<td></td>
<td>Show the whole orbits.</td>
</tr>
<tr>
<td><code>-recognize</code></td>
<td>$L$</td>
<td>Recognize the given object in the classified list and compute a transporter that maps the given object to the canonical form. Here, $L$ must be a list of integers (comma separated and enclosed in double quotes) encoding an object. This option can be repeated.</td>
</tr>
<tr>
<td><code>-export_</code></td>
<td><code>schreier_trees</code></td>
<td>Export all Schreier trees.</td>
</tr>
<tr>
<td><code>-draw_</code></td>
<td><code>schreier_trees</code></td>
<td>Draw all Schreier trees.</td>
</tr>
<tr>
<td><code>-preferred_choice</code></td>
<td>$n \ a \ b$</td>
<td>At node $n$, choose $b$ instead of $a$ as orbit representative.</td>
</tr>
</tbody>
</table>

Table 6.3: Options to control the poset classification algorithm (Part 2)
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:

\[
x, y \in \mathcal{P}(X) \text{ and } x \leq y \Rightarrow (y \in \mathcal{P} \rightarrow x \in \mathcal{P})
\]

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  ▶ ▶ -orbi
```
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 PGL(3, 2) on the subsets of lines of PG(2, 2).

PGL_3_2_on_lines:
▷ $(ORBITER) -v 3 \n▷ -orbiter_path $(ORBITER_PATH) \n▷ -define G -linear_group -PGL 3 2 -end \n▷ -define G_on_lines -modified_group -from G \n▷ -on_k_subspaces 2 \n▷ -end \n▷ -with G_on_lines -do \n▷ -group_theoretic_activity \n▷ -poset_classification_control \n▷ -problem_label PGL_3_2_lines -W -depth 7 \n▷ -draw_poset \n▷ -report -end \n▷ -end \n▷ -orbits_on_subsets 7 \n▷ -report \n▷ -end
▷ pdflatex PGL_3_2_lines_poset.tex
▷ open PGL_3_2_lines_poset.pdf

The following example computes the orbits of PGO(5, 2) on the power set lattice of points of Q(4, 2):

PGO_5_2_on_subsets:
▷ $(ORBITER) -v 3 \n▷ -orbiter_path $(ORBITER_PATH) \n▷ -define F -finite_field -q 2 -end \n▷ -define G -linear_group -PGO 5 F -end \n▷ -with G -do \n▷ -group_theoretic_activity \n▷ -poset_classification_control \n▷ -problem_label PGO_5_2 \n▷ -depth 15 \n▷ -draw_poset \n▷ -w \n▷ -end \n▷ -orbits_on_subsets 15 \n▷ -report \n▷ -end
▷ pdflatex 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:

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

$$P_0, P_1, P_2, P_3, P_4, P_5, P_7, P_9, P_{10}.$$  

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

```
subspaces_op_4_2:
   $\langle$ORBITER\rangle -v 5 \n   $\langle$orbiter_path $\langle$ORBITER_PATH\rangle \n   $\langle$define G -linear_group -PGL 4 2 -orthogonal 1 -end \n   $\langle$with G -do \n```
Figure 6.8: Hasse-diagram for the orbits of the orthogonal group $\text{PGO}^+(4, 2)$ on subspaces of $\text{PG}(3, 2)$

```
\begin{verbatim}
▷ ▷ -group_theoretic_activity \\
▷ ▷ -poset_classification_control \\
▷ ▷ ▷ -node_label_is_element \\
▷ ▷ ▷ -draw_poset -draw_options -radius 200 -end \\
▷ ▷ ▷ -problem_label Op_4_2 -W -depth 4 \\
▷ ▷ ▷ -report -end \\
▷ ▷ -end \\
▷ ▷ -orbits_on_subspaces 4 \\
▷ ▷ -report \\
▷ ▷ -end
\end{verbatim}
```

produces a classification of all subspaces of $\text{PG}(3, 2)$ under $\text{PGO}^+(4, 2)$. The option `-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

```
C6_on_partition:
  ▶ $(ORBITER) -v 5 \n  ▶ ▶ -orbiter_path $(ORBITER_PATH) \n  ▶ ▶ -define G -permutation_group -cyclic_group 6 -end \n  ▶ ▶ -with G -do \n  ▶ ▶ -group_theoretic_activity \n  ▶ ▶ ▶ -poset_classification_control \n  ▶ ▶ ▶ ▶ -problem_label C6 \n  ▶ ▶ ▶ ▶ -depth 2 \n  ▶ ▶ ▶ ▶ -W \n  ▶ ▶ ▶ ▶ -draw_options \n  ▶ ▶ ▶ ▶ ▶ -radius 200 -embedded \n  ▶ ▶ ▶ ▶ -end \n  ▶ ▶ ▶ -end \n  ▶ ▶ -orbits \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

```
PGL_2_17_on_partition:
  ▶ $(ORBITER) -v 5 \n  ▶ ▶ -define G -linear_group -PGL 2 17 -end \n  ▶ ▶ -with G -do \n  ▶ ▶ -group_theoretic_activity \n  ▶ ▶ ▶ -poset_classification_control \n  ▶ ▶ ▶ ▶ -problem_label PGL_2_17 \n  ▶ ▶ ▶ ▶ -depth 6 \n  ▶ ▶ ▶ ▶ -W \n  ▶ ▶ ▶ ▶ -end \n  ▶ ▶ ▶ -orbits_on_partition 6 \n  ▶ ▶ -end
```

209
computes the orbits of the group PGL(2, 17) on set-partitions of type 6 + 6 + 6. The number of set-partitions is
\[ \binom{18}{6} \cdot \binom{12}{6} / 3! = 2858856 \]

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>Args</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-q</td>
<td>q</td>
<td>Specify the size of the field $\mathbb{F}_q$.</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>-n</td>
<td>n</td>
<td>The size of the matrix group.</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>-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>-forbidden_point_set</td>
<td>set</td>
<td>The arc must not contain any of the given points.</td>
</tr>
</tbody>
</table>

Table 6.4: Commands for Classifying Arcs

### 6.6 Arcs and Caps in Projective Spaces

In $\text{PG}(n,q)$, 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 $\text{PG}(2,q)$. 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 $\text{PG}(2,16)$ (cf. [48]) and the Pellegrino cap in $\text{AG}(4,3)$. 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 very line intersects $S$ in at most $d$ points. Arcs are related to linear codes and other structures. Two arcs $S_1$ and $S_2$ are equivalent if there is a projectivity $\Phi$ 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 $\text{PG}(2,2^e)$ is a $(2^e + 2,2)$-arc. It is interesting to classify the hyperovals up to collineation equivalence under the group $\text{PGL}(3,2^e)$. The command

```
subspaces.Op.4.2:
▷ $(\text{ORBITER}) -v 5 \$
▷ ▷ -orbiter_path $(\text{ORBITER\_PATH}) \$
▷ ▷ -define G -linear_group -PGL 4 2 -orthogonal 1 -end \$
▷ ▷ -with G -do \$
```

211
performs the classification of hyperovals in PG(2, 16). There are exactly two hyperovals in this plane. Orbiter also finds the stabilizers of these arcs. They have orders 16320 and 144, respectively. The two hyperovals are the regular hyperoval and the Lunelli-Sce hyperoval. Here is the relevant output from the Orbiter report (in the output, the Lunelli-Sce hyperoval is orbit 0, and the regular hyperoval is orbit 1):

**Orbits at Level 18**

There are 2 orbits at level 18.

**Orbit 0 / 2 at Level 18**

Node number: 4212

\[
\{0, 1, 2, 3, 52, 67, 89, 106, 126, 141, 159, 176, 184, 199, 220, 235, 245, 262\}_{144}
\]

\begin{align*}
0 : 0 &= (1, 0, 0) & 10 : 159 &= (14, 8, 1) \\
1 : 1 &= (0, 1, 0) & 11 : 176 &= (15, 9, 1) \\
2 : 2 &= (0, 0, 1) & 12 : 184 &= (7, 10, 1) \\
3 : 3 &= (1, 1, 1) & 13 : 199 &= (6, 11, 1) \\
4 : 52 &= (3, 2, 1) & 14 : 220 &= (11, 12, 1) \\
5 : 67 &= (2, 3, 1) & 15 : 235 &= (10, 13, 1) \\
6 : 89 &= (8, 4, 1) & 16 : 245 &= (4, 14, 1) \\
7 : 106 &= (9, 5, 1) & 17 : 262 &= (5, 15, 1) \\
8 : 126 &= (13, 6, 1) \\
9 : 141 &= (12, 7, 1)
\end{align*}
Strong generators for a group of order 144:

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

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\}_{16320}

0 : 0 = ( 1, 0, 0 )
1 : 1 = ( 0, 1, 0 )
2 : 2 = ( 0, 0, 1 )
3 : 3 = ( 1, 1, 1 )
4 : 52 = ( 3, 2, 1 )
5 : 70 = ( 5, 3, 1 )
6 : 83 = ( 2, 4, 1 )
7 : 109 = ( 12, 5, 1 )
8 : 127 = ( 14, 6, 1 )
9 : 139 = ( 10, 7, 1 )
10 : 156 = ( 11, 8, 1 )
11 : 174 = ( 13, 9, 1 )
12 : 186 = ( 9, 10, 1 )
13 : 199 = ( 6, 11, 1 )
14 : 217 = ( 8, 12, 1 )
15 : 229 = ( 4, 13, 1 )
16 : 256 = ( 15, 14, 1 )
17 : 264 = ( 7, 15, 1 )

Strong generators for a group of order 16320:

\[
\begin{bmatrix}
\delta^6 & 0 & 0 \\
0 & \delta^3 & 0 \\
0 & 0 & 1
\end{bmatrix},
\begin{bmatrix}
\delta^9 & 0 & 0 \\
0 & \delta^7 & 0 \\
0 & 0 & 1
\end{bmatrix},
\begin{bmatrix}
\delta^2 & 0 & 0 \\
0 & \delta^{11} & 0 \\
\delta^4 & \delta^7 & 1
\end{bmatrix},
\begin{bmatrix}
\delta^{10} & 0 & 0 \\
0 & \delta^3 & 0 \\
\delta & \delta^{11} & 1
\end{bmatrix},
\begin{bmatrix}
\delta^{12} & 0 & 0 \\
\delta^{14} & \delta^{10} & 1 \\
\delta^{16} & \delta^8 & 1
\end{bmatrix},
\begin{bmatrix}
\delta^{12} & 1 & \delta^2 \\
\delta^{14} & \delta^{10} & 1 \\
\delta^6 & \delta^3 & 1
\end{bmatrix}
\]

213
In the theory of cubic surfaces, we are interested in non-conical arcs. These are arcs which do not lie on a conic. The following example demonstrates how this can be done in Orbiter:

\begin{verbatim}
nc_arcs_16:
   \$ (ORBITER) -v 4 \$
   \$(define F -finite_field -q 16 -end) \$
   \$(define P -projective_space -n 2 -field F -v 0 -end) \$
   \$(with P -do) \$
   \$(projective_space_activity) \$
   \$(classify_arcs) \$
   \$(poset_classification_control) \$
   \$(problem_label nc_arcs_q16_d2) \$
   \$(problem_depth 6) \$
   \$(report) \$
   \$(end) \$
   \$(target_size 6) \$
   \$(d 2) \$
   \$(conic_test) \$
   \$(end) \$
   \$ pdflatex nc_arcs_q16_d2_poset.tex \$
   \$ open nc_arcs_q16_d2_poset.pdf \$
\end{verbatim}

Cubic surfaces are associated with arcs of size 6 (in a many-to-one relationship when considering isomorphism classes). 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 $\mathbb{F}_{32}$:

\begin{verbatim}
nc_arcs_32_E13:
\end{verbatim}
\begin{verbatim}
$\text{(ORBITER)} -v 4 \\
  -orbiter_path $(ORBITER_PATH) \\
  -define F -finite_field -q 32 -end \\
  -define P -projective_space -n 2 -field F -v 0 -end \\
  -with P -do \\
  -projective_space_activity \\
  -classify_arcs \\
  -poset_classification_control \\
  -problem_label nc_arcs_q32_d2 \\
  -W -depth 6 \\
  -draw_poset -draw_options -end \\
  -report -end \\
  -end \\
  -target_size 6 \\
  -test_nb_Eckardt_points 13 \\
  -d 2 \\
  -conic_test \\
  -end \\
  -end
\end{verbatim}

$pdflatex\text{ nc_arcs_q32_d2_poset.tex}$

$open\text{ nc_arcs_q32_d2_poset.pdf}$
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 G \n  \> -define F -finite_field -q 8 -end \n  \> -define P -projective_space -n 2 -field F -v 0 -end \n  \> -with P -do \n  \> -projective_space_activity \n  \> -classify_cubic_curves -q 8 -target_size 9 -n 3 -d 3 \n  \> -poset_classification_control \n  \> -problem_classification_label cc_8 -W -depth 9 \n  \> -recognize "0,1,2,3,35,28" \n  \> -draw_poset \n  \> -draw_options -radius 200 -embedded -end \n  $(\text{ORBITER})$ -v 2 -draw_matrix \n  \> -input_csv_file cc_8_KM_6_9.csv \n  \> -box_width 50 -bit_depth 8 -end
```

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

# the 6-set is orbit 7
# the 9-set is orbit 1

\[216\]
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 7.1-7.2 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. Let us look at some examples. The next command creates the unique surface with 27 lines over the field $\mathbb{F}_4$, the Hirschfeld surface. 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:
$\langle$ ORBITER $\rangle$ -v 3 \n$\langle$ -define F -finite_field -q 4 -end \n$\langle$ -define P -projective_space -n 3 -field F -v 0 -end \n$\langle$ -with P -do \n$\langle$ -projective_space_activity \n$\langle$ -define_surface S -q 4 -catalogue 0 -end \n$\langle$ -end \n$\langle$ -with S -do \n$\langle$ -cubic_surface_activity \n$\langle$ -report \n$\langle$ -report_with_group \n$\langle$ -end \n$\langle$ pdflatex surface_catalogue_q4_iso0_report.tex \n$\langle$ open surface_catalogue_q4_iso0_report.pdf \n
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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 $\text{PG}(3,q)$. 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>-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>
<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_</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>abcd</td>
<td></td>
<td></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>
</tbody>
</table>

Table 7.1: Commands to create a known cubic surface (Part 1)
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-arc_lifting_with_two_lines</td>
<td>$A, L$</td>
<td>Create the surface associated with the arc $a_1, \ldots, a_6$ 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 two-skew-lines algorithm to create the surface. Here, $A$ is a comma-separated string containing the numerical ranks of the $P_i$ in $\text{PG}(3,q)$ and $L$ is a comma-separated string of the numerical ranks of two lines in $\text{PG}(3,q)$. If both of the lines are given as 0, the program will pick a suitable set of lines automatically.</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>-transform</td>
<td>$A$</td>
<td>Transform the surface by the projectivity (or collineation) defined by $A$. This option can be repeated.</td>
</tr>
<tr>
<td>-transform_inverse</td>
<td>$A$</td>
<td>Transform the surface by the inverse projectivity (or collineation) defined by $A$. This option can be repeated.</td>
</tr>
</tbody>
</table>

Table 7.2: Commands to create a known cubic surface (Part 2)
Two reports are created, one with information about the group and the other without it.

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

eckardt_{13,4,12}:

```
$\texttt{ORBITER} -v 6  \\
$\texttt{-define F -finite_field -q 13 -end  }  \\
$\texttt{-define P -projective_space -n 3 -field F -v 0 -end  }  \\
$\texttt{-with P -do  }  \\
$\texttt{-projective_space_activity  }  \\
$\texttt{-define_surface S_{2,1} -q 13  }  \\
$\texttt{-family_Eckardt 4 12 -end  }  \\
$\texttt{-end  }  \\
$\texttt{-with S_{2,1} -do  }  \\
$\texttt{-cubic_surface_activity  }  \\
$\texttt{-report  }  \\
$\texttt{-report_with_group  }  \\
$\texttt{-end  }  \\
```

creates the member of the Eckardt family described in [12] with parameters \((a,b) = (4,12)\) over the field \(\mathbb{F}_{13}\).

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

\[
F_{abcd\text{ eqn}}=-(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*(b - d)*X0*X0*X2 \\
+ (a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*(a + b - c - d)*X0*X1*X2 \\
+ (a*a*c - a*a*d - a*c*c + b*c*c + a*d - b*c)*(b - d)*X0*X1*X3 \\
- (a*d - b*c)*(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*X0*X2*X2 \\
- (a*a*c*d - a*b*c*c - a*a*d + a*b*d + b*c*c - b*c*d)*(b - d)*X0*X2*X3 \\
- (a - c)*(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*X1*X1*X2 \\
- (a - c)*(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*X1*X1*X3 \\
+ (a*d - b*c)*(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*X1*X2*X2 \\
+ ((1+1)*a*a*b*c*d - a*a*b*d*d - (1+1)*a*a*b*c*c) \\
- (1+1)*a*b*b*c*c + a*b*b*c*d + (1+1)*a*b*c*c*d + a*b*c*d*d \\
- b*b*c*c*d - a*a*b*c + a*a*c*d + a*a*d*d + a*b*b*c + a*b*c*c \\
- (1+1+1+1)*a*b*c*d - a*c*c*d + a*c*d*d + b*b*b*c)*X1*X2*X3 \\
+ c*a*(a*d - b*c - a + b + c - d)*(b - d)*X1*X3*X3
\]

The following command parses the formula and creates the surface with \((a,b,c,d) = (4,2,2,4)\):
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.

F_alpha_beta_gamma_delta_q7.override_group:
  $\$(ORBITER) -v 3 \
  -define F -finite_field -q 7 -end \n  -with F -do \n  -finite_field_activity \n  -parse_and_evaluate "Fabcd" "X0,X1,X2,X3" \n  $F_{abcd_eqn} "a=4,b=2,c=2,d=4" \n  -end

pdflatex surface_equation_F_alpha_beta_gamma_delta_q7_report.tex
open surface_equation_F_alpha_beta_gamma_delta_q7_report.pdf
pdflatex surface_equation_F_alpha_beta_gamma_delta_q7_with_group.tex
open surface_equation_F_alpha_beta_gamma_delta_q7_with_group.pdf
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.

Let us first look at the built-in catalogue. After that, we will look at the problem of classifi-
cation of quartic curves.

Suppose we want to study the (unique) quartic curve for \( q = 9 \). The following command pulls
the curve from the catalogue and produces a report:

```
quartic_curve_9_0_report:
  $(ORBITER) -v 3 \n  \quad -define F -finite_field -q 9 -end \n  \quad -define P -projective_space -n 2 -field F -v 0 -end \n  \quad -with P -do \n  \quad \quad -projective_space_activity \n  \quad \quad -define_quartic_curve C -q 9 \n  \quad \quad -catalogue 0 -end \n  \quad \quad -end \n  \quad -with C -do \n  \quad -quartic_curve_activity \n  \quad -report \n  \quad -end
  pdflatex quartic_curve_catalogue_q9_iso0_report.tex
  open quartic_curve_catalogue_q9_iso0_report.pdf
```

The report contains the following information:

**The equation**

The equation of the quartic curve is:

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

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

**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, 0)

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

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

General information

<table>
<thead>
<tr>
<th>Number of bitangents</th>
<th>28</th>
</tr>
</thead>
<tbody>
<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:

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

The points by rank are: (0, 1, 2, 3, 4, 5, 14, 17, 24, 25, 30, 32, 34, 38, 41, 44, 46, 51, 53, 57, 58, 62, 76, 77, 78, 82, 83, 84)

The Kovalevski points are:
0 : \( P_7 = (4, 1, 0) = c_{13} \cap c_{14} \cap c_{36} \cap c_{46} \)
1 : \( P_8 = (5, 1, 0) = a_2 \cap a_4 \cap c_{25} \cap c_{45} \)
2 : \( P_9 = (6, 1, 0) = b_1 \cap b_6 \cap c_{12} \cap c_{26} \)
$$P_{10} = (7, 1, 0) = a_3 \cap b_5 \cap c_{35} \cap d$$
$$P_{11} = (8, 1, 0) = b_2 \cap b_3 \cap c_{24} \cap c_{34}$$
$$P_{12} = (1, 0, 1) = a_3 \cap a_4 \cap c_{23} \cap c_{24}$$
$$P_{13} = (2, 0, 1) = c_{34} \cap c_{36} \cap c_{45} \cap c_{56}$$
$$P_{15} = (4, 0, 1) = b_3 \cap b_6 \cap c_{13} \cap c_{16}$$
$$P_{16} = (5, 0, 1) = a_5 \cap a_6 \cap c_{25} \cap c_{26}$$
$$P_{18} = (7, 0, 1) = a_2 \cap b_1 \cap c_{35} \cap c_{46}$$
$$P_{19} = (8, 0, 1) = b_4 \cap b_5 \cap c_{14} \cap c_{15}$$
$$P_{20} = (0, 1, 1) = a_2 \cap b_3 \cap c_{14} \cap c_{56}$$
$$P_{21} = (2, 1, 1) = b_2 \cap b_4 \cap c_{26} \cap c_{46}$$
$$P_{22} = (3, 1, 1) = a_4 \cap b_5 \cap c_{12} \cap c_{36}$$
$$P_{23} = (4, 1, 1) = a_6 \cap b_1 \cap c_{23} \cap c_{45}$$
$$P_{26} = (7, 1, 1) = c_{16} \cap c_{25} \cap c_{34} \cap d$$
$$P_{27} = (8, 1, 1) = a_3 \cap a_5 \cap c_{13} \cap c_{15}$$
$$P_{28} = (0, 2, 1) = c_{12} \cap c_{13} \cap c_{25} \cap c_{35}$$
$$P_{29} = (1, 2, 1) = b_1 \cap b_5 \cap c_{16} \cap c_{56}$$
$$P_{31} = (3, 2, 1) = a_3 \cap a_6 \cap c_{34} \cap c_{46}$$
$$P_{33} = (5, 2, 1) = a_2 \cap b_4 \cap c_{24} \cap d$$
$$P_{35} = (7, 2, 1) = b_2 \cap b_6 \cap c_{23} \cap c_{36}$$
$$P_{36} = (8, 2, 1) = a_4 \cap b_3 \cap c_{15} \cap c_{26}$$
$$P_{37} = (0, 3, 1) = a_5 \cap b_1 \cap c_{24} \cap c_{36}$$
$$P_{39} = (2, 3, 1) = a_2 \cap a_6 \cap c_{12} \cap c_{16}$$
$$P_{40} = (3, 3, 1) = b_3 \cap b_4 \cap c_{35} \cap c_{45}$$
$$P_{42} = (5, 3, 1) = a_4 \cap b_2 \cap c_{13} \cap c_{56}$$
$$P_{43} = (6, 3, 1) = a_3 \cap b_6 \cap c_{14} \cap c_{25}$$
$$P_{45} = (8, 3, 1) = c_{15} \cap c_{23} \cap c_{46} \cap d$$
$$P_{47} = (1, 4, 1) = a_6 \cap b_2 \cap c_{14} \cap c_{35}$$
$$P_{48} = (2, 4, 1) = b_3 \cap b_5 \cap c_{23} \cap c_{25}$$
$$P_{49} = (3, 4, 1) = a_5 \cap b_6 \cap c_{56} \cap d$$
$$P_{50} = (4, 4, 1) = a_2 \cap a_3 \cap c_{26} \cap c_{36}$$
$$P_{52} = (6, 4, 1) = b_1 \cap b_4 \cap c_{13} \cap c_{34}$$
$$P_{54} = (8, 4, 1) = c_{12} \cap c_{15} \cap c_{24} \cap c_{45}$$
$$P_{55} = (0, 5, 1) = a_4 \cap a_6 \cap b_4 \cap b_6$$
$$P_{56} = (1, 5, 1) = c_{13} \cap c_{26} \cap c_{45} \cap d$$
$$P_{59} = (4, 5, 1) = c_{24} \cap c_{25} \cap c_{46} \cap c_{56}$$
$$P_{60} = (5, 5, 1) = c_{12} \cap c_{14} \cap c_{23} \cap c_{34}$$
$$P_{61} = (6, 5, 1) = a_2 \cap a_5 \cap b_2 \cap b_5$$
$$P_{63} = (8, 5, 1) = c_{15} \cap c_{16} \cap c_{35} \cap c_{36}$$
$$P_{64} = (0, 6, 1) = a_1 \cap b_5 \cap c_{26} \cap c_{34}$$
$$P_{65} = (1, 6, 1) = a_1 \cap b_4 \cap c_{25} \cap c_{36}$$
$$P_{66} = (2, 6, 1) = a_1 \cap b_6 \cap c_{24} \cap c_{35}$$
$$P_{67} = (3, 6, 1) = a_1 \cap a_2 \cap c_{13} \cap c_{23}$$
$$P_{68} = (4, 6, 1) = a_1 \cap b_2 \cap c_{12} \cap d$$
$$P_{69} = (5, 6, 1) = a_1 \cap a_3 \cap b_1 \cap b_3$$
47 : $P_{70} = (6, 6, 1) = a_1 \cap a_4 \cap c_{16} \cap c_{46}$
48 : $P_{71} = (7, 6, 1) = a_1 \cap a_5 \cap c_{14} \cap c_{45}$
49 : $P_{72} = (8, 6, 1) = a_1 \cap a_6 \cap c_{15} \cap c_{56}$
50 : $P_{73} = (0, 7, 1) = a_3 \cap b_2 \cap c_{16} \cap c_{45}$
51 : $P_{74} = (1, 7, 1) = a_5 \cap b_3 \cap c_{12} \cap c_{46}$
52 : $P_{75} = (2, 7, 1) = a_1 \cap a_6 \cap c_{14} \cap c_{45}$
53 : $P_{79} = (6, 7, 1) = c_{23} \cap c_{26} \cap c_{35} \cap c_{56}$
54 : $P_{80} = (7, 7, 1) = a_6 \cap b_5 \cap c_{13} \cap c_{24}$
55 : $P_{81} = (8, 7, 1) = a_2 \cap b_6 \cap c_{15} \cap c_{34}$
56 : $P_{85} = (3, 8, 1) = c_{14} \cap c_{16} \cap c_{24} \cap c_{26}$
57 : $P_{86} = (4, 8, 1) = a_4 \cap a_5 \cap c_{34} \cap c_{35}$
58 : $P_{87} = (5, 8, 1) = b_5 \cap b_6 \cap c_{45} \cap c_{46}$
59 : $P_{88} = (6, 8, 1) = a_6 \cap b_3 \cap c_{36} \cap c_{d}$
60 : $P_{89} = (7, 8, 1) = a_3 \cap b_4 \cap c_{12} \cap c_{56}$
61 : $P_{90} = (8, 8, 1) = b_1 \cap b_2 \cap c_{15} \cap c_{25}$
62 : $P_{6} = (3, 1, 0) = a_5 \cap b_4 \cap c_{16} \cap c_{23}$

The Kovalevski points by rank are: ( 7, 8, 9, 10, 11, 12, 13, 15, 16, 18, 19, 20, 21, 22, 23, 26, 27, 28, 29, 31, 33, 35, 36, 37, 39, 40, 42, 43, 45, 47, 48, 49, 50, 52, 54, 55, 56, 59, 60, 61, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 79, 80, 81, 85, 86, 87, 88, 89, 90, 6 )

The points off the curve are:
The lines and their points of contact are:

<table>
<thead>
<tr>
<th>$n$</th>
<th>$P_n = (x, y, z)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$P_0 = (3, 1, 0)$</td>
</tr>
<tr>
<td>1</td>
<td>$P_1 = (4, 1, 0)$</td>
</tr>
<tr>
<td>2</td>
<td>$P_2 = (5, 1, 0)$</td>
</tr>
<tr>
<td>3</td>
<td>$P_3 = (6, 1, 0)$</td>
</tr>
<tr>
<td>4</td>
<td>$P_4 = (7, 1, 0)$</td>
</tr>
<tr>
<td>5</td>
<td>$P_5 = (8, 1, 0)$</td>
</tr>
<tr>
<td>6</td>
<td>$P_6 = (1, 0, 1)$</td>
</tr>
<tr>
<td>7</td>
<td>$P_7 = (2, 0, 1)$</td>
</tr>
<tr>
<td>8</td>
<td>$P_8 = (4, 0, 1)$</td>
</tr>
<tr>
<td>9</td>
<td>$P_9 = (5, 0, 1)$</td>
</tr>
<tr>
<td>10</td>
<td>$P_{10} = (7, 0, 1)$</td>
</tr>
<tr>
<td>11</td>
<td>$P_{11} = (8, 0, 1)$</td>
</tr>
<tr>
<td>12</td>
<td>$P_{12} = (0, 1, 1)$</td>
</tr>
<tr>
<td>13</td>
<td>$P_{13} = (2, 1, 1)$</td>
</tr>
<tr>
<td>14</td>
<td>$P_{14} = (3, 1, 1)$</td>
</tr>
<tr>
<td>15</td>
<td>$P_{15} = (4, 1, 1)$</td>
</tr>
<tr>
<td>16</td>
<td>$P_{16} = (7, 1, 1)$</td>
</tr>
<tr>
<td>17</td>
<td>$P_{17} = (8, 1, 1)$</td>
</tr>
<tr>
<td>18</td>
<td>$P_{18} = (0, 2, 1)$</td>
</tr>
<tr>
<td>19</td>
<td>$P_{19} = (1, 2, 1)$</td>
</tr>
<tr>
<td>20</td>
<td>$P_{20} = (3, 2, 1)$</td>
</tr>
<tr>
<td>21</td>
<td>$P_{21} = (5, 2, 1)$</td>
</tr>
</tbody>
</table>

(6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 18, 19, 20, 21, 22, 23, 26, 27, 28, 29, 31, 33, 35, 36, 37, 39, 40, 42, 43, 45, 47, 48, 49, 50, 52, 54, 55, 56, 59, 60, 61, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 79, 80, 81, 85, 86, 87, 88, 89, 90)

The lines and their points of contact are:

\[
\begin{align*}
\alpha_1 &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & \alpha^3 \\ 0 & \alpha^6 & 1 \end{bmatrix}^8 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 8 \end{bmatrix}^8 \quad P_0 = P(1, 0, 0) \quad 4 \\
\alpha_2 &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & \alpha^6 \\ 0 & 1 & 1 \end{bmatrix}^{51} = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 1 \end{bmatrix}^{51} \quad P_{83} = P(1, 8, 1) \quad 4 \\
\alpha_3 &= \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & \alpha^6 \\ 0 & 1 & 1 \end{bmatrix}^{15} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 5 \end{bmatrix}^{15} \quad P_{57} = P(2, 5, 1) \quad 4 \\
\alpha_4 &= \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & \alpha^2 \\ 0 & 1 & \alpha^6 \end{bmatrix}^{17} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 7 \end{bmatrix}^{17} \quad P_{53} = P(7, 4, 1) \quad 4 \\
\alpha_5 &= \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & \alpha^2 \\ 0 & 1 & \alpha^7 \end{bmatrix}^{74} = \begin{bmatrix} 1 & 0 & 7 \\ 0 & 1 & 4 \end{bmatrix}^{74} \quad P_{30} = P(2, 2, 1) \quad 4 \\
\alpha_6 &= \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & \alpha^2 \\ 0 & 1 & \alpha^2 \end{bmatrix}^{77} = \begin{bmatrix} 1 & 0 & 7 \\ 0 & 1 & 7 \end{bmatrix}^{77} \quad P_5 = P(2, 1, 0) \quad 4
\end{align*}
\]
\[
\begin{align*}
  b_1 &= \begin{bmatrix} 1 & 0 & \alpha^6 \\ 0 & 1 & \alpha^7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 4 \end{bmatrix} & P_{58} &= P(3, 5, 1) 4\times \\
  b_2 &= \begin{bmatrix} 1 & 0 & \alpha^7 \\ 0 & 1 & \alpha^6 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 4 \\ 0 & 1 & 5 \end{bmatrix} & P_{14} &= P(3, 0, 1) 4\times \\
  b_3 &= \begin{bmatrix} 1 & 0 & \alpha \\ 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 1 \end{bmatrix} & P_{62} &= P(7, 5, 1) 4\times \\
  b_4 &= \begin{bmatrix} 1 & 0 & \alpha^5 \\ 0 & 1 & \alpha^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 6 \\ 0 & 1 & 7 \end{bmatrix} & P_{77} &= P(4, 7, 1) 4\times \\
  b_5 &= \begin{bmatrix} 1 & 0 & \alpha^5 \\ 0 & 1 & \alpha^3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 8 \end{bmatrix} & P_{41} &= P(4, 3, 1) 4\times \\
  b_6 &= \begin{bmatrix} 1 & 0 & \alpha \\ 0 & 1 & \alpha^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 7 \end{bmatrix} & P_3 &= P(1, 1, 1) 4\times \\
  c_{12} &= \begin{bmatrix} 1 & 0 & \alpha^3 \\ 0 & 1 & \alpha^4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 8 \\ 0 & 1 & 2 \end{bmatrix} & P_{17} &= P(6, 0, 1) 4\times \\
  c_{13} &= \begin{bmatrix} 1 & 0 & \alpha \\ 0 & 1 & \alpha^4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 2 \end{bmatrix} & P_{84} &= P(2, 8, 1) 4\times \\
  c_{14} &= \begin{bmatrix} 1 & 0 & \alpha^5 \\ 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 6 \\ 0 & 1 & 1 \end{bmatrix} & P_{32} &= P(4, 2, 1) 4\times \\
  c_{15} &= \begin{bmatrix} 1 & 0 & \alpha^5 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 6 \\ 0 & 1 & 0 \end{bmatrix} & P_1 &= P(0, 1, 0) 4\times \\
  c_{16} &= \begin{bmatrix} 1 & 0 & \alpha \\ 0 & 1 & \alpha^6 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 5 \end{bmatrix} & P_{51} &= P(5, 4, 1) 4\times \\
  c_{23} &= \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & \alpha^5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 6 \end{bmatrix} & P_{82} &= P(0, 8, 1) 4\times \\
  c_{24} &= \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & \alpha^7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 4 \end{bmatrix} & P_{25} &= P(6, 1, 1) 4\times \\
  c_{25} &= \begin{bmatrix} 1 & 0 & \alpha^2 \\ 0 & 1 & \alpha^4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 7 \\ 0 & 1 & 2 \end{bmatrix} & P_{76} &= P(3, 7, 1) 4\times \\
  c_{26} &= \begin{bmatrix} 1 & 0 & \alpha^2 \\ 0 & 1 & \alpha^3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 7 \\ 0 & 1 & 8 \end{bmatrix} & P_{44} &= P(7, 3, 1) 4\times \\
  c_{34} &= \begin{bmatrix} 1 & 0 & \alpha^4 \\ 0 & 1 & \alpha^3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 8 \end{bmatrix} & P_{38} &= P(1, 3, 1) 4\times \\
  c_{35} &= \begin{bmatrix} 1 & 0 & \alpha^6 \\ 0 & 1 & \alpha^4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 2 \end{bmatrix} & P_{24} &= P(5, 1, 1) 4\times 
\end{align*}
\]
\[
c_{36} = \begin{bmatrix} 1 & 0 & \alpha^4 \\ 0 & 1 & \alpha^7 \end{bmatrix}_{24} = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 4 \end{bmatrix}_{24} \quad P_{78} = P(5, 7, 1) \times 4 \\
c_{45} = \begin{bmatrix} 1 & 0 & \alpha^4 \\ 0 & 1 & \alpha^6 \end{bmatrix}_{24} = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 5 \end{bmatrix}_{25} \quad P_{34} = P(6, 2, 1) \times 4 \\
c_{46} = \begin{bmatrix} 1 & 0 & \alpha^6 \\ 0 & 1 & \alpha \end{bmatrix}_{53} = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 3 \end{bmatrix}_{53} \quad P_{46} = P(0, 4, 1) \times 4 \\
c_{56} = \begin{bmatrix} 1 & 0 & \alpha^4 \\ 0 & 1 & 1 \end{bmatrix}_{21} = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 1 \end{bmatrix}_{21} \quad P_4 = P(1, 1, 0) \times 4 \\
d = \begin{bmatrix} 1 & \alpha^6 & 0 \\ 0 & 0 & 1 \end{bmatrix}_{59} = \begin{bmatrix} 1 & 5 & 0 \\ 0 & 0 & 1 \end{bmatrix}_{59} \quad P_2 = P(0, 0, 1) \times 4 \\
\]

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: 1²⁸

<table>
<thead>
<tr>
<th>28</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, 2, 3, 4, 5, 6, 7, 8, 9, 10,</td>
<td></td>
</tr>
<tr>
<td>11, 12, 13, 14, 15, 16, 17, 18, 19, 20,</td>
<td></td>
</tr>
<tr>
<td>21, 22, 23, 24, 25, 26, 27, 0</td>
<td></td>
</tr>
</tbody>
</table>

Point types: 1²⁸

<table>
<thead>
<tr>
<th>28</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, 2, 3, 4, 5, 6, 7, 8, 9, 10,</td>
<td></td>
</tr>
<tr>
<td>11, 12, 13, 14, 15, 16, 17, 18, 19, 20,</td>
<td></td>
</tr>
<tr>
<td>21, 22, 23, 24, 25, 26, 27, 0</td>
<td></td>
</tr>
</tbody>
</table>

Point types for points off the curve: 4⁶³

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

Lines on points off the curve:
Off point 0 = \( P_6 = (3, 1, 0) \) lies on 4 bisecants: \{ 4, 9, 16, 17 \}
Off point 1 = \( P_7 = (4, 1, 0) \) lies on 4 bisecants: \{ 13, 14, 23, 25 \}
Off point 2 = \( P_8 = (5, 1, 0) \) lies on 4 bisecants: \{ 1, 3, 19, 24 \}
Off point 3 = \( P_9 = (6, 1, 0) \) lies on 4 bisecants: \{ 6, 11, 12, 20 \}
Off point 4 = \( P_{10} = (7, 1, 0) \) lies on 4 bisecants: \{ 2, 10, 22, 27 \}
Off point 5 = \( P_{11} = (8, 1, 0) \) lies on 4 bisecants: \{7, 8, 18, 21\}
Off point 6 = \( P_{12} = (1, 0, 1) \) lies on 4 bisecants: \{2, 3, 17, 18\}
Off point 7 = \( P_{13} = (2, 0, 1) \) lies on 4 bisecants: \{21, 23, 24, 26\}
Off point 8 = \( P_{15} = (4, 0, 1) \) lies on 4 bisecants: \{8, 11, 13, 16\}
Off point 9 = \( P_{16} = (5, 0, 1) \) lies on 4 bisecants: \{4, 5, 19, 20\}
Off point 10 = \( P_{18} = (7, 0, 1) \) lies on 4 bisecants: \{1, 6, 22, 25\}
Off point 11 = \( P_{19} = (8, 0, 1) \) lies on 4 bisecants: \{9, 10, 14, 15\}
Off point 12 = \( P_{20} = (0, 1, 1) \) lies on 4 bisecants: \{1, 8, 14, 26\}
Off point 13 = \( P_{21} = (2, 1, 1) \) lies on 4 bisecants: \{7, 9, 20, 25\}
Off point 14 = \( P_{22} = (3, 1, 1) \) lies on 4 bisecants: \{3, 10, 12, 23\}
Off point 15 = \( P_{23} = (4, 1, 1) \) lies on 4 bisecants: \{5, 6, 17, 24\}
Off point 16 = \( P_{26} = (7, 1, 1) \) lies on 4 bisecants: \{16, 19, 21, 27\}
Off point 17 = \( P_{27} = (8, 1, 1) \) lies on 4 bisecants: \{2, 4, 13, 15\}
Off point 18 = \( P_{28} = (0, 2, 1) \) lies on 4 bisecants: \{12, 13, 19, 22\}
Off point 19 = \( P_{29} = (1, 2, 1) \) lies on 4 bisecants: \{6, 10, 16, 26\}
Off point 20 = \( P_{31} = (3, 2, 1) \) lies on 4 bisecants: \{2, 5, 21, 25\}
Off point 21 = \( P_{33} = (5, 2, 1) \) lies on 4 bisecants: \{1, 9, 18, 27\}
Off point 22 = \( P_{35} = (7, 2, 1) \) lies on 4 bisecants: \{7, 11, 17, 23\}
Off point 23 = \( P_{36} = (8, 2, 1) \) lies on 4 bisecants: \{3, 8, 15, 20\}
Off point 24 = \( P_{37} = (0, 3, 1) \) lies on 4 bisecants: \{4, 6, 18, 23\}
Off point 25 = \( P_{39} = (2, 3, 1) \) lies on 4 bisecants: \{1, 5, 12, 16\}
Off point 26 = \( P_{40} = (3, 3, 1) \) lies on 4 bisecants: \{8, 9, 22, 24\}
Off point 27 = \( P_{42} = (5, 3, 1) \) lies on 4 bisecants: \{3, 7, 13, 26\}
Off point 28 = \( P_{43} = (6, 3, 1) \) lies on 4 bisecants: \{2, 11, 14, 19\}
Off point 29 = \( P_{45} = (8, 3, 1) \) lies on 4 bisecants: \{15, 17, 25, 27\}
Off point 30 = \( P_{47} = (1, 4, 1) \) lies on 4 bisecants: \{5, 7, 14, 22\}
Off point 31 = \( P_{48} = (2, 4, 1) \) lies on 4 bisecants: \{8, 10, 17, 19\}
Off point 32 = \( P_{49} = (3, 4, 1) \) lies on 4 bisecants: \{4, 11, 26, 27\}
Off point 33 = \( P_{50} = (4, 4, 1) \) lies on 4 bisecants: \{1, 2, 20, 23\}
Off point 34 = \( P_{52} = (6, 4, 1) \) lies on 4 bisecants: \{6, 9, 13, 21\}
Off point 35 = \( P_{54} = (8, 4, 1) \) lies on 4 bisecants: \{12, 15, 18, 24\}
Off point 36 = \( P_{55} = (0, 5, 1) \) lies on 4 bisecants: \{3, 5, 9, 11\}
Off point 37 = \( P_{56} = (1, 5, 1) \) lies on 4 bisecants: \{13, 20, 24, 27\}
Off point 38 = \( P_{59} = (4, 5, 1) \) lies on 4 bisecants: \{18, 19, 25, 26\}
Off point 39 = \( P_{60} = (5, 5, 1) \) lies on 4 bisecants: \{12, 14, 17, 21\}
Off point 40 = \( P_{61} = (6, 5, 1) \) lies on 4 bisecants: \{1, 4, 7, 10\}
Off point 41 = \( P_{63} = (8, 5, 1) \) lies on 4 bisecants: \{15, 16, 22, 23\}
Off point 42 = \( P_{64} = (0, 6, 1) \) lies on 4 bisecants: \{0, 10, 20, 21\}
Off point 43 = \( P_{65} = (1, 6, 1) \) lies on 4 bisecants: \{0, 9, 19, 23\}
Off point 44 = \( P_{66} = (2, 6, 1) \) lies on 4 bisecants: \{0, 11, 18, 22\}
Off point 45 = \( P_{67} = (3, 6, 1) \) lies on 4 bisecants: \{0, 1, 13, 17\}
Off point 46 = \( P_{68} = (4, 6, 1) \) lies on 4 bisecants: \{0, 7, 12, 27\}
Off point 47 = \( P_{69} = (5, 6, 1) \) lies on 4 bisecants: \{0, 2, 6, 8\}
Off point 48 = \( P_{70} = (6, 6, 1) \) lies on 4 bisecants: \{0, 3, 16, 25\}
Regarding the problem of classification, we first fix the field order $q$ for which we want to classify the quartic curves. Next, we observe that quartic curves with 28 bitangents are related to cubic surfaces with 27 lines over the same field. This means we can exploit the previously classified list of cubic surfaces towards the goal of classifying quartic curves. For reasonably small field orders 2, the Orbiter knowledge base contains the classification of cubic surfaces with 27 lines over $\mathbb{F}_q$. So, because of this dependency, there is a restriction on the size of $q$ for which this algorithm can be applied. Next, we consider the list of cubic surfaces with 27 lines over the field $\mathbb{F}_q$. For each surface, and for each orbit on points not on lines, we perform a projection operation to create one quartic curve. This guarantees that every isomorphism type of quartic curve with 28 bitangents will be created.

Let us look at some examples of this algorithm. We try $q = 7$ and $q = 13$. In each case, we need a makefile variable to set 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

\[ \text{NB\_CUBIC\_SURFACES\_Q7=1} \]

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\} -v 3 \n  \$\{ORBITER\} -list_arguments \n  \$\{ORBITER\} -define F -finite_field -q 7 -end \n  \$\{ORBITER\} -define P -projective_space -n 3 -field F -v 0 -end \n```
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:

```bash
quartic_curves_q13:
  $(ORBITER) -v 3 \
  -list_arguments \
  -define F -finite_field -q 13 -end \
  -define P -projective_space -n 3 -field F -v 0 -end \
  -loop L 0 $(NB\_CUBIC\_SURFACES\_Q13) 1 \
  -with P -do \
  -projective_space_activity \
  -define_surface S13_\%L -q 13 -catalogue \%L -end \
  -end \
  -end_loop \
  -print_symbols \
  -loop L 0 $(NB\_CUBIC\_SURFACES\_Q13) 1 \
  -with S13_\%L -do \
  -cubic_surface_activity \
  -export_all_quartic_curves \
  -end \
```
The next command processes the curves that have been created and performs a classification up to isomorphism. The result is the classification of quartic curves with 28 bitangents over the field $\mathbb{F}_{13}$:

```bash
quartic_curves_q13_classify:
▷ $(ORBITER) -v 3 \\
▷ ▷ -list_arguments \\
▷ ▷ -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 \\
▷ ▷ ▷ ▷ surface_catalogue_q13_iso%d_quartics.csv \\
▷ ▷ ▷ ▷ $(NB_CUBIC_SURFACES_Q13) 4 4 quartic_curves_q13 \\
▷ ▷ ▷ -end \\
▷ ▷ -print_symbols
```

We find exactly two isomorphism types. The data is exported to C++ source code. The file `quartic_curves_q13.cpp` is created. This file is now part of Orbiter’s knowledge base of geometric objects.
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 [59]. 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:

```
surface_classify_q4:
▷ $(ORBITER) -v 5 \ 
▷ ▷ -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▷ ▷ ▷ -classify_surfaces_with_double_sixes Surf27 -W -end \n▷ ▷ -end \n▷ ▷ -with Surf27 -do \n▷ ▷ -classification_of_cubic_surfaces_with_double_sixes_activity \n▷ ▷ ▷ -report -end \n▷ ▷ -end \n▷ ▷ -print_symbols
▷ pdflatex Surfaces_q4.tex
▷ open Surfaces_q4.pdf
```

The `-report` option creates a latex report. After some redactions, the report contains the following elements.

The semilinear group

The Action

Group action $\text{PGL}(4,4)$ 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
:

**The orthogonal group**

**The Action**

Group action \( \mathcal{PG}(4, 4) \) on wedge of degree 1365

The group is a matrix group.
The base action is on projective space \( \mathcal{P}G(3, 4) \)

\[ q = 4 \]
\[ p = 2 \]
\[ e = 2 \]
\[ n = 3 \]
Number of points = 85
Number of lines = 357
Number of lines on a point = 21
Number of points on a line = 5
:

**The group stabilizing the fixed line**

**The Action**

Group action \( \mathcal{PG}(4, 4) \) on wedge res 100 of degree 100

:\

Strong generators for a group of order 5529600: :

**The classification of five-plus-ones**

Poset classification up to depth 5
The Orbits

Number of Orbits By Level

<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 7.3: 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>7</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>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>
</tbody>
</table>

235
| 10 | 5 | 3 | { 0, 3, 56, 80, 93 } | (120, 46080) | 7 |
Poset of Orbits in Detail

Classification of 5 + 1 Configurations in PG(3, 4)

The order of the group is 1974067200
The group has 4 orbits on five plus one configurations in 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 PG(3, 4) is: 46080

Flag orbits for double sixes

The number of primary orbits below is 4
The number of primary orbits above is 1
The number of flag orbits is 1
The flag orbits are:

(1) Flag orbit 0 / 1 down=(3,0) up=(0, -1) is (0, 3, 56, 80, 93, 16, 340, 38, 61, 156, 0, 16, 340, 38, 61, 156, 165, 72, 54, 25, 356, 0) with a stabilizer of order 120
Strong generators for a group of order 120:

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

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

The orbits are:

(1) 0/1 \{16, 340, 38, 61, 156, 165, 155, 72, 54, 25, 356, 0\} \text{ orbit length } 1370880

Strong generators for a group of order 1440:
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:

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

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

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

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

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

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

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} orbit length 38080

239
Strong generators for a group of order 51840:

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

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

The overall number of objects is: 38080

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

The order of the group is 1974067200

Cubic Surfaces with 27 Lines in 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:

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix},
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & \omega & 0 & 0 \\
0 & 0 & \omega & 0 \\
0 & 0 & 0 & 1
\end{bmatrix},
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & \omega^2 & 0 & 0 \\
0 & 0 & \omega^2 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix},
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & \omega & 0 & 0 \\
0 & 0 & \omega & 0 \\
0 & 0 & 0 & 1
\end{bmatrix},
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & \omega^2 & \omega & 1 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0
\end{bmatrix},
\begin{bmatrix}
\omega & \omega & 1 & 1 \\
0 & 1 & 0 & \omega \\
\omega & 0 & 0 & \omega \\
0 & 0 & 1 & 0
\end{bmatrix},
\begin{bmatrix}
\omega^2 & \omega & \omega^2 & 1 \\
\omega^2 & 0 & 1 & 0 \\
\omega & 1 & 0 & 0 \\
1 & 0 & 0 & 0
\end{bmatrix},
\begin{bmatrix}
\omega & \omega & 1 & 1 \\
0 & 1 & 0 & \omega \\
\omega & 0 & 0 & \omega \\
0 & 0 & 1 & 0
\end{bmatrix},
\begin{bmatrix}
1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1,
1,0,0,0,2,0,0,0,0,2,0,0,0,0,1,0,
1,0,0,0,3,0,0,0,0,3,0,1,0,0,1,0,
1,0,0,0,1,0,0,1,1,1,0,1,1,0,1,0,
1,0,0,0,3,2,2,0,0,0,2,0,1,0,3,1,0,
1,0,0,0,1,0,2,0,2,2,0,0,2,2,1,1,0,
1,3,1,2,1,0,2,0,3,2,0,0,2,0,0,0,0,
1,1,3,3,0,3,0,1,1,2,0,1,0,3,0,0,0,
\end{bmatrix}
\]

The overall number of objects is: 38080

**Surface 4#0**

**The equation**

The equation of the surface is:

\[X_0^2 X_3 + X_1^2 X_2 + X_1 X_2^2 + X_0 X_3^2 = 0\]

( 0, 0, 0, 0, 0, 0, 1, 0, 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:

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

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

General information

Points on lines: \[5^{27}\]

Lines on points: \[3^{45}\]

The 27 Lines

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

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

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

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

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

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

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

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

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

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

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

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

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

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

\[ \ell_{15} = c_{15} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}_{84} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}_{84} = P_l(1, 0, 0, 1, 0, 0)_{10} \]
\[ \ell_{16} = c_{16} = \begin{bmatrix} 1 & 0 & \omega^2 & 0 \\ 0 & 1 & 0 & \omega \end{bmatrix}, \quad \ell_{17} = c_{23} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}, \quad \ell_{18} = c_{24} = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}, \quad \ell_{19} = c_{25} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}, \quad \ell_{20} = c_{26} = \begin{bmatrix} 1 & 0 & \omega & 0 \\ 0 & 1 & 1 & \omega^2 \end{bmatrix}, \quad \ell_{21} = c_{34} = \begin{bmatrix} 1 & \omega & 0 & 1 \\ 0 & 0 & 1 & \omega^2 \end{bmatrix}, \quad \ell_{22} = c_{35} = \begin{bmatrix} 1 & 0 & \omega & 1 \\ 0 & 1 & 1 & \omega^2 \end{bmatrix}, \quad \ell_{23} = c_{36} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}, \quad \ell_{24} = c_{45} = \begin{bmatrix} 1 & \omega^2 & 0 & 0 \\ 0 & 0 & 1 & \omega \end{bmatrix}, \quad \ell_{25} = c_{46} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad \ell_{26} = c_{56} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix} \]

\[ \mathbf{P}(3, 2, 0, 0, 3, 1)_{299} \]
\[ \mathbf{P}(1, 1, 1, 1, 0, 0)_{16} \]
\[ \mathbf{P}(0, 1, 1, 1, 1, 1)_{202} \]
\[ \mathbf{P}(1, 0, 1, 1, 1, 1)_{199} \]
\[ \mathbf{P}(2, 3, 2, 0, 2, 1)_{244} \]
\[ \mathbf{P}(0, 3, 2, 3, 2, 1)_{271} \]
\[ \mathbf{P}(2, 0, 2, 3, 2, 1)_{267} \]
\[ \mathbf{P}(1, 1, 1, 0, 1, 1)_{180} \]
\[ \mathbf{P}(0, 0, 3, 2, 3, 1)_{332} \]
\[ \mathbf{P}(0, 1, 1, 0, 0, 0)_{6} \]
\[ \mathbf{P}(1, 0, 1, 0, 0, 0)_{3} \]

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 : $E_{32} = a_3 \cap b_2 \cap c_{23} = P_4 = P_4 = P(1,1,1,1) = P(1,1,1,1)$, $T = 27$
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 : $E_{33} = a_5 \cap b_3 \cap c_{35} = P_7 = P_7 = P(\omega^2,1,0,0) = P(3,1,0,0)$, $T = 3$
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 : $E_{65} = a_6 \cap b_5 \cap c_{56} = P_{13} = P_{13} = P(\omega,1,1,0) = P(2,1,1,0)$, $T = 12$
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 : $E_{25} = a_2 \cap b_6 \cap c_{25} = P_{26} = P_{48} = P(\omega^2,\omega,1,1) = P(3,2,1,1)$, $T = 80$
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 : $E_{35} = a_3 \cap b_5 \cap c_{35} = P_{31} = P_{59} = P(\omega,\omega,\omega,1) = P(2,1,2,1)$, $T = 49$
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 : $E_{13} = a_1 \cap b_3 \cap c_{13} = P_{35} = P_{67} = P(\omega,\omega^2,\omega,1) = P(2,3,2,1)$, $T = 50$
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 : $E_{12,36,45} = c_{12} \cap c_{36} \cap c_{45} = P_{39} = P_{75} = P(\omega,\omega^2,1) = P(2,1,3,1)$, $T = 59$
40 : $E_{34} = a_3 \cap b_4 \cap c_{34} = P_{40} = P_{76} = P(\omega^2,1,\omega^2,1) = P(3,1,3,1)$, $T = 71$
41: \( E_{24} = a_2 \cap b_4 \cap c_{24} = P_{41} = P_{79} = P(\omega, \omega, \omega^2, 1) = P(2, 2, 3, 1), \ T = 58 \)
42: \( E_{13,26,45} = c_{13} \cap c_{26} \cap c_{45} = P_{42} = P_{80} = P(\omega^2, \omega, \omega^2, 1) = P(3, 2, 3, 1), \ T = 70 \)
43: \( E_{14} = a_1 \cap b_4 \cap c_{14} = P_{43} = P_{81} = P(0, \omega^2, \omega^2, 1) = P(0, 3, 3, 1), \ T = 72 \)
44: \( E_{16,23,45} = c_{16} \cap c_{23} \cap c_{45} = P_{44} = P_{82} = P(1, \omega^2, \omega^2, 1) = P(1, 3, 3, 1). \ T = 33 \)
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:
\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 \\
0 & 0 & 1 & \omega
\end{bmatrix}
\]
:
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: \( E_{56}, E_{31}, E_{15,24,36}, E_{16,25,34}, E_{12}, E_{14,26,35}, E_{23}, E_{45}, E_{64} \)
:
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 = \( E_{23}, E_{31}, E_{12} \)
:
239: 239 = 119,1 = \( E_{12,36,45}, E_{14,26,35}, E_{13,25,46} \)

246
Tritangent planes

The 45 tritangent planes are:

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

\[ = V(\omega^2 X_0 + \omega^2 X_1 + X_2 + X_3) = V(3X_0 + 3X_1 + X_2 + X_3) \]

dual pt rank = 52 = (3, 3, 1, 1).

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

\[ = V(\omega X_0 + \omega X_1 + \omega^2 X_2 + X_3) = V(2X_0 + 2X_1 + 3X_2 + X_3) \]

dual pt rank = 79 = (2, 2, 3, 1).

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

```
surface classify_q4_arc_lifting_two_lines:
▷ $\$(ORBITER) -v 10 \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▷ ▷ ▷ -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. The only surface with 27 lines in PG(3, 4) is the Hirschfeld surface.
<table>
<thead>
<tr>
<th>Modifier</th>
<th>Arguments</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-surface_ identify_Eckardt</code></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><code>-surface_ identify_F13</code></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><code>-surface_ identify_Bes</code></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><code>-surface_ identify_general_ abcd</code></td>
<td>surface-descr-1 surface-descr-2</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><code>-surface_isomorphism_testing</code></td>
<td>surface-descr</td>
<td>Computes an isomorphism between two given surfaces or concludes that none exists.</td>
</tr>
<tr>
<td><code>-surface_recognize</code></td>
<td>surface-descr</td>
<td>Identifies the isomorphism type of the given surface.</td>
</tr>
<tr>
<td><code>-create_surface</code></td>
<td>surface-descr</td>
<td>Creates a surface from a description. See Section 7.1.</td>
</tr>
</tbody>
</table>

Table 7.4: Projective space activities related to the recognition of cubic surfaces

### 7.4 Isomorphism Testing and Recognition

Besides classification, Orbiter provides recognition, isomorphism testing and study of cubic surfaces. Table 7.4 lists the relevant Orbiter commands. These commands are projective space activities.

The `-surface_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,

```bash
surface_recognize_q7_abcd_2_3_3_4:
  $(ORBITER) -v 3 \n  $define F -finite_field -q 7 -end \n  $define P -projective_space -n 3 -field F -v 0 -end \n  $with P -do \n  $projective_space_activity \n  $with Surf -do \n  $classification_of_cubic_surfaces_with_double_sixes_activity \n  $recognize \n```

248
identifies the surface (cf. Table 4.3)

\[ X_0^2X_3 + X_1^2X_2 + X_1X_2^2 + X_0X_3^2 + X_1X_2X_3 = 0 \]  \hspace{1cm} (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

\[
\begin{bmatrix}
1 & 4 & 4 & 0 \\
6 & 0 & 0 & 0 \\
6 & 2 & 0 & 1 \\
7 & 0 & 4 & 0
\end{bmatrix}
\]  \hspace{1cm} (7.2)

Orbiter can compute isomorphism between two given surfaces. Both surfaces must have 27 lines. For instance, the command

```
surface_isomorph_16:
  \$ (ORBITER) -v 3 \n  -define F -finite_field -q 16 -end \n  -define P -projective_space -n 3 -field F -v 0 -end \n  -with P -do \n  -projective_space_activity \n  -classify_surfaces_with_double_sixes Surf27 -W -end \n  -end \n  -with Surf27 -do \n  -classification_of_cubic_surfaces_with_double_sixes_activity \n  -isomorphism_testing \n  -q 16 -by_coefficients \n  "1,5,1,8,1,9,1,10,1,11,1,12,6,14,6,15,7,18,7,19" -end \n  -q 16 -by_coefficients \n  "13,6,3,8,3,11,13,13,1,19" -end \n  -end \n  -print_symbols
```

249
computes an isomorphism between two cubic surfaces with 27 lines
\[X^2 \delta_0 X^2 + X^2 \delta_1 X^2 + X^2 \delta_2 X^2 + X_1 \delta_3 X^2 + X_2 \delta_4 X^2 + \delta \delta_7 X_1 X_2 X_3 + \delta \delta_7 X_1 X_2 X_3 = 0\]
and
\[\delta \delta_1 X^2 X_3 + \delta \delta_2 X^2 X_2 + \delta \delta_2 X^2 X_2 + \delta \delta_3 X_1 X_2 X_3 = 0\]
over the field \(F_{16}\).

Orbiter can recognize the isomorphism type of a cubic surface with 27 lines. This means that Orbiter can determine the Orbiter Catalogue Number of the surface in the catalogue which is isomorphic to the given surface. For instance, the following command determines the isomorphism type of the surface
\[X^2 \delta_0 X^2 + X^2 \delta_1 X^2 + X_1 \delta_2 X^2 + X_2 \delta_3 X^2 + X_1 X_2 X_3 = 0.\]

The command find that the surface is isomorphic to the surface with OCN=0. An isomorphism will be computed as well.

\[
\begin{bmatrix}
12 & 13 & 0 & 0 \\
8 & 13 & 0 & 0 \\
0 & 0 & 13 & 0 \\
12 & 13 & 11 & 1
\end{bmatrix}
\]
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 $\text{PG}(3, 2)$. 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 $\text{PG}(3, 2)$ were classified using this Orbiter command:

```
orbit_cubic_curves.q2:
  $(\text{ORBITER}) \ -v \ 4 \ \\
  \ -define \ G \ -linear_group \ -PGL \ 3 \ 2 \ -end \ \\
  \ -with \ G \ -do \ \\
  \ -group_theoretic_activity \ \\
  \ -orbits_on_polynomials \ 3 \ \\
  \ -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:

```
poly_orbits_d3_n3_q2_F2.csv: poly_orbits_d3_n3_q2.csv
  $(\text{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 \ \\
  \ -end
```

and

```
Dickson_q2_analyze: poly_orbits_d3_n3_q2_F2.csv
  $(\text{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 \ \\
  \ -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:

```
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

```
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 $q = 17$:

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

cubic_surfaces_tables_17:
```bash
$(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

```bash
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 factoring 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 \( \mathbb{F}_p[X] \).

The \(-\text{finite}_\text{field}_\text{activity}\) command can be used to define command requiring a finite field. The \(-q \) option can be used to specify the order of the finite field. The \(-\text{override}_\text{polynomial} \ a \) option can be used to specify the polynomial \( m(X) \) as integer \( a \) in the base \( p \) representation. This option can be ommitted, in which case Orbiter will use a precomputed and built-in polynomial. Table 8.1 lists Orbiter activities for polynomials over finite fields. For instance, the command

```
poly_division:
▷ $\text{ORBITER) -v 2 \n▷ ▷ -define F -finite_field -q 2 -end \n▷ ▷ -with F -do \n▷ ▷ -finite_field_activity \n▷ ▷ -polynomial_division "1,0,0,0,0,0,0,0,0,0,1" "1,0,1,1" -end
```

computes the polynomial long division of \( A(X) \) by \( B(X) \) over \( \mathbb{F}_2 \) where

\[
A(X) = X^{10} + 1, \quad B(X) = X^3 + X^2 + 1.
\]

The result is \( Q(X) \) and \( R(X) \) with

\[
A(X) = Q(X) \cdot B(X) + R(X)
\]

with

\[
Q(X) = X^7 + X^6 + X^5 + X^3 + 1, \quad R(X) = X^2.
\]

The coefficient lists in the arguments are from the lowest term up.
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Description</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>
<tr>
<td>-find_CRC_polynomials</td>
<td>$t \ n \ k$</td>
<td>Computes all CRC polynomials of degree $k$ over $\mathbb{F}_q$ who detect all error patterns of Hamming weight $t$ or less in messages of length $n$. See Section 10.4.</td>
</tr>
</tbody>
</table>

Table 8.1: Finite Field Activities Related to Polynomials
It is perhaps more convenient to use the vector builder from Section 2.7 to create the polynomials. The following example illustrates this. First, the coefficient vectors of the two polynomials are created using a define -define command. The vectors are symbolic variables named \( A \) and \( B \). After that, the division command is called as a finite field activity for \( F \). The division command creates the polynomials from the coefficient vectors automatically. Note the difference in how the vectors are created.

\[
\text{poly\_division2:}
\]
\[
\begin{align*}
\text{\&} \quad & \text{\$ (ORBITER) -v 2 } \\
\text{\&} \quad & \text{\&} \quad \text{-define } F \text{ -finite\_field -q 2 -end } \\
\text{\&} \quad & \text{\&} \quad \text{-define } A \text{ -vector -field } F \text{ -sparse 11 "1,0,1,10" -end } \\
\text{\&} \quad & \text{\&} \quad \text{-define } B \text{ -vector -field } F \text{ -dense "1,0,1,1" -end } \\
\text{\&} \quad & \text{\&} \quad \text{-with } F \text{ -do } \\
\text{\&} \quad & \text{\&} \quad \text{-finite\_field\_activity } \\
\text{\&} \quad & \text{\&} \quad \text{-polynomial\_division } A \text{ } B \text{ -end}
\end{align*}
\]

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,

\[
\text{poly\_gcd:}
\]
\[
\begin{align*}
\text{\&} \quad & \text{\$ (ORBITER) -v 2 } \\
\text{\&} \quad & \text{\&} \quad \text{-define } F \text{ -finite\_field -q 2 -end } \\
\text{\&} \quad & \text{\&} \quad \text{-with } F \text{ -do } \\
\text{\&} \quad & \text{\&} \quad \text{-finite\_field\_activity } \\
\text{\&} \quad & \text{\&} \quad \text{-extended\_gcd\_for\_polynomials } "1,0,0,0,0,0,0,0,0,0,1" \text{ } "1,0,1,1" \text{ -end}
\end{align*}
\]

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

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

\[
\text{poly\_mult\_mod1:}
\]
\[
\begin{align*}
\text{\&} \quad & \text{\$ (ORBITER) -v 2 } \\
\text{\&} \quad & \text{\&} \quad \text{-define } F \text{ -finite\_field -q 7 -end } \\
\text{\&} \quad & \text{\&} \quad \text{-with } F \text{ -do } \\
\text{\&} \quad & \text{\&} \quad \text{-finite\_field\_activity } \\
\text{\&} \quad & \text{\&} \quad \text{-polynomial\_mult\_mod } "1,2,3" \text{ } "3,4,5" \text{ } "6,0,0,1" \text{ -end}
\end{align*}
\]
which has a result of 
\[ X^2 + 4X + 4. \]

Observe how 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 \
```

258
The next command performs polynomial long division based on numerical polynomials:

\[
\text{polynomial.division.ranked.2.27.13:}
\]
\[
\text{\textbackslash $(ORBITER) -v 2 \textbackslash}
\]
\[
\text{\textbackslash \textbackslash -define F -finite_field -q 2 -end \textbackslash}
\]
\[
\text{\textbackslash \textbackslash -with F -do \textbackslash}
\]
\[
\text{\textbackslash \textbackslash -finite_field_activity \textbackslash}
\]
\[
\text{\textbackslash \textbackslash \textbackslash -polynomial_division.ranked 27 13 \textbackslash}
\]
\[
\text{\textbackslash \textbackslash -end}
\]
\[
\text{pdflatex polynomial_division.27.13.tex}
\]
\[
\text{open polynomial_division.27.13.pdf}
\]

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

\[
\text{mult.polynomials.1024.999.997:}
\]
\[
\text{\textbackslash $(ORBITER) -v 2 \textbackslash}
\]
\[
\text{\textbackslash \textbackslash -define F -finite_field -q 2 -end \textbackslash}
\]
\[
\text{\textbackslash \textbackslash -with F -do \textbackslash}
\]
\[
\text{\textbackslash \textbackslash -finite_field_activity \textbackslash}
\]
\[
\text{\textbackslash \textbackslash \textbackslash -mult.polynomials 999 997 \textbackslash}
\]
\[
\text{\textbackslash \textbackslash -end}
\]
\[
\text{pdflatex polynomial_mult.999.997.tex}
\]
\[
\text{open polynomial_mult.999.997.pdf}
\]

The next command performs division with remainder:

\[
\text{polynomial.division.ranked.2.349147.1033:}
\]
\[
\text{\textbackslash $(ORBITER) -v 2 \textbackslash}
\]
\[
\text{\textbackslash \textbackslash -define F -finite_field -q 2 -end \textbackslash}
\]
\[
\text{\textbackslash \textbackslash -with F -do \textbackslash}
\]
\[
\text{\textbackslash \textbackslash -finite_field_activity \textbackslash}
\]
\[
\text{\textbackslash \textbackslash \textbackslash -polynomial_division.ranked 349147 1033 \textbackslash}
\]
\[
\text{\textbackslash \textbackslash -end}
\]
\[
\text{pdflatex polynomial_division.349147.1033.tex}
\]
\[
\text{open polynomial_division.349147.1033.pdf}
\]
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:

```
mult_polynomials_1024_999_997_check:
  ▶ $(ORBITER) -v 3 \n  ▶ ▶ -define F -finite_field -q 1024 -end \n  ▶ ▶ -with F -do \n  ▶ ▶ -finite_field_activity -parse_and_evaluate \n  ▶ ▶ "test" "" "a*b" "a=999,b=997" -end
```

In this last command, the formula $a*b$ 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 using the `vector` object type. 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,\n1,10,1,8,1,7,1,5,1,4,1,2,1,1,1,0"
TWO_TO_THE_32_MINUS_2=4294967294
```

```
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
```

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:
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  ▶ ▶ -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

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

Berlekamp_matrix_2_3:
  ▶ $(ORBITER) -v 2 \n  ▶ ▶ -define F -finite_field -q 2 -end \n  ▶ ▶ -define v -vector -field F -dense "1,1,0,1" -end \n  ▶ ▶ -with F -do \n  ▶ ▶ -finite_field_activity \n  ▶ ▶ -Berlekamp_matrix v -end

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
search_primitive_poly_2:
▷ $(ORBITER) -v 3 \$
▷ ▷ -search_for_primitive_polynomial_in_range 2 2 2 10 #| grep //

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

```
"7", // X^2 + X + 1
"13", // X^3 + X^2 + 1
"25", // X^4 + X^3 + 1
"37", // X^5 + X^2 + 1
"97", // X^6 + X^5 + 1
"193", // X^7 + X^6 + 1
"285", // X^8 + X^7 + X^3 + X^2 + 1
"529", // X^9 + X^4 + 1
"1033", // X^10 + X^3 + 1
```

Primitive polynomials over the base field $\mathbb{F}_s$ are converted into integers, using the base-$s$ representation of integers. For instance, the polynomial $X^2 + X + 1$ is read as binary string 111, which in turn translates to the integer 7 (we use $s = 2$).

Regarding the problem of creating all irreducible polynomials, we can use the following command:

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

It produces a table of all irreducible polynomials of degree 3 over $\mathbb{F}_4$. The output is:

```
There are 20 irreducible polynomials of degree 3 over the field F4:
0 : 1123 : 91
1 : 1031 : 77
2 : 1213 : 103
3 : 1323 : 123
4 : 1322 : 122
5 : 1222 : 106
```
6 : 1021 : 73
7 : 1101 : 81
8 : 1333 : 127
9 : 1232 : 110
10 : 1113 : 87
11 : 1233 : 111
12 : 1301 : 113
13 : 1003 : 67
14 : 1112 : 86
15 : 1002 : 66
16 : 1312 : 118
17 : 1011 : 69
18 : 1132 : 94
19 : 1201 : 97
8.2 Multivariate Polynomial Rings

Orbiter can work with multivariate and graded polynomial rings. 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.

\[
\begin{align*}
\text{CREMONA}\_\text{MAP}\_\text{Y0} &= "3*y0*y0*y0*y0*y0*y2+4*y0*y0*y0*y1*y1*y1*y2+6*y0*y1*y1*y2*y2+9*y0*y2*y2*y2*y2*y2" \\
\text{CREMONA}\_\text{MAP}\_\text{Y1} &= "y0*y0*y0*y0*y0*y0*y1+5*y0*y0*y0*y1*y1*y1+3*y0*y0*y1*y1*y1*y1+5*y0*y0*y1*y2*y2*y2+y0*y0*y1*y2*y2*y2" \\
\text{CREMONA}\_\text{MAP}\_\text{Y2} &= "10*y0*y0*y0*y0*y0*y0+11*y0*y0*y0*y0*y0*y0*y0*y1*y1+11*y0*y0*y0*y0*y1*y2+4*y0*y0*y0*y0*y0*y2*y2+y0*y0*y0*y1*y2*y2+y0*y0*y1*y2*y2*y2" \\
\text{CREMONA}\_\text{MAP}\_\text{Y3} &= "0"
\end{align*}
\]

Cremona map:
\[
\text{\$\{\text{ORBITER}\} -v 3 \backslash}
\text{\text{-define F -finite_field -q 13 -end \backslash}}
\text{\text{-define P -projective_space -n 2 -field F -v 0 -end \backslash}}
\text{\text{-define R -polynomial_ring \backslash}}
\text{\text{-field F \backslash}}
\text{\text{-number_of_variables 3 \backslash}}
\text{\text{-homogeneous_of_degree 6 \backslash}}
\text{\text{-monomial_ordering_lex \backslash}}
\text{\text{-variables "y0,y1,y2" "y_0,y_1,y_2" \backslash}}
\text{\text{-end \backslash}}
\text{\text{-define Y0 -formula \backslash}}
\text{\text{-define Y1 -formula \backslash}}
\text{\text{-define Y2 -formula \backslash}}
\text{\text{-define Cremona -collection "Y0,Y1,Y2" \backslash}}
\text{\text{-with P -do \backslash}}
\]

264
Next, we will consider ideals. As an application, we classify arcs in a projective plane and see which conics we get. The next command classifies the $(5, 2)$-arcs in $\text{PG}(2, 11)$:

```
$\text{arcs}_5_2_\text{q11}$:
$\text{pdflatex arcs}_5_2_\text{q11}\_\text{poset}\_\text{poset}\_\text{tex}$
$\text{open arcs}_5_2_\text{q11}\_\text{poset}\_\text{pdf}$
```

It finds exactly two isomorphism types of arcs. The representative sets are

$$
\{0, 1, 2, 3, 37\}, \quad \{0, 1, 2, 3, 49\}.
$$

They are stored in the file $\text{arcs}_5_2_\text{q11}_1\_\text{lv1}_5$. Let us now create the ideal in the quadratic component of the polynomial ring in three variables over $\mathbb{F}_{11}$:

```
arcs_5_2_\text{q11}\_\text{ideal}$:
```

265
The ideals are generated by

\[ 7x_0x_1 + 5x_0x_2 + 10x_1x_2 \]
and

\[ 4x_0x_1 + 8x_0x_2 + 10x_1x_2, \]
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.

\texttt{PTS\_OF\_SURFACE\_ORBIT211\_Q3\_L9\_E4=\
0,1,2,5,7,8,10,14,9,12,\
15,3,16,37,31,34,20,19,17,32,36,33}\

Then, we create a ring and compute the ideal:

\texttt{surface\_9lines\_4E\_ideal:}\
\texttt{\$\{ORBITER\} -v 2 \}\
\texttt{\$\{PTS\_OF\_SURFACE\_ORBIT211\_Q3\_L9\_E4\} \}\
\texttt{-define F -finite_field -q 3 -end \}\
\texttt{-define R -polynomial_ring \}\
\texttt{-field F \}\
\texttt{-number_of_variables 4 \}\
\texttt{-homogeneous_of_degree 3 \}\
\texttt{-monomial_ordering.lex \}\
\texttt{-variables \"x0,x1,x2,x3\" \"x_0,x_1,x_2,x_3\" \}\
\texttt{-end \}\
\texttt{-with R -do \}\
\texttt{-ring_theoretic_activity \}\
\texttt{-ideal "surf_eqn" "surf\_eqn" Pts \}\
\texttt{-end}
We find a two-dimensional ideal. Generators are:

\[ x_0^2 x_1 + 2x_0 x_1^2 x_3 + 2x_0 x_1 x_3^2 \quad \text{and} \quad 2x_2^2 x_3^2 + 2x_2 x_3^3. \]

Let us take the sum of the two polynomials and create the cubic surface:

\[ \text{SURFACE}_F = "x_0 x_0 x_1 - x_0 x_1 x_1 - x_0 x_1 x_3 - x_2^2 x_2 x_3 - x_2 x_3 x_3" \]

In the next example, we wish to explore the relationship between conics and \((5, 2)\)-arcs. We consider the plane \(\text{PG}(2, 11)\). 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:

\[ \text{random}_k\text{-subsets}_\text{PG}_2\text{.11}: \]

\[ \$(\text{ORBITER}) -v 4 \]

\[ \$\text{create}_\text{random}_k\text{-subsets} 133 5 20 \]

The sets are stored in the file \texttt{random_k_subsets_n133_k5_nb20.csv}. Now, let’s compute the line type of these subsets, to see which ones are arcs:

\[ \text{line}_\text{type}_\text{in}_\text{PG}_2\text{.11}: \]

\[ \$(\text{ORBITER}) -v 3 \]
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 \n  -define F -finite_field -q 11 -end \n  -define R -polynomial_ring \n  -field F \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
```

The ideal is generated by

\[
10*x0*x0 + 3*x0*x1 + 8*x0*x2 + 2*x1*x1 + 10*x2*x2.
\]

The conic contains the following 12 points:

\( \{3, 15, 19, 33, 40, 42, 46, 50, 83, 88, 102, 108\} \).
Chapter 9

Applications

9.1 Number Theory

In Table 9.1, some number theoretic commands are shown. For instance,

\texttt{inverse}\_mod\_a:
\[\texttt{\$(ORBITER) \ -v 2 \ -inverse\_mod 18059241 \ 58014043}\]

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

\texttt{jacobi}\_a:
\[\texttt{\$(ORBITER) \ -v 5 \ -jacobi 2221 \ 7817}\]

269
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Description</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:

\[
\left( \frac{2221}{7817} \right) = \left( \frac{7817}{2221} \right) \cdot (-1)^{2221-1} \cdot \left( \frac{7817}{2} \right) \\
= \left( \frac{7817}{2221} \right) = \left( \frac{1154}{2221} \right) \\
= \left( \frac{2}{2221} \right) \cdot \left( \frac{577}{2221} \right) \\
= (-1)^{2221^2 - 1} \cdot \left( \frac{577}{2221} \right) \\
= (-1) \cdot \left( \frac{577}{2221} \right) \\
= (-1) \cdot \left( \frac{2221}{577} \right) \cdot (-1)^{577 - 1} \cdot \left( \frac{2221}{2} \right)^{-1} \\
= (-1) \cdot \left( \frac{2221}{577} \right) \\
= (-1) \cdot \left( \frac{490}{577} \right)
\]
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

```
sqrt_mod_7817:
```

yields that 7634 is a square root. Indeed,

\[ 7634^2 \equiv 2221 \mod 7817. \]
<table>
<thead>
<tr>
<th>Command</th>
<th>Args</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 \
  -define G -linear_group -PGL 4 3 -end \
  -with G -do \
  -group_theoretic_activity \
  -representation_on_polynomials 3 \
  -end
  $(ORBITER) -v 2 \
  -loop L 0 9 1 -draw_matrix \
  -input_csv_file PGL_4_3_rep_3_%L.csv \
  -box_width 40 -bit_depth 24 -partition 3 20 20 -end \
  -end_loop
```

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>Description</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>$d \ n \text{ list-of-integers}$</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,

**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

**EC_cyclic_subgroup:**

```
▷ $(ORBITER) -v 2 \\
▷ ▷ -define F -finite_field -q 11 -end \\
▷ ▷ -with F -do \\
▷ ▷ -finite_field_activity \\
▷ ▷ -EC_cyclic_subgroup 1 3 "1,4" -end
```
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-EC_add</td>
<td>$a \ b \ i_1 \ i_2$</td>
<td>On the elliptic curve $y^2 \equiv x^3 + ax + b$ in $\mathbb{F}_q$, add the points with indices $i_1$ and $i_2$, each given as a pair $x, y$.</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>$a \ b \ pt \ n \ cipher \ round-keys$</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
Figure 9.2: The elliptic curve $y^2 = x^3 + 5x + 7 \mod 199$

computes the cyclic subgroup generated by the point $(1, 4)$ on the curve $y^2 = x^3 + x + 3 \mod 11$. The command

```
EC_points_199:
  $(ORBITER) -v 2 \$
  $-define F -finite_field -q 199 -end \$
  $-with F -do \$
  $-finite_field_activity \$
  $-EC_points "EC_5_7_q199" 5 7 -end$
  $(ORBITER) -v 2 \$
  $-draw_matrix -input_csv_file EC_5_7_q199_points_xy.csv \$
  $-box_width 10 -bit_depth 24 \$
  $-partition 2 199 199 -end$
```

computes all points on the curve $y^2 = x^3 + 5x + 7 \mod 199$ and produces a bitmap drawing of the points in the affine plane 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

276
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:

\[\begin{array}{l}
\text{EC.bsgs:} \\
\text{EC.bsgs:} \\
\text{EC.bsgs:} \\
\text{EC.bsgs:} \\
\text{EC.bsgs:} \\
\text{EC.bsgs:} \\
\text{EC.bsgs:} \\
\text{EC.bsgs:} \\
\end{array}\]

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:

\[\begin{array}{l}
\text{EC.bsgs_decode:} \\
\text{EC.bsgs_decode:} \\
\text{EC.bsgs_decode:} \\
\text{EC.bsgs_decode:} \\
\text{EC.bsgs_decode:} \\
\text{EC.bsgs_decode:} \\
\text{EC.bsgs_decode:} \\
\text{EC.bsgs_decode:} \\
\end{array}\]
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 \(F_q\): The coefficients of \(A(X) = a_0 + a_1 X + \cdots + a_d X^d\) are listed as a sequence, starting with the constant term and ending with the leading coefficient. The cryptosystem requires coefficients \(a_i\) in the range \(-\frac{p}{2} \leq a_i \leq \frac{p}{2}\). So, in an extension to the conventions for field elements in \(F_q\), 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

\[ \text{NTRU\_N}=7 \]
\[ \text{NTRU\_P}=3 \]
\[ \text{NTRU\_Q}=41 \]
\[ \text{NTRU\_D}=2 \]
\[ \text{NTRU\_XN1}=-1,0,0,0,0,0,1,\]
\[ \text{ALICE\_PRIVATE\_F}=-1,0,1,1,-1,0,1\]
\[ \text{ALICE\_PRIVATE\_G}=0,-1,-1,0,1,0,1\]

are used to set up the appropriate variables according to these choices.

Regarding the NTRU set-up, Alice needs to compute her private keys \(F_p(x)\) and \(F_q(x)\). These two polynomials are defined as follows:
1. $F_p(x)$ is the inverse of $f(x)$ in $\mathbb{F}_p[x]/(x^n - 1)$,

2. $F_q(x)$ the inverse of $f(x)$ in $\mathbb{F}_q[x]/(x^n - 1)$.

To this end, we can use the `extended_gcd_for_polynomials` command from Table 9.1. The following two makefile commands do the job:

```
NTRU_Alice1:
  $(ORBITER) -v 2 \\
  -define F -finite_field -q $(NTRU_Q) -end \\
  -with F -do \\
  -finite_field_activity \\
  -extended_gcd_for_polynomials \\
  $(NTRU_XN1) $(ALICE_PRIVATE_F) \\
  -end

ALICE_PRIVATE_FQ="37,2,40,21,31,26,8"

NTRU_Alice2:
  $(ORBITER) -v 2 \\
  -define F -finite_field -q $(NTRU_P) -end \\
  -with F -do \\
  -finite_field_activity \\
  -extended_gcd_for_polynomials \\
  $(NTRU_XN1) $(ALICE_PRIVATE_F) \\
  -end

ALICE_PRIVATE_FP="1,1,1,1,0,2,1"
```

The resulting polynomials (indicated as comments by means of the # symbol) are again encoded as makefile variables.

```
The resulting polynomials (indicated as comments by means of the # symbol) are again encoded as makefile variables.
```

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:
  $(ORBITER) -v 2 \\
  -define F -finite_field -q $(NTRU_Q) -end \\
  -with F -do \\
  -finite_field_activity \\
  -polynomial_mult_mod $(ALICE_PRIVATE_F) \\
  $(ALICE_PRIVATE_G) $(NTRU_XN1) \\
  -end
```
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 \n  -define F -finite_field -q $(NTRU_Q) -end \n  -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 \n  -define F -finite_field -q $(NTRU_Q) -end \n  -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 \n  -define F -finite_field -q $(NTRU_Q) -end \
```
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

\[
\text{Allen\_Gates\_noise\_1\_percent:} \\
\text{\$(ORBITER) -v 3 \$} \\
\text{\$ random\_noise\_in\_bitmap\_file \$} \\
\text{\$ allen\_Gates\_bmp \$} \\
\text{\$ allen\_Gates\_1\_bmp \$} \\
\text{\$ 1 100 \$} \\
\text{\$ open allen\_Gates\_1\_bmp \$}
\]

simulates random noise at the 1 percent level applied to the file \text{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

\[
\text{Hamming\_space\_4\_2\_distance\_matrix:} \\
\text{\$(ORBITER) -Hamming\_space\_distance\_matrix 4 2 \$}
\]

creates the distance matrix of the Hamming graph $H(n,q)$. The data is written to the file \text{Hamming\_n4\_q2.csv}. The command

\[
\text{Hamming\_space\_4\_2\_distance\_matrix\_draw:} \\
\text{\$(ORBITER) -v 2 -draw\_matrix \$} \\
\text{\$ -input\_csv\_file Hamming\_n4\_q2.csv \$} \\
\text{\$ -box\_width 20 -bit\_depth 24 \$}
\]
### Table 10.1: Global Coding Theoretic Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Description</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 ≤ n ≤ n_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>

Figure 10.1: Random noise at the 1% level
Figure 10.2: The color-coded distance matrix of the Hamming graph $H(4, 2)$

\[
\begin{array}{ccc}
\text{⊿} & \text{⊿} & \text{⊿} \\
\text{⊿} & \text{⊿} & \text{open Hamming\_n4\_q2\_draw.bmp} \\
\end{array}
\]

produces the bitmap graphic `Hamming_n4_q2_draw.bmp` shown in Figure 10.2.

The command

```
$\text{Hamming\_code\_macwilliams:}$
```

```
\text{▷} \text{▷} \text{▷} \text{-partition} \text{ 4 16 16} \ \text{▷} \text{▷} \text{-end} \\
\text{▷} \text{open Hamming\_n4\_q2\_draw.bmp}
```

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 $\{0,7,25,30\}$:

\[\text{CODE\_5\_2\_3\_CODEWORDS} = "0,7,25,30"\]

code\_5\_2\_3\_diagram:
\[
\text{\$\(\text{ORBITER}\) -v 2 \ }
\text{\$\(\text{-define F -finite_field -q 2 -end}\) \ }
\text{\$\(\text{-with F -do -coding_theoretic_activity}\) \ }
\text{\$\(\text{-code_diagram } "\text{code\_5\_2\_3"}\) \ }
\text{\$\(\text{-metric_balls 1}\) \ }
\text{\$\(\text{-end}\) \ }
\text{\$\(\text{ORBITER}\) -v 2 \ }
\text{\$\(\text{-draw_matrix}\) \ }
\text{\$\(\text{-input_csv_file code\_5\_2\_3\_diagram.01\_5\_4.csv}\) \ }
\text{\$\(\text{-box_width 25 -bit_depth 24}\) \ }
\text{\$\(\text{-partition 4 8 4}\) \ }
\text{\$\(\text{-end}\) \ }
\]

The Hamming graph $H(5,2)$ can be created with the following command:

\[\text{Hamming\_5\_2\_graph:}\]
\[
\text{\$\(\text{ORBITER}\) -v 2 \ }
\text{\$\(\text{-define G -graph -Hamming 5 2 -end}\) \ }
\text{\$\(\text{-with G -do}\) \ }
\text{\$\(\text{-graph_theoretic_activity -export_csv -end}\) \ }
\]
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-BCH</td>
<td>$n$ $q$ $t$</td>
<td>Compute a table of BCH codes of length $n$ over $\mathbb{F}_q$.</td>
</tr>
<tr>
<td>-BCH_dual</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-general_code_binary</td>
<td>$n$ text</td>
<td></td>
</tr>
<tr>
<td>-code_diagram</td>
<td>label codewords</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-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>out nb-bits</td>
<td></td>
</tr>
<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>-crc32_file_based</td>
<td>fname-in fname-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>out block-length</td>
<td></td>
</tr>
<tr>
<td>-crc_new_file_based</td>
<td>fname</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>
</tbody>
</table>

Table 10.2: Coding Theoretic Activities (Part I)
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-minimum_distance</code></td>
<td>code-object-label</td>
<td>Compute the minimum distance of the linear code object.</td>
</tr>
<tr>
<td><code>-generator_matrix_cyclic_code</code></td>
<td>n poly</td>
<td></td>
</tr>
<tr>
<td><code>-nth_roots</code></td>
<td>n</td>
<td></td>
</tr>
<tr>
<td><code>-make_BCH_code_and_encode</code></td>
<td>n d text fname</td>
<td></td>
</tr>
<tr>
<td><code>-NTT</code></td>
<td>n q</td>
<td></td>
</tr>
<tr>
<td><code>-find_CRC_polynomials</code></td>
<td>nb-errors info-bits check-bits</td>
<td></td>
</tr>
<tr>
<td><code>-write_code_for_division</code></td>
<td>fname A B</td>
<td></td>
</tr>
<tr>
<td><code>-polynomial_division_from_file</code></td>
<td>fname r1</td>
<td></td>
</tr>
<tr>
<td><code>-polynomial_division_from_file_all_k_bit_error_patterns</code></td>
<td>fname r1 k</td>
<td></td>
</tr>
<tr>
<td><code>-export_magma</code></td>
<td>fname</td>
<td></td>
</tr>
<tr>
<td><code>-export_codewords</code></td>
<td>fname</td>
<td></td>
</tr>
<tr>
<td><code>-export_genma</code></td>
<td>fname</td>
<td></td>
</tr>
<tr>
<td><code>-export_checkma</code></td>
<td>fname</td>
<td></td>
</tr>
</tbody>
</table>

Table 10.3: Coding Theoretic Activities (Part II)
Figure 10.3: Drawing of the Hamming graph $H(5, 2)$

```
$ ORBITER -v 2 -draw_matrix \ 
$ -input_csv_file Hamming_5_2.csv \ 
$ -box_width 8 -bit_depth 24 -partition 4 32 32 -end \ 
$ dot -Tpng Hamming_5_2.gv >Hamming_5_2.png
```

Using the unix dot program, this command sequence creates the drawing of $H(5, 2)$ shown in Figure 10.3.
Table 10.4: Commands to Create Codes

<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Description</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>Code defined by a generator matrix.</td>
</tr>
<tr>
<td>-linear_code_from_projective_set</td>
<td>$nmg S$</td>
<td>Code defined by a projective set in the dual.</td>
</tr>
<tr>
<td>-linear_code_by_columns_of_parity_check</td>
<td>$nmg M$</td>
<td>Code defined by an affine set in the dual.</td>
</tr>
<tr>
<td>-first_order_Reed_Muller</td>
<td>$m$</td>
<td>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>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>Description</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 list 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 \
  ▶ ▶ -end \
  ▶ ▶ -with C -and F -do -coding_theoretic_activity \
```
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:

```plaintext
SIMPLEX_CODE_GENERATOR="\n1,0,1,0,1,0,1, \n0,1,0,0,1,1, \n0,0,0,1,1,1"
```

The following command computes the nullspace of this matrix, which is the Hamming code:

```plaintext
simplex_code:
  $(ORBITER) -v 2 \n  -define F -finite_field -q 2 -end \n  -define v -vector -field F -format 3 \n  -dense $(SIMPLEX_CODE_GENERATOR) \n  -end \n  -define C -code -field F \n  -linear_code_through_generator_matrix v \n  -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
Hamming_code:
  $(ORBITER) -v 2
  -define F -finite_field -q 2 -end
  -define v -vector -field F -format 3
  -dense $(SIMPLEX_CODE_GENERATOR)
  -define C -code -field F -linear_code
  -dual
  -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:

```plaintext
K<w> := GF(2);
V := VectorSpace(K, 7);
C := LinearCode(sub<V | [1,1,1,0,0,0,0], [1,0,0,1,1,0,0], [0,1,0,1,0,1,0], [1,1,0,1,0,0,1])
```

The next command creates the first order Reed-Muller code in 3 variables. All codewords are created. The codewords and the generator matrix are exported to files.

```bash
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
```

292
Alternatively, we can store the generator matrix in 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 \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
```

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 \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
```

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:
Hamming_singer:

\$\text{ORBITER} -v 3 \$
\$\text{define G } \text{-linear_group } \text{-PGL 3 2 } \text{-singer 1 } \text{-end } \$
\$\text{-with G } \text{-do } \$
\$\text{-group_theoretic_activity } \$
\$\text{-report } \$
\$\text{-orbits_on_points } \$
\$\text{-end } \$

\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

0

1

2

5

3

4

6

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="\
1,0,0,1,1,1,0, \
0,1,0,0,1,1,1, \
0,0,1,1,1,0,1"$
```

Hamming cyclic generator:
```
$($(ORBITER) -v 2 \n  -define F -finite_field -q 2 -end \n  -define v -vector -format 3 -field F \n  -dense $(SIMPLEX_CODE_GENMA_CYCLIC) \n  -end \n  -with F -do -finite_field_activity \n  -nullspace v \n  -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, \n0,1,0,1,0,1,1, \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  -with F -do \n  -coding_theoretic_activity \n  -weight Enumerator v \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 \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

The following command computes the minimum distance of the Golay code of length 23:

Golay23 minimum distance:
▷ $(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
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_23_COLUMN_RANKS_PROJECTIVELY="0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 132, 913, 1460, 1750, 1898, 2518, 2787, 2874, 3320, 3357, 3662"
```

Suppose we want to list the code words. The following command can be used:

```
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 \
  -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>Description</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 \n  "Hithere" "text.csv" 
  $(ORBITER) -v 2 \n  -define F -finite_field -q 2 -end \n  -with F -do \n  -coding_theoretic_activity \n  -polynomial_division_from_file \n  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 \ 
   \> -with F -do \ 
   \> -coding_theoretic_activity \ 
   \> -polynomial_division_from_file \ 
   \> text_with_1error.csv 13 \ 
   \> -end
   pdflatex polynomial_division_file_13.tex
   open polynomial_division_file_13.pdf
```

This creates the following output:

```
text.csv / 13 =
1010110100110101010111000010111100 / 1101 =

1101101110000101111101011110010101
=================================================================
1101 | 1010110100110101010111000010111100
   1101
====
111110100110101010111000010111100
   1101
====
10101001101010111000010111100
   1101
====
11110011010101011000010111100
   1101
====
1000110101010111000010111100
   1101
====
101110101010111000010111100
   1101
====
11010101011100010111100
   1101
====
11110111000010111100

300
```
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```
The following output is created:

```
01010110100110101010111000010111100
0 : 01010110100110101010111000010111101 : 100 : 4 : X^{-2}
1 : 01010110100110101010111000010111110 : 111 : 7 : X^{-2} + X + 1
2 : 01010110100110101010111000010111000 : 001 : 1 : 1
3 : 01010110100110101010111000010110100 : 000 : 0 : 0
4 : 01010110100110101010111000010110100 : 010 : 2 : X
5 : 01010110100110101010111000010110100 : 011 : 3 : X + 1
6 : 010101101001101010101110000101101100 : 100 : 4 : X^{-2}
7 : 010101101001101010101110000101101100 : 110 : 6 : X^{-2} + X
8 : 01010110100110101010111000010111000 : 111 : 7 : X^{-2} + X + 1
9 : 01010110100110101010111000010111100 : 001 : 1 : 1
10 : 01010110100110101010111000010111100 : 000 : 0 : 0
11 : 01010110100110101010111000010111100 : 010 : 2 : X
12 : 01010110100110101010111000010111100 : 011 : 3 : X + 1
13 : 010101101001101010101110000101111100 : 100 : 4 : X^{-2}
14 : 010101101001101010101110000101111100 : 110 : 6 : X^{-2} + X
15 : 010101101001101010101110000101111100 : 111 : 7 : X^{-2} + X + 1
16 : 010101101001101010101110000101111100 : 001 : 1 : 1
17 : 010101101001101010101110000101111100 : 000 : 0 : 0
18 : 010101101001101010101110000101111100 : 010 : 2 : X
19 : 010101101001101010101110000101111100 : 011 : 3 : X + 1
20 : 010101101001101010101110000101111100 : 100 : 4 : X^{-2}
22 : 010101101001101010101110000101111100 : 111 : 7 : X^{-2} + X + 1
23 : 010101101001101010101110000101111100 : 001 : 1 : 1
24 : 010101101001101010101110000101111100 : 000 : 0 : 0
25 : 010101101001101010101110000101111100 : 010 : 2 : X
26 : 010101101001101010101110000101111100 : 011 : 3 : X + 1
27 : 010101101001101010101110000101111100 : 100 : 4 : X^{-2}
29 : 010101101001101010101110000101111100 : 111 : 7 : X^{-2} + X + 1
30 : 010111101001101010101110000101111100 : 001 : 1 : 1
31 : 010111101001101010101110000101111100 : 000 : 0 : 0
32 : 011110100110101010111000101111100 : 010 : 2 : X
33 : 000101101001101010101110000101111100 : 110 : 6 : X^{-2} + X
```

302
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 $n = 128$.

```
CRC_3_128_10:
  $(ORBITER) -v 1 \n  -define F -finite_field -q 2 -end \n  -with F -do -coding_theoretic_activity \n  -find_CRC_polynomials 3 128 10 \n  -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"

CRC32ETHERNET="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:

```bash
$ (ORBITER) -v 2 \n  -define F -finite_field -q 2 -end \n  -define v -vector -field F -sparse 33 $(CRC32_ETHHERNET) -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 3 \n  -define F -finite_field -q 256 -end \n  -with F -do -coding_theoretic_activity \n  -make_BCH_code 771 2 \n  -end
```

The polynomial in dense coding

```plaintext
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  -define F -finite_field -q 256 -end \n  -define A -vector -field F -sparse 772 "1,771,1,0" -end \n  -define B -vector -field F -dense $(CRC_POLY_Q256_DEG2_DENSE) -end 
```
We create a BCH code of length 771 over $\mathbb{F}_{256}$ with designed distance 16:

```bash
$ (ORBITER) -v 3 \
  > -define F -finite_field -q 256 -end \
  > -with F -do -coding_theoretic_activity \
  > -make BCH_code 771 16 \
  > -end \
  > pdflatex BCH_codes_q256_n771_d16.tex \
  > open BCH_codes_q256_n771_d16.pdf
```

The polynomial in sparse coding is:

```
POLY_Q256_DEG30_SPARSE="1,0,26,1,210,2,24,3,\
138,4,148,5,160,6,58,7,108,8,199,9,95,10,56,\
11,9,12,205,13,194,14,193,15,3,16,248,17,110,\
18,150,19,24,20,169,21,192,22,212,23,112,24,\
144,25,97,26,109,27,174,28,253,29,1,30"
```

The polynomial in dense coding is:

```
POLY_Q256_DEG30_DENSE="1,26,210,24,138,148,\
160,58,108,199,95,56,9,205,194,193,3,248,110,\
150,24,169,192,212,112,144,97,109,174,253,1"
```

We generate C++ source code for the use of this polynomial:

```bash
$ (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 \n  > -coding_theoretic_activity \n  > -write_code_for_division \n  > check_q256_n771_r30.cpp A B \
```

305
We confirm that the polynomial divides $X^{771} - 1$ as it should:

\begin{verbatim}
F256_BCH_code_d16_division:
> $(ORBITER) -v 2 \n> -define F -finite_field -q 256 -end \n> -define A -vector -field F -sparse 772 "1,771,1,0" -end \n> -define B -vector -field F -dense $(POLY_Q256_DEG30_DENSE) -end \n> -with F -do \n> -finite_field_activity \n> -polynomial_division A B -end
\end{verbatim}

The next example introduces three errors. The remainder is not zero, so the errors are detected:

\begin{verbatim}
F256_BCH_code_d16_error:
> $(ORBITER) -v 2 \n> -define F -finite_field -q 256 -end \n> -define A -vector -field F -sparse 771 "2,30,3,31,55,770" -end \n> -define B -vector -field F -dense $(POLY_Q256_DEG30_DENSE) -end \n> -with F -do \n> -finite_field_activity \n> -polynomial_division A B -end
\end{verbatim}
10.5 Reed-Muller Codes

The following command creates the generator matrix of the first order Reed-Muller code in 3 dimensions, RM$_{3,1}$. The codewords are listed as well.

```
REED_MULLER_3_1_BASIS_IN_BINARY="255,170,204,240"
```

```
RM_3_1_code_words:
  $(ORBITER) -v 2 \\
  -define v -vector -dense $(REED_MULLER_3_1_BASIS_IN_BINARY) -end \\
  -linear_code_through_basis 8 v \\
  pdflatex code_n8_k4_q2.tex \\
  open code_n8_k4_q2.pdf
```

The output is shown in Figure 10.5.

The following command produces a diagram of the characteristic function of the Reed Muller code in the Hamming space.

```
RM_3_1_Hamming_space_diagram:
  $(ORBITER) -v 2 \\
  -define F -finite_field -q 2 -end \\
  -with F -do \\
  -coding_theoretic_activity \\
  -code_diagram "RM_3_1"
```
Figure 10.6: Boolean function representation of RM$_{3,1}$ in $H(8,2)$

produces a representation of the code as boolean function in the Hamming space $H(8,2)$, shown in Figure 10.6. The different codewords are given different colors.
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

$$\text{lcm}\left(m_{\beta^1, \mathbb{F}_q}, m_{\beta^2, \mathbb{F}_q}, \ldots, m_{\beta^{d-1}, \mathbb{F}_q}\right).$$

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

draw_cyclotomic_mod_21_q8:

```bash
$ (ORBITER) -v 2 \
  -draw_options \
  -radius 100 \
  -line_width 1.0 -embedded \
  -end \
  -draw_mod_n -n 21 -file mod_21_cyclotomic \
  -cyclotomic_sets 8 "1,2,4,5,7,10,13" -end \
  pdflatex mod_21_cyclotomic_draw.tex \
  open mod_21_cyclotomic_draw.pdf
```

The output is shown in Figure 10.7. We will try BCH-codes with minimum distances 3, 5 and 7. Here is distance 3:

```bash
F_8_BCH_code_d3:

```

The code is described in a latex output file:

BCH-code:

$n = 21$, $k = 17$, $d_0 = 3$, $q = 8$, $g(x) = m_1 m_2 = X^4 + 4X^3 + 4X^2 + 3X + 4$

Chosen cyclotomic sets:
Figure 10.7: 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:

\[
\begin{bmatrix}
4 & 3 & 4 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 4 & 3 & 4 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 4 & 3 & 4 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 4 & 3 & 4 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 4 & 3 & 4 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 4 & 3 & 4 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 4 & 3 & 4 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 3 & 4 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 3 & 4 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 3 & 4 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 3 & 4 & 4 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 3 & 4 & 4 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 3 & 4 & 4 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 3 & 4 & 4 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 3 & 4 & 4 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 3 & 4 & 4 & 1
\end{bmatrix}
\]

And now for \(d = 5\):

\texttt{F}_8\texttt{BCH\_code\_d5}: 
\begin{verbatim}
▷ $(\text{ORBITER})$ -v 3 \ 
▷ ▷ -define F -finite_field -q 8 -override_polynomial 11 -end \ 
▷ ▷ -with F -do -coding_theoretic_activity \ 
▷ ▷ ▷ -make_BCH_code 21 5 \ 
▷ ▷ -end 
▷ pdflatex BCH_codes_q8_n21_d5.tex 
▷ open BCH_codes_q8_n21_d5.pdf
\end{verbatim}

The output file is:

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 \}
The generator polynomial has degree 7

-dense "2,1,2,1,2,3,3,1"

-sparse "2,0,1,1,2,2,1,3,2,4,3,5,3,6,1,7"

The generator matrix is:

\[
\begin{bmatrix}
2 & 1 & 2 & 1 & 2 & 3 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 2 & 1 & 2 & 1 & 2 & 3 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 2 & 1 & 2 & 1 & 2 & 3 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 2 & 1 & 2 & 1 & 2 & 3 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 2 & 1 & 2 & 1 & 2 & 3 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 2 & 1 & 2 & 1 & 2 & 3 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 & 2 & 1 & 2 & 3 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 & 2 & 1 & 2 & 3 & 3 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 & 2 & 1 & 2 & 3 & 3 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 & 2 & 1 & 2 & 3 & 3 \\
\end{bmatrix}
\]

Finally, \(d = 7\):

F_8_BCH_code_d7:

```
$ (ORBITER) -v 3 \n$ -define F -finite_field -q 8 -override_polynomial 11 -end \n$ -with F -do -finite_field_activity -make_BCH_code 21 7 -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:
```

312
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:

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

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.

draw_mod_255_cyclotomic_1_and_3:
\[
\begin{bmatrix}
6 & 6 & 5 & 6 & 4 & 0 & 2 & 5 & 2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 6 & 6 & 5 & 6 & 4 & 0 & 2 & 5 & 2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 6 & 6 & 5 & 6 & 4 & 0 & 2 & 5 & 2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 6 & 6 & 5 & 6 & 4 & 0 & 2 & 5 & 2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 6 & 6 & 5 & 6 & 4 & 0 & 2 & 5 & 2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 6 & 6 & 5 & 6 & 4 & 0 & 2 & 5 & 2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 6 & 6 & 5 & 6 & 4 & 0 & 2 & 5 & 2 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & 6 & 5 & 6 & 4 & 0 & 2 & 5 & 2 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & 6 & 5 & 6 & 4 & 0 & 2 & 5 & 2 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & 6 & 5 & 6 & 4 & 0 & 2 & 5 & 2 & 1 & 1 & 0 \\
\end{bmatrix}
\]

The drawing is shown in Figure 10.8.
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:

BCH_F256_roots_771:

```
$\text{ORBITER} -v 3 \\
\text{-define F -finite_field -q 256 -end} \\
\text{-with F -do -coding_theoretic_activity} \\
\text{-nth_roots 771} \\
\text{-end}
```

The next command creates a BCH-code of length 771 over $\mathbb{F}_{256}$ with minimum distance at least 16:

BCH_F256_BCH_code_d16:

```
$\text{ORBITER} -v 3 \\
\text{-define F -finite_field -q 256 -end} \\
\text{-with F -do -coding_theoretic_activity} \\
\text{-make_BCH_code 771 16} \\
\text{-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

```plaintext
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

```
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
  -with F -do
  -coding_theoretic_activity
  -weight Enumerator v
  -end
```

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.

```
RS_8_field_reduction:
$ (ORBITER) -v 2
  -define F -finite_field -q 8 -end
```
The reduced matrix is shown in Figure 10.9. Let us compute the weight enumerator of the reduced code. The command

```latex
\begin{verbatim}
\$\text{RS}_8\text{reduced}="\n010001100000000000000000\n001110010000000000000000\n110011001000000000000000\n000010001100000000000000\n000001110010000000000000\n000110011001000000000000\n000000010001100000000000\n000000001110010000000000\n000000000111001000000000\n\end{verbatim}
```

317
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 \\
▷ ▷ -with F -do \\
▷ ▷ -coding_theoretic_activity \\
▷ ▷ ▷ -weight_enumerator v \\
▷ ▷ -end

computes the weight enumerator of the binary code. It is

\[
1y^{21} + 28x^3y^{18} + 84x^4y^{17} + 273x^5y^{16} + 924x^6y^{15} + 1956x^7y^{14} + \\
2982x^8y^{13} + 4340x^9y^{12} + 5796x^{10}y^{11} + 5796x^{11}y^{10} + 4340x^{12}y^{9} + \\
2982x^{13}y^{8} + 1956x^{14}y^{7} + 924x^{15}y^{6} + 273x^{16}y^{5} + 84x^{17}y^{4} + 28x^{18}y^{3} + \\
1x^{21}
\]

In particular, the field reduced Reed-Solomon code 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="\n1110001000000000000000 \n1101000100000000000000 \n1011000010000000000000 \n0111000001000000000000 \n1100100000010000000000 \n1010100000010000000000 \n0110100000010000000000 \n1001100000001000000000 \n0101100000000100000000 \n0011100000000010000000 \n1111100000000001000000 \n1100010000000001000000 \n10100100000000000100000 "

318
01100100000000000000010 \ 
1001010000000000000001"

CODE_21_15_4_store:
 DELAY $(ORBITER) -v 2 \ 
 DELAY 15 21 $(CODE_21_15_4) \ 
 DELAY $(ORBITER) -v 2 -draw_matrix \ 
 DELAY -input_csv_file code_21_15_4.csv \ 
 DELAY -box_width 40 -bit_depth 24 \ 
 DELAY -partition 4 "15" "21" \ 
 DELAY -end

We compute the weight enumerator

CODE_21_15_4_weight_enumerator:
 DELAY $(ORBITER) -v 2 \ 
 DELAY -define F -finite_field -q 2 -end \ 
 DELAY -define v -vector -format 15 -field F \ 
 DELAY -compact $(CODE_21_15_4) \ 
 DELAY -end \ 
 DELAY -with F -do \ 
 DELAY -coding_theoretic_activity \ 
 DELAY -weight_enumerator v \ 
 DELAY -end

which turns out to be

\[ 1y^{21} + 221x^4y^{17} + 1600x^6y^{15} + 6498x^8y^{13} + 10912x^{10}y^{11} + 9250x^{12}y^9 + 3584x^{14}y^7 + 669x^{16}y^5 + 32x^{18}y^3 + 1x^{20}y. \]

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

\[
\text{bounds for } d \text{ given } n_{15}, k_6, q_2:
\]

\[
\text{ORBITER} -v 2 -make \text{ bounds for } d \text{ given } n, k, \text{ and } q 15 6 2
\]

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:

\[
\begin{align*}
d_{\text{GV}} &= 5 \\
d_{\text{singleton}} &= 10 \\
d_{\text{hamming}} &= 6 \\
d_{\text{plotkin}} &= 7 \\
d_{\text{griesmer}} &= 6
\end{align*}
\]

This shows that \( 5 \leq d_{\text{max}} \leq 6 \). The command

\[
\text{coding theory bounds q2:}
\]

\[
\text{ORBITER} -v 2 -table of bounds 20 2
\]

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

\[
\text{GV n15 k6 d5:}
\]

\[
\text{ORBITER} -v 2 \text{ -define F -finite_field -q 2 -end \ -with F -do \ -coding_theoretic_activity \ -make_gilbert_varshamov_code 15 6 5 \ -end}
\]

creates a \([15, 6, d]_2\) with minimum distance \( g \geq 5 \) using a greedy algorithm based on the proof of the Gilbert-Varshamov bound. The code that is produced has the following generator matrix:
To compute the minimum distance of the code, we do:

```
CODE GV N15 K6="\n111111111100000\n111110000010000\n111001100001000\n1101010000100\n1010110000010\n1011010010001"
```

The weight enumerator is

```
1 y^{15} + 27 x^6 y^9 + 24 x^8 y^7 + 9 x^{10} y^5 + 3 x^{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

```plaintext
codes_8_4_4:
  $\text{ORBITER} -v 6 \$
  $\text{-orbiter_path ORBITER_PATH} \$
  $\text{-define G} \$
  $\text{-linear_group -PGL 4 2 -end} \$
  $\text{-with G -do} \$
  $\text{-group_theoretic_activity} \$
  $\text{-poset_classification_control} \$
  $\text{-problem_label codes_8_4_4} \$
  $\text{-draw_poset} \$
  $\text{-draw_options -embedded -radius 250} \$
  $\text{-line_width 1.0 -spanning_tree -end} \$
  $\text{-report -end} \$
  $\text{-end} \$
  $\text{-linear_codes 3 8} \$
  $\text{-end} \$
pdflatex codes_8_4_4_poset_lvl_8.tex
open codes_8_4_4_poset_lvl_8.pdf
pdflatex codes_8_4_4_poset.tex
open codes_8_4_4_poset.pdf
```

classifies linear codes with redundancy 4 and minimum distance at least 4. Orbiter confirms that there is exactly one such code, and it computes the code together with the projective stabilizer. Orbiter creates the action of the group $\text{PGL}(4, 2)$ on the poset $\Lambda_{3,3}(2)$. Using poset classification, Orbiter then produces the poset of orbits shown in Figure 10.10. 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
Figure 10.10: Orbits of PGL(4, 2) on the poset $A_{3,3}(2)$
8 in the poset of orbits tells us that the code is unique up to equivalence. Let us look at the code. The elements of the set \{0, 1, 2, 3, 8, 11, 13, 14\} are points in PG(3, 2). We write the coordinate vectors in the columns of a matrix \( H \):

\[
H = \begin{bmatrix}
1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 & 1 & 1
\end{bmatrix}.
\]

This matrix is the parity check matrix \( H \) of the code \( C \). 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 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. Global means that the commands are not associated with an activity related to an object.

The command

\[
\text{Sym}_{10}\text{.conj\_classes:}
\]

\[
\text{\gt $(ORBITER)$ -v 2 -conjugacy\_classes\_Sym\_n 10}
\]

\[
\text{\gt open\_classes\_Sym\_10.csv}
\]

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):

\[
\text{Char\_Sym\_4:}
\]

\[
\text{\gt $(ORBITER)$ -v 2 -character\_table\_symmetric\_group 4}
\]

The command produces the following output:

The character table of Sym(4) is the matrix

\[
\begin{bmatrix}
1 & -1 & 1 & 1 & -1 \\
3 & 1 & 0 & -1 & -1 \\
2 & 0 & -1 & 2 & 0 \\
3 & -1 & 0 & -1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
\end{bmatrix}
\]
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-random_permutation</code></td>
<td><code>n fname</code></td>
<td>Creates a random permutation in $\text{Sym}(n)$ and stores it in the given file.</td>
</tr>
<tr>
<td><code>-create_random_k_subsets</code></td>
<td><code>n k N</code></td>
<td>Creates $N$ random $k$-subsets of an $n$-set.</td>
</tr>
<tr>
<td><code>-read_poset_file</code></td>
<td><code>fname</code></td>
<td>Reads a poset from the given file.</td>
</tr>
<tr>
<td><code>-read_poset_file_with_grouping</code></td>
<td><code>fname x-stretch</code></td>
<td>Reads a poset from the given file and sets stretch factor for orbit grouping.</td>
</tr>
<tr>
<td><code>-list_parameters_of_SRG</code></td>
<td><code>v_{max}</code></td>
<td>Performs a sift for putative parameter sets of SRGs.</td>
</tr>
<tr>
<td><code>-conjugacy_classes_Sym_n</code></td>
<td><code>n</code></td>
<td>Compute a list of conjugacy classes of $\text{Sym}(n)$.</td>
</tr>
<tr>
<td><code>-tree_of_all_k_subsets</code></td>
<td><code>n k</code></td>
<td>Creates a tree-file for all $k$-subsets of an $n$-set.</td>
</tr>
<tr>
<td><code>-Delandtsheer_Doyen</code></td>
<td></td>
<td>See Section 11.7.</td>
</tr>
<tr>
<td><code>-tdo_refinement</code></td>
<td></td>
<td>See Section 11.8.</td>
</tr>
<tr>
<td><code>-tdo_print</code></td>
<td></td>
<td>See Section 11.8.</td>
</tr>
<tr>
<td><code>-convert_stack_to_tdo</code></td>
<td></td>
<td>See Section 11.8.</td>
</tr>
<tr>
<td><code>-maximal_arc_parameters</code></td>
<td></td>
<td>See Section 11.8.</td>
</tr>
<tr>
<td><code>-arc_parameters</code></td>
<td></td>
<td>See Section 11.8.</td>
</tr>
<tr>
<td><code>-pentomino_puzzle</code></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>Description</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 conjugacy classes of $\text{Sym}(10)$ arranged by size

The following command creates the character table of $\text{Sym}(4)$.

```
Char_Sym_4:
▷ $(\text{ORBITER}) \ -v\ 2\ -\text{character\_table\_symmetric\_group\ 4}$
```

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:
▷ $(\text{ORBITER}) \ -v\ 4\ \$
▷ ▷ \ -\text{create\_random\_k\_subsets\ 10\ 5\ 20}$
```

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:
▷ $(\text{ORBITER}) \ -v\ 2\ \$
▷ ▷ \ -\text{rank\_k\_subset\ 10\ 3\ 0,1,2,0,3,4,1,3,5,2,4,5,3,6,7,1,6,8,0,6,9}$
```
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:

```plaintext
Walsh_matrix_4:
▷ $(ORBITER) -v 3 \
▷ -define F -finite_field -q 2 -end \
▷ -with F -do -finite_field_activity \
▷ -Walsh_matrix 4 -end
▷ $(ORBITER) -v 2 -draw_matrix \
▷ -input_csv_file Walsh_01_4.csv \
▷ -box_width 10 -bit_depth 24 -partition 3 16 16 -end
▷ #pdflatex GF_2.tex
▷ #open GF_2.pdf
```

The following command creates the matrix of Dedekind numbers of order at most 10:

```plaintext
Dedekind_10_10:
▷ $(ORBITER) -v 3 -Dedekind_numbers 2 10 2 10
```

The following command creates the elementary symmetric functions in 4 variables.

```plaintext
elementary_symmetric_functions_4:
▷ $(ORBITER) -make_elementary_symmetric_functions 4 4
```

The output is:

```
k=1 :
x0 + x1 + x2 + x3
k=2 :
x0*x1 + x0*x2 + x0*x3 + x1*x2 + x1*x3 + x2*x3
k=3 :
x0*x1*x2 + x0*x1*x3 + x0*x2*x3 + x1*x2*x3
k=4 :
x0*x1*x2*x3
```

Orbiter can compute domino portraits. To do so, we need an input file in r/g/b format of size $(D + 1)s \times Ds$, where $D = 7$ for double-six dominos.

```plaintext
domino_portrait:
▷ $(ORBITER) -v 3 -domino_portrait 7 4 anton_28x32 -end
```
Figure 11.2: Domino Portrait
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.

```bash
domino
portrait
input:
⊿ $(ORBITER) -v 2 
⊿ ⊿ -define all_one_r -vector -repeat 1 28 -end 
⊿ ⊿ -define all_one_c -vector -repeat 1 32 -end 
⊿ ⊿ -draw_matrix 
⊿ ⊿ -grayscale 
⊿ ⊿ -invert_colors 
⊿ ⊿ -input_csv_file anton_28x32.m.csv 
⊿ ⊿ -box_width 20 -bit_depth 8 
⊿ ⊿ -partition 3 
⊿ ⊿ ⊿ all_one_c all_one_r 
⊿ ⊿ -end
open anton_28x32_m_draw.bmp
```

The grayscale version of the input file is 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

\[
Ax = 1
\]

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 \( \text{TEST\_SYSTEM} \) for the coefficient matrix and \( \text{TEST\_RHS} \) for the right hand side.

\[
\text{TEST\_SYSTEM}="\ \\
0,1,0,1,0,0, \\
0,0,1,0,1,0, \\
1,0,1,0,0,0, \\
0,1,0,1,0,1, \\
1,0,0,0,1,1, \\
1,0,1,0,0,0, \\
0,1,0,0,1,1"
\]

\[
\text{TEST\_RHS}="1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1"
\]

```
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
```
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Description</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 $\text{PG}(2,q)$. 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 $\text{PG}(2,q)$ for maximal arcs.</td>
</tr>
<tr>
<td>-override_polynomial</td>
<td>a</td>
<td>Use polynomial numerically coded as $a$ for creating $\mathbb{F}_q$.</td>
</tr>
</tbody>
</table>

Table 11.3: Orbiter Commands to create Diophantine systems
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Description</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.</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 \n▷ ▷ -diophant -label test_system \n▷ ▷ ▷ -coefficient_matrix A \n▷ ▷ ▷ -RHS $(TEST_RHS) \n▷ ▷ ▷ ▷ -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

\[
\binom{n}{2} = \sum_{j=2}^{n} a_j \binom{j}{2}
\]

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

```
linsp6:
$ (ORBITER) -v 4 \n  -define A -vector -format 1 -dense "15,10,6,3,1" -end \n  -define D -diophant -label linsp6 \n  -coefficient_matrix A \n  -RHS "15,15,1" \n  -x_min_global 0 \n  -x_max_global 15 \n  -end \n  -with D -do \n  -diophant_activity -solve_mckay \n  -end
#
```

solves the system using McKay’s program possolve [50]. The program finds 15 solutions, written to the file \texttt{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

\[p_j = \#j\text{-lines though } P.\]
We are trying to precompute the matrix of point types

\[
(p_{ij})
\]

where \( j = 7, 5, 4 \) and \( i \) belongs to an index set of all possible point types. Fixing a point \( P \), counting points \( Q \neq P \) collinear with \( P \) yields

\[
6p_7 + 4p_5 + 3p_4 = 29, \quad p_7 \leq 1, \quad p_5 \leq 27, \quad p_4 \leq 24.
\]

Using the Orbiter commands

\[
\text{linsp30.pt.types:}
\]
\[
\begin{align*}
\text{\( $(\text{ORBITER}) -v 4 \) \}} \& \text{-define A -vector -format 1 -dense "6,4,3" -end \}} \& \\
\text{-define D -diophant \}} \& \\
\text{-label linsp30.pt.types \}} \& \\
\text{-coefficient_matrix A \}} \& \\
\text{-RHS "29,29,1" -x_bounds "0,1,0,27,0,24" \}} \& \\
\text{-end \}} \& \\
\text{-with D -do \}} \& \\
\text{-diophant_activity -solve_mckay \}} \& \\
\text{-end \}}
\end{align*}
\]

we determine the possibilities

\[
(p_7, p_5, p_4) = \begin{cases} 
1 & 5 & 1 \\
1 & 2 & 5 \\
0 & 5 & 3 \\
0 & 2 & 7 
\end{cases}
\]

The rows in this matrix are called the point types \((i = 0, 1, 2, 3)\). Let \( b_i \) be the number of points of type \( i \). By counting points, incident (point,line) pairs by \( j \)-lines and pairs of intersecting \( j \)-lines, we arrive at the following system:

\[
\begin{align*}
b_0 + b_1 + b_2 + b_3 &= 30 \\
b_0 + b_1 &= 7 \\
5b_0 + 2b_1 + 5b_2 + 2b_3 &= 135 = 27 \cdot 5 \\
b_0 + 5b_1 + 3b_2 + 7b_3 &= 96 = 24 \cdot 4 \\
10b_0 + b_1 + 10b_2 + b_3 &\leq 351 = \binom{27}{2} \\
10b_1 + 3b_2 + 21b_3 &\leq 276 = \binom{24}{2}
\end{align*}
\]

Using the Orbiter commands
we determine the possibilities

\[
(b_0, b_1, b_2, b_3) = \begin{pmatrix}
2 & 5 & 23 & 0 \\
3 & 4 & 22 & 1 \\
4 & 3 & 21 & 2 \\
5 & 2 & 20 & 3 \\
6 & 1 & 19 & 4 \\
7 & 0 & 18 & 5
\end{pmatrix}
\]
11.4 Classification of Configurations and Geometries

A partial linear space is a set system on a fixed set $V$. We write $\mathcal{L} = (V, \mathcal{B})$, where $\mathcal{B}$ is a set of distinct subsets of $V$, called lines. The members of $V \cup \mathcal{B}$ are called elements. For two elements $x, y$, we say that $x$ is incident with $y$, written $xIy$, if either $x \in y$ or $y \in x$. We require that any line has at least two points and any two points are contained in at most one line. A decomposition of a linear space is a partition $\Pi = (C_1, \ldots, C_n)$ of $V \cup \mathcal{B}$ such that each $C_i$ either is a subset of $V$ or a subset of $\mathcal{B}$. A decomposition is called tactical if for all $i$, the incidence number

$$\iota(C_i, C_j) = \#\{y \in C_j, xIy\}$$

does not depend on the choice of $x \in C_i$. Any linear space has a tactical decomposition, as the discrete partition (every element is in its own class) is tactical. Let $\text{Aut}(\mathcal{L})$ be the automorphism group of the linear space, which is the subgroup of $\text{Sym}(V)$ which preserves incidence. For $\alpha \in \text{Aut}(\mathcal{L})$ we say that the decomposition $\Pi$ preserves $\alpha$ if $\alpha$ fixes every class of $\Pi$. For $A \leq \text{Aut}(\mathcal{L})$, we say that $\Pi$ preserves $A$ is $\Pi$ preserves every element $\alpha \in A$. Mostly, we are interested in those decompositions $\Pi$ which preserve $\text{Aut}(\mathcal{L})$. In light of this, the discrete decomposition is not that interesting.

Any linear space has a coarsest tactical decomposition that preserves its automorphism group: The orbit partition of the automorphism group acting on $V \cup \mathcal{B}$ will do. Up to ordering of the classes, the coarsest tactical refinement is unique. Computing the orbit decomposition is challenging as it involves computing the automorphism group. Computationally, there are easier ways to get to admissible decompositions. One 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 $\mathcal{P}$. Simply start with the decomposition of two classes, one the set of points and one the set of blocks, and refine. The output may or may not be equal to the decomposition arising from the orbit partition of $\text{Aut}(\mathcal{L})$.

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 -geometry_builder option can answer these kinds of questions.

The command

```
geo_10_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  ▶ ▶ ▶ -test Test_lines \n```

339
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 \textit{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. A file $10\_3$\_inc is written which contains all the partial linear spaces admitting the tactical decomposition. 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 last row start with $-1$. 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 66 69 74 78 79 85 87 89 96 97 98
0 1 2 10 13 14 20 25 26 31 33 35 41 44 47 52 54 58 62 66 69 73 78 79 85 87 89 96 97 98
0 1 2 10 13 14 20 25 26 31 33 35 41 44 47 52 54 58 62 66 69 73 76 79 85 88 89 96 97 98
0 1 2 10 13 14 20 25 26 31 33 35 41 44 47 52 56 58 62 67 69 73 78 79 84 86 89 95 97 98
0 1 2 10 13 14 20 25 26 31 33 35 41 44 47 52 54 57 62 66 69 73 77 79 84 86 89 95 98 99
0 1 2 10 13 14 20 25 26 31 33 35 41 47 48 52 54 57 62 66 68 73 77 79 84 86 89 95 98 99
0 1 2 10 13 14 20 25 26 31 33 35 41 47 48 52 54 57 62 66 68 73 78 79 84 86 89 95 97 99
0 1 2 10 13 14 20 25 26 31 33 35 41 47 48 52 54 57 62 66 69 73 78 79 84 86 88 95 97 99
0 1 2 10 13 14 20 25 26 31 33 37 41 45 48 52 54 57 62 66 68 73 75 79 84 86 89 97 98 99
0 1 2 10 13 14 20 25 26 31 33 37 41 45 48 52 54 57 62 66 68 73 75 79 84 88 89 96 97 99
0 1 2 10 13 14 20 25 26 31 33 37 41 45 48 52 54 57 62 66 68 73 76 79 84 88 89 95 97 99
-1 10
120, 24, 12, 10, 6, 4^2, 3^2, 2
```

Two further files are written, containing the lines of the incidence geometry. The file $10\_3$\_blocks:

```
10 10 3
0 15 26 44 51 68 81 109 114 116
0 15 26 46 49 68 81 109 114 116
0 15 26 46 49 68 83 106 115 116
0 15 26 46 52 69 77 106 114 116
0 15 26 46 56 69 80 101 106 119
0 15 26 46 56 69 80 103 104 119
0 15 26 46 56 69 80 103 107 117
0 15 26 46 56 72 80 93 106 119
0 15 26 46 56 72 81 93 105 119
0 15 26 46 56 74 79 93 105 119
-1 10
120, 24, 12, 10, 6, 4^2, 3^2, 2
```

contains the blocks as ranked 3-subsets of a 10-element set. The file $10\_3$\_blocks\_long contains the list of blocks written out.

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 \texttt{fname\_GEO} option. Here, the \texttt{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 green nodes are nodes that are accepted. The red nodes are nodes that are rejected. This means they represent geometries that have been seen before. The 10 green nodes at the very bottom of the diagram represent the 10 10_3 configurations.

geo_10_3_tree:
  ▶ $(ORBITER) -v 20 \n  ▶ ▶ -define Test_lines -set -loop 0 11 1 -end \n  ▶ ▶ -define GEO -geometry_builder \n  ▶ ▶ ▶ -V 10 -B 10 -TDO 3 -fuse 1 \n  ▶ ▶ ▶ -fname GEO 10_3 \n  ▶ ▶ ▶ -search_tree \n  ▶ ▶ ▶ -test Test_lines \n  ▶ ▶ -end \n  ▶ $(ORBITER) -v 20 \n  ▶ ▶ -draw_options -embedded -radius 20 \n  ▶ ▶ ▶ -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  ▶ ▶ -end \n  ▶ -tree_draw \n  ▶ ▶ -file 10_3_tree.txt \n  ▶ ▶ -end \n  ▶ pdflatex 10_3_tree_draw.tex \n  ▶ open 10_3_tree_draw.pdf

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 if 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:
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 -girth 4 \
  \  \  \ -search_tree \
  \  \  \ -test Test_lines \
  \  \ -end
```
This command takes about 8 minutes of time to complete. The next command classifies the 15₃ with girth 4. Only one configuration arises, the Cremona Richmond, with automorphism group Sym(6) of order 720.

geo_15.3_g4:

$\texttt{(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 \\
-girth 4 \\
-search_tree \\
-test Test_lines \\
-end \\
\texttt{(ORBITER)} -v 2 \\
-draw_options -embedded -radius 50 \\
-nodes_empty \\
-scale 0.5 -line_width 0.3 -end \\
-tree_draw -file 15_3_g4_tree.txt -end \\
pdflatex 15_3_g4_tree_draw.tex \\
open 15_3_g4_tree_draw.pdf
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 \ndefine D -design -q 3 -family PG_2.q -end \n-define D -do \ndesign_activity \ndelete_inc \ndelete -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:

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

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

\[
\begin{array}{c|c|c}
\rightarrow & 13_1 & \downarrow 13_1 \\
13_0 & 4 & 13_0 & 4
\end{array}
\]

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 $\text{Sym}(4) \triangleright \text{Sym}(2)$. The design has 12 blocks of size 4. The command

\begin{verbatim}
wreath_product_designs_n8_k6_inc.txt:
$ (ORBITER) -v 8 \\
    define D -design -wreath_product_designs 8 6 -end \\
\end{verbatim}

345
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

$$M_{t,k}(G) \cdot \mathbf{x} = \lambda \mathbf{1},$$

where $\mathbf{x}$ is a column vector of zeros and ones and $\mathbf{1}$ is the column vector of all ones. The length of $\mathbf{x}$ is the number of $k$-orbits of $G$ on $V$, while the length of $\mathbf{1}$ is the number of $t$-orbits of $G$ on $V$. Any vector $\mathbf{x}$ satisfying the matrix equation corresponds to a design invariant under $G$. Simply take the blocks of the design to be the union of those orbits of $G$ on $k$-subsets whose associated entry in $\mathbf{x}$ is one. We assume the group $\text{PGL}(2,32)$ in the action on points of the projective line $\text{PG}(1,32)$ over the field $\mathbb{F}_{32}$. The parameters of the design are $7$-$(33,8,10)$, that is, each 7-subset of $\text{PG}(1,32)$ 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

$$\text{KM\_PGGL\_2\_32\_KM\_7\_8.csv}.$$

The second command produces the graphical representation of the matrix shown in Figure 11.5 (different colors represent different values of entries in the matrix). The third Orbiter command creates the diophantine system associated with the Kramer-Mesner matrix.

346
Figure 11.5: Kramer-Mesner matrix $M_{7,8}(PGL(2,32))$

```
KM_PGGL_2_32:
  $(ORBITER) -v 3 \n  -define G -linear_group -PGGL 2 32 -end \n  -with G -do \n  -group_theoretic_activity \n  -poset_classification_control \n  -problem_label KM_PGGL_2_32 -W -depth 8 \n  -Kramer_Mesner_matrix 7 8 \n  -draw_poset \n  -draw_options -embedded -sideways -radius 50 \n  -scale 0.5 -line_width 0.3 -end \n  -orbits_on_subsets 8 \n  -end
$(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
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
$(ORBITER) -v 4 \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  -with D -do \n```
The last command performs a complete enumeration of all solutions by solving the system and producing the solution vectors \( \mathbf{x} \) 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 40 43 46 49 53 56 59 62 64 69 71 74 78 80 82 87 89 93 94 99 102 103 107
-1 1
432
```

This shows that the design is unique, and has an automorphism group of order 432. For the following 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"
```

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:

349
The files $\text{AG}_2_3\text{design_table_disjoint_sets_sol.txt}$ and $\text{AG}_2_3\text{design_table_disjoint_sets_sol.csv}$ 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:

```bash
LS_AG_2_3_export_solutions:
  $\text{ORBITER} -v 20 \
  $\text{define D -design -list_of_blocks 9 3} \$
  $\text{define Sym9 -permutation_group -symmetric_group 9} \$
  $\text{define T -design_table D "AG}_2_3" \text{Sym9} \$
  $\text{with D -do} \$
  $\text{-design activity} \$
  $\text{-extract_solutions_by_index "AG}_2_3" \text{Sym9} \$
  $\text{AG}_2_3\text{design_table_disjoint_sets_sol.csv} \$
  $\text{solutions.csv} \$
  "$" \$
  $\text{-end} \$
```

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 $q_{d_1}^1 \times a_{d_2}^2$ 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 $(\tau_1, \tau_2) : \mathbb{Z}_3 \times \mathbb{Z}_7 \rightarrow \mathbb{Z}_3 \times \mathbb{Z}_7$,

where

$\tau_1 : \mathbb{Z}_3 \rightarrow \mathbb{Z}_3, x \mapsto x + 1 \mod 3, \quad \tau_2 : \mathbb{Z}_7 \rightarrow \mathbb{Z}_7, y \mapsto y + 1 \mod 7.$

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=/
  -nb_orbits_on_blocks 1 \
  -depth 5 \
  -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 \
  > -Delandtsheer_Doyen $(PP4) $(PP4_GROUP1) $(PP4_MASK1) \
  > end \
```

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 11.5 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} |
```

353
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-lambda3</td>
<td>λ₃ s</td>
<td>Refine as 3-design with λ₃ and with block-size s</td>
</tr>
<tr>
<td>-solution</td>
<td>i fname</td>
<td>Use solutions to system i from file frame.</td>
</tr>
<tr>
<td>-range</td>
<td>f l</td>
<td>Refine cases i with f ≤ 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 x₀ ≤ 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 11.5: 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
> max_arc_16_4r_print:
> $(ORBITER) -v 4 -tdo_print max_arc_q16_r4r.tdo
```

is used. Because the incidence structure is a projective plane, the dual is a linear space also. Hence the option `-dual_is_linear_space` can be used, which is helpful to reduce possibilities. As it turns out, there is exactly one refinement, and it is tactical. The file `max_arc_q16_r4r.tdo` is produced. Note the added letter `r` at the end of the file name (`r` for refinement). We can use the following command to display the decomposition stack in the file:

```
max_arc_16_4r_print:
> $(ORBITER) -v 4 -tdo_print max_arc_q16_r4r.tdo
```

This produces the following output:

```
decomposition 0.1:
lambda_scheme at level 2 :
is 1 x 1
| 273_{ 1} |
====================
273_{ 0} | 17 17
```

```
| 52_{ 0} | 17 | 0 |
| 221_{ 3} | 13 | 4 |

col_scheme at level 4 :
is 2 x 2
| 221_{ 1} | 52_{ 2} |

| 52_{ 0} | 4 | 0 |
| 221_{ 3} | 13 | 17 |

extra_col_scheme at level 3 :
is 1 x 2
| 221_{ 1} | 52_{ 2} |

| 273_{ 0} | 17 | 17 |
Chapter 12

Finite Geometry

12.1 Spreads

A $t$-spread of $PG(n, q)$ is a set of disjoint $PG(t, q)$ that cover all of $PG(n, q)$ pointwise. $t$-spreads in $PG(n, q)$ exist if and only if $t + 1$ divides $n + 1$. 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:
\[
\begin{align*}
\text{create} & \quad \text{spread} & \quad 9a: \\
\text{-} & \quad (\text{ORBITER}) \ -v \ 3 \ \\
\text{-} & \quad \text{-define} \ F \ -finite\_field \ -q \ 3 \ \text{-end} \\
\text{-} & \quad \text{-define} \ G \ -linear\_group \ -PGL \ 4 \ F \ \text{-end} \\
\text{-} & \quad \text{-define} \ S \ -spread \ -kernel\_field \ F \\
\text{-} & \quad \text{-group} \ G \ -k \ 2 \ -catalogue \ 0 \\
\text{-} & \quad \text{-end}
\end{align*}
\]

create spread 9b:
\[
\begin{align*}
\text{create} & \quad \text{spread} & \quad 9b: \\
\text{-} & \quad (\text{ORBITER}) \ -v \ 3 \ \\
\text{-} & \quad \text{-define} \ F \ -finite\_field \ -q \ 3 \ \text{-end} \\
\text{-} & \quad \text{-define} \ G \ -linear\_group \ -PGL \ 4 \ F \ \text{-end} \\
\text{-} & \quad \text{-define} \ S \ -spread \ -kernel\_field \ F \\
\text{-} & \quad \text{-group} \ G \ -k \ 2 \ -catalogue \ 1 \\
\text{-} & \quad \text{-end}
\end{align*}
\]

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 $F_q$ 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 [52] can be defined using the following makefile variable.
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Description</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 spread-set $S$. 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

```
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,
2,0,2,0,2,1,1,0,
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,
2,1,2,0,2,0,2,0,1,
0,1,2,2,0,1,0,1,1,
1,0,0,0,1,0,0,0,1,
2,1,0,1,2,1,0,2,0,
0,2,0,0,2,2,1,1,2,
0,0,1,0,1,2,2,1,1,
2,0,1,2,2,1,1,0,1,
0,1,1,1,0,1,2,2,1,
2,2,0,2,0,0,0,2,2,
2,1,1,1,1,2,2,1,2,
2,2,1,2,1,0,2,0,0,
1,2,0,2,0,2,1,0,0,
```

358
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:
  $\text{ORBITER} -v 3 \$
  ▶ -define F -finite_field -q 3 -end \\
  ▶ -define SS -vector -dense $(\text{SPREAD\_SET\_27\_RAO\_RAO}) -end \\
  ▶ -define G -linear_group -PGL 6 F -end \\
  ▶ -define S -spread -kernel_field F \\
  ▶ ▶ -group G -k 3 -spread_set SS \\
  ▶ ▶ -end
```

The following command creates the Desarguesian line-spread in PG(3, 2):

```
desarguesian_spread_in_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 \\
  ▶ -cheat_sheet_desarguesian_spread_3_2.tex \\
  ▶ open Desarguesian_Spread_3_2.pdf
```

The cheat sheet contains the following spread:

<table>
<thead>
<tr>
<th>Spread element 0 is (1, 0) =</th>
<th>1 0 0 0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0 1 0 0</td>
</tr>
<tr>
<td></td>
<td>0 0 1 0</td>
</tr>
<tr>
<td></td>
<td>0 0 0 1</td>
</tr>
<tr>
<td>Spread element 1 is (0, 1) =</td>
<td>1 0 1 0</td>
</tr>
<tr>
<td></td>
<td>0 1 0 1</td>
</tr>
<tr>
<td></td>
<td>1 1 1 0</td>
</tr>
<tr>
<td></td>
<td>1 0 0 1</td>
</tr>
<tr>
<td>Spread element 2 is (1, 1) =</td>
<td>1 0 1 0</td>
</tr>
<tr>
<td></td>
<td>0 1 0 1</td>
</tr>
<tr>
<td></td>
<td>1 1 1 0</td>
</tr>
<tr>
<td></td>
<td>1 0 0 1</td>
</tr>
<tr>
<td>Spread element 3 is (2, 1) =</td>
<td>1 0 0 1</td>
</tr>
</tbody>
</table>

359
Spread element 4 is \((3, 1) = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}\).

Spread elements by rank: \((0, 34, 9, 17, 22)\).

The following command creates the Desarguesian plane-spread in \(\text{PG}(5, 2)\):

```
desarguesian_spread_in_PG_5_2:
  $\text{ORBITER} -v 3$
  $\text{-define FQ -finite_field -q 8 -end}$
  $\text{-define Fq -finite_field -q 2 -end}$
  $\text{-with FQ -and Fq -do -finite_field_activity}$
  $\text{-cheat_sheet_desarguesian_spread 2 -end}$
  pdflatex Desarguesian_Spread_5_2.tex
  open Desarguesian_Spread_5_2.pdf
```

Spread element 0 is \((1, 0) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}\).

Spread element 1 is \((0, 1) = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}\).

Spread element 2 is \((1, 1) = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}\).

Spread element 3 is \((2, 1) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}\).

Spread element 4 is \((3, 1) = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}\).

Spread element 5 is \((4, 1) = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}\).
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 \( \mathbb{F}_4 \).

```
classify_spreads16_4:
  > $(ORBITER) -v 6 \n  >   -orbiter_path $(ORBITER_PATH) \n  >   -define F -finite_field -q 4 -end \n  >   -define P -projective_space 3 F -end \n  >   -with P -do \n  >   -projective_space_activity \n  >   -spread_classify 2 \n  >     -problem_label spreads_4_2 \n  >     -W -depth 17 -draw_poset \n  >     -draw_options -radius 20 \n  >     -nodes_empty -line_width 0.2 -embedded \n  >     -end \n  >   -report \n  >   -end
```

\[
\text{Spread element 6 is } (5,1) = \begin{bmatrix}
0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 \\
\end{bmatrix}, \quad 792
\]

\[
\text{Spread element 7 is } (6,1) = \begin{bmatrix}
0 & 0 & 1 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 \\
\end{bmatrix}, \quad 1161
\]

\[
\text{Spread element 8 is } (7,1) = \begin{bmatrix}
1 & 0 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 \\
\end{bmatrix}, \quad 373
\]

Spread elements by rank: ( 0, 1394, 189, 671, 562, 1040, 792, 1161, 373 )
can be used. The command builds the poset of orbits for the group $G = \text{PGL}(4, 4)$ acting on the poset of partial spreads in PG(3, 4), shown in Figure 12.1. Up to isomorphism, there are exactly three line-spreads in PG(3, 4) (corresponding to the three nodes at the bottom of the poset of orbits in the figure). These three spreads are the dearguesian spread, the Hall spread, and the semifield spread, respectively. Here is the relevant output taken from the latex report:

There are 3 orbits at level 17.

**Orbit 0 / 3 at Level 17**

Node number: 1126

\{0, 25, 50, 75, 90, 107, 122, 140, 144, 157, 179, 204, 213, 238, 268, 334, 345\}_{1200}

Strong generators for a group of order 1200:

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

1, 0, 0, 0, 1, 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

**Orbit 1 / 3 at Level 17**

Node number: 1127

\{0, 25, 50, 75, 90, 107, 140, 157, 179, 204, 213, 238, 265, 282, 299, 316, 356\}_{81600}

Strong generators for a group of order 81600:

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

1, 0, 0, 0, 1, 0, 2, 3, 0, 2, 1, 1, 3, 2, 0,
Figure 12.1: The poset of orbits of partial spreads in $PG(3, 4)$
\[
\begin{bmatrix}
\omega^2 & 0 & 0 & 0 \\
\omega & \omega^2 & 0 & 0 \\
0 & 0 & \omega^2 & 0 \\
0 & 0 & \omega & 1 \\
\end{bmatrix}, \quad 
\begin{bmatrix}
\omega^2 & 0 & 0 & \omega \\
\omega & \omega^2 & 1 & \omega^2 \\
0 & 0 & 1 & 0 \\
0 & 0 & \omega^2 & 1 \\
\end{bmatrix}, \quad 
\begin{bmatrix}
0 & \omega^2 & \omega^2 & 0 \\
1 & 0 & \omega^2 & \omega^2 \\
0 & 0 & 1 & \omega^2 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}, \quad 
\begin{bmatrix}
0 & 0 & 0 & 1 \\
0 & \omega & 1 \\
0 & 1 & \omega^2 \\
\omega & 1 & 1 & 1 \\
\end{bmatrix}
\]

1,0,0,0,0,1,0,0,0,0,3,0,0,0,0,0,3,0,
1,0,0,0,0,1,0,0,0,0,2,3,0,0,1,1,0,
1,0,0,0,0,1,0,0,2,1,3,1,2,3,2,2,0,
1,0,0,0,1,0,0,0,0,0,1,0,0,3,2,1,
1,0,0,0,3,1,2,1,0,0,2,0,0,0,1,2,1,
0,1,1,0,2,0,1,1,0,2,1,0,0,0,2,0,
0,0,0,1,0,0,2,1,0,1,2,3,2,1,1,1,0,
There are 0 extensions
Number of generators 7

## Orbit 2 / 3 at Level 17

Node number: 1128

\{0, 25, 50, 75, 90, 108, 122, 140, 158, 183, 199, 217, 233, 250, 268, 312, 345\}_{576}

Strong generators for a group of order 576:

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

1,0,0,0,0,2,0,0,0,0,2,0,0,0,0,0,0,3,1,
1,0,0,0,0,1,0,0,3,0,3,2,1,0,0,2,0,
1,0,0,0,3,0,0,0,2,2,1,0,1,2,0,
1,1,1,2,0,2,0,2,0,2,1,0,2,2,3,0,
1,0,3,1,1,3,1,0,1,0,2,0,0,0,1,1,
0,1,1,0,0,0,1,2,0,1,1,3,2,3,2,0,
| OCN | |Aut| | Name           |
|-----|-----|-----|----------------|
| 0   | 1200|     | Hall spread    |
| 1   | 81600|    | Desarguesian spread |
| 2   | 576 |     | Semifield spread |

Table 12.2: Spreads in \( \text{PG}(3, 4) \) in the Orbiter Catalogue

There are 0 extensions
Number of generators 6

The three spreads in \( \text{PG}(3, 4) \) can be distinguished by their stabilizer orders. Table 12.2 lists the line spreads in \( \text{PG}(3, 4) \) according to their orbiter catalogue number (OCN).

Let us now look at a more difficult problem. We wish to classify the spreads in \( \text{PG}(3, 5) \). 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 \( \text{PG}(3, 5) \) 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 \( \text{PG}(3, 5) \). 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

```bash
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 \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  ▶ ▶ ▶ ▶ ▶ -embedded \n  ▶ ▶ ▶ ▶ ▶ -end \n```
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 \
  -with G -do \
  -graph_theoretic_activity \
  -find_cliques -rainbow -target_size 21 -end \
  -end
```

can be used.

The command

```
classify_spreads_25_starter_lift_all_cases:
  $(ORBITER) -v 10 \
  -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 \
```

366
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:
  $(ORBITER) -v 2 \\
  -loop L 0 29 1 \\
  -define G -graph -load spreads_25_graph_%L.bin -end \\
  -with G -do \\
  -graph_theoretic_activity \\
  -find_cliques -rainbow -target_size 21 -end \\
  -end \\
  -end_loop
```

The resulting cliques are again stored in files. The command

```
classify_splits_25_isomorph:
  $(ORBITER) -v 3 \\
```

367
-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 \ 
-end \ 
-draw_poset \ 
-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 \ 
."/" \ 
."/" \ 
-use_database_for_starter \ 
-build_db \ 
-solution_prefix "" \ 
-base_fsize "" \ 
-end \ 
-end \ 
-with C -do -spread_classify_activity \
-isomorph \ 
."/" \ 
."/" \ 
-use_database_for_starter \ 
-read_solutions \ 
-solution_prefix "" \ 
-base_fsize "spreads_25_graph" \ 

Table 12.3: Spreads in PG(7, 2) in the Orbiter Catalogue

| OCN | |Aut| | Name |
|-----|-------|---------|
| 0   | 1008  |
| 1   | 1008  |
| 2   | 1728  |
| 3   | 216   |
| 4   | 360   |
| 5   | 288   |
| 6   | 3600  |
| 7   | 244800|

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).
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 $F = \mathbb{F}_q$ 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 \
  -define F -finite_field -q 3 -end \
  -define G -linear_group -PGL 4 F -end \
  -define G1 -linear_group -PGL 5 F -end \
  -define S -spread -kernel_field F \
  -group G -k 2 -catalogue 1 \
  -define T -translation_plane S G G1 -end \
  $(ORBITER) -v 2 \
  -draw_matrix \
  -input_csv_file plane_catalogue_q3_k2_1_incma.csv \
  -box_width 6 -bit_depth 8 \
  -partition 2 91 91 \
  -end 
open plane_catalogue_q3_k2_1_incma_draw.bmp
```

creates the (projective) Hall plane of order 9 from the Hall spread. In this example, we use the fact that $\text{PGL}(n, q) = \text{PGL}(n, q)$ if $q$ is prime. The example also creates a bitmap drawing of the incidence matrix of the plane, shown in Figure 12.2.
Figure 12.2: Incidence matrix of the Hall plane of order 9
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.

A packing is regular if it consists solely of regular spreads. The smallest regular packings exist in \( \text{PG}(3, 5) \). They were first described by Prince [58] and later placed into an infinite family by Penttila and Williams [55]. Up to isomorphism, there are exactly two regular packings in \( \text{PG}(3, 5) \). Let us construct these packings. We start by making a table of all regular packings:

\[
\text{spread\_table\_PG\_3\_5\_regular:}
\]

There are 155,000 packings. In the command, we rely on the classification of spreads in \( \text{PG}(3, 5) \) which is built into Orbiter. The spread with orbiter catalogue number 12 is the regular spread.

We consider the projectivity of order 31 given by the matrix

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

The next command computes the normalizer of the cyclic subgroup of order 31 generated by this element:
<table>
<thead>
<tr>
<th>Modifier</th>
<th>Args</th>
<th>Meaning</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 Tables 6.2-6.3.</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>Modifier</th>
<th>Args</th>
<th>Meaning</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
PG_3_5_element_of_order_31_ME_normalizer:
▷ $(ORBITER) -v 6 -define G \
▷  -linear_group -PGL 4 5 -end \
▷  -with G -do \
▷  -group_theoretic_activity \
▷  ▷ -normalizer_of_cyclic_subgroup "31" \
▷  ▷  "1,0,0,0, 0,3,4,3, 0,3,3,4, 0,3,2,3" \
▷  ▷ -end \
▷ mv normalizer_of_31_in_PGL_4_5.tex normalizer_of_31_ME_in_PGL_4_5.tex  
▷ pdflatex normalizer_of_31_ME_in_PGL_4_5.tex  
▷ open normalizer_of_31_ME_in_PGL_4_5.pdf

The normalizer is a group of order 372. We encode the group and its normalizer as makefile variables:

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,"

Let $H$ be the subgroup of order 31 and let $N$ be its normalizer. Then we compute the orbits of $H$ on the regular spreads:

PG_3_5_assume_31_graph:
▷ $(ORBITER) -v 5 \
▷  ▷ -define F -finite_field -q 5 -end \
▷  ▷ -define P -projective_space 3 F -end \
▷  ▷ -define T -spread_table P 2 "12" "SPREAD_TABLES_5_REG/" \
▷  ▷  ▷ -H "H31" $(PGL_4_5_SUBGROUP_31_ME) -end \
▷  ▷  ▷ -N "N31" $(PGL_4_5_SUBGROUP_31_ME_NORMALIZER) -end \
▷  ▷  ▷ -end \
▷  ▷ -define PWF -packing_choose_fixed_points PW 0 -end \
▷  ▷ -define L -packing_long_orbits PWF \
▷  ▷  ▷ -orbit_length 31 -clique_size 1 -create_graphs -end \
▷  ▷ -print_symbols \
▷ pdflatex H31_reduced_spread_orbits_orbits_report.tex

375
The command produces reports about the orbits of both $H$ and $N$ on points, lines and spreads. The following command searches all cliques of size 1 in the graph on long orbits.

```bash
PG_3_5.assume_31_fpc0_lo cliques:
  $(ORBITER) -v 2 \
  -define G -graph -load H31_fpc0_lo.graph -end \
  -with G -do \
  -graph_theoretic_activity \
  -find_cliques -target_size 1 -end -end \
  -print_symbols
```

There are exactly 8 cliques of size 1. The next command builds the packings arising from these 8 cliques:

```bash
PG_3_5.assume_31_read_again:
  $(ORBITER) -v 5 \
  -define F -finite_field -q 5 -end \
  -define P -projective_space 3 F -end \
  -define T -spread_table P 2 "12" "SPREAD_TABLES_5_REG/" \n  -define PW -packing_with_symmetry_assumption T \n  -H "H31" $(PGL_4_5_SUBGROUP_31_ME) -end \n  -N "H31" $(PGL_4_5_SUBGROUP_31_ME) -end \n  -end \
  -define PWF -packing_choose_fixed_points PW 0 -end \
  -define L -packing_long_orbits PWF \n  -orbit_length 31 -clique_size 1 \n  -read_solutions \n  -end \
```

376
<table>
<thead>
<tr>
<th>Modifier</th>
<th>Arguments</th>
<th>Meaning</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 $\text{PG}(3, q)$, to flocks of the quadratic cone in $\text{PG}(3, q)$, 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 [54], 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

```
BLT_11_0:
  ▶ $(ORBITER) -v 2 \n  ▶ ▶ -define F -finite_field -q 11 -end \n  ▶ ▶ -define O -orthogonal_space 0 5 F -end \n  ▶ ▶ -with O -do -orthogonal_space_activity \n  ▶ ▶ ▶ -create_BLT_set -catalogue 0 -end \n  ▶ ▶ -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

```
BLT_11_Mondello:
  ▶ $(ORBITER) -v 2 \n  ▶ ▶ -define F -finite_field -q 11 -end \n  ▶ ▶ -define O -orthogonal_space 0 5 F -end \n  ▶ ▶ -with O -do -orthogonal_space_activity \n```

377
<table>
<thead>
<tr>
<th>$F$</th>
<th>Condition</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td></td>
<td>Linear BLT-set.</td>
</tr>
<tr>
<td>Fisher</td>
<td>$q \equiv \pm 1 \mod 10$</td>
<td>Mondello BLT-set due to Penttila [53].</td>
</tr>
<tr>
<td>Mondello</td>
<td></td>
<td>Mondello BLT-set [27].</td>
</tr>
<tr>
<td>FTWKB</td>
<td>$q \equiv \pm 2 \mod 3$</td>
<td>Fisher, Thas, Walker [66], Kantor, Betten [13] BLT-set.</td>
</tr>
<tr>
<td>Mondello</td>
<td>$q \equiv \pm 1 \mod 10$</td>
<td>Mondello BLT-set due to Penttila [53].</td>
</tr>
<tr>
<td>FTWKB</td>
<td>$q \equiv \pm 2 \mod 3$</td>
<td>Fisher, Thas, Walker [66], Kantor, Betten [13] BLT-set.</td>
</tr>
<tr>
<td>Ketan</td>
<td>$q = p^e, e &gt; 1$</td>
<td>Kantor’s first family.</td>
</tr>
<tr>
<td>Ketan</td>
<td>$q \equiv \pm 2 \mod 5$</td>
<td>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

\[ X_0^2 + X_1X_2 + X_3X_4 = 0 \]

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)

\[
\begin{array}{c|c|c}
\rightarrow & 18_1 & \\
\downarrow & 12_0 & 4 \\
\end{array}
\]

\[C_0 = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}_{12}\]
\[C_1 = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17\}_{18}\]

\[
\begin{array}{c|c|c}
\rightarrow & 18_1 & \\
\downarrow & 12_0 & 4 \\
\end{array}
\]

\[C_0 = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}_{12}\]
\[C_1 = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17\}_{18}\]

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 $Q(4, 13)$:

\textbf{BLT\_13\_deep\_search}:
\quad $\$(\text{ORBITER})$ -v 2 \$
\quad $\quad -\text{define F} -\text{finite\_field} -q 13 -\text{end} \$
\quad $\quad -\text{define O} -\text{orthogonal\_space} 0 5 F -\text{end} \$
\quad $\quad -\text{define C} -\text{BLT\_set\_classifier} 0 -\text{starter\_size} 14 -\text{end} \$

379
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:

```latex
\begin{verbatim}
BLT_13.classify_starter:
  $(ORBITER) -v 2 \\
  -define F -finite_field -q 13 -end \\
  -define O -orthogonal_space 0 5 F -end \\
  -define C -BLT_set_classifier O -starter_size 5 -end \\
  -with C -do -BLT_set.classify_activity \\
  -compute_starter \\
  -problem_label BLT_q13 \\
  -W -depth 5 \\
  -end \\
  -end \\
  -with C -do -BLT_set.classify_activity \\
  -create_graphs \\
  -end
\end{verbatim}
```

In the next step, we compute all rainbow cliques in each of the graphs:

```latex
\begin{verbatim}
BLT_13.clique:\n  $(ORBITER) -v 2 \\
  -loop L 0 38 1 \\
  -define G -graph -load BLT_q13_graph_5_%.bin -end \\
  -with G -do \\
  -graph.theoretic_activity \\
  -find_cliques -rainbow -target_size 9 -end \\
  -end \\
  -end_loop
\end{verbatim}
```

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:
The next step is to read the rainbow cliques from the clique finding process:

```
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 \
>     "./" \
>     "./" \
>     -use_database_for_starter \
>     -build_db \
>     -solution_prefix "" \
>     -base_fname "" \
>     -end \
>   -end 
```

```
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 \
>     "./" \
>     "./" \
>     -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 
```
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  > ./" \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  > report -end \n  > end \n  > with C -do -BLT_set_classify_activity \n  > isomorph \n  > "/" 
```
This last command results in three isomorphism types of BLT-sets of order 13.
Chapter 13

Graph Theory

13.1 Creating Graphs

Table 13.1 shows some Orbiter commands to create graphs.

For instance, the command

```
Cycle_graph_13:
  $(ORBITER) -v 2 \
  ▶ -define Gamma -graph \
  ▶ -cycle 13 \
  ▶ -end
```

creates the cycle graph of degree 13.

There are many ways to read graphs from file. One way is by means of an adjacency matrix stored as a csv file. Consider an example. The `-load_csv_no_border` command can be used to create a graph from a csv file containing the adjacency matrix. The following command sequence uses a makefile variable to store the adjacency matrix of a graph. The matrix is then copied into a file and the graph is created from the file. Here is the makefile variable containing the adjacency matrix:

```
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:

```
make_triangle_graph:
  ▶ echo $(TRIANGLE_GRAPH) >triangle_graph.csv
  ▶ $(ORBITER) -v 6 \
  ▶ ▶ -define G -graph \
  ▶ ▶ ▶ -load_csv_no_border \
  ▶ ▶ ▶ ▶ triangle_graph.csv \
```
<table>
<thead>
<tr>
<th>Key</th>
<th>Arguments</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-load</td>
<td>filename</td>
<td>Read a graph from file in Orbiter format.</td>
</tr>
<tr>
<td>-load_csv_no_border</td>
<td>filename</td>
<td>Read a graph from a csv file. Ignore the first row and first column.</td>
</tr>
<tr>
<td>-load_dimacs</td>
<td>filename</td>
<td>Read a graph from file in dimacs format.</td>
</tr>
<tr>
<td>-edge_list</td>
<td>$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>-triheral_pair_disjointness_graph</td>
<td></td>
<td>Triheral 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>
<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 13.1: Orbiter commands to define graphs

386
This will create the three-cycle graph.

The command

\texttt{Chain\_232:}
\begin{verbatim}
\$ (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{verbatim}

creates the chain graph with respect to the partitions \((2, 3, 2)\) and \((2, 3, 2)\).

The command

\texttt{Paley\_13\_graph:}
\begin{verbatim}
\$ (ORBITER) -v 2 \\
\define Gamma -graph -Paley 13 -end \\
\end{verbatim}

creates the Paley graph on 13 vertices.

The command

\texttt{triheiral\_pair\_graph:}
\begin{verbatim}
\$ (ORBITER) -v 2 \\
\define Gamma \\
\graph -triheiral\_pair\_disjointness\_graph \\
\end{verbatim}

creates the graph of trihedral pairs. Two vertices are adjacent if the associated trihedral pairs are line-disjoint.

The command

\texttt{small\_graph:}
\begin{verbatim}
\$ (ORBITER) -v 2 \\
\define G -graph -edges\_as\_pairs \\
\graph 5 "0,1,0,2,0,3,0,4,1,3,1,4,2,4" \\
\end{verbatim}
creates a graph by listing the edges in pairs. In this case, the graph

\begin{center}
\begin{tikzpicture}
  \node[shape=circle,draw=black] (A) at (0,0) {1};
  \node[shape=circle,draw=black] (B) at (1,1.732) {2};
  \node[shape=circle,draw=black] (C) at (1.732,0) {3};
  \node[shape=circle,draw=black] (D) at (0,-1.732) {0};
  \node[shape=circle,draw=black] (E) at (-1.732,0) {4};

  \draw (A) -- (B);
  \draw (B) -- (C);
  \draw (C) -- (D);
  \draw (D) -- (E);
  \draw (E) -- (A);
\end{tikzpicture}
\end{center}

is created.

The command

```
petersen:
  $ (ORBITER) -v 2 \
  -define G -graph -Johnson 5 2 0 -end
```

creates the Petersen graph.

The command

```
Johnson_6_2_0:
  $ (ORBITER) -v 2 \
  -define G -graph -Johnson 6 2 0 -end
```

creates the Johnson graph \( J(6,2,0) \).

The command

```
Hamming_graph_3:
  $ (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 \( \text{Aut}(HJ) \), 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:

```
```
### Key Arguments Meaning

<table>
<thead>
<tr>
<th>Key</th>
<th>Arguments</th>
<th>Meaning</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 13.2: Orbiter commands to modify graphs

```
HJ_graph:
▷ $(ORBITER) -v 6 \n▷ ▷ -define G -graph \n▷ ▷ ▷ -load_csv_no_border \n▷ ▷ ▷ halljanko315.csv \n▷ ▷ -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:
▷ $(ORBITER) -v 2 \n▷ ▷ -define gens -vector -file \n▷ ▷ ▷ halljanko315_gens.csv -end \n▷ ▷ -define G -permutation_group \n▷ ▷ ▷ -bsgs halljanko315 "File\haljanko315" \n▷ ▷ ▷ 315 1209600 "0,1,2" 6 gens \n▷ ▷ -end \n▷ ▷ -define Gamma -graph \n▷ ▷ ▷ -orbital_graph G 3 \n▷ ▷ -end \n```

Table 13.2 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:
▷ $(ORBITER) -v 6 \n```

389
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:

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:

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>Key</th>
<th>Arguments</th>
<th>Meaning</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 13.3: Graph Theoretic Activities

## 13.2 Graph Theoretic Activities

Graph theoretic activities allow us to perform tasks on graphs. Table 13.3 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

```bash
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
```

392
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:

```
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:

```
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:
<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 energy is 11.5175
Eigenvalues: $\lambda_i$
Laplace eigenvalues: $\theta_i$
Figure 13.1: The Cayley graph in $\mathbb{Z}_{11}$

Cayley_Sym5_coxeter_draw:
\begin{verbatim}
$($ORBITER$) -v 2 \\
   -draw_options -xin 1000000 -yin 1000000 \\
   -embedded -radius 10000 -nodes_empty -end \\
   -define S -vector -dense \\
   "1,0,2,3,4, 0,2,1,3,4, 0,1,3,2,4, 0,1,2,4,3" -end \\
   -define G -permutation_group -symmetric_group 5 \\
   -end \\
   -define Gamma -graph \\
   -Cayley_graph G S \\
   -end \\
   -with Gamma -do \\
   -graph_theoretic_activity -draw -end \\
pdflatex Cayley_graph_Perm5_draw.tex \\
open Cayley_graph_Perm5_draw.pdf
\end{verbatim}

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:

PG0_5.2_collinearity_graph: 0.5.2_incidence_matrix.csv
\begin{verbatim}
$($ORBITER$) -v 3 \\
\end{verbatim}
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 13.4: Options for classifying graphs

<table>
<thead>
<tr>
<th>Option</th>
<th>Arguments</th>
<th>Meaning</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 (-\text{treatment}))</td>
</tr>
</tbody>
</table>

### 13.3 Classification of Graphs and Tournaments

Table 13.4 lists the Orbiter commands to classify graphs and tournaments. The following command classifies all graphs on 5 vertices:

```latex
graph_classify_5:
\begin{verbatim}
▷ $(ORBITER) -v 2 \n▷▷ -oribter_path $(ORBITER_PATH) \\
▷▷ -define GC -graph_classification \\
▷▷▷ -n 5 \\
▷▷▷ -poset_classification_control \\
▷▷▷▷ -problem_label graphs_v5 \\
▷▷▷▷ -depth 10 -draw_poset \\
▷▷▷▷ -draw_options -radius 250 \\
▷▷▷▷ -embedded -end \\
▷▷▷▷ -report -end \\
▷▷▷ -end \\
▷▷ -end \\
▷▷ -with GC -do \\
▷▷ -graph_classification_activity \\
▷▷▷ -list_graphs_at_level 5 5 \\
▷▷ -end \\
▷▷ -with GC -do \\
▷▷ -graph_classification_activity \\
▷▷▷ -draw_options \\
▷▷▷▷ -radius 300 -nodes_empty \\
▷▷▷▷ -line_width 1.5 \\
▷▷▷▷ -scale 0.1 \\
▷▷▷ -end \\
▷▷▷ -draw_graphs_at_level 5 \\
▷▷ -end \\
▷▷ -print_symbols
\end{verbatim}
```

397
After the classification, the graphs with 5 edges are shown. The file contains the following graph drawings:

![Graphs with 5 edges](image)

The next command classifies all tournaments on 4 vertices:

```bash
tournament_classify_4:
  $(ORBITER) -v 2 \\
  -define GC -graph_classification \\
  -n 4 -tournament \\
  -poset_classification_control \\
  -problem_label tournament_4 \\
  -depth 6 -draw_poset \\
  -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 \\
pdflatex tournament_4_level_6_reps.tex \\
open tournament_4_level_6_reps.pdf
```

There are four tournaments. The following graph drawings are produced:
The next command classifies all cubic graphs on 8 vertices:

```bash
graph_classify_8_r3:
> $(ORBITER) -v 3 \
>   -define GC -graph_classification \
>   -n 8 -regular 3 \
>   -poset_classification_control \
>   -problem_label graphs_v8_r3 \
>   -depth 12 -draw_poset \
>   -draw_options -radius 250 \
>   -line_width 0.2 -embedded \
>   -end \
> -end \
> -with GC -do \
> -graph_classification_activity \
>   -draw_options \
>   -radius 400 \
>   -line_width 2 -scale 0.10 \
>   -end \
> -draw_graphs_at_level 12 \
> -end \
> -print_symbols \
> #pdflatex graphs_v8_r3_poset_lvl_12.tex \
> #open graphs_v8_r3_poset_lvl_12.pdf
```

There are six cubic graphs. The following graph drawings are produced:
### Table 13.5: Clique Finding Options

<table>
<thead>
<tr>
<th>Key</th>
<th>Arguments</th>
<th>Meaning</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 13.3 can be used to find all cliques in a graph. Additional options for this command are shown in Table 13.5. 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 \
```

401
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 
  -define Gamma -graph -Paley 13 -end 
  -with G -do 
  -group_theoretic_activity 
  -poset_classification_control 
  -w 
  -problem_label Paley13_cliques 
  -clique_test Gamma 
  -depth 5 
  -end 
  -orbits_on_subsets 5 
  -report 
  -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 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 
  -with G -do 
  -group_theoretic_activity 
  -poset_classification_control 
  -w 
  -problem_label Paley13_cliques 
  -clique_test Gamma 
  -depth 5 
  -end 
  -orbits_on_subsets 5 
  -report 
  -end
```

For comparison, the command

```
Paley_13_cliques_all:
  $(ORBITER) -v 10 
  -define Gamma -graph -Paley 13 -end 
```

402
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:

```plaintext
HJ64_cliques5:
$ (ORBITER) -v 6 \
  -define Gamma -graph \
  -load Group_perm315_orbital_3.colored_graph \
  -end \
  -with Gamma -do \
  -graph_theoretic_activity \
  -find_cliques -target_size 5 -end \
  -end
```

It turns out that there are exactly 1008 5-cliques. Concerning the classification with respect to the automorphism group of the graph, we apply the following command:

```plaintext
HJ64_cliques5_classify:
$ (ORBITER) -v 6 \
  -define Gamma -graph \
  -load Group_perm315_orbital_3.colored_graph \
  -end \
  -define gens -vector -file halljanko315_gens.csv \
  -end \
  -define G -permutation_group \
  -bsgs halljanko315 "File\halljanko315" \
  315 1209600 "0,1,42,95" 6 gens -end \
  -with G -do \
  -group_theoretic_activity \
  -poset_classification_control \
  -w \
  -problem_label HJ64_cliques \
  -clique_test Gamma \
```

403
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 \n> -define Inc -vector -file 0_5_2_incidence_matrix.csv -end \n> -define Gamma -graph -collinearity_graph Inc -end \n> -with Gamma -do \n> -graph_theoretic_activity \n> -find_cliques -target_size 3 -end \n> -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 $ (HIRSCHFELD_SURFACE_Q4_SET_OF_POINTS) \n$ -end \n$ -define C -combinatorial_objects \n```
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Description</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:

\[
\text{HYPEROVAL}_{16}^{144} = "0, 1, 2, 3, 52, 67, 89, 106, 126, \ldots \frac{141}{159}, 176, 184, 199, 220, 235, 245, 262"
\]

\[
\text{HYPEROVAL}_{16}^{16320} = "0, 1, 2, 3, 52, 70, 83, 109, 127, \ldots \frac{139}{156}, 174, 186, 199, 217, 229, 256, 264"
\]

\[
\text{hyperoval}_{16}^{create}:
\]

\[
\text{\$(ORBITER)$ -v 2 \-define C -combinatorial_objects \-set_of_points $(HYPEROVAL_{16}^{16320}) \-set_of_points $(HYPEROVAL_{16}^{144}) \-end}
\]

In the next example, we read the points of an elliptic curve from file (see Section 4.2):

\[
\text{EC_read: elliptic_curve_b1_c3_q11.txt}
\]

\[
\text{\$(ORBITER)$ -v 4 \-define C -combinatorial_objects \-file_of_points elliptic_curve_b1_c3_q11.txt \-end}
\]

In the next example, we read a packing, using a previously defined table of spreads, stored in a csv file.

\[
\text{PG}_{3.5}^{31} \text{assume_31_read}:
\]

\[
\text{\$(ORBITER)$ -v 2 \-define C -combinatorial_objects \-file_of_packings_through_spread_table \-H31_packings.csv \-SPREAD_TABLES.5_REG/spread_25_spreads.csv \-5 \-end}
\]

The following command reads a file of large sets of designs:

407
LS_AG_2.3_read:
\[\texttt{\$(ORBITER) -v 2} \]
\[\texttt{\$\textbackslash define C \textbackslash -combinatorial\_objects} \]
\[\texttt{\$\textbackslash -file\_of\_designs} \]
\[\texttt{\$\textbackslash solutions.csv 9 84 3 12} \]
\[\texttt{\$\textbackslash -end} \]

The next command reads incidence geometries from a file containing the flags:

\texttt{geo_7.3_read:}
\[\texttt{\$(ORBITER) -v 10} \]
\[\texttt{\$\textbackslash -draw\_incidence\_structure\_description} \]
\[\texttt{\$\textbackslash -width 60 \textbackslash -with\_10 6 \textbackslash -end} \]
\[\texttt{\$\textbackslash -define C \textbackslash -combinatorial\_objects} \]
\[\texttt{\$\textbackslash -file\_of\_incidence\_geometries} \]
\[\texttt{\$\textbackslash 7.3.inc 7 7 21} \]
\[\texttt{\$\textbackslash -end} \]

The next command creates incidence geometries from a file containing row-ranks:

\texttt{Desargues_path_lex_least_read:}
\[\texttt{\$(DESARGUES\_PATH\_LEX\_LEAST) >Desargues\_path\_lex\_least.inc} \]
\[\texttt{\$(ORBITER) -v 10} \]
\[\texttt{\$\textbackslash -draw\_incidence\_structure\_description} \]
\[\texttt{\$\textbackslash -width 60 \textbackslash -with\_10 6 \textbackslash -end} \]
\[\texttt{\$\textbackslash -define C \textbackslash -combinatorial\_objects} \]
\[\texttt{\$\textbackslash -file\_of\_incidence\_geometries\_by\_row\_ranks} \]
\[\texttt{\$\textbackslash Desargues\_path\_lex\_least.inc 10 10 3} \]
\[\texttt{\$\textbackslash -end} \]
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
```

409
$$\begin{array}{l|llll}
\ell_0 & \ell_1 & \ell_2 & \ell_3 \\
P_0 & 1 & 1 & 0 & 0 \\
P_1 & 1 & 0 & 1 & 0 \\
P_2 & 1 & 0 & 0 & 1 \\
P_3 & 0 & 1 & 1 & 0 \\
P_4 & 0 & 1 & 0 & 1 \\
P_5 & 0 & 0 & 1 & 1 \\
\end{array}$$

Figure 14.2: The incidence matrix of the Pasch configuration

$$\begin{array}{l|llllllllllllllllll}
\ell_0 & \ell_1 & \ell_2 & \ell_3 \\
P_0 & 0 & 1 & 2 & 3 \\
P_1 & 4 & 5 & 6 & 7 \\
P_2 & 8 & 9 & 10 & 11 \\
P_3 & 12 & 13 & 14 & 15 \\
P_4 & 16 & 17 & 18 & 19 \\
P_5 & 20 & 21 & 22 & 23 \\
\end{array}$$

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:
$$\texttt{($\text{ORBITER}) -v 10 \ \}
\ \texttt{\ -define C -combinatorial_objects \}
\ \texttt{\ -file_of_incidence_geometries \}
\ \texttt{\ -pasch.inc 6 4 12 \}
\ \texttt{\ -end}
$$
```

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_pasch_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.
Table 15.1: Orbiter commands related to canonical labelings

<table>
<thead>
<tr>
<th>Modifier</th>
<th>Args</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-max_TDO_depth</td>
<td>$d$</td>
<td>Limit TDO depth to $d$ in the report.</td>
</tr>
<tr>
<td>-classification_prefix</td>
<td>prefix</td>
<td>Use the given prefix when writing files related to the classification.</td>
</tr>
<tr>
<td>-save_ago</td>
<td></td>
<td>Save the automorphism group orders to file.</td>
</tr>
<tr>
<td>-save_transversal</td>
<td></td>
<td>Save the indices of the elements chosen for the transversal.</td>
</tr>
</tbody>
</table>

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 refers to the file containing the Orbiter point ranks of points on the curve.

```bash
EC_canon: elliptic_curve_b1_c3_q11.txt
▷ $(ORBITER) -v 40 \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 ▷  -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 \n ▷  -end \n[end]
```

Orbiter shows that the curve has a collineation stabilizer of order 6, generated by

```
Orbiter shows that the curve has a collineation stabilizer of order 6, generated by
```

415
The following example computes the canonical form and the automorphism group of the Hirschfeld surface in PG(3,4). Using indexing of points in PG(3,4), we encode the surface as a set of points using Orbiter ranks. We use a makefile variable to provide these ranks as input for the canonical form procedure.

\[
\begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix},
\begin{bmatrix}
1 & 1 & 8 \\
5 & 9 & 5 \\
8 & 1 & 1
\end{bmatrix}.
\]

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 \
  -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 \n  -end \n  -end
```

Hirschfeld_q4_set_c:
In the next example, we compute the canonical form of the two hyperovals in PG(2, 16).

hyperoval_16.canonical_form:
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:**

```plaintext
$\text{ORBITER} -v 3 \$
$+define C -combinatorial_objects \$
$+set_of_points "0,1,2,5,6" \$
$+set_of_points "0,1,2,3,6" \$
$+set_of_points "0,1,2,3,4" \$
$+set_of_points "0,1,2,3,8" \$
$+set_of_points "0,1,2,3,5,6,9" \$
$+set_of_points "0,1,2,3,5,6,10" \$
$+set_of_points "0,1,2,3,5,6,4" \$
$+set_of_points "0,1,2,3,8,11,13" \$
$+set_of_points "3,6,9,7,10,12,8,11,13,14,4" \$
$+set_of_points "3,5,6,9,7,10,12,11,13,14,4" \$
$+set_of_points "0,1,2,3,5,6,9,7,10,12,4" \$
$+end \$
$+define F -finite_field -q 2 -end \$
$+define P -projective_space -n 3 -field F -v 0 -end \$
$+with C -do \$
$+combinatorial_object_activity \$
$+canonical_form_PG P \$
```

418
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:

```bash
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 \
  -dont_show_incidence_matrices \
  -export_group \
  -end \
  -end
```

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
```

419
We can store the generators in a makefile variable as follows:

SUZUKI_8_GENERATORS="
1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1, \\
1,0,0,0,0,6,0,0,0,0,2,0,0,0,0,3,0, \\
1,0,0,0,1,1,0,0,0,0,1,0,0,0,1,0,1, \\
1,0,0,0,3,6,2,2,5,0,2,0,3,0,6,3,2, \\
0,1,0,0,1,0,0,0,0,0,0,1,0,1,0,1,0,2"

We can now recover the Suzuki group using the command:

Suzuki_8:
  $(ORBITER) -v 6 \\
  -define F -finite_field -q 8 -end \\
  define gens -vector -field F \\
  -compact $(SUZUKI_8_GENERATORS) -end \\
  define G -linear_group -PGGL 4 8 \\
  -subgroup_by_generators "Sz8" "87360" 5 gens \\
  -end \
\begin{verbatim}
\$\$ \\
\$\$ -with G -do \ \\
\$\$ -group_theoretic_activity \ \\
\$\$  -report \ \\
\$\$  -end \\
\$\$ pdflatex PGGL_4_S_Subgroup_Sz8_87360_report.tex \\
\$\$ open PGGL_4_S_Subgroup_Sz8_87360_report.pdf
\end{verbatim}
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 if 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 PG(2, 2), which is a configuration 7,3.

```
geo_7_3_c:
  >>> $(ORBITER) -v 10 \
  >>> -draw.incidence_structure_description \ 
  >>> -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:

<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
Isomorphism type 0 / 1

Isomorphism type 0 / 1 is original object 0 and appears 1 times:
This isomorphism type appears 1 times, namely for the following 1 input objects:
{0}
incidence structure:
( 0, 1, 2, 7, 10, 11, 14, 19, 20, 22, 24, 26, 29, 32, 34, 37, 38, 41, 44, 46, 47 )
Column sets of the encoded object:
{ 0, 1, 2 }
{ 0, 3, 4 }
{ 0, 5, 6 }
{ 1, 3, 5 }
{ 1, 4, 6 }
{ 2, 3, 6 }
{ 2, 4, 5 }
Row sets of the encoded object:
{ 0, 1, 2 } = 0
{ 0, 3, 4 } = 9
{ 0, 5, 6 } = 14
{ 1, 3, 5 } = 20
{ 1, 4, 6 } = 23
{ 2, 3, 6 } = 27
{ 2, 4, 5 } = 28
Generators for the automorphism group:
The stabilizer of order 168 is generated by:
$g_1 = (3, 5)(4, 6)(8, 9)(12, 13)$ of order 2 and with 6 fixed points.
$g_2 = (3, 4)(5, 6)(10, 11)(12, 13)$ of order 2 and with 6 fixed points.
$g_3 = (1, 2)(5, 6)(10, 12)(11, 13)$ of order 2 and with 6 fixed points.
$g_4 = (1, 3)(2, 4)(7, 8)(11, 12)$ of order 2 and with 6 fixed points.
$g_5 = (0, 1)(4, 5)(8, 10)(9, 11)$ of order 2 and with 6 fixed points.

Canonical labeling:
canonical row = 6
canonical orbit number = 0
Flags : ( 0, 1, 2, 7, 10, 11, 14, 19, 20, 22, 24, 26, 30, 31, 34, 36, 39, 41, 44, 46, 47 )

<table>
<thead>
<tr>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>*0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Flag orbits:
orbit length : number of orbits of that length:

21 1

Anti-Flag orbits:
orbit length : number of orbits of that length:

28 1

The following command computes the canonical form and a report of the affine plane
AG(2,3), which is a configuration 94123.

```
AG_2.3.c: AG_2.3.inc
  $(ORBITER) -v 2 \n  -define C -combinatorial_objects \n  -file_of_incidence_geometries \n  AG_2.3.inc 9 12 36 \n  -end \n  -with C -do \n  -combinatorial_object_activity \n  -canonical_form \n  -classification_prefix AG_2.3 \n  -label AG_2.3 \n```
Figure 15.2: The affine plane AG(2, 3) is a configuration $9_412_3$

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:

\[
\begin{array}{c|c|c|c|c}
\rightarrow & | & | & | \\
9_0 & 12_1 & 4 & \\
\downarrow & | & | & |
\end{array}
\begin{array}{c|c|c|c|c}
\rightarrow & | & | & | \\
9_0 & 12_1 & 3 & \\
\end{array}
\]
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 \$
  $ $ -width 60 -with_10 6 -end \$
  $ $ -define Test_lines -set -loop 4 11 1 -end \$
  $ $ -define C -combinatorial_objects \$
  $ $ -file_of_incidence_geometries 10_3.inc 10 10 30 \$
  $ $ -end \$
  $ $ -with C -do \$
  $ $ -combinatorial_object_activity \$

428
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, 3^2, 4^2, 6, 10, 12, 24, 120

Isomorphism type 0 / 10

Isomorphism type 0 / 10 is original object 9 and appears 1 times:
This isomorphism type appears 1 times, namely for the following 1 input objects:
{9}
incidence structure:
( 0, 1, 2, 10, 13, 14, 20, 25, 26, 31, 33, 35, 41, 44, 47, 52, 53, 58, 62, 66, 69, 74, 78, 79, 85, 87, 89, 96, 97, 98 )

Generators for the automorphism group:
The stabilizer of order 3 is generated by:
\( g_1 = (0, 1, 3)(2, 5, 4)(6, 7, 8)(10, 13, 11)(12, 14, 15)(16, 18, 17) \) of order 3 and with 2 fixed points.
Decomposition by automorphism group:

10131112141516181719

Canonical labeling:
canonical row = 5
canonical orbit number = 1
Flags : 0,1,2,16,17,18,25,27,29,34,38,39,40,43,45,51,56,62,63,64,70,74,77,82,86,89,91,95,98,

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:

\[
\begin{array}{ccccccccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\
\end{array}
\]

Canonical labeling:
canonical row = 0
canonical orbit number = 0
Flags : 0, 1, 2, 15, 18, 19, 24, 26, 29, 33, 37, 39, 40, 43, 44, 50, 55, 56, 61, 67, 68, 72, 75, 77, 82, 84, 88, 91, 93, 96,

\[
\begin{array}{ccccccccccccccc}
5 & 8 & 1 & 4 & *6 & 2 & 9 & 7 & *0 & 14 & 12 & 18 & 10 & 13 & 17 & 11 & 16 & 19 & 15 \\
\end{array}
\]

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"
\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  324  327  342  354  357 \
367  379  381  392  398  400  417  428  429  442  443  450  466  471 \
479  492  497  502  517  519  521  542  548  551  571  574  575  48 \
\n0  1  2  24  27  28  48  53  54  73  79  80  97  105  106  122  131 \
132  146  157  158  171  175  183  195  203  208  220  225  233  244 \
258  259  269  272  281  293  301  308  318  324  327  342  354  357 \
```

432
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\n\n0 1 2 24 27 28 48 53 54 73 79 80 97 105 106 122 131 \
132 146 157 158 171 175 179 195 201 207 220 226 232 244 \
257 258 269 274 277 293 300 307 318 323 329 342 352 356 \
367 374 381 392 397 406 416 423 431 441 450 454 468 476 \
477 494 499 503 519 521 525 544 547 550 570 572 575 

\n-1 3"

TFC_24_3_c:
  echo $(FILE_24_3_TFC_INC) >24_3_TFC.inc
  $(ORBITER) -v 6 \
    -define C -combinatorial_objects \
    -file_of_incidence_geometries 24_3_TFC.inc 24 24 72 \
    -end \
    -with C -do \
    -combinatorial_object_activity \
    -canonical_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 \
  -end
  pdflatex 24_3_TFC_classification.tex
  open 24_3_TFC_classification.pdf
  $(ORBITER) -v 2 -draw_matrix \
    -input_csv_file 24_3_TFC_object2_TDA_flag_orbits.csv \
    -secondary_input_csv_file 24_3_TFC_object2_TDA.csv \
    -box_width 40 -bit_depth 24 \
    -end
  open 24_3_TFC_object2_TDA_flag_orbits_draw.bmp

The command also computes the tactical decomposition induced by the automorphism group. In addition, the command also computes the orbits on flags and on anti-flags. The third of
the three geometries is flag transitive. A bitmap drawing is produced, shown 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 PG(2,3):

design\_PG\_2\_3\_canonical:
> $(\text{ORBITER}) -v 3 \$
> ▶ -define D -design -q 3 -family PG\_2\_q -end \$
> ▶ -with D -do \$
> ▶ ▶ -design\_activity \$
> ▶ ▶ ▶ -export\_inc \$
> ▶ ▶ -end \$
> ▶ -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 \$
> ▶ ▶ -end \$
> ▶ -end \$
> 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
> 

435
The command

```
wreath_product_designs_n4_k2_c: wreath_product_designs_n4_k2_inc.txt
  ▶ $(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 \n  ▶ ▶ ▶ ▶ -end \n  ▶ ▶ -end
```

computes the automorphism group of the design on 8 points created in Section 11.5. The group is Sym(4) \( \wr \) Sym(2). The command

```
wreath_product_designs_n8_k6_c: wreath_product_designs_n8_k6_inc.txt
  ▶ $(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  ▶ ▶ -with C -do \n```

436
computes the automorphism group of the design on 16 points created in Section 11.5. The group is \(\text{Sym}(8) \wr \text{Sym}(2)\).

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 `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).

```
$ORBITER -v 2
  -draw_incidence_structure_description 
  -width 20 -width_10 2 -end 
  -define C -combinatorial_objects 
  -file_of_designs 
  solutions.csv 9 84 3 12 
  -end 
  -with C -do 
  -combinatorial_object_activity 
  -canonical_form 
  -save_ag 
  -save_transversal 
  -classification_prefix LS_AG_2_3 
  -label LS_AG_2_3 
  -max_TDO_depth 10 
  -end 
  -report 
  -prefix LS_AG_2_3 
  -export_flag_orbits 
```

437
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:

```latex
$(\text{ORBITER})$ -v 20 \\
$\implies$ -define F -finite_field -q 2 -end \\
$\implies$ -define genma -vector -field F -format 2 \\
$\implies$ -dense $(\text{CODEN3K2Q2GENMA})$ \\
$\implies$ -end \\
$\implies$ -define P -projective_space -n 1 -field F -v 0 -end \\
$\implies$ -with P -do \\
$\implies$ -projective_space_activity \\
$\implies$ -canonical_form_of_code \\
$\implies$ -save Ago -label "3_2" \\
$\implies$ -classification_prefix "3_2" \\
$\implies$ -end \\
$\implies$ -end
```

The code has a semilinear automorphism group of order 6. The following report is written:

The code has a semilinear automorphism group of order 6. The following report is written:

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>

439
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:

<table>
<thead>
<tr>
<th>2_1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>4_0</td>
<td>3</td>
</tr>
</tbody>
</table>

Canonical labeling:
- canonical row = 3
- canonical orbit number = 1
Flats : ( 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:
▷ $(ORBTER) -v 2 \n▷ -define F -finite_field -q 2 -end \n▷ -define genma -vector -field F -format 4 \n▷ -compact $(CODE_RM_3_1_GENMA) \n▷ -end \n```

441
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:

```
$ORBITER -v 2 -input_csv_file RM_3_1_object0_INP_flag_orbits.csv
$ORBITER -v 2 -input_csv_file RM_3_1_object0_TDA_flag_orbits.csv
```

open RM_3_1_object0_INP_flag_orbits_draw.bmp
The drawing in Figure 15.4 is created.

The command

```
open RM_3_1_object0_TDA_flag_orbits_draw.bmp
```

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,

\[
\begin{array}{c|cc}
& 2850_1 & 1_2 \\
401_0 & 57 & 1
\end{array}
\]

The command

\texttt{GV\_n15\_k6\_d5\_group:}
\texttt{\\
\& \& \& \$\texttt{\$(ORBITER) -v 20 \backslash
\& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \& \}&
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) -v 2 \
▶ ▶ -define G -graph -Hamming 7 2 \
▶ ▶ ▶ -subset "_Hamming_code" "\\_with\\_Hamming\\_code" \ 
▶ ▶ ▶ $(HAMMING_CODE_CODEWORDS) -end \ 
▶ ▶ -with G -do \ 
▶ ▶ -graph_theoretic_activity -export_csv -end \ 
▶ ▶ -with G -do \ 
▶ ▶ -graph_theoretic_activity -export_graphviz -end \ 
▶ ▶ -with G -do \ 
▶ ▶ -graph_theoretic_activity -save -end \ 
▶ ▶ -with G -do \ 
▶ ▶ -graph_theoretic_activity -automorphism_group -end
▶ pdflatex Hamming_7_2_Hamming_code_report.tex
▶ open Hamming_7_2_Hamming_code_report.pdf
```

computes the set stabilizer of the Hamming code inside the automorphism group of the Hamming graph. The group has order \(2688 = 16 \cdot 168\).
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 \\
  -define Gamma -graph -cycle 13 -end \\
  -with Gamma -do \\
  -graph_theoretic_activity -automorphism_group \\
  -end \\
```

The output is two files: The first one, `Cycle_13_group.makefile`, is a makefile containing an Orbiter command to create the automorphism group: The second file is `Cycle_13_gens.csv`, which contains the permutation representation of the group, and which is needed for the makefile.

The next command computes the automorphism group of the chain graph with respect to the partition (2, 3, 2).

```
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 \\
  -with Gamma -do \\
  -graph_theoretic_activity -automorphism_group \\
  -end \\
  pdflatex chain_graph_report.tex \\
  open chain_graph_report.pdf
```

The following report is written:

```
The automorphism group of chain_graph has order 1152 and is generated by:
Strong generators for a group of order 1152:

(12, 13),
(3, 4),
(2, 3),
```

447
Junttila and Kaski maintain a collection of graphs that can be used as test cases. The graphs are stored in Dimacs format and can be read in through the Orbiter -load_dimacs command. For instance, the following command computes the automorphism group of the Levi graph of the desarguesian projective plane of order 16:

```bash
JK_graph_pp16_1:
    $(ORBITER) -v 2 \n    -define Gamma -graph -load_dimacs \n    ../JUNTTILA_KASKI/benchmarks/pp/pp16-1 \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 command shows a group of order 34217164800. As a measurement of the complexity, the number of backtrack nodes in Nauty is recorded:

- `nb_backtrack1 = 6`
- `nb_backtrack2 = 134`
- `nb_backtrack3 = 134`
- `nb_backtrack4 = 1`

Here, `nb_backtrack1` is the number of calls to `firstpathnode`, `nb_backtrack2` is the number of calls to `othernode`, `nb_backtrack3` is the number of calls to `processnode`, and `nb_backtrack4` is the number of calls to `try_edge`.
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
```

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 $\text{PG}(2, q)$ 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.makefilests-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)

449
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:

HJ_aut:

```
$ (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:

```
$ (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 \
  -with G -do \
  -group_theoretic_activity \
  -poset_classification_control \
  -W \
  -problem_label HJ_orbits \
  -depth 2 \
  -end \
  -orbits_on_subsets 2 \
  -report \
  -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:

```
$ (ORBITER) -v 2 \
```

450
-define gens -vector -file halljanko315_gens.csv -end
-define G -permutation_group
-bsgs halljanko315 "File\_halljanko315" 315 1209600 "0,1,2" 6 gens
-end
-define Gamma -graph
-orbital_graph G 3
-end
-with Gamma -do
-graph_theoretic_activity
-properties
-end
-with Gamma -do
-graph_theoretic_activity
-save
-end

The graph is regular of degree 64.

The next command computes the automorphism group of the collinearity graph of the $Q(4, 2)$ quadric.

```
PGO_5_2_graph_group: 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
-automorphism_group
-end
-with Gamma -do
-graph_theoretic_activity
-eigenvalues
-end
```

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 $\mathbb{F}_{17}$

\[ F_{17,\text{edge}}: \]
\[ \text{define F -finite_field -q 17 -end} \]
\[ \text{with F -do -finite_field_activity} \]
\[ \text{-cheat_sheet_GF -end} \]
\[ \text{pdflatex GF} \]
\[ \text{open GF_17.pdf} \]

Next, we compute the canonical form of the Edge quartic. This command also computes generators for the automorphism group of the curve.

\[ \text{Edge_curve_17_nauty:} \]
\[ \text{define C -combinatorial_objects} \]
\[ \text{-file_of_points Edge_q17.txt} \]
\[ \text{-end} \]
\[ \text{-define F -finite_field -q 17 -end} \]
\[ \text{-define P -projective_space -n 2 -field F -v 0 -end} \]
\[ \text{-with C -do} \]
\[ \text{-combinatorial_object_activity} \]
\[ \text{-canonical_form_PG P} \]
\[ \text{-classification_prefix Edge_curve_q17} \]
\[ \text{-label Edge_curve_q17} \]
\[ \text{-save_ago} \]
\[ \text{-save_transversal} \]
\[ \text{-max_TDO_depth 10} \]
\[ \text{-end} \]
\[ \text{-report} \]
\[ \text{-prefix Edge_curve_q17} \]
\[ \text{-export_flag_orbits} \]
\[ \text{-show_TDO} \]
\[ \text{-show_TDA} \]
\[ \text{-dont_show_incidence_matrices} \]
\[ \text{-export_group} \]
\[ \text{-end} \]
\[ \text{pdflatex Edge_curve_q17_classification.tex} \]
\[ \text{open Edge_curve_q17_classification.pdf} \]
\[ \text{$(ORBITER) \ -v \ 2 \ -draw_matrix} \]
\[ \text{-input_csv_file Edge_curve_q17_object0_TDA_flag_orbits.csv} \]

452
Using the generators that have just been computed, we can recreate the group of the quartic curve:

```
Edge_curve_17_group:
  $(ORBITER) -v 3 \n  -define G -linear_group -PGL 3 17 \n  -subgroup_by_generators "Stab_Edge" "24" 3 \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  -with G -do \n  -group_theoretic_activities \n  -print_elements_tex \n  -group_table \n  -report \n  -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 [64],
2. Metapost [34],
3. Bitmap files (.bmp) [67],
4. Povray, see Section 16.2.

Bitmaps can be created using the \texttt{-draw_matrix} command. The input is an integer-valued matrix in csv format. The matrix entries are translated into colors. The possible commands after \texttt{-draw_matrix} are shown in Table 16.1. Suppose we want to create a graphical representation of the addition table of the finite field $\mathbb{F}_7$. The following command sequence first creates the addition and multiplication tables of the field, and then produces a bitmap graphic for the addition table:

\begin{table}[h]
\centering
\begin{tabular}{|c|c|l|}
\hline
\textbf{Command} & \textbf{Arguments} & \textbf{Description} \\
\hline
\texttt{-input_csv_file} & csv-file & Specify the input csv file \\
\hline
\texttt{-partition} & $w \ R \ C$ & Specify a partition $R$ of rows and $C$ of columns. Use separating lines of with $w$. \\
\hline
\texttt{-box_width} & $w$ & Use $w$ pixels per matrix entry. \\
\hline
\texttt{-bit_depth} & $d$ & Use color bit depth of $d$ bits ($d = 8$ or $d = 24$). \\
\hline
\texttt{-invert_colors} & & Use an inverted color scheme. \\
\hline
\end{tabular}
\caption{Commands to Create Bitmap Graphics}
\end{table}
F_7.tables: 
▷ $(ORBITER) -v 3 \ 
▷ ▷ -define F -finite_field -q 7 -end \ 
▷ ▷ -with F -do -finite_field_activity \ 
▷ ▷ ▷ -cheat_sheet_GF \ 
▷ ▷ -end 
▷ $(ORBITER) -v 2 \ 
▷ ▷ -draw_matrix \ 
▷ ▷ ▷ -input_csv_file GF_q7_addition_table.csv \ 
▷ ▷ ▷ -box_width 40 \ 
▷ ▷ ▷ -bit_depth 24 \ 
▷ ▷ ▷ -partition 3 7 7 \ 
▷ ▷ -end 
▷ open GF_q7_addition_table_draw.bmp

The finite field activity -cheat_sheet_GF creates the file

GF_q7_addition_table.csv

which is used as the input for the second command. The file content is:

<table>
<thead>
<tr>
<th>Row</th>
<th>C0</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>C5</th>
<th>C6</th>
</tr>
</thead>
<tbody>
<tr>
<td>0,0</td>
<td>0,1</td>
<td>2,3</td>
<td>4,5</td>
<td>6,0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1,1</td>
<td>2,3</td>
<td>4,5</td>
<td>6,0</td>
<td>1,2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2,2</td>
<td>3,4</td>
<td>5,6</td>
<td>0,1</td>
<td>2,3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3,3</td>
<td>4,5</td>
<td>6,0</td>
<td>1,2</td>
<td>3,4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4,4</td>
<td>5,6</td>
<td>0,1</td>
<td>2,3</td>
<td>4,5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5,5</td>
<td>6,0</td>
<td>1,2</td>
<td>3,4</td>
<td>5,6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6,6</td>
<td>0,1</td>
<td>2,3</td>
<td>4,5</td>
<td>6,0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>END</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The second command creates the diagram in Figure 16.1. The -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 PG(2, q) admit a cyclic automorphism group known as the Singer cycle. The command

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 \ 

456
produces a cyclically ordered incidence matrix of the plane 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 \texttt{-draw_poset} option in the poset classification control command package, see Table 6.2. The posets are stored in a file with extension \texttt{.layered_graph}. These files can be drawn using the \texttt{-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 PGL(4, 2) in its action on the wedge product. The command

\begin{verbatim}
$ORBITER) -v 4 \n$list_arguments \n(define R -vector -repeat 1 21 -end \n(define C -vector -repeat 1 21 -end \n(draw_matrix \n(input_csv_file PG_2_4_singer_incma_cyclic.csv \n(box_width 40 -bit_depth 24 \n(partition 3 R C \n(end \nopen PG_2_4_singer_incma_cyclic_draw.bmp
\end{verbatim}

Figure 16.1: Addition table of $\mathbb{F}_7$
<table>
<thead>
<tr>
<th>Modifier</th>
<th>Args</th>
<th>Meaning</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 $\text{PG}(2, 4)$

<table>
<thead>
<tr>
<th>Modifier</th>
<th>Args</th>
<th>Meaning</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)
The command

```
schreier_tree_graphical_output:
  $\$(ORBITER) -v 4 \\n  \>-define G -linear_group -PGL 4 2 \\n  \>-yout 500000 \\n  \>-radius 15 -nodes_empty \\n  \>-line_width 0.5 -y_stretch 0.25 \\n  \>-end \\n  \>-define G -linear_group -PGL 4 2 -end \n  \>-with G -do \n  \>-group_theoretic_activity \n  \>-report \n  \>-end
```

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  \>-define G -linear_group -PGL 4 2 \\n  \>-yout 500000 \\n  \>-radius 15 -nodes_empty \\n  \>-line_width 0.5 -y_stretch 0.25 \\n  \>-end \\n  \>-define G -linear_group -PGL 4 2 -end \n  \>-with G -do \n  \>-group_theoretic_activity \n```
Figure 16.4: A Schreier tree in the action on polynomials

\begin{verbatim}
\texttt{-orbits_on_polynomials 3 \}
\texttt{-orbits_on_polynomials_draw_tree 6 \}
\texttt{-end}
\texttt{pdflatex poly_orbits_d3_n3_q2.tex}
\texttt{open poly_orbits_d3_n3_q2.pdf}
\end{verbatim}

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 420 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 [57]. 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:
▷ $(ORBITER) -v 2 -povray
  ▷ -round 0 -nb_frames_default 30
  ▷ -output_mask cube_%d_%03d.pov
  ▷ -video_options -W 1024 -H 768
  ▷ -global_picture_scale 0.5
  ▷ -default_angle 75
  ▷ -clipping_radius 2.7
  ▷ -scene_objects
  ▷   -obj_file cube_centered.obj
  ▷   -edge "0, 1"
  ▷   -edge "0, 2"
  ▷   -edge "0, 4"
  ▷   -edge "1, 3"
  ▷   -edge "1, 5"
  ▷   -edge "2, 3"
  ▷   -edge "2, 6"
  ▷   -edge "3, 7"
  ▷   -edge "4, 5"
<table>
<thead>
<tr>
<th>Option</th>
<th>Arguments</th>
<th>Meaning</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</td>
<td>s</td>
<td>Set font size to $s$.</td>
</tr>
<tr>
<td>-stroke_width</td>
<td>s</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</td>
<td>w</td>
<td>Set output dimension to $w$ pixels wide.</td>
</tr>
<tr>
<td>-H</td>
<td>h</td>
<td>Set output dimension to $h$ pixels height.</td>
</tr>
<tr>
<td>-nb_frames</td>
<td>n</td>
<td>Set number of frames to $n$. One revolution around the axis is split into $n$ frames.</td>
</tr>
<tr>
<td>-zoom</td>
<td>r $a_s$ $a_t$ $c_s$ $c_t$</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</td>
<td>r $F$ $T$ $C$</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</td>
<td>r $F$ $T$ $C$</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</td>
<td>r</td>
<td>Remove bottom plane in round $r$.</td>
</tr>
<tr>
<td>-camera</td>
<td>r $S$ $C$ $L$</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>Option</th>
<th>Arguments</th>
<th>Meaning</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>$r\ a\ \text{text}$</td>
<td>In round $r$, produce running text $\text{text}$ with sustain value $a$.</td>
</tr>
<tr>
<td>-label</td>
<td>$r\ s\ a\ g\ \text{text}$</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>$r\ s\ a\ \text{praeamble}\ g\ \text{text}\ l\ \text{fname}$</td>
<td>In round $r$, produce running latex text $\text{text}$ with start value $s$, sustain $s$ and gravity $g$. Put $\text{praeamble}$ 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>$r\ d\ \text{fname}\ \text{options}$</td>
<td>In round $r$, place picture $\text{fname}$ scaled by $d$ using options.</td>
</tr>
<tr>
<td>-picture</td>
<td>$r\ d\ \text{fname}\ \text{options}$</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 $Ax y z + B(x^2 + y^2 + z^2) + C = 0$</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>i₁ i₂</td>
<td>Create a point from the intersection of two lines $i₁$ and $i₂$</td>
</tr>
<tr>
<td>-edge</td>
<td>i₁ i₂</td>
<td>Create an edge (line segment) between points $i₁$ and $i₂$</td>
</tr>
<tr>
<td>-text</td>
<td>i₁ s</td>
<td>Create a label s located at the point $i₁$</td>
</tr>
<tr>
<td>-triangular_face_given_by_three_lines</td>
<td>i₁ i₂ i₃</td>
<td>Create a triangular face given by three lines $i₁, i₂, i₃$</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>i₁ i₂ i₃</td>
<td>Create a quadric through three skew lines</td>
</tr>
<tr>
<td>-plane_defined_by_three_points</td>
<td>i₁ i₂ i₃</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>$i r$ 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>$i r$ 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>$i d$ 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>$i$ prop</td>
<td>For each element in plane group $i$, create a plane with given Povray properties.</td>
</tr>
<tr>
<td>-lines</td>
<td>$i r$ 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>$i$ 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>$i$ 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>$i$ 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>$i$ 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>$i d s$ 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:

![First frame of the cube rotation](image)

The coordinates of the cube are stored in an object file `cube_centered.obj`. The content of this file is:

```
v -1 -1 -1  
v 1 -1 -1  
v -1 1 -1  
v 1 1 -1  
v -1 -1 1  
v 1 -1 1  
v -1 1 1
```
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.

\[ \text{MONKEY_SADDLE_CUBIC} = "1,0,0,0,-3,0,0,0,0,0,0,0,0,0,0,0,0,-1,0" \]

monkey:
\[
\begin{verbatim}
  $(ORBITER) -v 2 -povray \\
  -cubic_lex $(MONKEY_SADDLE_CUBIC) \\
  -plane_by_dual_coordinates "0,0,1,0" \\
  -group_of_things "0" \\
  -group_of_things "0" \\
  -cubics 0 $(COLOR_GOLD) \\
  -planes 1 $(COLOR_BLUE) \\
  -scene_objects_end \\
  -povray_end \\
  -rm -rf POV \\
  mkdir POV \\
  mv monkey_0*.*.pov POV \\
  mv makefile_animation POV
\end{verbatim}
\]
Here is one of the frames that are created:

![Eckardt Surface](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:**

```plaintext
$\$(ORBITER) -v 2 -povray \\
  -round 0 -nb_frames_default 30 \ 
  -output_mask Eckardt_%d_03d.pov \ 
  -video_options -W 1024 -H 768 \ 
  -global_picture_scale 0.9 \ 
  -default_angle 75 \ 
  -clipping_radius 2.4 \ 
  -camera 0 "1,1,1" "-3,1,3" "0.12,0.12,0.12" \ 
  -end \ 
  -scene_objects \ 
  -Hilbert_Cohn_Vossen_surface \ 
  -group_of_things "0" \ 
  -cubics 0 $(SURFACE_COLOR) \ 
  -group_of_things_as_interval 0 6 \ 
  -group_of_things_as_interval 6 6 \ 
  -group_of_things_as_interval_with_exceptions 12 15 \ 
  "14,19,23" \ 
  -lines 1 0.02 $(COLOR_RED_SHINY) \ 
  -lines 2 0.02 $(COLOR_BLUE_SHINY) \ 
  -lines 3 0.02 $(COLOR_YELLOW_SHINY) \ 
  -label 0 "a1" \ 
```

470
Figure 16.5 shows the final product.

The Endrass octic [26] is the algebraic surface given by the equation

\[ x^8 := 64 ( -w^2 + x^2 ) ( -w^2 + y^2 ) ((x+y)^2 - 2w^2) ( (x-y)^2 - 2w^2 ) - \left( -4 (1 + \sqrt{2}) (x^2 + y^2)^2 + (8 (2 + \sqrt{2}) z^2 + 2 (2 + 7 \sqrt{2}) w^2) (x^2 + y^2) - 16 z^4 + 8 (1 - 2 \sqrt{2}) z^2 w^2 - (1 + 12 \sqrt{2}) w^4 \right)^2 \]

The following Orbiter command creates a povray graphics of the octic, shown in Figure 16.6:
Figure 16.5: The Eckardt surface
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,-687.529,0,0,1582.59,0,1186.94,0,0,0,0,-1055.06,0,\ 
-1582.59,0,-593.47,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-309.019,\ 
0,0,1582.59,0,1186.94,0,0,0,0,-2110.12,0,-3165.17,0,-1186.94,0,0,0,0,0,\ 
0.874.039,0,1560.63,0,1677.92,0,343.362,0,0,0,0,0,0,0,0,0,0,0,0,0,\ 
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-93.2548,0,0,527.529,0,395.647,\ 
0,0,0,0,-1055.06,0,-1582.59,0,-593.47,0,0,0,0,0,0,0,0,0,0,0,0,0,0,\ 
1677.92,0,343.362,0,0,0,0,0,0,0,-256.0,-468.077,0,-789.019,0,\ 
-525.726,0,0.941125"

derrass8:
▷ $(ORBITER) -v 2 -povray \n▷ -round 0 -nb_frames_default 30 \n▷ -output_mask endrass.octic.%d.%03d.pov \n▷ -video_options -W 1024 -H 768 \n▷ -global_picture_scale 0.75 \n▷ -default_angle 75 \n▷ -clipping_radius 3.7 \n▷ -no_bottom_plane \n▷ -camera 0 "1,1,1" "6,6,3" "0,0,0" \n▷ -rotate_about_111 \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▷▷ -group_of_things_as_interval 3 39 \n▷▷ -lines 0 0.15 $(COLOR.RED_SHINY) \n▷▷ -lines 1 0.15 $(COLOR.GREEN_SHINY) \n▷▷ -lines 2 0.15 $(COLOR.BLUE_SHINY) \n▷▷ -lines 3 0.05 $(COLOR.BLACK_SHINY) \n▷▷ -octic_lex_165 $(ENDRASS_OCTIC_LEX_165) \n▷▷ -plane_by_dual_coordinates "0,0,1,0" \n▷▷ -group_of_things "0" \n▷▷ -group_of_things "0" \n▷▷ -octics 4 $(SURFACE_COLOR_SEETHROUGH) \n▷▷ -planes 5 " texture{ pigment{ color Blue transmit 0.5 } \nfinish { diffuse 0.9 phong 1} }" \n▷ -scene_objects_end \n▷ -povray_end \n▷ -rm -rf POV \n▷ mkdir POV

473
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_%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_%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:

\[ \sin(x) \rightarrow \text{push } x \text{ sin}, \]
\[ a + b \rightarrow \text{push } a \text{ push } b \text{ add}, \]
\[ a \cdot b \rightarrow \text{push } a \text{ push } b \text{ mult}. \]

The coordinate functions are enclosed between -code and -code_end commands. Each coordinate function is described in RPN and terminated using a return keyword. By the time the return keyword is reached, the RPN expression must have exactly one value on the stack which is considered the value of the expression. Constants are declared between the -const and -const_end keywords. Likewise, variables are declared between the -var and -var_end keywords. Picking \( a = 3, b = 2, c = \pi/2 \) and \( r = 7 \), the function is computed using

```
\text{lissajous:}
\text{\> \$ (ORBITER) \text{-v 2 \"\text{-smooth_curve "lissajous" 0.07 2000 15 0 18.85 \}-code \text{-const a 3 b 2 c 1.57 r 7 -const_end \}-code \text{-var t -var_end \}-code \text{-code_end \}}}}
```

The sequence

\[ \text{push } t \text{ push } a \text{ push } c \text{ add sin push } r \text{ mult return \}} \]
\[ \text{push } t \text{ push } b \text{ mult sin push } r \text{ mult return \}} \]
\[ \text{-code_end \}} \]

is \( r \sin (at + c) \) expressed in RPN. The constants are defined in the line

\[-\text{const a 3 b 2 c 1.57 r 7 -const_end} \]
The input variable is defined using the line

```
-var t -var_end
```

The sequence

```
-smooth_curve "lissajous" 0.07 2000 15 0 18.85
```

defines the name of the output file, the fact that two consecutive points are never further
than \( \epsilon = 0.07 \) away, the fact that points that are 15 or more away from the origin should
be ignored, and the fact that the variable \( t \) loops over the range \([0, 18.85]\) with a default of
2000 steps. The evaluator automatically reduces the step-size if consecutive image points
are more than \( \epsilon \) apart. The code to produce the plot is

```
lissajous_plot:
  ▶ $(ORBITER) -v 2 -povray 
  ▶ ▶ -round 0 -nb_frames_default 1 
  ▶ ▶ -output_mask lissajous.%d_%03d.pov 
  ▶ ▶ -video_options -W 1024 -H 768 
  ▶ ▶ -global_picture_scale 0.40 
  ▶ ▶ -default_angle 45 
  ▶ ▶ -clipping_radius 5 
  ▶ ▶ -omit_bottom_plane 
  ▶ ▶ -camera 0 "0,-1,0" "0,0,12" "0,0,0" 
  ▶ ▶ -rotate_about_z_axis 
  ▶ ▶ -end 
  ▶ ▶ -scene_objects 
  ▶ ▶ ▶ -line_through_two_points_recentered_from_csv_file 
  ▶ ▶ ▶ coordinate_grid.csv 
  ▶ ▶ ▶ -group_of_things "0" 
  ▶ ▶ ▶ -group_of_things "1" 
  ▶ ▶ ▶ -group_of_things "2" 
  ▶ ▶ ▶ -lines 0 0.09 "texture{ pigment{ color Yellow } }" 
  ▶ ▶ ▶ -lines 1 0.09 "texture{ pigment{ color Yellow } }" 
  ▶ ▶ ▶ -lines 2 0.09 "texture{ pigment{ color Yellow } }" 
  ▶ ▶ ▶ -group_of_things_as_interval 3 39 
  ▶ ▶ ▶ -lines 3 0.02 "texture{ pigment{ color Black } }" 
  ▶ ▶ ▶ -point_list_from_csv_file 
  ▶ ▶ ▶ function_lissajous_N2000_points.csv 
  ▶ ▶ ▶ -group_of_things_as_interval 0 6524 
  ▶ ▶ ▶ -spheres 4 0.1 "texture{ pigment{ color Red } }" 
  ▶ ▶ ▶ finish { diffuse 0.9 phong 1}"
  ▶ ▶ -plane_by_dual_coordinates "0,0,1,0" 
  ▶ ▶ -group_of_things "0" 
  ▶ ▶ -planes 5 "texture{ pigment{ color Blue*0.5 
  transmit 0.5 } }"
  ▶ ▶ -scene_objects_end 
```

477
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 \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 \n```

The code to produce the 3D plot is

```
lissajous_3d_plot:
```

```
$(ORBITER) -v 2 -povray \n  -round 0 -nb_frames_default 30 \n```

478
The 3D curve is shown in Figure 16.8.
Figure 16.8: Lissajous Spacecurve
Chapter 17

Miscellaneous

17.1 Miscellaneous

Table 17.1 list miscellaneous Orbiter commands. The command -csv_file_select_rows can be used to select rows from a csv file. The command -csv_file_select_cols can be used to select columns from a csv file. The command -csv_file_select_rows_and_cols selects rows and columns. Here is an example. We create the multiplication table of the finite field \( \mathbb{F}_7 \), ordered according to the powers of a primitive element:

\[
\alpha^0, \alpha^1, \alpha^2, \alpha^3, \alpha^4, \alpha^5.
\]

After that, we pull the rows and columns corresponding to even powers \( \alpha^0, \alpha^2, \alpha^4 \).

```
$ (ORBITER) -v 3 \\
 $ $(ORBITER) -v 4 -csv_file_select_rows_and_cols \\
 GF.q7_multiplication_table_reordered.csv \\
 "$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 3.3 in Section 3.2). Here is the file GF.q7_multiplication_table_reordered.csv

<table>
<thead>
<tr>
<th>Row, C0, C1, C2, C3, C4, C5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0, 1, 3, 2, 6, 4, 5</td>
</tr>
<tr>
<td>1, 3, 2, 6, 4, 5, 1</td>
</tr>
<tr>
<td>2, 2, 6, 4, 5, 1, 3</td>
</tr>
<tr>
<td>3, 6, 4, 5, 1, 3, 2</td>
</tr>
<tr>
<td>4, 4, 5, 1, 3, 2, 6</td>
</tr>
</tbody>
</table>

481
<table>
<thead>
<tr>
<th>Option</th>
<th>Arguments</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-csv_file_select_rows</code></td>
<td>fname R</td>
<td>Selects rows listed in $R$ from the csv-file fname.</td>
</tr>
<tr>
<td><code>-csv_file_select_cols</code></td>
<td>fname R</td>
<td>Selects columns listed in $R$ from the csv-file fname.</td>
</tr>
<tr>
<td><code>-csv_file_select_rows_and_cols</code></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><code>-csv_file_join</code></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><code>-csv_file_latex</code></td>
<td>fname</td>
<td>Produces a latex table from the given csv-file.</td>
</tr>
<tr>
<td><code>-store_as_csv_file</code></td>
<td>fname m 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:

```
Row, "C0", "C2", "C4"
0, "1", "2", "4"
1, "2", "4", "1"
2, "4", "1", "2"
END
```
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 $s = \text{sizeof(int)}$.

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 than $2^{8s-1}$ where $s = \text{sizeof(long int)}$.

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 than 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.
4. Install additional software using your own GNU/Linux distribution package manager.
5. Invoke Windows applications using a Unix-like command-line shell.
6. 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 web, go to
  [https://github.com/abetten/orbiter](https://github.com/abetten/orbiter)
• Click on the green icon “Code” that opens a drop-down menu
• You want to copy HTTPS URL
Orbiter - installation

- In command prompt, once you are in C:\Users\Joel type the command
  - `wsl.exe git clone https://github.com/abetten/orbiter.git`
  - Hit enter
- Now, orbiter will begin the cloning process

![Cloning into orbiter]

Orbiter - compile

- After cloning orbiter, run the command
  - `dir`
- You will find a new directory created called “orbiter”
- Change directory to “orbiter”
  - `cd orbiter`

![Directory of C:\Users\Joel]
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

![Screenshot of make command output]

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/](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
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30

The Makefile

#MY PATH=../orbiter
MY PATH=~/DEV.22/orbiter
#MY PATH=/scratch/betten/COMPILE/orbiter

# uncomment exactly one of the following two lines.
# uncomment the first if you want to run orbiter through docker.
# uncomment the second if you have an installed copy of orbiter and you want to r
un it directly
#ORBITER PATH=docker run -it --volume ${PWD}:/mnt -w /mnt abetten/orbiter
ORBITER PATH=$(MY PATH)/src/apps/orbiter/
ORBITER=$(ORBITER PATH)orbiter.out
SANDBOX=$(MY PATH)/src/apps/sandbox/sandbox.out
###############################################################################
# additional configurations for when you want to
# compile automatically generated code
###############################################################################
SRC=$(MY PATH)/src
MY CPP = g++
MY CC = gcc
CPPFLAGS = -Wall -I../../DEV.22/orbiter/src/lib -std=c++14
LIB = $(SRC)/lib/liborbiter.a -lpthread
LFLAGS = -lm -Wl,-rpath -Wl,/usr/local/gcc-8.2.0/lib64

###############################################################################
# End of configuration part
###############################################################################

495


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

#############################################################################
# Makefile Variables

#MAGMA_PATH=/usr/local/magma
MAGMA_PATH=

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

#Co3 from Conway et al., 1985 (ATLAS)
#order = 49576656000
#Co3 from the paper by Suleiman and Wilson 1997

CONWAY_GEN1="\%
110111000100001010000\%
1111010111110100001011\%
00000100000100010100\%
1111010110110001001110\%
010101000000010011101"
CONWAY

`GEN2="1,2,3,4,0,6,7,8,9,5,10,11,12"

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

# large sets of PG(2,3):
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, \
1,2,3,4,0,6,7,9,8,5,10,11,12, \
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="\n3000300030 \n2000201230 \n1000100111 \n1000220200 \n1002312010 \n0331003211 \n2200011331"
GENERATORS_WEYL_GROUP_E8="\n  -1,-1,-1,-1,0,0,0,0, \n  0,0,0,1,0,0,0,0, \n  1,0,0,0,0,0,0,0, \n  0,0,1,0,0,0,0,0, \n  0,1,0,1,1,0,0,0, \n  0,0,0,0,0,1,0,0, \n  0,0,0,0,0,0,1,0, \n  -1,0,-1,-1,-1,-1,-1,-1, \n  0,1,0,1,1,1,1, \n  1,0,0,0,0,0,0,0, \n  0,0,1,0,0,0,0,0, \n  0,0,0,1,0,0,0,0, \n  0,0,0,0,1,0,0,0, \n  0,0,0,0,0,1,0,0, \n  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, \n  14,18,1,5,13,6,7, 3,3,2,1,24,16,3, 17,3,22,10,16,24,26, \n  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, \n  9,7,2,15,17,5,11, 3,3,6,21,4,24,16, 25,8,6,24,21,12,7, \n  24,15,2,13,11,14,24"

HIRSCHFELD_SURFACE_Q4_SET_OF_POINTS="0,1,2,3,4,5,6,7,8,9,\n  10,11,12,13,14,23,26,27,30,31,34,35,38,39,42,47,48,51,52,\n  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, \n  141, 159, 176, 184, 199, 220, 235, 245, 262"

HYPEROVAL_16_16320="0, 1, 2, 3, 52, 70, 83, 109, 127, \n  139, 156, 174, 186, 199, 217, 229, 256, 264"

FILE_24_3.TFC_INC="24 24 72\n  n0 1 2 24 27 48 53 54 73 79 80 97 105 106 122 131 \n  132 146 157 158 171 175 183 195 203 208 220 225 233 244 \n  258 259 269 272 282 293 300 308 318 325 333 342 352 358 \n  367 379 381 392 398 400 417 428 429 442 443 450 466 471 \n  479 492 497 502 517 519 521 542 548 551 571 574 575 48\n"
\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 \\
148 \n0 1 2 24 27 28 48 53 54 73 79 80 97 105 106 122 131 \\
132 146 157 158 171 175 179 195 201 207 220 226 232 244 \\
257 258 269 274 277 293 300 307 318 323 329 342 352 356 \\
367 374 381 392 397 406 416 423 431 441 450 454 468 476 \\
479 494 499 503 519 521 525 544 547 550 570 572 575 \\
144 \n-1 3" DOILY="Row,C0,C1,C2\n0,0,12,13\n1,1,12,14\n2,8,9,12\n3,4,6,8\n4,6,10,14\n5,3,7,8\n6,7,10,13\n7,4,11,13\n8,3,11,14\n9,0,5,6\n10,1,5,7\n11,5,9,11\n12,0,2,3\n13,1,2,4\n14,2,9,10\nEND"

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_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 \n062100 \n006210 \n000621"

CODE_RS_10_8_11="\n7,2,1,0,0,0,0,0,0,0,0, \n0,7,2,1,0,0,0,0,0,0,0, \n0,0,7,2,1,0,0,0,0,0,0, \n0,0,0,7,2,1,0,0,0,0,0, \n0,0,0,0,7,2,1,0,0,0,0, \n0,0,0,0,0,7,2,1,0,0,0, \n0,0,0,0,0,0,7,2,1,0,0, \n0,0,0,0,0,0,0,7,2,1,0,0, \n0,0,0,0,0,0,0,0,7,2,1,0,0, \n0,0,0,0,0,0,0,0,0,7,2,1""

# Normal form for 15 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"
\[
\begin{align*}
+ (0+\alpha*\delta+\alpha*\gamma -\beta*\gamma -\beta + \delta - \gamma)*x_0*x_2*x_2 \\
- (\alpha*\delta + \beta - \delta)*x_0*x_2*x_3 \\
- (\delta + 1)*(\alpha - 1)*x_1*x_1*x_2 \\
+ (\alpha*\delta - \alpha*\gamma + \beta*\gamma + \beta - \delta + \gamma)*x_1*x_2*x_2 \\
+ (\alpha*\beta*\gamma + \alpha*\beta + \alpha*\delta - \alpha*\gamma + \beta*\gamma + \beta - \delta + \gamma)*x_1*x_2*x_3 \\
+ \alpha*\beta*(\gamma + 1)*x_1*x_3*x_3
\end{align*}
\]

# general normal form for surfaces with 27 lines:

\[
F_{abcd\_eqn}=-(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*(b - d)*X0*X0*X2 \\
+ (a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*(a + b - c - d)*X0*X1*X2 \\
+ (a*a*c - a*a*d - a*c*c + b*c*c + a*d - b*c)*(b - d)*X0*X1*X3 \\
- (a*d - b*c)*(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*X0*X2*X2 \\
- (a*a*c*d - a*b*c*c - a*a*d + a*b*d + b*c*c - b*c*d)*(b - d)*X0*X2*X3 \\
- (a - c)*(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*X1*X1*X2 \\
- (a - c)*(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*X1*X1*X3 \\
+ (a*d - b*c)*(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*X1*X2*X2 \\
+ ((1+1)*a*a*b*c*d - a*a*b*d*d - (1+1)*a*a*c*d*d - b*b*c*c*d + a*b*c*c*d + a*b*d*d)*X1*X2*X3 \\
+ c*a*(a*d - b*c - a + b + c - d)*(b - d)*X1*X3*X3
\]

KNECHT_13_1_AS_PAIRS="1,0,1,1,2,12,9"

KNECHT_13_1_AS_VECTOR="1,1,1,0,0,0,0,0,0,0,0,0"

KNECHT_13_2_AS_PAIRS="1,0,1,1,2,8,9,8,10,8,11"

KNECHT_13_2_AS_VECTOR="1,1,1,0,0,0,0,0,8,0,0,0"

# 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,\n1,5,1,4,1,2,1,1,1,0"
CRC32_CASTAGNOLI="1,32,1,28,1,27,1,26,1,25,1,23,1,22,1,20,1,19,1,\n18,1,14,1,13,1,11,1,10,1,9,1,8,1,6,1,0"
CRC64_ECMA182="1,64,1,62,1,57,1,55,1,54,1,53,1,52,1,47,1,46,1,45,\n1,40,1,39,1,38,1,37,1,35,1,33,1,32,1,31,1,29,1,27,1,24,1,23,1,22,\n1,21,1,19,1,17,1,13,1,12,1,10,1,9,1,7,1,4,1,1,1,0"
CRC64_ROCKSOFT="1,64,1,63,1,61,1,59,1,58,1,56,1,55,1,52,1,49,1,48,\n1,47,1,46,1,44,1,41,1,37,1,36,1,34,1,32,1,31,1,28,1,26,1,23,1,22,1,\n19,1,16,1,13,1,12,1,10,1,9,1,6,1,4,1,3,1,0"

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="\n11111111\n01010101\n00110011\n00001111"

CODE_RM_4_1_GENMA="\n11111111111111\n01010101010101\n0011001100110011\n0000011100001111\n0000000011111111"
CODE_RS_8="\
5610000\
0561000\
0056100\
0005610\
0000561"
CODE_RS_11_RREF="\
1,0,0,0,0,0,0,0,7,2,\
0,1,0,0,0,0,0,0,8,3,\
0,0,1,0,0,0,0,0,1,2,\
0,0,0,1,0,0,0,0,8,8,\
0,0,0,1,0,0,0,0,10,3,\
0,0,0,0,1,0,0,0,1,4,\
0,0,0,0,0,1,0,0,5,4,\
0,0,0,0,0,0,1,5,8"
RS_8_reduced="\
010001100000000000000\
001110010000000000000\
110011001000000000000\
000100011000000000000\
000011100100000000000\
000110011001000000000\
000000010001100000000\
000000001110010000000\
000000001110010000000\
000000000100011000000\
000000010000110010000\
000000000001110010000\
000000000000110011001"
# there are 5 [15,6,6]

# ago=12
CODE_15_6_6_A="  
  111111111100000 \
  111110000010000 \
  111001100001000 \
  110101100001000 \
  101010110000100 \
  101101001000001"

# ago=12
CODE_15_6_6_B="  
  111111111100000 \
  111110000010000 \
  111001100001000 \
  110101010000100 \
  101010110000010 \
  101101001000001"

# ago=720:
CODE_15_6_6_C="  
  111111111110000 \
  111110000010000 \
  111001100001000 \
  110101010000100 \
  101101001000010 \
  100010111000001"

# ago=96:
CODE_15_6_6_D="  
  111111111110000 \
  111110000010000 \
  111001100001000 \
  110101010000100 \
  101010100100010 \
  011001011000001"

# ago=360
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507

508
509

510
511
512
513
514
515
516
517
518
519
520

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"

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"
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"

506


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 + 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 + 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 + x2*x3*x4 + x2*x3*x5 + x2*x3*x6 + x2*x3*x7 + x2*x4*x5 + x2*x4*x6 + x2*x4*x7 + x3*x4*x5 + x3*x4*x6 + x3*x4*x7 + x3*x5*x6 + x3*x5*x7 + x3*x6*x7 + x4*x5*x6 + x4*x5*x7 + x4*x6*x7 + x5*x6*x7"
\texttt{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*x6 + x0*x1*x2*x4*x5*x7 + x0*x1*x2*x4*x6*x7 + x0*x1*x2*x5*x6*x7 + x0*x1*x2*x5*x7 + x0*x1*x2*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*x3*x5*x7 + x0*x1*x3*x6*x7 + x0*x1*x4*x5*x6*x7 + x0*x1*x4*x5*x7 + x0*x1*x4*x6*x7 + x0*x1*x5*x6*x7 + x0*x1*x5*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*x3*x5*x7 + x0*x2*x3*x6*x7 + x0*x2*x4*x5*x6*x7 + x0*x2*x4*x5*x7 + x0*x2*x4*x6*x7 + x0*x2*x5*x6*x7 + x0*x2*x5*x7 + x0*x2*x6*x7 + x0*x3*x4*x5*x6 + x0*x3*x4*x5*x7 + x0*x3*x4*x6*x7 + x0*x3*x5*x6*x7 + x0*x3*x5*x7 + x0*x3*x6*x7 + x0*x4*x5*x6*x7 + x0*x4*x5*x7 + x0*x4*x6*x7 + x0*x5*x6*x7 + x0*x5*x7 + x0*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*x3*x5*x7 + x1*x2*x3*x6*x7 + x1*x2*x4*x5*x6*x7 + x1*x2*x4*x5*x7 + x1*x2*x4*x6*x7 + x1*x2*x5*x6*x7 + x1*x2*x5*x7 + x1*x2*x6*x7 + x1*x3*x4*x5*x6*x7 + x1*x3*x4*x5*x7 + x1*x3*x4*x6*x7 + x1*x3*x5*x6*x7 + x1*x3*x5*x7 + x1*x3*x6*x7 + x1*x4*x5*x6*x7 + x1*x4*x5*x7 + x1*x4*x6*x7 + x1*x5*x6*x7 + x1*x5*x7 + x1*x6*x7 + x2*x3*x4*x5*x6*x7 + x2*x3*x4*x5*x7 + x2*x3*x4*x6*x7 + x2*x3*x5*x6*x7 + x2*x3*x5*x7 + x2*x3*x6*x7 + x2*x4*x5*x6*x7 + x2*x4*x5*x7 + x2*x4*x6*x7 + x2*x5*x6*x7 + x2*x5*x7 + x2*x6*x7 + x3*x4*x5*x6*x7 + x3*x4*x5*x7 + x3*x4*x6*x7 + x3*x5*x6*x7 + x3*x5*x7 + x3*x6*x7 + x4*x5*x6*x7 + x4*x5*x7 + x4*x6*x7 + x5*x6*x7 + x5*x7 + x6*x7"

\texttt{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*x3*x5*x7 + x0*x1*x2*x3*x6*x7 + x0*x1*x2*x4*x5*x6*x7 + x0*x1*x2*x4*x5*x7 + x0*x1*x2*x4*x6*x7 + x0*x1*x2*x5*x6*x7 + x0*x1*x2*x5*x7 + x0*x1*x2*x6*x7 + x0*x1*x3*x4*x5*x6*x7 + x0*x1*x3*x4*x5*x7 + x0*x1*x3*x4*x6*x7 + x0*x1*x3*x5*x6*x7 + x0*x1*x3*x5*x7 + x0*x1*x3*x6*x7 + x0*x1*x4*x5*x6*x7 + x0*x1*x4*x5*x7 + x0*x1*x4*x6*x7 + x0*x1*x5*x6*x7 + x0*x1*x5*x7 + x0*x1*x6*x7 + x0*x2*x3*x4*x5*x6*x7 + x0*x2*x3*x4*x5*x7 + x0*x2*x3*x4*x6*x7 + x0*x2*x3*x5*x6*x7 + x0*x2*x3*x5*x7 + x0*x2*x3*x6*x7 + x0*x2*x4*x5*x6*x7 + x0*x2*x4*x5*x7 + x0*x2*x4*x6*x7 + x0*x2*x5*x6*x7 + x0*x2*x5*x7 + x0*x2*x6*x7 + x0*x3*x4*x5*x6*x7 + x0*x3*x4*x5*x7 + x0*x3*x4*x6*x7 + x0*x3*x5*x6*x7 + x0*x3*x5*x7 + x0*x3*x6*x7 + x0*x4*x5*x6*x7 + x0*x4*x5*x7 + x0*x4*x6*x7 + x0*x5*x6*x7 + x0*x5*x7 + x0*x6*x7 + x1*x2*x3*x4*x5*x6*x7 + x1*x2*x3*x4*x5*x7 + x1*x2*x3*x4*x6*x7 + x1*x2*x3*x5*x6*x7 + x1*x2*x3*x5*x7 + x1*x2*x3*x6*x7 + x1*x2*x4*x5*x6*x7 + x1*x2*x4*x5*x7 + x1*x2*x4*x6*x7 + x1*x2*x5*x6*x7 + x1*x2*x5*x7 + x1*x2*x6*x7 + x1*x3*x4*x5*x6*x7 + x1*x3*x4*x5*x7 + x1*x3*x4*x6*x7 + x1*x3*x5*x6*x7 + x1*x3*x5*x7 + x1*x3*x6*x7 + x1*x4*x5*x6*x7 + x1*x4*x5*x7 + x1*x4*x6*x7 + x1*x5*x6*x7 + x1*x5*x7 + x1*x6*x7 + x2*x3*x4*x5*x6*x7 + x2*x3*x4*x5*x7 + x2*x3*x4*x6*x7 + x2*x3*x5*x6*x7 + x2*x3*x5*x7 + x2*x3*x6*x7 + x2*x4*x5*x6*x7 + x2*x4*x5*x7 + x2*x4*x6*x7 + x2*x5*x6*x7 + x2*x5*x7 + x2*x6*x7 + x3*x4*x5*x6*x7 + x3*x4*x5*x7 + x3*x4*x6*x7 + x3*x5*x6*x7 + x3*x5*x7 + x3*x6*x7 + x4*x5*x6*x7 + x4*x5*x7 + x4*x6*x7 + x5*x6*x7 + x5*x7 + x6*x7"

\texttt{ELEMENTARY\_SYMMETRIC\_8\_8}="x0*x1*x2*x3*x4*x5*x6*x7"
CLASS_2A=-centralizer_of_element \\
"1,0,0,0, 0,1,0,0, 0,0,1,0, 0,0,0,1, 1" \\
-label "2A"

# Baer involution

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 \\
> "1,0,0,0, 0,1,0,0, 0,0,1,0, 0,0,0,1, 1"

PGGL_4_4_SUBGROUP_2A_NORMALIZER=-PGGL 4 4 \\
> -subgroup_by_generators "centralizer_2A" "40320" 10 \\
> "1,0,0,0,0,1,0,0,0,1,0,0,0,0,1,1, \\
> 1,0,0,0,0,1,0,0,0,1,0,0,1,1,0,1,1,1, \\
> 1,0,0,0,0,1,0,0,0,1,0,1,1,1,1,0, \\
> 1,0,0,0,0,1,0,0,1,1,0,1,1,1,1,0,0, \\
> 1,0,0,0,0,1,0,0,0,1,0,1,0,1,1,1,0, \\
> 1,0,0,0,0,1,0,0,0,1,1,1,0,1,0,1,0, \\
> 1,0,0,0,0,1,1,1,1,1,0,1,1,1,1,0, \\
> 1,0,0,0,0,1,1,1,1,1,0,1,1,1,1,0, \\
> 0,1,0,0,0,1,0,1,1,1,1,0,1,0,1,1,1,1

# 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, 0,0,0,1, 1"
PGGL_4_4_SUBGROUP_2C_NORMALIZER=-PGGL 4 4 \n> -subgroup_by_generators "centralizer_2C" "30720" 9 \n> "1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1, \n> 1,0,0,0,0,1,0,0,0,0,2,0,0,0,0,2,1, \n> 1,0,0,0,0,1,0,0,0,0,1,0,2,0,3,1,0, \n> 1,0,0,0,0,1,0,0,0,0,1,0,1,0,3,1,0, \n> 1,0,0,0,1,1,0,0,0,0,1,0,0,0,1,1,0, \n> 1,0,0,0,2,1,0,0,0,0,1,0,1,0,1,0, \n> 1,0,0,0,1,1,2,0,0,0,1,0,0,0,1,0, \n> 1,0,3,0,1,1,1,3,0,0,2,0,0,0,0,2,1," \n
PGGL_4_4_SUBGROUP_5A=-PGGL 4 4 \n> -subgroup_by_generators "5A" 5 1 \n> "0,2,0,0, 1,1,0,0, 0,0,3,0, 0,0,0,3, 0" \n
PGGL_4_4_SUBGROUP_5A_NORMALIZER=-PGGL 4 4 \n> -subgroup_by_generators "normalizer_5A" "3600" 6 \n> "1,0,0,0,0,1,0,0,0,0,2,0,0,0,0,2,0, \n> 1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,2,0, \n> 1,0,0,0,0,1,0,0,0,0,1,0,0,1,1,0, \n> 1,0,0,0,0,1,0,0,0,0,2,0,0,2,0,0, \n> 1,0,0,0,2,2,0,0,0,0,1,0,0,0,0,1,1, \n> 0,1,0,0,3,3,0,0,0,0,2,0,0,0,0,2,0" \n
PGGL_4_4_SUBGROUP_5B=-PGGL 4 4 \n> -subgroup_by_generators "5B" 5 1 \n> "0,2,0,0,1,1,0,0,0,0,0,2,0,1,1,0" \n
PGGL_4_4_SUBGROUP_5B_NORMALIZER=-PGGL 4 4 \n> -subgroup_by_generators "normalizer_5B" "81600" 6 \n> "1,0,0,0,0,1,0,0,0,0,2,0,0,0,0,2,0, \n> 1,0,0,0,0,1,0,0,0,0,1,0,0,0,1,1,0, \n> 1,0,0,0,0,1,0,0,0,0,2,0,0,2,0,0, \n> 1,0,0,0,2,2,0,0,0,0,1,0,0,0,0,1,1, \n> 0,1,0,0,3,3,0,0,0,0,1,0,0,3,3,0, \n> 0,0,1,0,0,0,1,2,2,0,0,0,2,3,0,0,1 \n
PGGL_4_4_SUBGROUP_2Cx2_0=-PGGL 4 4 \n> -subgroup_by_generators "2Cx2_0" 4 2 \n> "1,0,0,0,0,1,0,0,0,0,1,0,0,0,1,1,0 \n> 1,0,0,0,0,1,0,0,0,0,1,0,0,0,1,0, \n> 1,0,0,0,0,1,0,0,0,0,1,0,0,0,1,0, \n> 1,0,0,0,0,1,0,0,0,0,1,0,0,0,1,0, \n> 1,0,0,0,0,1,0,0,0,0,1,0,0,0,1,0, \n> 1,0,0,0,0,1,0,0,0,0,1,0,0,0,1,0, \n> 1,0,0,0,0,1,0,0,0,0,1,0,0,0,1,0, \n> 1,0,0,0,0,1,0,0,0,0,1,0,0,0,1,0, \n> 1,0,0,0,0,1,0,0,0,0,1,0,0,0,1,0, \n> 1,0,0,0,0,1,0,0,0,0,1,0,0,0,1,0, \n> 1,0,0,0,0,1,0,0,0,0,1,0,0,0,1,0, \n> 1,0,0,0,0,1,0,0,0,0,1,0,0,0,1,0, \n> 1,0,0,0,0,1,0,0,0,0,1,0,0,0,1,0,
-subgroup_by_generators "normalizer_2Cx2_0" "768" 8 \ 
 1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1 \ 
 1,0,0,0,0,1,0,0,0,1,0,0,0,1,0,1,1 \ 
 1,0,0,0,1,0,0,0,1,0,0,1,0,1,0,1,1 \ 
 1,0,0,0,2,1,1,0,1,0,1,1,1,2,1,1 \ 
 1,0,0,0,0,0,1,0,0,1,0,0,0,0,0,1,0 \ 
 1,0,0,0,3,1,0,0,1,0,0,1,3,1,1" 

#PGL 4 5_SUBGROUP_3B=-PGL 4 5 
#-subgroup_by_generators "3B" 3 1 
"1,0,0,0,0,1,0,0,0,0,2,1,0,0,3,2" 

#PGL 4 5_SUBGROUP_3B_NORMALIZER=-PGL 4 5 
#-subgroup_by_generators "normalizer_3B" "5760" 8 
"1,0,0,0,0,4,0,0,0,0,1,0,0,0,0,1," 
"1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,4," 
"1,0,0,0,0,4,0,0,0,0,4,0,0,0,0,4," 
"1,0,0,0,0,1,0,0,0,0,3,0,0,0,0,3," 
"1,0,0,0,0,3,0,0,0,0,1,0,0,0,0,1," 
"1,0,0,0,0,1,0,0,0,1,4,0,0,2,1," 
"1,0,0,0,4,4,0,0,0,0,1,0,0,0,0,1," 
"0,1,0,0,1,0,0,0,0,0,4,0,0,0,0,4," 

#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, 
1,0,0,0,0,1,0,0,0,1,0,0,0,0,4, 
1,0,0,0,0,1,0,0,0,1,0,0,0,0,4,"
PGL_4_5_SUBGROUP_31_ME=-PGL 4 5 \n> subgroup_by_generators "31" 31 1 \n> "1,0,0,0,0,4,0,0,0,0,4,0,0,0,0,4, \n> 1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1, \n> 1,0,0,0,0,3,0,0,0,0,3,0,0,0,0,3, \n> 1,0,0,0,0,1,0,0,0,0,2,4,0,0,2,3, \n> 1,0,0,0,4,4,0,0,0,1,0,0,0,0,1, \n> 0,1,0,0,1,0,0,0,0,4,0,0,0,0,4,"

PGL_4_5_SUBGROUP_31_ME_NORMALIZER=-PGL 4 5 \n> subgroup_by_generators "normalizer_31" "372" 4 \n> "1,0,0,0,0,4,0,0,0,0,4,0,0,0,0,4, \n> 1,0,0,0,3,0,0,0,0,3,0,0,0,0,3, \n> 1,0,0,0,4,0,0,0,0,2,1,0,3,2,4, \n> 1,0,0,0,0,1,0,0,0,0,1,0,1,1,3,"

# subgroup of order 31 for the construction of regular packings in PG_3_5:

PGL_4_5_SUBGROUP_31=-PGL 4 5 \n> subgroup_by_generators "31" 31 1 \n> "2,0,0,0,0,0,1,0,0,0,0,1,0,1,1,3,"

PGL_4_5_SUBGROUP_31_NORMALIZER=-PGL 4 5 \n> subgroup_by_generators "normalizer_31" "372" 4 \n> "1,0,0,0,0,4,0,0,0,0,4,0,0,0,0,4, \n> 1,0,0,0,3,0,0,0,0,3,0,0,0,0,3, \n> 1,0,0,0,4,0,0,0,0,2,1,0,3,2,4, \n> 1,0,0,0,0,1,0,0,0,0,1,0,1,1,3,"

#372:

#"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, " 
#"1,0,0,0,0,1,0,0,0,0,1,0,1,1,3,"
# Exterior square roots:

elt of order 3:
the exterior square root of f is X =

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 2 & 2 \\
0 & 0 & 4 & 2 \\
\end{bmatrix}
\]

elt of order 31:
the exterior square root of g is Z =

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

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, \\
0,1,0,0,0,1,0, \\
0,0,1,0,1,1,0, \\
0,0,0,1,1,1,1"

GOLAY23_CODE_GENERATOR="
1,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,0,1,0,1,0, \\
0,1,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,1,0,1,0, \\
0,0,1,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,1,1,1,1,0, \\
0,0,0,1,0,0,0,0,0,0,0,0,1,1,1,1,1,1,0,1,1,1,1,0, \\
0,0,0,0,1,0,0,0,0,0,0,0,1,1,1,1,1,1,1,0,1,1,1,0, \\
0,0,0,0,0,1,0,0,0,0,0,0,1,1,1,1,1,1,1,1,0,1,1,0, \\
0,0,0,0,0,0,1,0,0,0,0,0,1,1,1,1,1,1,1,1,1,0,1,0, \\
0,0,0,0,0,0,0,1,0,0,0,0,1,1,1,1,1,1,1,1,1,1,0,0, \\
0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,1,1,1,1,1,1,0, \\
0,0,0,0,0,0,0,0,0,1,0,0,1,1,1,1,1,1,1,1,1,1,1,0, \\
0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,1,1,1,1,1,1,1,1,0, \\
0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,1,1,1,1,1,1,1,0, \\
0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,0, \\
0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,0, \\
0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,0, \\
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,0, \\
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,0, \\
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,0, \\
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0, \\
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0, \\
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0, \\
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0, \\
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0, \\
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1, \\
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
\]
HAMMING_CODE_ROWS_IN_BINARY_RANKS="67,37,22,15"

SIMPLEX_CODE_GENMA_CYCLIC="\n1,0,0,1,1,1,0, \n0,1,0,0,1,1,1, \n0,0,1,1,1,1,0,1""
AG_2_3_BLOCKS="0,13,22,27,35,41,47,53,55,59,71,76"
LARGE_SET_AG_2_3_NEIGHBOR_SET="129,130,133,134,136,139,141,142,153,154,156,160,16
5,166,178,179,183,184,185,190,192,194,197,203,204,206,218,221,222,225,227,231,248
,251,252,255,256,259,261,262,272,277,279,283,285,287,299,301,303,305,306,309,313,
315,319,320,323,325,341,342,343,344,345,349,368,371,375,378,381,383,392,393,397,4
0,403,405,416,419,421,422,425,426,429,430,440,443,447,449,453,454,464,467,468,47
3,474,479,490,493,494,497,500,503,513,517,518,520,523,527,536,539,541,542,544,547
,548,551,563,566,567,571,572,573,585,589,590,593,595,596,600,601,603,611,614,615,
625,629,631,635,637,638,657,659,661,667,668,671,681,683,686,689,691,693,705,706,7
09,710,712,715,717,718,720,723,724,729,733,735,747,748,750,752,754,757,777,780,78
1,784,790,791,802,804,807,808,811,814,824,827,828,831,832,835,837,838"

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, \n1,0,0,0,0,1, \n1,0,1,0,0,0, \n0,1,0,0,1,1"\n
TEST_RHS="1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1"

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=\
  > -nb_orbits_on_blocks 1 \\n  > -depth 5 \\n  > -mask_label "no_mask"

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=\n  -nb_orbits_on_blocks 3 \
  -depth 6 \
  -mask_label "no_mask"

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 \n    "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=\n  -masktest 3 x ge 1 \
  -masktest 4 x+y ge 3 \
  -depth 4 \
  -mask_label "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 \n    "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,\n  75363,27729,32139,5244,60442,142811,111115,94209,120678,89533,13798,\n  103994,129953,82168,136838,19253,23017,145985,134996,54705,36267,\n  55066,117542,96699,69154,72460"

PENTTILA_WILLIAMS_PRINCE_REG_PACKING_1="616,42728,152655,48576,105431,\n  79607,28634,32817,9799,62356,141176,110085,92557,122136,86312,13975,\n  101942,126869,81478,139352,18028,24325,147284,130370,52074,36843,\n  55602,118454,95973,69642,74036"

PENTTILA_WILLIAMS_PRINCE_REG_PACKING_0_DUAL="3938, 66740, 56555, \n  93538, 107785, 64917, 47567, 54483, 141012, 138602, 18308, 6880, \n  131351, 88788, 125484, 102075, 21234, 99392, 119149, 80640, 124839, 
```
# consider the binary code with generator matrix:
# 1 0 1
# 0 1 1
CODE_N3_K2_Q2_GENMA="1,0,1, 0,1,1"
CODE_N6_K3_Q2_GENMA="111100, 110010, 101001"

TRIANGLE_GRAPH="0,1,1
1,0,1
1,1,0"

# q=17:
# 3 is p.e. mod 17.
# so we pick f=3.
# then, 2f^2=18=1
# 4f = 12
# X^4 -Y^4 -Z^4 +2f^2Y^2Z^2 +4fX^2YZ
#endif
EDGE_CURVE_Q17_EQUATION="1,16,16,0,0,0,0,0,0,0,0,1,12,0,0"
EDGE_CURVE_Q17_AS_POINTS="4, 7, 16, 19, 20, 23, 32, 35, 89, 100, 244, 251"
FILE_Q17="orbit,curve,pts_on_curve,bitangents,go\n\n","$(EDGE_CURVE_Q17_EQUATION)","$(EDGE_CURVE_Q17_AS_POINTS)","",-1\n\nEND"

DESARGUES_PATH_LEX_LEAST="10 10 3\n0\n1 0 15 26\n4 0 15 26 46\n5 0 1 5 26 46 56\n6 0 15 26 46 56 72\n7 0 15 26 46 56 72 80\n8 0 15 26 46 56 72 80 93\n9 0 15 26 46 56 72 80 93 106\n10 0 15 26 46 56 72 80 93 106 119\n-1"

# Povray:
# povray colors:
POLISHED_CHROME_WHITE="\ntexture{ Polished_Chrome pigment{quick_color White} }

YELLOW_TRANSPARENT="\ntexture{ pigment{ color Yellow transmit 0.7 } \nfinish {diffuse 0.9 phong 0.6} }

COLOR_RED="\ntexture{ pigment{ color Red } \nfinish {diffuse 0.9 phong 0.6} }

COLOR_RED_SHINY="\ntexture{ pigment{ color Red } \nfinish { diffuse 0.9 phong 1} }

COLOR_GREEN_SHINY="\ntexture{ pigment{ color Green } \nfinish { diffuse 0.9 phong 1} }

COLOR_BLUE_SHINY="\ntexture{ pigment{ color Blue } \nfinish { diffuse 0.9 phong 1} }

COLOR_YELLOW_SHINY="\ntexture{ pigment{ color Yellow } \nfinish { diffuse 0.9 phong 1} }

COLOR_BLACK_SHINY="\ntexture{ pigment{ color Black } \nfinish { diffuse 0.9 phong 1} }"
COLOR_RED SEE THROUGH=
  "texture{ pigment{ color Red transmit 0.5 } \n  finish { diffuse 0.9 phong 1}}"

COLOR_GREEN SEE THROUGH=
  "texture{ pigment{ color Green transmit 0.5 } \n  finish { diffuse 0.9 phong 1}}"

COLOR_BLUE SEE THROUGH=
  "texture{ pigment{ color Blue transmit 0.5 } \n  finish { diffuse 0.9 phong 1}}"

COLOR_YELLOW SEE THROUGH=
  "texture{ pigment{ color Yellow transmit 0.5 } \n  finish { diffuse 0.9 phong 1}}"

COLOR_YELLOW_THICK=
  "texture{ pigment{ color Yellow } \n  finish { diffuse 0.9 phong 1}}"

COLOR_BLACK NO_SHADOW=
  "texture{ pigment{Black} } no_shadow"

SURFACE_COLOR=
  "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=
  "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=
  "texture{ pigment{ Gold } finish \n  {ambient 0.4 diffuse 0.5 roughness 0.001 \n  reflection 0.1 specular .8} }"

COLOR_TURQUOISE=
  "texture{ pigment{Cyan*1.3} \n  finish {ambient 0.4 diffuse 0.6 roughness 0.001 \n  reflection 0.1 specular .8} }"

MONKEY_SADDLE_CUBIC="1,0,0,0,-3,0,0,0,0,0,0,0,0,0,0,0,0,0,-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="-2,0,0,0,2,0,0,2,0,2,0,0,\0,0,0,0,0,0,0,0,-2,0,2,0,-2"


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,-687.529,0,0,1582.59,0,1186.94,0,0,0,0,-1055.06,0,\0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,"
# Section 2.3: Makefiles and Shell Scripts

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

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

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

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 2.4: Objects and Activities

# Section 2.5: Mathematical Data

```bash

```
create_BLT_5_1: $(ORBITER) -v 2 \n-define F -finite_field -q 5 -end \n-define O -orthogonal_space 0 5 F -end \n-define P -projective_space -n 3 -field F -v 0 -end \n-define S4 -set -loop 0 64 1 -end \n-define S -set -here "2,3,5,7,11,13" -end \n-define S -set -catalogue 1 -end \ncreate_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 S4 -set -loop 0 64 1 -end \n-define S -set -catalogue 1 -end

# Section 2.6: Set Builder

SECTION_SET_BUILDER:

set_of_primes: $(ORBITER) -v 2 \n-define S -set -here "2,3,5,7,11,13" -end \n-define S -set -catalogue 0 -end

set_interval: $(ORBITER) -v 2 -define S -set -loop 0 64 1 -end \n-define S -set -catalogue 0 -end

# Section 2.7: Vector Builder

SECTION_VECTOR_BUILDER:
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

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

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_example1:
$(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_co1:
$(ORBITER) -v 2 \
define F -finite_field -q 2 -end \
define v -vector -field F -format 22 \
 compact $(CONWAY_GEN1) -end \
print_symbols
SECTION_FORMULA_BUILDER:

TEST_FORMULA="(-a+b*b)*x*x+a*b*x"

formula_example:

$\text{dot} -Tpng \text{test\_formula.gv} > \text{test\_formula.png}$

open \text{test\_formula.png}

formula_evaluate:

$\text{dot} -Tpng \text{test\_formula.gv} > \text{test\_formula.png}$

open \text{test\_formula.png}

# 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

PR29:

$\text{orbiter} -v 1 -\text{smallest\_primitive\_root} 29$

PR31:

$\text{orbiter} -v 1 -\text{smallest\_primitive\_root} 31$

PR37:
$(ORBITER) -v 1 -smallest_primitive_root 37

PR_100:
$(ORBITER) -v 1 -smallest_primitive_root_interval 2 100

# randomized algo:

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

# a primitive root modulo 915839 is 43085

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

# the power of 43085 to the 49842 mod 915839 is 487320

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

# The discrete log is 49842 since 487320 = 43085^49842 mod 915839, time: 0:22

IM_723:
$(ORBITER) -v 5 -inverse_mod 723 4060

IM_3_19:
$(ORBITER) -v 5 -inverse_mod 3 19

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

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

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

sqrt_mod:
$(ORBITER) -v 2 -square_root_mod 33 41
\[ \sqrt{5} \mod 11: \]
\[
\text{\texttt{ORBITER}} -v 2 -\text{square} \text{\texttt{root}} \mod 5 11
\]
\[ \sqrt{5} \mod 19: \]
\[
\text{\texttt{ORBITER}} -v 2 -\text{square} \text{\texttt{root}} \mod 5 19
\]
\[ \sqrt{20} \mod 31: \]
\[
\text{\texttt{ORBITER}} -v 2 -\text{square} \text{\texttt{root}} \mod 20 31
\]
\[ \text{order of } 2 \mod n: \]
\[
\text{\texttt{ORBITER}} -v 3 -\text{order of q \mod n 2 3 151}
\]
\[ \text{PR}_{1000}: \]
\[
\text{\texttt{ORBITER}} -v 1 -\text{smallest} \text{\texttt{primitive root}} \text{\texttt{interval 2 1000}}
\]
\[ \text{PE number}_{1000}: \]
\[
\text{\texttt{ORBITER}} -v 1 -\text{number of primitive roots} \text{\texttt{interval 2 1000}}
\]
\[ \text{Eulerfunction}_{10000}: \]
\[
\text{\texttt{ORBITER}} -v 1 -\text{number of primitive roots} \text{\texttt{interval 10000 10001}}
\]
1402 power_function_2_mod_11:
1403 ▶ $(ORBITER) -v 5 -power_function_mod_n 2 11
1404 ▶ $(ORBITER) -v 1 -csv_file_latex 1 power_function_k2_n11.csv
1405 ▶ pdflatex power_function_k2_n11.tex
1406 ▶ open power_function_k2_n11.pdf
1407
1408 draw_mod_13:
1410 ▶ $(ORBITER) -v 2 \n1412 ▶ -draw_options -embedded -end \n1413 ▶ -draw_mod_n -n 13 -file mod_13 -power_cycle 2 -end
1414 ▶ pdflatex mod_13.draw.tex
1415 ▶ open mod_13.draw.pdf
1418
1420 # Section 3.2: Prime Fields
1421
1422 SECTION_PRIME_FIELDS:
1423 ▶
1425 F_2:
1426 ▶ $(ORBITER) -v 3 -list_arguments \n1428 ▶ -with F -do -finite_field_activity -cheat_sheet_GF -end
1429 ▶ pdflatex GF_2.tex
1430 ▶ open GF_2.pdf
1433 F_3:
1434 ▶ $(ORBITER) -v 3 \n1436 ▶ -with F -do -finite_field_activity -cheat_sheet_GF -end
1437 ▶ #pdflatex GF_3.tex
1438 ▶ #open GF_3.pdf
1440 F_5:
1441 ▶ $(ORBITER) -v 3 \n1443 ▶ -with F -do -finite_field_activity -cheat_sheet_GF -end
1444 ▶ pdflatex GF_5.tex
1445 ▶ open GF_5.pdf
1446
527
F_127:
$(ORBITER) -v 3 \\
-define F -finite_field -q 127 -end \\
-with F -do -finite_field_activity -cheat_sheet_GF -end

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

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

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

F_1021_wo:
$(ORBITER) -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:
\$(ORBITER) -v 3 \n\$define F -finite_field -q 4 -end \n\$with F -do -finite_field_activity -cheat_sheet_GF -end \npdflatex GF.4.tex \nopengf.4.pdf

**F\_4\_tables:**
\$(ORBITER) -v 3 \n\$define F -finite_field -q 4 -end \n\$with F -do -finite_field_activity -cheat_sheet_GF -end \n\$draw_matrix -input_csv_file GF.q4\_addition\_table.csv \n\$box_width 40 -bit_depth 24 -partition 3 4 4 -end \n\$draw_matrix -input_csv_file GF.q4\_multiplication\_table.csv \n\$box_width 40 -bit_depth 24 -partition 3 3 3 -end

#pdflatex GF.4.tex
#open GF.4.pdf

**F\_16\_tables:**
\$(ORBITER) -v 3 \n\$define F -finite_field -q 16 -end \n\$with F -do -finite_field_activity -cheat_sheet_GF -end \npdflatex GF.16.tex

529
# Section 3.4: Linear Algebra over Finite Fields

**SECTION LINEAR ALGEBRA:**

**RREF:**

```
$($\text{ORBITER}) -v 2 \n$ -define F -finite_field -q 2 -end \n$ -define v -vector -field F -format 2 \n$ -dense "1,1,1,0,1,1,0,1,0,1" \n$ -end \n$ -with F -do -finite_field_activity \n$ -RREF v -normalize_from_the_right \n$ -end
```

**RREF.V7:**

```
$($\text{ORBITER}) -v 2 \n$ -define F -finite_field -q 7 -end \n$ -define V -vector -format 7 \n$ -dense $(\text{V7\_VANDERMONDE\_EXTENDED}) \n$ -end \n$ -with F -do -finite_field_activity \n$ -RREF V \n$ -end
```

**nullspace:**

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

530
eigenstuff:

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

classes_GL_3_2:

```
$ (ORBITER) -v 7 \n  -define F -finite_field -q 2 -end \n  -all_rational_normal_forms F 3 \n  pdflatex Class_reps_GL_3_2.tex \n  open Class_reps_GL_3_2.pdf
```

classes_GL_4_2:

```
$ (ORBITER) -v 7 \n  -define F -finite_field -q 2 -end \n  -all_rational_normal_forms F 4 \n  pdflatex Class_reps_GL_4_2.tex \n  open Class_reps_GL_4_2.pdf
```

# 252 classes

RREF_demo_4_4_q5:

```
$ (ORBITER) -v 2 \n  -define F -finite_field -q 5 -end \n  -with F -do \n  -finite_field_activity -RREF_demo 4 4 -end \n  pdflatex RREF_example_q5_4_4.tex \n  open RREF_example_q5_4_4.pdf \n  gs -sDEVICE=png16 -dFIXEDMEDIA \n  -dDEVICEWIDTHPOINTS=500 -dDEVICEHEIGHTPOINTS=450 \n  -r240 -oRREF_example_q5_4_4_page%02d.png \n  RREF_example_q5_4_4.pdf
```

RREF_demo_4_6_q7:

```
$ (ORBITER) -v 2 \n  -define F -finite_field -q 7 -end 
```
# Section 3.5: Advanced Topics in finite fields

SECTION 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 

-els -end

$\$(ORBITER) -v 2 -draw_matrix \$

-input_csv_file F64_over_F8.csv 

-box_width 40 -bit_depth 24 

533
1729 ▶ ▶ -partition 4 "2,2,2,2,2,2,2,2" "2,2,2,2,2,2,2,2" -end
1730 ▶ open F64_over_F8_draw.bmp
1731 ▶ #pdflatex field_reduction_Q64_q8_8_8.tex
1732 ▶ #open field_reduction_Q64_q8_8_8.pdf
1733
1734
1735 F_64_over_F4_field_reduction:
1736 ▶ $(ORBITER) -v 2 \n1737 ▶ ▶ -define F -finite_field -q 64 -end \n1738 ▶ ▶ -define elts -vector -field F -loop 0 64 1 -end \n1739 ▶ ▶ -with F -do \n1740 ▶ ▶ -finite_field_activity \n1741 ▶ ▶ ▶ -field_reduction "F64_over_F4" 4 8 8 elts -end
1742 ▶ $(ORBITER) -v 2 -draw_matrix \n1743 ▶ ▶ -input_csv_file F64_over_F4.csv \n1744 ▶ ▶ -box_width 40 -bit_depth 24 \n1745 ▶ ▶ -partition 4 "3,3,3,3,3,3,3,3" "3,3,3,3,3,3,3,3" -end
1746 ▶ open F64_over_F4_draw.bmp
1747 ▶ #pdflatex field_reduction_Q64_q4_8_8.tex
1748 ▶ #open field_reduction_Q64_q4_8_8.pdf
1749
1750
1751 F_64_over_F2_field_reduction:
1752 ▶ $(ORBITER) -v 2 \n1753 ▶ ▶ -define F -finite_field -q 64 -end \n1754 ▶ ▶ -define elts -vector -field F -loop 0 64 1 -end \n1755 ▶ ▶ -with F -do \n1756 ▶ ▶ -finite_field_activity \n1757 ▶ ▶ ▶ -field_reduction "F64_over_F2" 2 8 8 elts -end
1758 ▶ $(ORBITER) -v 2 -draw_matrix \n1759 ▶ ▶ -input_csv_file F64_over_F2.csv \n1760 ▶ ▶ -box_width 40 -bit_depth 24 \n1761 ▶ ▶ -partition 4 "6,6,6,6,6,6,6,6" "6,6,6,6,6,6,6,6" -end
1762 ▶ open F64_over_F2_draw.bmp
1763 ▶ #pdflatex field_reduction_Q64_q2_8_8.tex
1764 ▶ #open field_reduction_Q64_q2_8_8.pdf
1765
1766
1767
1768
1769
1770 F_8_Nth_roots_21:
1771 ▶ $(ORBITER) -v 3 \n1772 ▶ ▶ -define F -finite_field -q 8 -override_polynomial 11 -end \n1773 ▶ ▶ -with F -do -coding_theoretic_activity \n1774 ▶ ▶ ▶ -nth_roots 21 \n1775 ▶ ▶ -end

534
# Section 3.6: Basic Ring Theory

```
1776 \> pdflatex Nth_roots_q8_n21.tex
1777 \> open Nth_roots_q8_n21.pdf
1778
1779
1780
1781
1782 F_8_vandermonde:
1783 \> $(ORBITER) -v 3 \ 
1784 \> \> -define F -finite_field -q 8 -end \ 
1785 \> \> -with F -do -finite_field_activity \ 
1786 \> \> \> -Vandermonde_matrix \ 
1787 \> \> -end
1788
1789
1790
1791 F_1024_vandermonde:
1792 \> $(ORBITER) -v 3 \ 
1793 \> \> -define F -finite_field -q 1024 -end \ 
1794 \> \> -with F -do -finite_field_activity \ 
1795 \> \> \> -Vandermonde_matrix \ 
1796 \> \> -end
1797 \> rm Vandermonde_1024.csv
1798 \> rm Vandermonde_inv_1024.csv
1799
1800 \#User time: 0:46
1801
1802
1803
1804
1805 NTT_k4_q17.cpp:
1806 \> $(ORBITER) -v 3 \ 
1807 \> \> -define F -finite_field -q 17 -end \ 
1808 \> \> -with F -do -coding_theoretic_activity \ 
1809 \> \> \> -NTT 4 17 \ 
1810 \> \> -end
1811
1812 F_17_NTT_compile: NTT_k4_q17.cpp
1813 \> $(MY_CPP) NTT_k4_q17.cpp $(CPPFLAGS) \ 
1814 \> \> $(LIB) $(LFLAGS) -o NTT_k4_q17.out
1815 \> ./NTT_k4_q17.out
1816
1817 \>
1818
1819
1820 # Section 3.6: Basic Ring Theory
1821
1822
```
Polynomial ring:
\begin{verbatim}
  (ORBITER) -v 3 \
  define F -finite_field -q 4 -end \
  define R -polynomial_ring -field F \
  number_of_variables 4 \
  homogeneous_of_degree 3 \
  variables "x0,x1,x2,x3" "x_0,x_1,x_2,x_3" \
  end
\end{verbatim}

\begin{verbatim}
PG_3_2_easy:
  (ORBITER) -v 2 \
  define F -finite_field -q 2 -end \
  define P -projective_space -n 3 -field F -end
\end{verbatim}

\begin{verbatim}
PG_1_16:
  (ORBITER) -v 2 \
  define F -finite_field -q 16 -end \
  define P -projective_space -n 1 -field F -v 0 -end \
  with P -do -projective_space.activity \
  cheat_sheet \
  -end
\end{verbatim}

\begin{verbatim}
pdflatex PG_1_16.tex
open PG_1_16.pdf
\end{verbatim}
1919 \( \text{PG}_3.4: \)
1920 \& \> $(\text{ORBITER}) -v 2 \ \$
1921 \& \& \> -define F -finite_field -q 4 -end \$
1922 \& \& \> -define P -projective_space -n 3 -field F -v 0 -end \$
1923 \& \& \> -with P -do -projective_space_activity \$
1924 \& \& \> \> -cheat_sheet \$
1925 \& \& \> -end
1926 \& \> pdflatex PG_3.4.tex
1927 \& \> open PG_3.4.pdf
1928
1929 \( \text{PG}_3.5: \)
1930 \& \> $(\text{ORBITER}) -v 2 \$
1931 \& \& \> -define F -finite_field -q 5 -end \$
1932 \& \& \> -define P -projective_space -n 3 -field F -v 0 -end \$
1933 \& \& \> -with P -do -projective_space_activity \$
1934 \& \& \> \> -cheat_sheet \$
1935 \& \& \> -end
1936 \& \> pdflatex PG_3.5.tex
1937 \& \> open PG_3.5.pdf
1938
1939
1940 \( \text{PG}_3.7: \)
1941 \& \> $(\text{ORBITER}) -v 2 \$
1942 \& \& \> -define F -finite_field -q 7 -end \$
1943 \& \& \> -define P -projective_space -n 3 -field F -v 0 -end \$
1944 \& \& \> -with P -do -projective_space_activity \$
1945 \& \& \> \> -cheat_sheet \$
1946 \& \& \> -end
1947 \& \> pdflatex PG_3.7.tex
1948 \& \> open PG_3.7.pdf
1949
1950
1951
1952
1953 \( \text{PG}_3.8: \)
1954 \& \> $(\text{ORBITER}) -v 2 \$
1955 \& \& \> -define F -finite_field -q 8 -end \$
1956 \& \& \> -define P -projective_space -n 3 -field F -v 0 -end \$
1957 \& \& \> -with P -do -projective_space_activity \$
1958 \& \& \> \> -cheat_sheet \$
1959 \& \& \> -end
1960 \& \> pdflatex PG_3.8.tex
1961 \& \> open PG_3.8.pdf
1962
1963
1964  PG_3_16:
1965  $ (ORBITER) -v 2 \
1966  $ -define F -finite_field -q 16 -end \n1967  $ -define P -projective_space -n 3 -field F -v 0 -end \n1968  $ -with P -do -projective_space_activity \n1969  $ -cheat_sheet \n1970  $ -end
1971  pdflatex PG_3_16.tex
1972  open PG_3_16.pdf
1973
1974
1975
1976
1977  PG_3_25:
1978  $ (ORBITER) -v 2 \
1979  $ -define F -finite_field -q 25 -end \n1980  $ -define P -projective_space -n 3 -field F -v 0 -end \n1981  $ -with P -do -projective_space_activity \n1982  $ -cheat_sheet \n1983  $ -end
1984  pdflatex PG_3_25.tex
1985  open PG_3_25.pdf
1986
1987
1988
1989
1990  PG_4_3:
1991  $ (ORBITER) -v 2 \
1992  $ -define F -finite_field -q 3 -end \n1993  $ -define P -projective_space -n 4 -field F -v 0 -end \n1994  $ -with P -do -projective_space_activity \n1995  $ -cheat_sheet \n1996  $ -end
1997  pdflatex PG_4_3.tex
1998  open PG_4_3.pdf
1999
2000
2001  PG_8_2:
2002  $ (ORBITER) -v 2 \
2003  $ -define F -finite_field -q 2 -end \n2004  $ -define P -projective_space -n 8 -field F -v 0 -end \n2005  $ -with P -do -projective_space_activity \n2006  $ -cheat_sheet \n2007  $ -end
2008  pdflatex PG_8_2.tex
2009  open PG_8_2.pdf
2010
# Section 4.2: Indexing Points

SECTION INDEXING POINTS:

PG_2.4_rank_point:

-define F -finite_field -q 4 -end \
-with F -do -finite_field_activity \
-rank_point_in_PG 2 "3,3,1" -end

# geometry_global::do_rank_point_in_PG coeff: ( 3, 3, 1 ) has rank 20

elliptic_curve_b1_c3_q11.txt:

-define F -finite_field -q 11 -end \
-define P -projective_space -n 2 -field F -v 0 -end \
-define EC -geometric_object P \
-elliptic_curve 1 3 \
-end \
-with EC -do -combinatorial_object_activity -save \
-end

PG_2.2_incidence_matrix:

-define F -finite_field -q 2 -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 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:

```bash
$\textsc{ORBITER} -v 2 \\\n\textbf{-define F -finite_field -q 4 -end} \\\n\textbf{-define P -projective_space -n 2 -field F -v 0 -end} \\\n\textbf{-with P -do -projective_space_activity} \\\n\textbf{-export_point_line_incidence_matrix} \\\n\textbf{-end}
```

```bash
$\textsc{ORBITER} -v 2 \\\n\textbf{-define all_one -vector -repeat 1 21 -end} \\\n\textbf{-draw_matrix} \\\n\textbf{-input_csv_file PG\_n2\_q4\_incidence\_matrix.csv} \\\n\textbf{-box.width 20 -bit.depth 8} \\\n\textbf{-partition 3} \\\n\textbf{-all_one all_one} \\\n\textbf{-end}
```

```bash
open PG\_n2\_q4\_incidence\_matrix\_draw.bmp
```

```bash
# writes PG\_n2\_q4\_incidence\_matrix.csv
```

PG_2.8.incidence_matrix:

```bash
$\textsc{ORBITER} -v 2 \\\n\textbf{-define F -finite_field -q 8 -end} \\\n\textbf{-define P -projective_space -n 2 -field F -v 0 -end} \\\n\textbf{-with P -do -projective_space_activity} \\\n\textbf{-export_point_line_incidence_matrix} \\\n\textbf{-end}
```

```bash
$\textsc{ORBITER} -v 2 \\\n\textbf{-define all_one -vector -repeat 1 73 -end} \\\n\textbf{-draw_matrix} \\\n\textbf{-input_csv_file PG\_n2\_q8\_incidence\_matrix.csv} \\\n\textbf{-box.width 20 -bit.depth 8} \\\n\textbf{-partition 3} \\\n\textbf{-all_one all_one} \\\n\textbf{-end}
```

```bash
open PG\_n2\_q8\_incidence\_matrix\_draw.bmp
```

# Section 4.3: Finite Desarguesian Projective Planes
SECTION_Finite_Desarguesian_Projective_Planes:

PG_2_16:

```
$\text{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 \
-define F -finite_field -q 4 -end \
.define P -projective_space -n 2 -field F -v 0 -end \
.with P -do -projective_space_activity \
-define F -finite_field -q 4 -end \
.define P -projective_space -n 2 -field F -v 0 -end \
.with P -do -projective_space_activity \
-define F -finite_field -q 4 -end \
.define P -projective_space -n 2 -field F -v 0 -end \
.with P -do -projective_space_activity \
-define F -finite_field -q 4 -end \
.define P -projective_space -n 2 -field F -v 0 -end \
.with P -do -projective_space_activity \
-define F -finite_field -q 4 -end \
.define P -projective_space -n 2 -field F -v 0 -end \
.with P -do -projective_space_activity \
```

PG_2_4_with_decomposition:

```
$\text{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 \
-define F -finite_field -q 4 -end \
.define P -projective_space -n 2 -field F -v 0 -end \
.with P -do -projective_space_activity \
.define F -finite_field -q 4 -end \
.define P -projective_space -n 2 -field F -v 0 -end \
.with P -do -projective_space_activity \
.define F -finite_field -q 4 -end \
.define P -projective_space -n 2 -field F -v 0 -end \
.with P -do -projective_space_activity \
```

PG_2_4_incma_cyclic:

```
$\text{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 \
```

PG_2_4_singer:

```
$\text{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 \
```
# Section 4.4: The Grassmannian

SECTION_GRASSMANNIAN:

GR_3_2_2:

\$(ORBITER) -v 2 \n
-define F -finite_field -q 2 -end \
-with F -do -finite_field_activity \
-cheat_sheet_Gr 3 2 -end \
pdflatex Gr_3_2_2.tex
rank_lines:

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

planes_in_pencil:

```bash
$ (ORBITER) -v 2 \
  -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  -planes_through_line 0 \n  -end
```

# Section 4.5: Algebraic Sets

SECTION_ALGEBRAIC_SETS:

EC_11.txt:

```bash
$ (ORBITER) -v 2 \
  -define F -finite_field -q 11 -end
```
define R -polynomial_ring 
  -field F
-define P -projective_space 
  -n 2 -field F 
-define EC -geometric_object P 
-define EC11 -projective_variety R 
end 
define P -projective_space 
  -n 3 -field F 
-define H16 -geometric_object P 
-end 
with P -do -combinatorial_object_activity -save 
end 

Hirschfeld_surface.q4.txt:
$ORBITER -v 2 
#define F -finite_field -q 4 -end 
#define R -polynomial_ring 
  -field F 
-number_of_variables 4 
-homogeneous_of_degree 3 
end 
#define P -projective_space 
  -n 3 -field F 
-number_of_variables 3 
homogeneous_of_degree 3 
end 
define H4 -geometric_object P 
-projective_variety R 
"Hirschfeld_surface.q4" 
"Hirschfeld_surface.q4" 
$(HIRSCHFELD_SURFACE_EQUATION) 
end 
with H4 -do -combinatorial_object_activity -save 
end 

Hirschfeld_surface.q16.txt:
$ORBITER -v 2 
#define F -finite_field -q 16 -end 
#define R -polynomial_ring 
  -field F 
-number_of_variables 4 
-homogeneous_of_degree 3 
end 
#define P -projective_space 
  -n 3 -field F 
-number_of_variables 3 
homogeneous_of_degree 3 
end 
define H16 -geometric_object P 
-projective_variety R 
"Hirschfeld_surface.q16" 
"Hirschfeld_surface.q16" 

# creates Hirschfeld_surface.q4.txt 
Hirschfeld_surface.q16.txt:
# the coefficient vector is given as a list of pairs.

# 165 = \text{binomial}(11,3)

Endrass\_F7.txt:

```plaintext
> $(ORBITER) -v 2 \\
> -define F -finite_field -q 7 -end \\
> -define R -polynomial_ring -field F \\
> -number_of_variables 4 \\
> -homogeneous_of_degree 8 \\
> -end \\
> -define eqn -vector -field F -sparse 165 \\
> $(ENDRASS\_SPARSE) -end \\
> -define P -projective_space -n 3 -field F -v 0 -end \\
> -define Endrass\_F7 -geometric_object P \\
> -projective_variety R \\
> "Endrass\_F7" \\
> "Endrass\_F7" \\
> eqn \\
> -end \\
> -with Endrass\_F7 -do \\
> -combinatorial_object_activity -save \\
> -end
```

# we created a set of 33 points, called Endrass\_F7.txt

```
```

octic\_prepare:

```plaintext
> $(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 \\
```
# Section 4.6: The Klein Quadric and Plücker coordinates

```
#diophant_activity -solve_mckay
#end
sort -r octic_monomials.sol >octic_monomials_sorted.txt
#Found 165 solutions with 210 backtrack steps
# 165=binomial(11,3)
```

---

# Section 4.7: Orthogonal spaces

```
SECTION_ORTHOGONAL_SPACES:
Op_2:
$(ORBITER) -v 2
#define F -finite_field -q 2 -end
#define O -orthogonal_space 1 4 F -without_group -end
#define O -orthogonal_space_activity
#define O -cheat_sheet.orthogonal -end
pdflatex 0_1_4_2_report.tex
open 0_1_4_2_report.pdf
```

---

```
0_5_2_incidence_matrix.csv:
$(ORBITER) -v 2
```

---

```
0_5_2_incidence_matrix.csv:
```
-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  

# recall that $W(3,2)$ and $Q(4,q)$ are self dual if $q$ is even

Op_6_2:

$(ORBITER) -v 2 \  
-define F -finite_field -q 2 -end \  
-define O -orthogonal_space 1 6 F -without_group -end \  
-with 0 -do -orthogonal_space_activity \  
-cheat_sheet_orthogonal -end  

pdflatex 0_1_6_2.report.tex  

open 0_1_6_2.report.pdf

Op_6_2.incidence_matrix.csv:

$(ORBITER) -v 2 \  
-define F -finite_field -q 2 -end \  
-define O -orthogonal_space 1 6 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 35 -end \  
-define all_one_c -vector -repeat 1 105 -end \  
-draw_matrix \  
-input_csv_file Op_6_2.incidence_matrix.csv \  
-box_width 20 -bit_depth 8 \  
-partition 2 \  
-all_one_r all_one_c \  
-end  

open Op_6_2.incidence_matrix_draw.bmp
Op_6.2: with group:
$\text{(ORBITER) -v 2 \-define F -finite_field -q 2 -end \-define 0 -orthogonal_space 1 6 F -end \-with 0 -do -orthogonal_space_activity \-cheat_sheet_orthogonal -end}
\text{pdflatex 0.1_6.2_report.tex}
\text{open 0.1_6.2_report.pdf}

# problem:
# error message:
\text{stabilizer_chain_base_data::allocate_base_data degree is too large}

Op_6.8:
$\text{(ORBITER) -v 2 \-define F -finite_field -q 8 -end \-define 0 -orthogonal_space 1 6 F -without_group -end \-with 0 -do -orthogonal_space_activity \-cheat_sheet_orthogonal \-end}
\text{pdflatex 0.1_6.8_report.tex}
\text{open 0.1_6.8_report.pdf}

Op_8.2:
$\text{(ORBITER) -v 2 \-define F -finite_field -q 2 -end \-define 0 -orthogonal_space 1 8 F -without_group -end \-with 0 -do -orthogonal_space_activity \-cheat_sheet_orthogonal \-end}
\text{pdflatex 0.1_8.2_report.tex}
\text{open 0.1_8.2_report.pdf}

Op_6.64:
$\text{(ORBITER) -v 2 \-define F -finite_field -q 64 -end \-define 0 -orthogonal_space 1 6 F -without_group -end \-with 0 -do -orthogonal_space_activity \-cheat_sheet_orthogonal \-end}
\text{pdflatex 0.1_6.64_report.tex}
\text{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 \\
\text{define } F \text{ -finite_field -q 64 -end } \\
\text{define } O \text{ -orthogonal_space 1 6 F -end } \\
\text{with } O \text{ -do -orthogonal_space_activity } \\
\text{unrank_line_through_two_points 15447347 15225451 } \\
\text{end}
```

# use option -without_group to skip the group. This will work:

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

# this will create a basic report without the group:

```
Op_6_64_report:
$\text{(ORBITER)} -v 4 \\
\text{define } F \text{ -finite_field -q 64 -end } \\
\text{define } O \text{ -orthogonal_space 1 6 F -without_group -end } \\
\text{with } O \text{ -do -orthogonal_space_activity } \\
\text{unrank_line_through_two_points 15447347 15225451 } \\
\text{end}
```

```
pdflatex O_1_6_64_report.tex
open O_1_6_64_report.pdf
```

# Section 4.8: Hermitian varieties

```
SECTION_HERMITIAN_VARIETIES:
```

```
H_2_4:
$\text{(ORBITER)} -v 2 \\
\text{define } F \text{ -finite_field -q 4 -end } \\
\text{with } F \text{ -do -finite_field_activity } \\
\text{cheat_sheet_hermitian 2 -end}
```

550
H₂₉:

$(\text{ORBITER}) -v 2 \ \
-define \text{F} -\text{finite\_field} -q 9 -end \ 
-\text{with } F -\text{do} -\text{finite\_field\_activity} \ 
-\text{cheat\_sheet\_hermitian} 2 -end \ 
\text{pdflatex H₂₉.tex} \ 
\text{open H₂₉.pdf} \ 
\text{pdflatex H₂₉.tex} \ 
\text{open H₂₉.pdf} \ 

# 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

H₃₄:

$\text{(ORBITER)} -v 2 \ 
-define \text{F} -\text{finite\_field} -q 4 -end \ 
-\text{with } F -\text{do} -\text{finite\_field\_activity} \ 
-\text{cheat\_sheet\_hermitian} 3 -end \ 
\text{pdflatex H₃₄.tex} \ 
\text{open H₃₄.pdf} \ 
\text{pdflatex H₃₄.tex} \ 
\text{open H₃₄.pdf} \ 

# H₃₄ = the Hirschfeld surface

# Section 4.9: Projective Space Advanced Topics

SECTION_PROJECTIVE_SPACE_ADVANCED_TOPICS:

fix\_structure\_2A:

$\text{(ORBITER)} -v 2 \ 
-define \text{F} -\text{finite\_field} -q 4 -end \ 
-define \text{P} -\text{projective\_space} -n 3 -\text{field } F -v 0 -end \ 
-\text{with } P -\text{do} \ 
-\text{projective\_space\_activity} \ 
-\text{cheat\_sheet\_for\_decomposition\_by\_element\_PG 1} \ 
-\text{"1,0,0,0, 0,1,0,0, 0,0,1,0, 0,0,0,1, 1"} \ 
-\text{fix\_structure\_2A} \ 
-\text{end}
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, 1,1,0,0, 0,0,1,0, 0,0,0,1, 0" \%
  fix_structure_2A \%
  -end \%
\%(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_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, 0,0,0,1, 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 \%

del_Pezzo_F13ab_report:
\$(ORBITER) -v 3 \n-define F -finite_field -q 13 -end \n-define P -projective_space -n 3 -field F -v 0 -end \n-define f3 -formula \n"del\_Pezzo\_F13a" "x,y,z" \n"x*x*x*x*y*y*y*y+z*z+z+8*x*x*y*y+8*x*x*z+z+8*y*y*z*z" \n-define f4 -formula \n"del\_Pezzo\_F13b" "x,y,z" \n"x*x*x*x+y*y*y*y+z*z+z-x*x*y*y" \n-define del\_Pezzo13 -collection "f3,f4" \n-with P -do \n-projective_space_activity \n-analyze\_del\_Pezzo\_surface del\_Pezzo13 '' \n-end \npdflatex del\_Pezzo\_F13a_report.tex \npdflatex del\_Pezzo\_F13b_report.tex \nopen del\_Pezzo\_F13a_report.pdf \nopen del\_Pezzo\_F13b_report.pdf

del\_Pezzo\_F13a_points.txt:
\$(ORBITER) -v 3 \n-define F -finite_field -q 13 -end \n-define P -projective_space -n 3 -field F -v 0 -end \n-define f1 -formula "del\_Pezzo\_F9" \n"del\_Pezzo\_F9" "x,y,z" \n"x*x*x*x+y*y*y*y+z*z+z" \n-define f2 -formula "del\_Pezzo\_F11" \n"del\_Pezzo\_F11" "x,y,z" \n"x*x*x*x+y*y*y*y+z*z+z+x*x*y*y+x*x*z+z+y*y*z*z" \n-define f3 -formula "del\_Pezzo\_F13a" \n"del\_Pezzo\_F13a" "x,y,z" \n"x*x*x*x+y*y*y*y+z*z+z+8*x*x*y*y+8*x*x*z+z+8*y*y*z*z" \n-define f4 -formula "del\_Pezzo\_F13b" \n"del\_Pezzo\_F13b" "x,y,z" \n"x*x*x*x+y*y*y*y+z*z+z-x*x*y*y" \n-define del\_Pezzo9 -collection "f1" \n-define del\_Pezzo11 -collection "f2" \n-define del\_Pezzo13 -collection "f3,f4" \n-with P -do \n-projective_space_activity \n-analyze\_del\_Pezzo\_surface del\_Pezzo13 '' \n-end \npdflatex del\_Pezzo\_F13a_report.tex \npdflatex del\_Pezzo\_F13b_report.tex \nopen del\_Pezzo\_F13a_report.pdf \nopen del\_Pezzo\_F13b_report.pdf
#dot -Tpng del\_Pezzo\_F13a.gv >del\_Pezzo\_F13a.png

553
# Section 4.10: Geometric Objects

SECTION GEOMETRIC OBJECTS:

elliptic_quadric_ovoid_q8:

$\text{(ORBITER)} -v 2$

$\text{define } F \text{ finite field } -q 8 \text{ end}$

$\text{define } P \text{ projective space } -n 3 \text{ field } F \text{ v 0 end}$

$\text{define } O \text{ geometric object } P$

$\text{define } O \text{ elliptic quadric ovoid}$

$\text{with } O \text{ do } \text{combinatorial object activity } -\text{save}$

# ovoid_q8.txt

##############################################################################

#ovoid_q8.txt
2715 # 65 points
2716
2717 ovoid_ST_q8:
2718 ▷ $(ORBITER) -v 2 \n2719 ▷ ▷ -define F -finite_field -q 8 -end \n2720 ▷ ▷ -define P -projective_space -n 3 -field F -v 0 -end \n2721 ▷ ▷ -define O -geometric_object P \n2722 ▷ ▷ ▷ -ovoid_ST \n2723 ▷ ▷ -end \n2724 ▷ ▷ -with 0 -do -combinatorial_object_activity -save \n2725 ▷ ▷ -end
2726
2727 # ovoid_ST_q8.txt
2728
2729
2730 Baer_PG_2.4:
2731 ▷ $(ORBITER) -v 2 \n2732 ▷ ▷ -define F -finite_field -q 4 -end \n2733 ▷ ▷ -define P -projective_space -n 2 -field F -v 0 -end \n2734 ▷ ▷ -define O -geometric_object P \n2735 ▷ ▷ ▷ -Baer_substructure \n2736 ▷ ▷ -end \n2737 ▷ ▷ -with 0 -do -combinatorial_object_activity -save \n2738 ▷ ▷ -end
2739
2740 Baer_PG_3.4:
2741 ▷ $(ORBITER) -v 2 \n2742 ▷ ▷ -define F -finite_field -q 4 -end \n2743 ▷ ▷ -define P -projective_space -n 3 -field F -v 0 -end \n2744 ▷ ▷ -define O -geometric_object P \n2745 ▷ ▷ ▷ -Baer_substructure \n2746 ▷ ▷ -end \n2747 ▷ ▷ -with 0 -do -combinatorial_object_activity -save \n2748 ▷ ▷ -end
2749
2750 BLT_database_5.0:
2751 ▷ $(ORBITER) -v 2 \n2752 ▷ ▷ -define F -finite_field -q 5 -end \n2753 ▷ ▷ -define P -projective_space -n 4 -field F -v 0 -end \n2754 ▷ ▷ -define O -geometric_object P \n2755 ▷ ▷ ▷ -BLT_database 0 \n2756 ▷ ▷ -end \n2757 ▷ ▷ -with 0 -do -combinatorial_object_activity -save \n2758 ▷ ▷ -end
2759
2760 # writes BLT_5.0.txt
2761

555
BLT_database_7.0:

```
$ (ORBITER) -v 2 \n(define F finite_field -q 7 -end \n(define P projective_space -n 4 -field F -v 0 -end \n(define O geometric_object P \n(with O -do combinatorial_object_activity -save \n-end \n
# writes BLT_7.0.txt
```

BLT_database_7.1:

```
$ (ORBITER) -v 2 \n(define F finite_field -q 7 -end \n(define P projective_space -n 4 -field F -v 0 -end \n(define S geometric_object P \n(with S -do combinatorial_object_activity -save \n-end \n(define S set -file orbiter_format BLT_7.1.txt -end \n(with O -do orthogonal_space_activity \n-print_points S -end \n(ORBITER) -v 2 \n(define F finite_field -q 7 -end \n(define O orthogonal_space 0 5 F -without_group -end \n(define S set -file orbiter_format BLT_7.1.txt -end \n(with O -do orthogonal_space_activity \n-print_points S -end \n-pdflatex S_set_report.tex \n-open S_set_report.pdf
```

BLT_database_7.1_print:

```
$ (ORBITER) -v 2 \n(define F finite_field -q 7 -end \n(define O orthogonal_space 0 5 F -without_group -end \n(define S set -file orbiter_format BLT_7.1.txt -end \n(with O -do orthogonal_space_activity \n-print_points S -end \n-pdflatex S_set_report.tex \n-open S_set_report.pdf
```

BLT_database_67.4:

```
$ (ORBITER) -v 2 \n(define F finite_field -q 67 -end \n(define P projective_space -n 4 -field F -v 0 -end \n(define Obj geometric_object P \n(with Obj -do combinatorial_object_activity -save 
```
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 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 \\
# the output defines doily.csv

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 \\
-end \\

557
2856  ▶  ▶ -print_symbols
2857  ▶  $(ORBITER) -v 2 \%
2858  ▶  ▶ union doily.csv doily_cliques.txt doily_cliques_union.csv
2859
2860  # 80 cliques
2861
2862  Doily_disjoint_sets_graph_cliques_5:
2863  ▶  echo $(DOILY) >doily.csv
2864  ▶  $(ORBITER) -v 2 \%
2865  ▶  ▶ define G -disjoint_sets_graph \%
2866  ▶  ▶  ▶ doily.csv \%
2867  ▶  ▶ -end \%
2868  ▶  ▶ -with G -do \%
2869  ▶  ▶  ▶  ▶ graph_theoretic_activity \%
2870  ▶  ▶  ▶  ▶ -find_cliques \%
2871  ▶  ▶  ▶  ▶  ▶ target_size 5 \%
2872  ▶  ▶  ▶  ▶  ▶ -output_file doily_cliques_5 \%
2873  ▶  ▶  ▶ -end \%
2874  ▶  ▶ -end \%
2875  ▶  ▶ -print_symbols
2876
2877  # 6 cliques
2878  # doily_cliques_5.csv
2879
2880
2881  PG_3_2_test:
2882  ▶  $(ORBITER) -v 2 \%
2883  ▶  ▶ define F -finite_field -q 2 -end \%
2884  ▶  ▶ -define P -projective_space -n 3 -field F -v 0 -end \%
2885  ▶  ▶ -with P -do -projective_space_activity \%
2886  ▶  ▶  ▶ -cheat_sheet \%
2887  ▶  ▶ -end
2888  ▶  pdflatex PG_3_2.tex
2889  ▶  open PG_3_2.pdf
2890
2891
2892  Edge_curve_17:
2893  ▶  $(ORBITER) -v 2 \%
2894  ▶  ▶ define F -finite_field -q 17 -end \%
2895  ▶  ▶ -define R -polynomial_ring -field F \%
2896  ▶  ▶  ▶ number_of_variables 3 \%
2897  ▶  ▶  ▶ -homogeneous_of_degree 4 \%
2898  ▶  ▶ -end \%
2899  ▶  ▶ -define P -projective_space -n 2 -field F -v 0 -end \%
2900  ▶  ▶ -define C -geometric_object P \%
2901  ▶  ▶  ▶ -projective_variety R \%
2902  ▶  ▶  ▶  ▶ "Edge_q17" "Edge\_q17" \%

558
2903 ▷ ▷ ▷ ▷ $(EDGE_CURVE_Q17_EQUATION) \ 
2904 ▷ ▷ -end \ 
2905 ▷ ▷ -with C -do -combinatorial_object_activity -save \ 
2906 ▷ ▷ -end 
2907
2908 #Edge_q17.txt 
2909 #combinatorial_object.create::init created a set of size 12 
2910 #( 4, 7, 16, 19, 20, 23, 32, 35, 89, 100, 244, 251 ) 
2911
2912
2913
2914
2915
2916
2917
2918 Edge_curve_17_line_type: 
2919 ▷ echo $(FILE_Q17) >edge_q17.csv 
2920 ▷ $(ORBITER) -v 2 \ 
2921 ▷ ▷ -define F -finite_field -q 17 -end \ 
2922 ▷ ▷ -define R -polynomial_ring -field F \ 
2923 ▷ ▷ ▷ -number_of_variables 3 \ 
2924 ▷ ▷ ▷ -homogeneous_of_degree 4 \ 
2925 ▷ ▷ ▷ -end \ 
2926 ▷ ▷ ▷ -define P -projective_space -n 2 -field F -v 0 -end \ 
2927 ▷ ▷ ▷ -define C -geometric_object P \ 
2928 ▷ ▷ ▷ ▷ -projective_variety R \ 
2929 ▷ ▷ ▷ ▷ "Edge_q17" "Edge\_q17" \ 
2930 ▷ ▷ ▷ ▷ $(EDGE_CURVE_Q17_EQUATION) \ 
2931 ▷ ▷ ▷ ▷ -end \ 
2932 ▷ ▷ ▷ -with C -do \ 
2933 ▷ ▷ ▷ -combinatorial_object_activity \ 
2934 ▷ ▷ ▷ ▷ -line_type \ 
2935 ▷ ▷ ▷ -end \ 
2936 ▷ ▷ ▷ -print_symbols 
2937
2938 #( 4^6, 2^30, 1^132, 0^139 ) 
2939
2940
2941 Edge_curve_23_line_type: 
2942 ▷ $(ORBITER) -v 2 \ 
2943 ▷ ▷ -define F -finite_field -q 23 -end \ 
2944 ▷ ▷ -define P -projective_space -n 2 -field F -v 0 -end \ 
2945 ▷ ▷ -define E -geometric_object P \ 
2946 ▷ ▷ ▷ -set $(EDGE_CURVE_Q23_AS_POINTS) \ 
2947 ▷ ▷ ▷ -end \ 
2948 ▷ ▷ ▷ -with E -do \ 
2949 ▷ ▷ ▷ -combinatorial_object_activity \
Section 5.1: Permutation groups

Cyclic_6:

```bash
$ORBITER -v 3
```

```bash
define G -permutation_group -cyclic_group 6 -end
```

```bash
with G -do
```

```bash
-exports_group_theoretic_activity
```

```bash
-report
```

```bash
-end
```

```bash
pdflatex Perm6_report.tex
```

```bash
open Perm6_report.pdf
```

Cyclic_6_group_table:

```bash
$ORBITER -v 3
```

```bash
define G -permutation_group -cyclic_group 6 -end
```

```bash
with G -do
```

```bash
-group_theoretic_activity
```

```bash
-export_group_table
```
Symmetric_3:

Symmetric_3_group_table:

Symmetric_3.elements:
3044 $\$(\textsc{Orbiter}) -v 2 \$
3045 \> -define Sym3_elts -vector -load_csv_data_column \
3046 \> Symmetric3_elts.csv 1 -end \\
3047 \> -save_matrix.csv Sym3_elts \\
3048 $\$(\textsc{Orbiter}) -v 2 \$
3049 \> -define all_one_r -vector -repeat 1 6 -end \\
3050 \> -define all_one_c -vector -repeat 1 3 -end \\
3051 \> -draw_matrix \
3052 \> \> -input_csv_file Sym3_elts_matrix.csv \\
3053 \> \> -box_width 50 -bit_depth 8 \\
3054 \> \> \> -partition 3 \\
3055 \> \> \> \> all_one_r all_one_c \\
3056 \> \> \> -end \\
3057 \> open Sym3_elts_matrix.draw.bmp
3058 
3059 Symmetric_3_long:
3060 $\$(\textsc{Orbiter}) -v 3 \$
3061 \> -define G -permutation_group -symmetric_group 3 -end \\
3062 \> -with G -do \\
3063 \> -group_theoretic_activity \\
3064 \> \> -export_orbiter \\
3065 \> \> -end \\
3066 \> \> -with G -do \\
3067 \> \> -group_theoretic_activity \\
3068 \> \> \> -print_elements.tex \\
3069 \> \> \> -end \\
3070 \> \> \> -with G -do \\
3071 \> \> \> -group_theoretic_activity \\
3072 \> \> \> \> -report \\
3073 \> \> \> \> -end \\
3074 \> \> \> \> -with G -do \\
3075 \> \> \> \> -group_theoretic_activity \\
3076 \> \> \> \> \> -save_elements_csv "Symmetric3_elts.csv" \\
3077 \> \> \> \> -end \\
3078 $\$(\textsc{Orbiter}) -v 3 \$
3079 \> -draw_options \\
3080 \> \> -nodes \\
3081 \> \> \> -embedded -radius 250 \\
3082 \> \> \> -xin 10000 -yin 10000 \\
3083 \> \> \> -xout 1000000 -yout 600000 \\
3084 \> \> \> -scale 0.3 -line_width 1.0 \\
3085 \> \> \> -end \\
3086 \> \> -tree_draw -file Perm3_elements_tree.txt -end
3087 $\$(\textsc{Orbiter}) -v 2 \$
3088 \> -define M -vector -load_csv_data_column \\
3089 \> \> Symmetric3_elts.csv 1 -end \\
3090 \> \> -save_matrix.csv M
3091 $(ORBITER) -v 2 \ 
3092 \> -define all_one_r -vector -repeat 1 6 -end \ 
3093 \> -define all_one_c -vector -repeat 1 3 -end \ 
3094 \> -draw_matrix \ 
3095 \> \> -input_csv_file M_matrix.csv \ 
3096 \> \> -box_width 50 -bit_depth 8 \ 
3097 \> \> -partition 3 \ 
3098 \> \> \> all_one_r all_one_c \ 
3099 \> \> -end \ 
3100 \> pdflatex Perm3_elements_tree_draw.tex \ 
3101 \> open Perm3_elements_tree_draw.pdf \ 
3102 \> #pdflatex Perm3_report.tex \ 
3103 \> #open Perm3_report.pdf \ 
3104 \ 
3105 \ 
3106 Symmetric_4: \ 
3107 $(ORBITER) -v 3 \ 
3108 \> -define G -permutation_group -symmetric_group 4 -end \ 
3109 \> \> -with G -do \ 
3110 \> \> -group_theoretic_activity \ 
3111 \> \> \> -report \ 
3112 \> \> \> -end \ 
3113 \> pdflatex Perm4_report.tex \ 
3114 \> open Perm4_report.pdf \ 
3115 \ 
3116 \ 
3117 Symmetric_4_group_table: \ 
3118 $(ORBITER) -v 3 \ 
3119 \> -define G -permutation_group -symmetric_group 4 -end \ 
3120 \> \> -with G -do \ 
3121 \> \> -group_theoretic_activity \ 
3122 \> \> \> -export_group_table \ 
3123 \> \> \> -end \ 
3124 $(ORBITER) -v 2 \ 
3125 \> -define all_one_r -vector -repeat 1 24 -end \ 
3126 \> -define all_one_c -vector -repeat 1 24 -end \ 
3127 \> -draw_matrix \ 
3128 \> \> -input_csv_file Perm4_group_table.csv \ 
3129 \> \> -box_width 50 -bit_depth 24 \ 
3130 \> \> -partition 3 all_one_r all_one_c \ 
3131 \> \> -end \ 
3132 \> open Perm4_group_table_draw.bmp \ 
3133 \ 
3134 \ 
3135 \ 
3136 Symmetric_4_long: \ 
3137 $(ORBITER) -v 3 \ 

563
3138  \>  \>  \>  -define G -permutation_group -symmetric_group 4 -end \\
3139  \>  \>  \>  -with G -do \\
3140  \>  \>  \>  -group_theoretic_activity \\
3141  \>  \>  \>  \>  -export_orbiter \\
3142  \>  \>  \>  -end \\
3143  \>  \>  \>  -with G -do \\
3144  \>  \>  \>  -group_theoretic_activity \\
3145  \>  \>  \>  \>  -export_group_table \\
3146  \>  \>  \>  -end \\
3147  \>  \>  \>  -with G -do \\
3148  \>  \>  \>  -group_theoretic_activity \\
3149  \>  \>  \>  \>  -print_elements.tex \\
3150  \>  \>  \>  -end \\
3151  \>  \>  \>  -with G -do \\
3152  \>  \>  \>  -group_theoretic_activity \\
3153  \>  \>  \>  \>  -report \\
3154  \>  \>  \>  -end \\
3155  \>  \>  \>  -with G -do \\
3156  \>  \>  \>  -group_theoretic_activity \\
3157  \>  \>  \>  \>  -save_elements_csv "Symmetric4_elts.csv" \\
3158  \>  \>  \>  -end \\
3159  \>  \>  \>  -with G -do \\
3160  \>  \>  \>  -group_theoretic_activity \\
3161  \>  \>  \>  \>  -export_inversion_graphs "Symmetric4_inversion_graphs.csv" \\
3162  \>  \>  \>  -end \\
3163  \>  \>  \>  $(ORBITER) -v 2 \\
3164  \>  \>  \>  -draw_options \\
3165  \>  \>  \>  \>  -nodes \\
3166  \>  \>  \>  \>  \>  -embedded -radius 175 \\
3167  \>  \>  \>  \>  \>  -xin 10000 -yin 10000 \\
3168  \>  \>  \>  \>  \>  -xout 1500000 -yout 600000 \\
3169  \>  \>  \>  \>  \>  -scale 0.3 -line_width 1.0 \\
3170  \>  \>  \>  -end \\
3171  \>  \>  \>  -tree_draw -file Perm4_elements_tree.txt -end \\
3172  \>  \>  \>  $(ORBITER) -v 2 -draw_matrix \\
3173  \>  \>  \>  -input_csv_file Perm4_group_table.csv \\
3174  \>  \>  \>  -box_width 50 -bit_depth 24 -end \\
3175  \>  \>  \>  $(ORBITER) -v 2 \\
3176  \>  \>  \>  -define M -vector -load_csv_data_column \\
3177  \>  \>  \>  Symmetric4_elts.csv 1 -end \\
3178  \>  \>  \>  -save_matrix.csv M \\
3179  \>  \>  \>  $(ORBITER) -v 2 \\
3180  \>  \>  \>  -define all_one_r -vector -repeat 1 24 -end \\
3181  \>  \>  \>  -define all_one_c -vector -repeat 1 4 -end \\
3182  \>  \>  \>  -draw_matrix \\
3183  \>  \>  \>  -input_csv_file M_matrix.csv \\
3184  \>  \>  \>  -box_width 50 -bit_depth 8 \\

564
# Section 5.2: Linear Groups

SECTION LINEAR_GROUPS:

PGL_3_2:
$(ORBITER) -v 2
$define F -finite_field -q 2 -end
$define G -linear_group -PGL 3 F -end
-with G -do
-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 -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

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

# created by PGL_4_2_export

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
L₅₃:
\$\text{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 PSL₅₃ report.tex
open PSL₅₃ report.pdf

#PSL(5,3): Order 237783237120 = 121 * 120 * 117 * 108 * 81 * 16

L₄₅:
\$\text{ORBITER} -v 2 \\
-define F -finite_field -q 5 -end \\
-define G -linear_group -PSL 4 F -end \\
-with G -do \\
-group_theoretic_activity \\
-report \\
-end

pdflatex PSL₄₅ report.tex
open PSL₄₅ report.pdf

#PSL(4,5): Order 7254000000

PGL₄₅:
\$\text{ORBITER} -v 2 \\
-define F -finite_field -q 5 -end \\
-define G -linear_group -PGL 4 F -end \\
-with G -do \\
-group_theoretic_activity \\
-report \\
-end

pdflatex PGL₄₅ report.tex
open PGL₄₅ report.pdf
PGGL_3.4:
$(ORBITER) -v 2 
```
$define G linear_group -PGGL 3 4 -end 
$with G -do 
$group_theoretic_activity 
REPORT 
$sylyou 
-END 
```
pdflatex PGGL_3.4_report.tex
open PGGL_3.4_report.pdf

PGGL_3.8:
$(ORBITER) -v 2 
```
$define G linear_group -PGGL 3 8 -end 
```
pdflatex PGGL_3.8_report.tex
open PGGL_3.8_report.pdf

AGL_1.27:
$(ORBITER) -v 2 
```
$define F finite_field -q 27 -end 
$define G linear_group -AGL 1 F -end 
$with G -do 
$group_theoretic_activity 
REPORT 
-END 
```
pdflatex AGL_1.27_report.tex
open AGL_1.27_report.pdf

SP_4.2:
```
$define G linear_group -PGGL 3 4 -end 
$with G -do 
$group_theoretic_activity 
-END 
```
pdflatex SP_4.2_report.tex
open SP_4.2_report.pdf
PSP_{4.4}:
\begin{verbatim}
$\text{(ORBITER)} -v 2 \\
-define F -finite_field -q 4 -end \\
-define G -linear_group -GL 4 F \\
-symplectic_group \\
-end \\
-with G -do \\
-group_theoretic_activity \\
-report \\
-end
\end{verbatim}

PGL{4.4}:
\begin{verbatim}
$\text{(ORBITER)} -v 2 \\
-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}

PGO_{5.2}:
\begin{verbatim}
$\text{(ORBITER)} -v 2 \\
-define F -finite_field -q 2 -end \\
-define G -linear_group -PGO 5 F -end \\
-with G -do \\
-group_theoretic_activity \\
-report \\
-end
\end{verbatim}

PGGO_{5.4}:
\begin{verbatim}
$\text{(ORBITER)} -v 2 \\
-define F4 -finite_field -q 4 -end \\
-define G -linear_group -PGGO 5 F4 -end \\
-with G -do \\
\end{verbatim}
# the following two groups are isomorphic:

```
3456 # the following two groups are isomorphic:

3457 >> >> >> $(_ORBITER) --v 2 \\n3458 >> >> >> -define F -finite_field -q 2 -end \\n3459 >> >> >> -define G -linear_group -PGL 6 F \#symplectic_group \\
3460 >> >> >> -end \\n3461 >> >> >> -with G -do \\n3462 >> >> >> -group_theoretic_activity \\n3463 >> >> >> -report \\
3464 >> >> >> -end \\n3465 >> >> >> 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

PGO_7_2:

$(ORBITER) -v 2 \n-define F -finite_field -q 2 -end \n-define G -linear_group -PGO 7 F -end \n-with G -do 
-group_theoretic_activity \n-report \n-end

pdflatex PGO_7_2_report.tex
open PGO_7_2_report.pdf

# group order 1451520, acting on 63 points

##############################################################################
# Section 5.3: Subgroups

SECTION_SUBGROUPS:

C13:

$(ORBITER) -v 2 \n-define gens -vector -dense $(GEN_C13) -end \n-define G -permutation_group \n-bsgs C13 C.{13} 13 13 0 1 \ngens \n-end \n-with G -do 
-group_theoretic_activity \n-export_orbiter \n-end \n-with G -do 
-group_theoretic_activity \n-export_group_table \n-end
C13 as subgroup:
$\Rightarrow$(ORBITER) -v 2 \
$\Rightarrow$ -define G -permutation_group -symmetric_group 13 \
$\Rightarrow$ -subgroup_by_generators C13 13 1 $(\text{GEN.C13})$ -end \
$\Rightarrow$ -with G -do \
$\Rightarrow$ -group_theoretic_activity \
$\Rightarrow$ -report \
$\Rightarrow$ -end \
$\Rightarrow$ -with G -do \
$\Rightarrow$ -group_theoretic_activity \
$\Rightarrow$ -save_elements_csv "C13_elts.csv" \
$\Rightarrow$ -end 

J1:
$\Rightarrow$ $(\text{ORBITER})$ -v 2 \
$\Rightarrow$ -define G -linear_group -PGL 7 11 -Janko1 -end \
$\Rightarrow$ -with G -do \
$\Rightarrow$ -group_theoretic_activity \
$\Rightarrow$ -save_elements_csv "C13_elts.csv" \
$\Rightarrow$ -end 

#pdflatex Perm13_Subgroup_C13_13_report.tex
#open Perm13_Subgroup_C13_13_report.pdf

572
PGL_3.11_singer:
$p(ORBITER) -v 2 \$
$\text{-define G -linear_group -PGL 3 11 -singer 19 -end }$
$\text{-with G -do }$
$\text{-group_theoretic_activity }$
$\text{-report }$
$\text{-end}$

PGL_3.11_singer_and_frobenius:
$p(ORBITER) -v 2 \$
$\text{-define G -linear_group -PGL 3 11 -singer_and_frobenius 19 -end }$
$\text{-with G -do }$
$\text{-group_theoretic_activity }$
$\text{-report }$
$\text{-end}$

PG_2.4_order_21:
$p(ORBITER) -v 2 \$
$\text{-define G -linear_group -PGL 3 4 -end }$
$\text{-with G -do }$
$\text{-group_theoretic_activity }$
$\text{-search_element_of_order 21 }$
$\text{-end}$

quaternion:
$p(ORBITER) -v 2 \$
$\text{-define G -linear_group -SL 2 3 }$
$\text{-subgroup_by_generators "quaternion" "8" 3 }$
$\text{-"1,1,1,2, 2,1,1,1, 0,2,1,0" }$
$\text{-end }$
$\text{-with G -do }$
$\text{-group_theoretic_activity }$
$\text{-print_elements.tex }$
$\text{-group_table }$
$\text{-report }$
cube_group:

$(ORBITER) -v 2 \
-define gens -vector -dense \
"0,1,0,2,0,0,0,1, \
0,0,1,0,1,0,2,0,0, \ 
2,0,0,0,1,0,0,0,1" \
-end \\
 define G -linear group -GL 3 3 \\
-subgroup_by_generators "cube" "48" 3 \\
-gens \\
-end \\
-with G -do \\
-group_theoretic_activity \\
-print_elements.tex \\
-report \\
-end


tetra_group:

$(ORBITER) -v 3 \
-define G -linear group -GL 3 3 \\
-subgroup_by_generators "tetra" "12" 2 \\
"0,1,0,0,0,1,1,0,0, 0,0,1,2,0,0,0,2,0" \\
-end \\
-with G -do \\
-group_theoretic_activity \\
-print_elements.tex \\
-report \\
-end
Hesse_group:

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

Weyl_E8:

# Hesse group:

test_subgroup:

```
$\text{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:

```
$\text{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
```

coset_reps_read:

```
$\text{ORBITER} -v 2 \
-define F -finite_field -q 2 -end \
-define G1 -linear_group -PGOp 6 F -end \
-define G2 -linear_group -PGL 6 F \
symplectic_group \
-end \
-define CR -vector_ge -action G2 \
-read_csv \
PGOp_6_2_coset_reps.csv Element \
-end
```

# Section 5.4: Linear Groups, Advanced Topics
SECTION_LINEAR_GROUPS_ADVANCED_TOPICS:

U_3_3:

```bash
$ (ORBITER) -v 3 \n  -define F -finite_field -q 9 -override_polynomial "17" -end \n  -define G -linear_group -PGL 3 F \n  -subgroup_by_generators "U_3_3" "6048" 2 \n  "1,6,4, 5,0,6, 8,5,1, \n  6,2,1, 7,8,4, 0,6,6" \n  -end \n  -with G -do \n  -group_theoretic_activity \n  -report \n  -end
```

PGL_2_3:

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

```
#Co3 from Conway et al., 1985 (ATLAS)  
#order = 495766656000  
#Co3 from the paper by Suleiman and Wilson 1997
```

Co3:

```bash
$ (ORBITER) -v 2 \n```
3795  ▶▶ -define F -finite_field -q 2 -end \ 
3796  ▶▶ -define g1 -vector -field F -format 22 -compact $(CONWAY_GEN1) -end \ 
3797  ▶▶ -define g2 -vector -field F -format 22 -compact $(CONWAY_GEN2) -end \ 
3798  ▶▶ -define G -linear_group -PGL 22 2 \ 
3799  ▶▶ -subgroup_by_generators "Co3" "495766656000" 2 gens \ 
3800  ▶▶ ▶ ▶  -end \ 
3801  ▶▶ ▶ ▶  -with G -do \ 
3802  ▶▶ ▶ ▶ ▶ -group_theoretic_activity \ 
3803  ▶▶ ▶ ▶ ▶ -report \ 
3804  ▶▶ ▶ ▶ ▶ -end 
3805  ▶▶ ▶ ▶ -pdflatex PGL_22_2(Subgroup_Co3_495766656000_report.tex 
3806  ▶▶ ▶ ▶ open PGL_22_2(Subgroup_Co3_495766656000_report.pdf 
3807  ▶▶ ▶ ▶ # needs a lot of memory to run! 
3808  
3809  # needs a lot of memory to run! 
3810  
3811  Ree_27: 
3812  ▶ $(ORBITER) -v 2 \ 
3813  ▶ ▶ -define F -finite_field -q 27 -override_polynomial "34" -end \ 
3814  ▶ ▶ -define g1 -vector -field F -format 7 -dense $(Ree_gen1) -end \ 
3815  ▶ ▶ -define g2 -vector -field F -format 7 -dense $(Ree_gen2) -end \ 
3816  ▶ ▶ -define gens -vector -concatenate g1 -concatenate g2 -end \ 
3817  ▶ ▶ -define G -linear_group -PGL 7 F \ 
3818  ▶ ▶ ▶ -subgroup_by_generators "Ree_27" "10073444472" 2 gens \ 
3819  ▶ ▶ ▶ ▶  -end \ 
3820  ▶ ▶ ▶ ▶ -with G -do \ 
3821  ▶ ▶ ▶ ▶ ▶ -group_theoretic_activity \ 
3822  ▶ ▶ ▶ ▶ ▶ -report \ 
3823  ▶ ▶ ▶ ▶ ▶ -end 
3824  ▶ ▶ ▶ ▶ # needs a lot of memory to run! 
3825  
3826  
3827  
3828  # Section 5.5: Induced Actions 
3829  
3830  
3831  SECTION_INDUCTED_ACTIONS: 
3832  
3833  Symmetric_4_on_pairs: 
3834  ▶ $(ORBITER) -v 3 \ 
3835  ▶ ▶ -define G -permutation_group -symmetric_group 4 -end \ 
3836  ▶ ▶ -define G_on_2subsets -modified_group -from G \ 
3837  ▶ ▶ ▶ -on_k_subsets 2 \ 
3838  ▶ ▶ ▶  -end \ 
3839  ▶ ▶ ▶ -with G_on_2subsets -do \ 
3840  ▶ ▶ ▶ ▶ -group_theoretic_activity \ 
578
T3 on tensors:

$(\texttt{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 

pdflatex GL_{2 2 wreath Sym3 report.tex}

open GL_{2 2 wreath Sym3 report.pdf}

T3r1:

$(\texttt{ORBITER}) \ -v \ 4 \$

-define G \n
-linear_group -GL_{d q wr Sym n 2 2 3} \n
-on_rank_one_tensors -end \n
-with G -do \n
-group_theoretic_activity \n
-report \n
-end 

pdflatex GL_{2 2 wreath Sym3 report.tex}

open GL_{2 2 wreath Sym3 report.pdf}

T4 on tensors:

$(\texttt{ORBITER}) \ -v \ 4 \$

-define G \n
-linear_group -GL_{d q wr Sym n 2 2 4} \n
-on_tensors -end \n
-with G -do \n
-group_theoretic_activity \n
-report \n
-end 

pdflatex GL_{2 2 wreath Sym4 report.tex}

open GL_{2 2 wreath Sym4 report.pdf}
T4r1:

```bash
$ (ORBITER) -v 4 \n\ndefine G \nlinear_group -GL_d_q_wr_Sym_n 2 2 4 \non_rank_one_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
```

```
PGGL_2.8_on_conic:
$ (ORBITER) -v 4 \ndefine G \nlinear_group -PGGL 2 8 -PGL2OnConic -end \nwith G -do \ngroup_theoretic_activity \nREPORT \nend
```

```
pdflatex PGGL_2.8_OnConic_2.8_report.tex
open PGGL_2.8_OnConic_2.8_report.pdf
```

```
SURFACE q13
STAB="1,0,0,0,0,12,0,0,0,0,12,0,0,0,0,1, 1,0,0,0,0,12,0,0,0,0,12,0,0,0,0,1, 0,1,0,0,1,0,0,0,0,0,1,0,0,0,0,1"
```

```
surface_q13_stab_on_tritangents_orbits:
$ (ORBITER) -v 30 \ndefine F -finite_field -q 13 -end \ndefine P -projective_space -n 3 -field F -v 0 -end \nwith P -do \nprojective_space_activity \ndefine_surface S -q 13 \narc_lifting "0,1,2,3,43,113" -end \nend \nwith S -do \ncubic_surface_activity \nREPORT \nend
```

580
-with S -do 
-cubic_surface_activity 
-export_tritangent_planes 
-end

$(ORBITER) -v 2 
-define TriP -set -file 
family_Eckardt_q13_a2_b1_tritangent_planes.csv 
-end 

define G -linear -PGL 4 13 -subgroup_by_generators "SURF_STAB" 
24 4 $(SURFACE_q13_STAB) 
-end 

define G_on_planes -modified_group -from G -on_k_subspaces 3 
-end 

define Gr -modified_group -from G_on_planes -restricted_action TriP 
-end 
-with Gr -do 
-group_theoretic_activity 
-report 
-end 

-with Gr -do 
-group_theoretic_activity 
-orbits_on_points 
-stabilizer 
-end

PGL_4_2 wd:

$(ORBITER) -v 12 
-define G -linear_group -PGL 4 2 -wedge_detached -end 
-with G -do 
-group_theoretic_activity 
-report 
-end

pdflatex PGL_4_2_Wedge_4_0_detached_report.tex
open PGL_4_2_Wedge_4_0_detached_report.pdf

PGL_4_2 wd_reverse:

$(ORBITER) -v 12 
-linear_group -PGL 4 2 -wedge_detached -end 
-group_theoretic_activity 

# Section 5.6: Group Theoretic Activities

SECTION_GROUP_THEORETIC_ACTIVITIES:

PGL\(_3\)\(_2\) elements:

\>(ORBITER) -v 5 \n\>(define G -linear_group -PGL 3 2 -end \n\>(with G -do \n\>(group_theoretic_activity \n\>(save_elements_csv "PGL\(_3\)\(_2\) elements.csv" \n\>(end \n
# creates PGL\(_3\)\(_2\) elements.csv

Sym\(_3\) elements:

\>(ORBITER) -v 3 \n\>(define G -permutation_group -symmetric_group 3 -end \n\>(with G -do \n\>(group_theoretic_activity \n\>(print_elements.tex \n\>(end \n
\>(ORBITER) -v 2 \n\>(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 Perm3.elements_tree.txt -end \n\>(pdflatex Perm3.elements_tree_draw.tex \n\>(open Perm3.elements_tree_draw.pdf

Cycle\(_{13}\) power:
4030  $(ORBITER) -v 5 \n4031  \texttt{-define G -permutation\_group -symmetric\_group 13 -end} \n4032  \texttt{-with G -do} \n4033  \texttt{-group\_theoretic\_activity} \n4034  \texttt{-consecutive\_powers} \n4035  \texttt{"1,2,3,4,5,6,7,8,9,10,11,12,0" 13} \n4036  \texttt{-end} \n4037  pdflatex Perm13\_all\_powers.tex \n4038  open Perm13\_all\_powers.pdf \n4039  
4040  
4041  Cycle\_12\_power: \n4042  $(ORBITER) -v 5 \n4043  \texttt{-define G -permutation\_group -symmetric\_group 12 -end} \n4044  \texttt{-with G -do} \n4045  \texttt{-group\_theoretic\_activity} \n4046  \texttt{-consecutive\_powers} \n4047  \texttt{"1,2,3,4,5,6,7,8,9,10,11,0" 12} \n4048  \texttt{-end} \n4049  pdflatex Perm12\_all\_powers.tex \n4050  open Perm12\_all\_powers.pdf \n4051  
4052  
4053  PGL\_3\_4\_singer: \n4054  $(ORBITER) -v 5 \n4055  \texttt{-define G -linear\_group -PGL 3 4 -end} \n4056  \texttt{-with G -do} \n4057  \texttt{-group\_theoretic\_activity} \n4058  \texttt{-find\_singer\_cycle} \n4059  \texttt{-end} \n4060  
4061  
4062  GL\_2\_8\_multiply: \n4063  $(ORBITER) -v 5 \n4064  \texttt{-define G -linear\_group -GL 2 8 -end} \n4065  \texttt{-with G -do} \n4066  \texttt{-group\_theoretic\_activity} \n4067  \texttt{-multiply "0,1,2,3" "4,5,6,7"} \n4068  \texttt{-end} \n4069  pdflatex GL\_2\_8\_mult.tex \n4070  open GL\_2\_8\_mult.pdf \n4071  
4072  
4073  GL\_2\_7\_multiply: \n4074  $(ORBITER) -v 5 \n4075  
4076  
583
define G -linear_group -GL 2 7 -end \n-with G -do \ngroup_theoretic_activity \nmultiply "0,1,2,3" "4,5,6,0" \n-end
pdflatex GL_2_7_mult.tex
open GL_2_7_mult.pdf

GL_2_7_inv:
$\text{ORBITER}) -v 5 \n-define G -linear_group -GL 2 7 -end \n-with G -do \ngroup_theoretic_activity \n-inverse "0,1,2,3" \n-end
pdflatex GL_2_7_inv.tex
open GL_2_7_inv.pdf

GL_2_7_power:
$\text{ORBITER}) -v 5 \n-define G -linear_group -GL 2 7 -end \n-with G -do \ngroup_theoretic_activity \n-raise_to_the_power "0,1,2,3" 2 \n-end
pdflatex GL_2_7_power.tex
open GL_2_7_power.pdf

PGL_3_2_classes:
$\text{ORBITER}) -v 3 \n-define G -linear_group -PGL 3 2 -end \n-with G -do \ngroup_theoretic_activity \n-classes_based_on_normal_form \n-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

#-classes 

PGL_4_2_classes_based_on_normal_form:
$\text{ORBITER}) -v 3 \n
4124 $define G -linear\_group -PGL 4 2 -end$
4125 $with G -do$
4126 $group\_theoretic\_activity$
4127 $classes\_based\_on\_normal\_form$
4128 $end$
4129 latex PGL.4.2_classes_normal_form.tex
4130 open PGL.4.2_classes_normal_form.pdf
4131
4132 PGL.10.2_classes_based_on_normal_form:
4133 $((ORBITER) -v 3$
4134 $define G -linear\_group -PGL 10 2 -end$
4135 $with G -do$
4136 $group\_theoretic\_activity$
4137 $classes\_based\_on\_normal\_form$
4138 $end$
4139 latex PGL.10.2_classes_normal_form.tex
4140 open PGL.10.2_classes_normal_form.pdf
4141
4145 normal_forms_PGL.4.4:
4146 $((ORBITER) -v 7$
4147 $define G -linear\_group -PGL 4 4 -end$
4148 $with G -do$
4149 $group\_theoretic\_activity$
4150 $classes\_based\_on\_normal\_form$
4151 $end$
4152 latex PGL.4.4_classes_normal_form.tex
4153 open PGL.4.4_classes_normal_form.pdf
4154
4158 PGL.4.4.2A_rank:
4159 $((ORBITER) -v 6$
4160 $define G -linear\_group -PGL 4 4 -end$
4161 $with G -do$
4162 $group\_theoretic\_activity$
4163 $element\_rank$
4164 $"1,0,0,0, 0,1,0,0, 0,0,1,0, 0,0,0,1, i"$
4165 $end$
4166
4168 PGL.4.4.2A_unrank:
4169 $((ORBITER) -v 6$
4170 $define G -linear\_group -PGL 4 4 -end$
normal forms $PGL_4\_5$:

```bash
d $(ORBITER) -v 7 
```

```bash
d -define G -linear_group -PGL 4 5 -end 
```

```bash
d -with G -do 
```

```bash
d -group_theoretic_activity 
```

```bash
d -element_rank "0,0,0,1, 2,3,0,1, 0,3,4,4, 0,1,2,1" 
```

```bash
d -end 
```

```bash
PGL_4\_5.Brank: 
```

```bash
d $(ORBITER) -v 6 
```

```bash
d -define G -linear_group -PGL 4 5 -end 
```

```bash
d -with G -do 
```

```bash
d -group_theoretic_activity 
```

```bash
d -element_rank "0,0,0,1, 2,3,0,1, 0,3,4,4, 0,1,2,1" 
```

```bash
d -end 
```

```bash
PGL_4\_5.Bunrank: 
```

```bash
d $(ORBITER) -v 6 
```

```bash
d -define G -linear_group -PGL 4 5 -end 
```

```bash
d -with G -do 
```

```bash
d -group_theoretic_activity 
```

```bash
d -element_rank "701459351" 
```

```bash
d -end 
```

```bash
# related to planes in pencil: 
```

```bash
# we are computing the action on the planes through the line 0. 
```

```bash
on_planes: 
```

586
Section 5.7: Group Theoretic Activities Based on Magma

PGGL₂₄_classes:

$\text{define } G \text{ -linear group -PGL 2 4} \text{-end} \text{with } G \text{-do -group_theoretic_activity -classes}$

pdflatex PGGL₂₄_classes_out.tex
open PGGL₂₄_classes_out.pdf
open PGGL₂₄_classes_out.csv
PGL_7_2_classes:
$\text{ORBITER} -v 3$
\> \text{-define G}
\> \text{-linear_group -PGL 7 2}
\> \text{-end}
\> \text{-with G -do}
\> \text{-group_theoretic_activity}
\> \text{-classes}
\> \text{-end}
$(\text{MAGMA PATH})\text{magma PGL_7_2_classes.magma}$

PGL_8_2_classes:
$\text{ORBITER} -v 3$
\> \text{-define G}
\> \text{-linear_group -PGL 8 2}
\> \text{-end}
\> \text{-with G -do}
\> \text{-group_theoretic_activity}
\> \text{-classes}
\> \text{-end}
$(\text{MAGMA PATH})\text{magma PGL_8_2_classes.magma}$

PGL_10_2_classes:
$\text{ORBITER} -v 3$
\> \text{-define G}
\> \text{-linear_group -PGL 10 2}
\> \text{-end}$
Normalizer of H5:

```
>
$($ORBITER) -v 2 \
> -define G -permutation_group -symmetric_group 13 \
> -subgroup_by_generators H5 5 1 \
> $($GENERATORS_H5) -end \
> -with G -do \
> -group_theoretic_activity \
> -normalizer \
> -end
```

```
Normalizer of H5: $(MAGMA_PATH)magma element_2A_centralizer.magma
```

```
Normalizer of H5: $(ORBITER) -v 3 \
define G -linear_group -PGL 10 2 -end \
-groupp_theoretic_activity \
-centralizer_of_element "2A" "1,0, 0,1, 1" -report -end
```

```
Normalizer of H5: $(ORBITER) -v 6 \
define G -linear_group -PGGL 2 4 -end \
-groupp_theoretic_activity \
-centralizer_of_element "2A" "1,0, 0,1, 1" -report -end
```

```
Normalizer of H5: $(MAGMA_PATH)magma PGL_10_2_classes.magma
```

```
Normalizer of H5: $(ORBITER) -v 3 \
define G -linear_group -PGL 10 2 -end \
group_theoretic_activity \
classes -end
```

```
Normalizer of H5: $(ORBITER) -v 3 \
define G -linear_group -PGL 10 2 -end \
group_theoretic_activity \
classes -end
```

```
Normalizer of H5: $(ORBITER) -v 3 \
define G -linear_group -PGL 10 2 -end \
group_theoretic_activity \
classes -end
```

```
Normalizer of H5: $(ORBITER) -v 3 \
define G -linear_group -PGL 10 2 -end \
group_theoretic_activity \
classes -end
```

```
Normalizer of H5: $(ORBITER) -v 3 \
define G -linear_group -PGL 10 2 -end \
group_theoretic_activity \
classes -end
```

```
Normalizer of H5: $(ORBITER) -v 3 \
define G -linear_group -PGL 10 2 -end \
group_theoretic_activity \
classes -end
```

```
Normalizer of H5: $(ORBITER) -v 3 \
define G -linear_group -PGL 10 2 -end \
group_theoretic_activity \
classes -end
```

```
Normalizer of H5: $(ORBITER) -v 3 \
define G -linear_group -PGL 10 2 -end \
group_theoretic_activity \
classes -end
```

```
Normalizer of H5: $(ORBITER) -v 3 \
define G -linear_group -PGL 10 2 -end \
group_theoretic_activity \
classes -end
```

```
Normalizer of H5: $(ORBITER) -v 3 \
define G -linear_group -PGL 10 2 -end \
group_theoretic_activity \
classes -end
```

```
Normalizer of H5: $(ORBITER) -v 3 \
define G -linear_group -PGL 10 2 -end \
group_theoretic_activity \
classes -end
```

```
Normalizer of H5: $(ORBITER) -v 3 \
define G -linear_group -PGL 10 2 -end \
group_theoretic_activity \
classes -end
```

```
Normalizer of H5: $(ORBITER) -v 3 \
define G -linear_group -PGL 10 2 -end \
group_theoretic_activity \
classes -end
```

```
Normalizer of H5: $(ORBITER) -v 3 \
define G -linear_group -PGL 10 2 -end \
group_theoretic_activity \
classes -end
```

```
Normalizer of H5: $(ORBITER) -v 3 \
define G -linear_group -PGL 10 2 -end \
group_theoretic_activity \
classes -end
```

```
Normalizer of H5: $(ORBITER) -v 3 \
define G -linear_group -PGL 10 2 -end \
group_theoretic_activity \
classes -end
```

```
Normalizer of H5: $(ORBITER) -v 3 \
define G -linear_group -PGL 10 2 -end \
group_theoretic_activity \
classes -end
```
4359  ▶ pdflatex Perm13_Subgroup_H5_5_normalizer.tex
4360  ▶ open Perm13_Subgroup_H5_5_normalizer.pdf
4361
4362
4363
4364
4365  PGGL_3_4_classes:
4366  ▶ \$(ORBITER) -v 3 \
4367  ▶ ▶ ~define G \
4368  ▶ ▶ ~linear_group ~PGGL 3 4 \
4369  ▶ ▶ ~end \
4370  ▶ ▶ ~with G ~do \
4371  ▶ ▶ ~group_theoretic_activity \
4372  ▶ ▶ ▶ ~classes \
4373  ▶ ▶ ~end
4374  ▶ pdflatex PGGL_3_4_classes_out.tex
4375  ▶ open PGGL_3_4_classes_out.pdf
4376
4377
4378
4379
4380
4381
4382
4383  classes_PGGL_4_4:
4384  ▶ \$(ORBITER) -v 3 \
4385  ▶ ▶ ~magma_path $(MAGMA_PATH) \
4386  ▶ ▶ ~define G \
4387  ▶ ▶ ~linear_group ~PGGL 4 4 ~end \
4388  ▶ ▶ ~with G ~do \
4389  ▶ ▶ ~group_theoretic_activity \
4390  ▶ ▶ ▶ ~classes \
4391  ▶ ▶ ~end
4392
4393  # group order 1974067200 = 2^13 * 3^4 * 5^2 * 7 * 17
4394
4395
4396
4397
4398
4399  # the ~find_subgroup command is too specialized
4400
4401  subgroups_PGL_4_5:
4402  ▶ \$(ORBITER) -v 6 \
4403  ▶ ▶ ~define G \
4404  ▶ ▶ ~linear_group ~PGL 4 5 ~end \
4405  ▶ ▶ ~with G ~do \

590
classes_PGL_4_5:

# 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 = 2880 Normalizer order = 5760

# of order 3 and with 6 fixed points.
# 0,0,0,1,2,3,0,1,0,3,4,4,0,1,2,1,

PGL_4_5_3B_class_again:

# OK, we found an irreducible and primitive polynomial X^3 + X^2 + 2
GL_3_5_singer_power:
$ $(ORBITER) -v 6 -define G \\
$ $-linear_group -GL 3 5 -end \\
$ $-with G -do \\
$ $-group_theoretic_activity \\
$ $-raise_to_the_power \\
$ "$0,1,0, 0,0,1, 3,0,4$" 31 \\
$ $-end \\
pdflatex GL_3_5_power.tex \\
open GL_3_5_power.pdf \\
PGL_4_5_norm_31:
$ $(ORBITER) -v 6 -define G \\
$ $-linear_group -PGL 4 5 -end \\
$ $-with G -do \\
$ $-group_theoretic_activity \\
$ $-normalizer_of_cyclic_subgroup "$31" \\
$ "$2,0,0,0, 0,0,0,1, 0,3,0,4$" \\
$ $-end \\
pdflatex normalizer_of_31_in_PGL_4_5.tex \\
open normalizer_of_31_in_PGL_4_5.pdf \\

Normalizer_of_Z22_in_PGL_2_9:
$ $(ORBITER) -v 2 \$
$ $-define G -linear_group -PGL 2 9 \$
$ $-subgroup_by_generators Z22 4 2 \$
$ "$2,0,0,1, 0,1,1,0" -end \$
$ $-normalizer \$
$ $-end \\
pdflatex PGL_2_9_Subgroup_Z22_4_normalizer.tex \\
open PGL_2_9_Subgroup_Z22_4_normalizer.pdf
SECTION_ORBIT_ALGORITHMS_SCHREIER_TREES:

orbits_PGL_4_2.on_points:
-define G -linear_group -PGL 4 2 -end
-with G -do
-group_theoretic_activity
-report
-orbits_on_points
-export_trees
-end

$\text{ORBITER} -v 3$
-draw_layered_graph
-PGL_4_2_0.layered_graph
-radius 500 -spanning_tree -embedded
-line_width 1.1 -x_stretch 1.4 -scale 0.25
-end

\text{pdflatex} \text{PGL}_4_2_0\text{draw.tex}
\text{open} \text{PGL}_4_2_0\text{draw.pdf}

\text{pdflatex} \text{PGL}_4_2_0\text{orbits_report.tex}
\text{open} \text{PGL}_4_2_0\text{orbits_report.pdf}

T3r1.orbits:
$\text{ORBITER} -v 4$
-define G
-linear_group -GL_d_q_wr_Sym_n 2 2 3
-on_rank_one_tensors -end
-with G -do
-group_theoretic_activity
-report
-orbits_on_points
-export_trees
-end

\text{pdflatex} \text{GL}_2_2\text{wreath_Sym3_orbits_report.tex}
\text{open} \text{GL}_2_2\text{wreath_Sym3_orbits_report.pdf}

T3r1.orbits_draw:
$\text{ORBITER} -v 3$
-draw_layered_graph
-GL_2_2\text{wreath_Sym3.res27.0.layered_graph}
-radius 500 -spanning_tree -embedded
-line_width 1.1 -x_stretch 1.4 -scale 0.25
-end

#\text{pdflatex} \text{GL}_2_2\text{wreath_Sym3.report.tex}
#\text{open} \text{GL}_2_2\text{wreath_Sym3.report.pdf}
\text{pdflatex} \text{GL}_2_2\text{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:

>>> $(ORBITER) -v 6 \
>>> > -define G -linear_group -PGGL 4 4 -end \
>>> > > -with G -do \
>>> > > > -group_theoretic_activity \
>>> > > > -conjugacy_class_of_element \
>>> > > > "2C" "1,0,0,0, 1,1,0,0, 0,0,1,0, 0,0,1,1, 0" \
>>> > > > -end 

# 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

# 1:33 on Mac

#User time: 2:59 on Mac

PGGL_4_4_subgroups_of_type_2C_2C: 2C_orbit_under_PGGL_4_4_elements_coded.csv

>>> $(ORBITER) -v 6 \
>>> > -define G -linear_group -PGGL 4 4 \
>>> > > -subgroup_by_generators "centralizer_2C" "30720" 9 \
>>> > > > "1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1,1," \
>>> > > > "1,0,0,0,0,1,0,0,0,0,2,0,0,0,0,2,1,1," 

594
# 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 Classes.compute_all_point_orbits

# found 29 conjugacy classes

# User time: 0:57

orbits_on_conics_q13:
  $(ORBITER) -v 4 \
  -define G -linear_group -PGL 3 13 -end \
  -with G -do \
  -group_theoretic_activity \
  -orbits_on_polynomials 2 \
  -end
  pdflatex poly_orbits_d2.n2.q13.tex
  open poly_orbits_d2.n2.q13.pdf

orbits_cubic_curves_q2:
  $(ORBITER) -v 4 \
  -define G -linear_group -PGL 3 2 -end \
  -with G -do \
  -group_theoretic_activity \
  -orbits_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:
  $(ORBITER) -v 4 \
  -draw_options -yout 500000 -radius 15 -nodes_empty \
  -line_width 0.5 -y_stretch 0.25 -embedded -end \
  -define G -linear_group -PGL 3 2 -end \
  -with G -do \
  -group_theoretic_activity \
  -orbits_on_polynomials 3 \
  -orbits_on_polynomials_draw_tree 6 \
  -end

poly_orbits_d3.n3.q2.csv:
  $(ORBITER) -v 4 \

595
poly_orbits_d3_n3_q2_get_ranks:

T4.orbits:

T4r1.orbits:
T4r1_orbits_draw:

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

\$\text{-draw\_layered\_graph }\$

\$\text{-define }G \quad \text{-linear\_group } \text{-GL}_d_q \quad \text{wr} \quad \text{Sym}_n \ 2 \ 2 \ 4 \$

\$\text{-on\_rank\_one\_tensors }\$

\$\text{-with }G \quad \text{-do }\$

\$\text{-group\_theoretic\_activity }\$

\$\text{-poset\_classification\_control } -\text{-problem\_label } T4r1 -W \$

\$\text{-bit\_depth } 4 \quad \text{-draw\_options }\$

\$\text{-draw\_poset } -\text{-report } -\text{-end }\$

\$\text{-end }\$

\$\text{-orbits\_on\_subsets } 4 \$

\$\text{-report }\$

\$\text{-end }\$

pdflatex T4r1_poset.tex

open T4r1_poset.pdf

pggl_2_8_on_conic_orbits:

\$(\text{ORBITER}) -v 4 \$

\$\text{-define }G \$

\$\text{-linear\_group } \text{-PGGL} \ 2 \ 8 \quad \text{-PGL2OnConic} \quad -\text{-end }\$

\$\text{-with }G \quad \text{-do }\$

\$\text{-group\_theoretic\_activity }\$

\$\text{-orbits\_on\_points}\$

\$\text{-report }\$

\$\text{-end }\$

pdflatex PGGL_2_8_OnConic_2_8_orbits_report.tex

open PGGL_2_8_OnConic_2_8_orbits_report.pdf

open GL_2_2_wreath_Sym4_res81_0_draw.pdf
# example from the Fining manual, page 107:

```
pggl_7_8_orbits:
  ➤ $(ORBITER) -v 4 \\
  ➤ ➤ -define G \\
  ➤ ➤ ➤ -linear_group -PGGL 7 8 -end \\
  ➤ ➤ ➤ -with G -do \\
  ➤ ➤ ➤ -group_theoretic_activity \\
  ➤ ➤ ➤ ➤ -report \\
  ➤ ➤ ➤ ➤ -orbits_on_points \\
  ➤ ➤ ➤ -end

# 1 min 31 sec on Mac
```

# Section 6.2: Poset Classification

```
SECTION_POSET_CLASSIFICATION:

poset_of_4subsets:
  ➤ $(ORBITER) -v 3 \\
  ➤ ➤ -orbiter_path $(ORBITER_PATH) \\
  ➤ ➤ -define G -linear_group -PGL 2 3 -identity_group -end \\
  ➤ ➤ -with G -do \\
  ➤ ➤ -group_theoretic_activity \\
  ➤ ➤ ➤ -poset_classification_control \\
  ➤ ➤ ➤ ➤ -problem_label poset_4 \\
  ➤ ➤ ➤ ➤ -W -depth 4 \\
  ➤ ➤ ➤ ➤ -draw_options -radius 200 -end \\
  ➤ ➤ ➤ ➤ -report -end \\
  ➤ ➤ ➤ ➤ -draw_poset \\
  ➤ ➤ ➤ -end \\
  ➤ ➤ ➤ -orbits_on_subsets 4 \\
  ➤ ➤ ➤ -report \\
  ➤ ➤ -end

pdflatex PGL_2_3_Identity_2_3_report.tex
pdflatex poset_4.poset.tex
open PGL_2_3_Identity_2_3_report.pdf
open poset_4.poset.pdf

poset_of_4subsets_draw:
  ➤ $(ORBITER) -v 3 \\
  ➤ -draw_layered_graph \\
```
V_3_2_trivial:
$\text{orbiter path }$ $(\text{ORBITER}$ $\text{PATH})$
$\text{define G -linear_group -PGL 3 2 -identity_group -end }$
$\text{group_theoretic_activity }$
$\text{poset_classification_control }$
$\text{problem_label V_3_2_trivial }$
$\text{draw_options }$
$\text{draw_poset }$
$\text{orbits_on_subspaces 3 }$
$\text{report }$
$\text{end }$
$\text{orbiter_path }$ $(\text{ORBITER}$ $\text{PATH})$
$\text{define G -linear_group -PGL 4 2 -identity_group -end }$
$\text{with G -do }$
$\text{group_theoretic_activity }$
$\text{poset_classification_control }$
$\text{problem_label V_4_2_trivial }$
$\text{draw_options }$
$\text{draw_poset }$
$\text{orbits_on_subspaces 3 }$
$\text{report }$
SECTION ORBITS ON SUBSETS:

PG_2_2_subsets:

$\text{orbiter -v 3}$

$\text{define G -linear_group -PGL 3 F}$

$\text{with G -do}$

$\text{-group_theoretic_activity}$

$\text{-poset_classification_control}$

$\text{-problem_label PGL_3_2}$

$\text{-depth 7}$

$\text{-draw_options}$

$\text{-radius 200 -embedded}$

$\text{-end}$

$\text{-report -end}$

$\text{-draw_poset}$

$\text{-end}$

$\text{-orbits_on_subspaces 7}$

$\text{-report}$

# Section 6.3: Orbits on Subsets
# PG(3,2) has \(2^3+2^2+2^1+1 = 15\) points:
# PG(3,3) has \(3^3+3^2+3^1+1 = 27 + 9 + 3 + 1 = 40\) points.

PG\(_{3,2}\) subsets:

```bash
$(ORBITER) -v 3 \\
-orbiter_path $(ORBITER_PATH) \\
-define F -finite_field -q 2 -end \\
-define G -linear_group -PGL 4 F -end \\
-with G -do \\
-group_theoretic_activity \\
-poset_classification_control \\
-problem_label PGL_4_2 \\
-depth 15 \\
-draw_options \\
-radius 200 -embedded \\
-end \\
-report -end \\
-draw_poset \\
-end \\
-orbits_on_subsets 15 \\
-report \\
-end
```

dfdlatex PGL_4_2_poset.tex
open PGL_4_2_poset.pdf

PGL\(_{3,2}\) singer:

```bash
$(ORBITER) -v 3 \\
-orbiter_path $(ORBITER_PATH) \\
-define G -linear_group -PGL 3 2 -singer 1 -end \\
-with G -do \\
-group_theoretic_activity \\
-poset_classification_control \\
-problem_label PGL_3_2_singer_1 -W -depth 7 \\
-draw_poset \\
```
PGL\_3\_2 on lines:
\$(\text{ORBITER}) \ -v\ 3 \$
\$(\text{ORBiter\_path}\ $(\text{ORBITER\_PATH}) \$
\$(\text{define}\ G \ -\text{linear\_group} \ -\text{PGL}\ 3\ 2\ -\text{end} \$
\$(\text{define}\ G\ \text{on\_lines} \ -\text{modified\_group} \ -\text{from}\ G \$
\$(\text{on\_k\_subspaces}\ 2 \$
\$(\text{with}\ G\ \text{on\_lines} \ -\text{do} \$
\$(\text{group\_theoretic\_activity} \$
\$(\text{poset\_classification\_control} \$
\$(\text{problem\_label} \text{PGL}\_3\_2\_lines} \ -w\ -\text{depth}\ 7 \$
\$(\text{draw\_poset} \$
\$(\text{report} \ -\text{end} \$
\$(\text{end} \$
\$(\text{orbits\_on\_subsets}\ 7 \$
\$(\text{report} \$

PGL\_2\_5 on subsets:
\$(\text{ORBITER}) \ -v\ 10 \$
\$(\text{orbiter\_path}\ $(\text{ORBITER\_PATH}) \$
\$(\text{define}\ G \ -\text{linear\_group} \ -\text{PGL}\ 2\ 5\ -\text{end} \$
\$(\text{with}\ G \ -\text{do} \$
\$(\text{group\_theoretic\_activity} \$
\$(\text{poset\_classification\_control} \$
\$(\text{problem\_label} \text{PGL}\_2\_5} \ -w\ -\text{depth}\ 6 \$
\$(\text{draw\_poset} \$
\$(\text{draw\_options} \ -\text{radius}\ 200 \ -\text{end} \$
\$(\text{report} \ -\text{end} \$
\$(\text{end} \$
\$(\text{orbits\_on\_subsets}\ 6 \$
\$(\text{report} \$

\text{pdflatex PGL\_3\_2\_singer\_1\_poset.tex}
\text{open PGL\_3\_2\_singer\_1\_poset.pdf}

\text{pdflatex PGL\_3\_2\_lines\_poset.tex}
\text{open PGL\_3\_2\_lines\_poset.pdf}

\text{pdflatex PGL\_2\_5\_lines\_poset.tex}
\text{open PGL\_2\_5\_lines\_poset.pdf}
PGL_2.7_on_subsets:
$$(\text{ORBITER}) -v 10 \backslash$
-\text{orbiter_path } $(\text{ORBITER_PATH}) \backslash$
-\text{define G -linear_group -PGL 2 7 -end } \backslash$
-\text{with G -do } \backslash$
-\text{group_theoretic_activity } \backslash$
-\text{poset_classification_control } \backslash$
-\text{problem_label PGL_2.7 -W -depth 8 } \backslash$
-\text{draw_poset } \backslash$
-\text{draw_options -radius 200 -end } \backslash$
-\text{report -end } \backslash$
-\text{orbits_on_subsets 8 } \backslash$
-\text{report } \backslash$
-\text{end}$

PGGL_2.8_on_subsets:
$$(\text{ORBITER}) -v 10 \backslash$
-\text{orbiter_path } $(\text{ORBITER_PATH}) \backslash$
-\text{define G -linear_group -PGGL 2 8 -end } \backslash$
-\text{with G -do } \backslash$
-\text{group_theoretic_activity } \backslash$
-\text{poset_classification_control } \backslash$
-\text{problem_label PGGL_2.8 -W -depth 9 } \backslash$
-\text{draw_poset } \backslash$
-\text{draw_options -radius 200 -end } \backslash$
-\text{report -end } \backslash$
-\text{orbits_on_subsets 9 } \backslash$
-\text{report } \backslash$
-\text{end}$

PGGL_2.9_on_subsets:
$$(\text{ORBITER}) -v 10 \backslash$
-\text{orbiter_path } $(\text{ORBITER_PATH}) \backslash$
-\text{define G -linear_group -PGGL 2 9 -end } \backslash$
-\text{with G -do } \backslash$
-\text{group_theoretic_activity } \backslash$
-\text{poset_classification_control } \backslash$
5061 \(\text{-problem\_label \textit{PGGL} \_2\_9 \text{-W \text{-depth 10 \ -draw\_poset \ -draw\_options \ -radius 200 \ -end \ -report \ -end \ -end \ -orbits\_on\_subsets 10 \ -report \ -end \ -orbits\_on\_subsets 10 \ -report} \)}

5062 \(\text{pdflatex PGGL\_2\_9\_poset.tex} \)

5063 \(\text{open PGGL\_2\_9\_poset.pdf} \)

5064 PGGL\_2\_11\_on\_subsets:

5065 \(\text{-problem\_label \textit{PGGL} \_2\_11 \text{-W \text{-depth 12 \ -draw\_poset \ -draw\_options \ -radius 200 \ -end \ -report \ -end \ -orbits\_on\_subsets 12 \ -report} \)}

5066 \(\text{pdflatex PGGL\_2\_11\_poset.tex} \)

5067 \(\text{open PGGL\_2\_11\_poset.pdf} \)

5068 PGGL\_2\_16\_on\_subsets:

5069 \(\text{-problem\_label \textit{PGGL} \_2\_16 \text{-W \text{-depth 10 \ -draw\_poset \ -report \ -end \ -orbits\_on\_subsets 10 \ -report \ -end} \)}

5070 \(\text{pdflatex PGGL\_2\_16\_poset.tex} \)
5108 ▶ open PGGL_2_16_poset.pdf
5109
5110
5111 PGGL_2_32_on_subsets:
5112 ▶ $(ORBITER) -v 3 \n5113 ▶ ▶ -orbiter_path $(ORBITER_PATH) \n5114 ▶ ▶ -define G -linear_group -PGGL 2 32 -end \n5115 ▶ ▶ -with G -do \n5116 ▶ ▶ -group_theoretic_activity \n5117 ▶ ▶ ▶ -poset_classification_control \n5118 ▶ ▶ ▶ ▶ -problem_label PGGL_2_32 -W -depth 8 \n5119 ▶ ▶ ▶ ▶ -draw_poset \n5120 ▶ ▶ ▶ ▶ -report -end \n5121 ▶ ▶ ▶ -end \n5122 ▶ ▶ ▶ -orbits_on_subsets 8 \n5123 ▶ ▶ ▶ -report \n5124 ▶ ▶ -end
5125 ▶ pdflatex PGGL_2_32_poset.tex
5126 ▶ open PGGL_2_32_poset.pdf
5127
5128
5129 PG_3_4_subsets:
5130 ▶ $(ORBITER) -v 3 \n5131 ▶ ▶ -orbiter_path $(ORBITER_PATH) \n5132 ▶ ▶ -define G -linear_group -PGGL 4 4 -end \n5133 ▶ ▶ -with G -do \n5134 ▶ ▶ -group_theoretic_activity \n5135 ▶ ▶ ▶ -poset_classification_control \n5136 ▶ ▶ ▶ ▶ -problem_label PGGL_4_4 \n5137 ▶ ▶ ▶ ▶ -depth 5 \n5138 ▶ ▶ ▶ ▶ -draw_poset \n5139 ▶ ▶ ▶ ▶ -draw_options \n5140 ▶ ▶ ▶ ▶ ▶ -radius 200 \n5141 ▶ ▶ ▶ ▶ ▶ -end \n5142 ▶ ▶ ▶ ▶ -report -end \n5143 ▶ ▶ ▶ -end \n5144 ▶ ▶ ▶ -orbits_on_subsets 5 \n5145 ▶ ▶ ▶ -report \n5146 ▶ ▶ -end
5147 ▶ pdflatex PGGL_4_4_poset.tex
5148 ▶ open PGGL_4_4_poset.pdf
5149
5150
5151 PGGL_2_9_orbits:
5152 ▶ $(ORBITER) -v 3 \n5153 ▶ ▶ -orbiter_path $(ORBITER_PATH) \n5154 ▶ ▶ -define G -linear_group -PGGL 2 9 -end \n5155
5156
with G -do \\
group_theoretic_activity \\
- poset_classification_control \\
-problem_label PGGL_2_9 -W -depth 5 \\
-report -end \\
-draw_poset \\
-draw_options -radius 200 -end \\
-end \\
-orbits_on_subsets 5 \\
-report \\
-end \\
 pdflatex PGGL_2_9_poset.tex \\
open PGGL_2_9_poset.pdf \\

PGO_5_2_on_subsets: \\
$ (ORBITER) -v 3 \\
-orbiter_path $(ORBITER_PATH) \\
-define F -finite_field -q 2 -end \\
-define G -linear_group -PGO 5 F -end \\
-with G -do \\
-group_theoretic_activity \\
-poset_classification_control \\
-problem_label PGO_5_2 \\
-depth 15 \\
-report -end \\
-draw_poset \\
-w \\
-end \\
-orbits_on_subsets 15 \\
-report \\
-end \\
pdflatex PGO_5_2_poset.tex \\
open PGO_5_2_poset.pdf \\

# Section 6.4: Orbits on Subspaces \\
SECTION ORBITS ON SUBSPACES: \\

subspaces_{Op.4.2}:

- $(\text{ORBITER}) -v 5 \$
- orbiter_path $(\text{ORBITER\_PATH}) \$
- define \text{G} -linear\_group -PGL 4 2 -orthogonal 1 -end \$
- with \text{G} -do \$
- group\_theoretic\_activity \$
- poset\_classification\_control \$
- node\_label\_is\_element \$
- draw\_poset -draw\_options -radius 200 -end \$
- problem\_label Op.4.2 -W -depth 4 \$
- report -end \$

- orbits\_on\_subspaces 4 \$
- report \$
- end \$

pdflatex PGL.4.2_Orthogonal\_plus.4.2_poset.tex
open PGL.4.2_Orthogonal\_plus.4.2_poset.pdf

PGL.4.2_singer_on\_subspaces:

- $(\text{ORBITER}) -v 5 \$
- orbiter_path $(\text{ORBITER\_PATH}) \$
- define \text{G} -linear\_group -PGL 4 2 -singer 1 -end \$
- with \text{G} -do \$
- group\_theoretic\_activity \$
- poset\_classification\_control \$
- node\_label\_is\_element \$

pdflatex PGL.4.2_report.tex
open PGL.4.2_report.pdf
pdflatex PGL.4.2_poset.tex
open PGL.4.2_poset.pdf
5249 \> \> \> \> -draw_poset \n5250 \> \> \> \> -draw_options -end \n5251 \> \> \> \> -problem_label PGL_4_2_singer -W -depth 4 \n5252 \> \> \> \> -report -end \n5253 \> \> \> -end \n5254 \> \> \> -orbits_on_subspaces 4 \n5255 \> \> \> -report \n5256 \> \> -end \n5257 \> pdflatex PGL_4_2_Singer_4_2_1_poset.tex \n5258 \> open PGL_4_2_Singer_4_2_1_poset.pdf \n5259 \n5260 \n5261 \n5262  PGL_8_2_singer_on_subspaces: \n5263 \> $(ORBITER) -v 5 \n5264 \> \> -orbiter_path $(ORBITER_PATH) \n5265 \> \> -define G -linear_group -PGL 8 2 -singer 1 -end \n5266 \> \> -with G -do \n5267 \> \> -group_theoretic_activity \n5268 \> \> \> -poset_classification_control \n5269 \> \> \> \> -node_label_is_element \n5270 \> \> \> \> -draw_poset \n5271 \> \> \> \> -draw_options -radius 150 -end \n5272 \> \> \> \> -problem_label PGL_8_2_singer \n5273 \> \> \> \> -W -depth 8 -report -end \n5274 \> \> \> -end \n5275 \> \> \> -orbits_on_subspaces 8 \n5276 \> \> \> -report \n5277 \> \> \> -end \n5278 \> pdflatex PGL_8_2_Singer_8_2_1_poset.tex \n5279 \> open PGL_8_2_Singer_8_2_1_poset.pdf \n5280 \n5281  # May 7, 2020: 16 sec on Mac \n5282 # 1643 orbits in total \n5283 \n5284  Op_6_2_orbits_on_subspaces: \n5285 \> $(ORBITER) -v 5 \n5286 \> \> -orbiter_path $(ORBITER_PATH) \n5287 \> \> -define G -linear_group -PGL 6 2 -orthogonal 1 -end \n5288 \> \> -with G -do \n5289 \> \> -group_theoretic_activity \n5290 \> \> \> -poset_classification_control \n5291 \> \> \> \> -node_label_is_element \n5292 \> \> \> \> -draw_poset \n5293 \> \> \> \> -draw_options -radius 200 -end \n5294 \> \> \> \> -problem_label Op_6_2 -W \n5295 \> \> \> \> -depth 6 -report -end \n
609
Op_6_3_orbits_on_subspaces:

$(ORBITER) -v 5 \n-orbiter_path $(ORBITER_PATH) \n-define G -linear_group -PGL 6 3 -orthogonal 1 -end \n-with G -do \n-group_theoretic_activity \n-poset_classification_control \n-node_label_is_element \n-draw_poset \n-draw_options -radius 200 -end \n-problem_label Op_6_3 -W \n-depth 6 -report -end \n
Op_6_11_orbits_on_subspaces:

$(ORBITER) -v 5 \n-orbiter_path $(ORBITER_PATH) \n-draw_options -nodes_empty -end \n-define G -linear_group -PGL 6 11 -orthogonal 1 -end \n-with G -do \n-group_theoretic_activity \n-poset_classification_control \n-node_label_is_element \n-draw_poset \n-draw_options -radius 200 -end \n-problem_label Op_6_11 -W \n-depth 6 -report -end \n
610
5343  ▶  ▶  ▶  -end \n5344  ▶  ▶  -orbits_on_subspaces 6 \n5345  ▶  ▶  -report \n5346  ▶  ▶  -end
5347  ▶  pdflatex PGL_6_11_Orthogonal_plus_6_11_report.tex
5348  ▶  open PGL_6_11_Orthogonal_plus_6_11_report.pdf
5349
5350
5351  # June 3, 2020 on Mac: 12 sec
5352
5353
5354  Op_8_2_orbits_on_subspaces:
5355  ▶  \$(ORBITER) -v 5 \n5356  ▶  ▶  -orbiter_path \$(ORBITER\_PATH) \n5357  ▶  ▶  -define G -linear_group -PGL 8 2 -orthogonal 1 -end \n5358  ▶  ▶  -with G -do \n5359  ▶  ▶  -group_theoretic_activity \n5360  ▶  ▶  ▶  -node_label_is_element \n5361  ▶  ▶  ▶  -draw_poset -draw_options -radius 200 -end \n5362  ▶  ▶  ▶  -problem_label Op_8_2 -W -depth 8 -report -end \n5363  ▶  ▶  ▶  -end \n5364  ▶  ▶  -orbits_on_subspaces 8 \n5365  ▶  ▶  -report \n5366  ▶  ▶  -end
5367  ▶  ▶  -end
5368  ▶  pdflatex PGL_8_2_Orthogonal_plus_8_2_poset.tex
5369  ▶  open PGL_8_2_Orthogonal_plus_8_2_poset.pdf
5370
5371
5372
5373  PGO_7_2_on_subspaces:
5374  ▶  \$(ORBITER) -v 20 \n5375  ▶  ▶  -orbiter_path \$(ORBITER\_PATH) \n5376  ▶  ▶  -define F -finite_field -q 2 -end \n5377  ▶  ▶  -define G -linear_group -PGL 7 F -orthogonal 0 -end \n5378  ▶  ▶  -with G -do \n5379  ▶  ▶  -group_theoretic_activity \n5380  ▶  ▶  ▶  -node_label_is_element \n5381  ▶  ▶  ▶  -draw_poset \n5382  ▶  ▶  ▶  -draw_options -radius 200 -end \n5383  ▶  ▶  ▶  -problem_label 0_7_2 \n5384  ▶  ▶  ▶  -W -depth 7 \n5385  ▶  ▶  ▶  -report -end \n5386  ▶  ▶  ▶  -end \n5387  ▶  ▶  ▶  -orbits_on_subspaces 7 \n5388  ▶  ▶  ▶  -report \n5389  ▶  ▶  ▶  -end
# Section 6.5: Orbits on set partitions

SECTION ORBITS ON SET_partitions:

C6 on partition:

```bash
$(ORBITER) -v 5 \n-define G -permutation_group -cyclic_group 6 -end \n-with G -do \n-group_theoretic_activity \n-poset_classification_control \n-problem_label C6 \n-depth 2 \n-W \n-draw_options \n-radius 200 -embedded \n-end \n-orbits_on_partition 2 \n-end
```

PGL_2.17 on partition:

```bash
$(ORBITER) -v 5 \n-define G -linear_group -PGL 2 17 -end \n-with G -do \n-group_theoretic_activity \n-poset_classification_control \n-problem_label PGL_2.17 \n-depth 6 \n-W \n-orbits_on_partition 6 \n-end
```
SECTION_ARCS_ANDCaps_IN_PROJECTIVE_SPACES:

PGL_3_27:

\$(ORBITER) -v 5 \\n\$define G \\
-linear_group -PGL 3 27 -end \\
-with G -do \\
group_theoretic_activity \\
\report \\
end

pdflatex PGL_3_27_report.tex
open PGL_3_27_report.pdf

AGGL_2_27:

\$(ORBITER) -v 5 \\n\$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:

\$(ORBITER) -v 4 \\n\$define F -finite_field -q 4 -end \\
\$define P -projective_space -n 2 -field F -v 0 -end \\
-with P -do \\
-projective_space_activity \\
-classify_arcs \\
poset_classification_control \\
-problem_label hyperoval_q4 \\
-W -depth 6 \\
\report -end
5484 \> \> \> \> -end \ 
5485 \> \> \> \> -target_size 6 \ 
5486 \> \> \> \> -d 2 \ 
5487 \> \> \> -end \ 
5488 \> \> -end 
5489 \> pdflatex hyperoval_q4.poset.tex 
5490 \> open hyperoval_q4.poset.pdf 
5491 
5492 
5493 
5494 
5495 \> hyperoval_8_classify: 
5496 \> \> $(\text{ORBITER}) -v 4 \ 
5497 \> \> -orbiter_path $(\text{ORBITER\_PATH}) \ 
5498 \> \> -define F -finite_field -q 8 -end \ 
5499 \> \> -define P -projective_space -n 2 -field F -v 0 -end \ 
5500 \> \> -with P -do \ 
5501 \> \> -projective_space_activity \ 
5502 \> \> \> -classify_arcs \ 
5503 \> \> \> \> -poset_classification_control \ 
5504 \> \> \> \> \> -problem_label hyperoval_q8 \ 
5505 \> \> \> \> \> \> -W -depth 10 \ 
5506 \> \> \> \> \> \> -report -end \ 
5507 \> \> \> \> \> \> -draw_poset \ 
5508 \> \> \> \> \> \> -draw_options \ 
5509 \> \> \> \> \> \> \> -radius 200 \ 
5510 \> \> \> \> \> \> \> \> -end \ 
5511 \> \> \> \> \> \> \> \> -end \ 
5512 \> \> \> \> \> \> \> \> -target_size 10 \ 
5513 \> \> \> \> \> \> \> \> -d 2 \ 
5514 \> \> \> \> \> \> \> \> -end \ 
5515 \> \> \> \> \> \> -end 
5516 \> pdflatex hyperoval_q8.poset.tex 
5517 \> open hyperoval_q8.poset.pdf 
5518 
5519 
5520 
5521 
5522 \> frame_stabilizer_PGGL: 
5523 \> \> $(\text{ORBITER}) -v 4 \ 
5524 \> \> -define G \ 
5525 \> \> -linear_group -PGGL 3 8 -end \ 
5526 \> \> -with G -do \ 
5527 \> \> -group_theoretic_activity \ 
5528 \> \> \> -poset_classification_control \ 
5529 \> \> \> \> -problem_label frame_q8 -W -depth 4 \ 
5530 \> \> \> \> -draw_options -end \ 

614
5531  
5532  
5533  
5534  
5535  
5536  
5537  
5538  
5539  
5540  
5541  
5542  
5543  frame_stabilizer_PGL:  
5544  
5545  
5546  
5547  
5548  
5549  
5550  
5551  
5552  
5553  
5554  
5555  
5556  
5557  
5558  
5559  
5560  
5561  
5562  
5563  
5564  
5565  
5566  hyperoval_16_classify:  
5567  
5568  
5569  
5570  
5571  
5572  
5573  
5574  
5575  
5576  
5577  

615
We found 17028 non-conical 6 subsets

C Eckardt point number distribution: $13^{-252}, 9^{-720}, 5^{-2304}, 3^{-13752}$

hyperoval

⊿ $(ORBITER) -v 2$

⊿ ⊿ -define F -finite_field -q 16 -end

⊿ ⊿ -define P -projective_space -n 2 -field F -v 0 -end

⊿ ⊿ -define H_16_2 -geometric_object P

⊿ ⊿ -set $(HYPEROVAL_16_16320) -end

⊿ ⊿ -with H_16_2 -do

⊿ ⊿ -combinatorial_object_activity

⊿ ⊿ -save

⊿ ⊿ -end

⊿ ⊿ -with H_16_2 -do

⊿ ⊿ -combinatorial_object_activity

⊿ ⊿ -non_conical_type

⊿ ⊿ -end

⊿ ⊿ -print_symbols

We found 6188 = \binom{17}{5} non-conical 6 subsets

# Eckardt point number distribution: $45^{-68}, 13^{-2040}, 5^{-4080}$

# neighbors of 0 with 4 removed.csv

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
#5,8,9,11,12
#6,7,11,13,17
#7,7,10,14,16
#8,1,9,13,16
#9,2,8,13,14
#10,1,2,15,17
#11,6,8,10,17
#12,6,7,9,15
#13,2,6,11,16
#14,5,9,14,17
#15,5,8,15,16
#16,1,6,12,14
#17,2,5,7,12
#18,3,12,16,17
#19,3,11,14,15
hyperoval

⊿ $(ORBITER) -v 2 \ 
⊿ ⊿ -define G -graph -disjoint_sets_graph \ 
⊿ ⊿ ⊿ neighbors_of_0_with_4_removed.csv \ 
⊿ ⊿ -end \ 
⊿ ⊿ -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

# 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:
nc_arcs_16:

\$\text{ORBITER}$ -v 4 \n
\text{-define F -finite_field -q 16 -end} \n
\text{-define P -projective_space -n 2 -field F -v 0 -end} \n
\text{-with P -do} \n
\text{-projective_space_activity} \n
\text{-classify_arcs} \n
\text{-poset_classification_control} \n
\text{-problem_label nc_arcs_q16_d2} \n
\text{-W -depth 6} \n
\text{-report -end} \n
\text{-end} \n
\text{-target_size 6} \n
\text{-d 2} \n
\text{-conic_test} \n
\text{-end} \n
\text{pdflatex nc_arcs_q16_d2_poset.tex} \n
\text{open nc_arcs_q16_d2_poset.pdf} \n
\text{nc_arcs_32_E13:} \n
\$\text{ORBITER}$ -v 4 \n
\text{-orbiter_path $(\text{ORBITER\_PATH})} \n
\text{-define F -finite_field -q 32 -end} \n
\text{-define P -projective_space -n 2 -field F -v 0 -end} \n
\text{-with P -do} \n
\text{-projective_space_activity} \n
\text{-classify_arcs} \n
\text{-poset_classification_control} \n
\text{-problem_label nc_arcs_q32_d2} \n
\text{-W -depth 6} \n
\text{-draw_poset -draw_options -end} \n
\text{-report -end} \n
\text{-end} \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{pdflatex nc_arcs_q32_d2_poset.tex}
open nc_arcs_q32_d2_poset.pdf

#User time: 0:00

F64_work:

F64_frob:

# surfaces with 13 Eckardt points have OCN=0,98,99

surface_64_0:

#makes it slow:

#makes it slow:
\texttt{multiple_of 24 -end} \ 
\texttt{#User time: 0:3} 
\texttt{5812} 
\texttt{5813} 
\texttt{5814} 
\texttt{5815} 
\texttt{5816 nc_arcs_{128}:} 
\texttt{5817} \texttt{\%} \texttt{(ORBITER) -v 4} \ \texttt{\%} 
\texttt{5818} \texttt{\%} \texttt{-define F -finite_field -q 128 -end} \ 
\texttt{5819} \texttt{\%} \texttt{-define P -projective_space -n 2 -field F -use_projectivity_subgroup -v 0 -end} 
\texttt{5820} \texttt{\%} \texttt{-with P -do} \ 
\texttt{5821} \texttt{\%} \texttt{-projective_space_activity} \ 
\texttt{5822} \texttt{\%} \texttt{-classify_arcs} \ 
\texttt{5823} \texttt{\%} \texttt{-poset_classification_control} \ 
\texttt{5824} \texttt{\%} \texttt{-problem_label nc_arcs_q128_d2 -W -depth 6} \ 
\texttt{5825} \texttt{\%} \texttt{-report -select_orbits_by_level 6} \ 
\texttt{5826} \texttt{\%} \texttt{-select_orbits_by_stabilizer_order_multiple_of 24} \ 
\texttt{5827} \texttt{\%} \texttt{-end} \ 
\texttt{5828} \texttt{\%} \texttt{-end} \ 
\texttt{5829} \texttt{\%} \texttt{-target_size 6} \ 
\texttt{5830} \texttt{\%} \texttt{-d 2} \ 
\texttt{5831} \texttt{\%} \texttt{-conic_test} \ 
\texttt{5832} \texttt{\%} \texttt{-end} \ 
\texttt{5833} \texttt{\%} \texttt{-end} 
\texttt{5834} \texttt{\%} \texttt{pdflatex nc_arcs_q128_d2_poset.tex} 
\texttt{5835} \texttt{\%} \texttt{open nc_arcs_q128_d2_poset.pdf} 
\texttt{5836} 
\texttt{5837} 
\texttt{5838} \texttt{#User time: 0:52} 
\texttt{5839} 
\texttt{5840} 
\texttt{5841} 
\texttt{5842} 
\texttt{5843 nc_arcs_{256}_{E13}:} 
\texttt{5844} \texttt{\%} \texttt{(ORBITER) -v 8} \ 
\texttt{5845} \texttt{\%} \texttt{-define F -finite_field -q 256 -end} 
\texttt{5846} \texttt{\%} \texttt{-define P -projective_space -n 2 -field F -use_projectivity_subgroup -v 0 -end} 
\texttt{5847} \texttt{\%} \texttt{-with P -do} \ 
\texttt{5848} \texttt{\%} \texttt{-projective_space_activity} \ 
\texttt{5849} \texttt{\%} \texttt{-classify_arcs} \ 
\texttt{5850} \texttt{\%} \texttt{-poset_classification_control} \ 
\texttt{5851} \texttt{\%} \texttt{-problem_label nc_arcs_q256_d2 -W -depth 6} \ 
\texttt{5852} \texttt{\%} \texttt{-report -end} \ 
\texttt{5853} \texttt{\%} \texttt{-end} \ 
\texttt{5854} \texttt{\%} \texttt{-target_size 6} \ 
\texttt{5855} \texttt{\%} \texttt{-test_nb_Eckardt_points 13} 

\texttt{621}
Example F64:

```
$(ORBITER) -v 3 \
-define F -finite_field -q 64 -end \
-define P -projective_space -n 3 -field F -v 0 -end \
-with P -do \ 
-projective_space_activity \ 
-define_surface S64_abcd_52_8_8_52 -q 64 \ 
-family_general_abcd 52 8 8 52 -end \ 
-with S64_abcd_52_8_8_52 -do \ 
-cubic_surface_activity \ 
-report \ 
-end \
```

```
pdflatex surface_family_general_abcd_q64_a52_b8_c8_d52_report.tex
```

six_arcs_4_nbE13:

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

```
$(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 \ 
\text{define}\ F -\text{finite\ field}\ -q\ 16 \end 
\text{define}\ P -\text{projective\ space}\ -n\ 2 -\text{field}\ F -v\ 0 \end 
\text{with}\ P \text{-do} \ 
\text{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 \ 
\text{define}\ F -\text{finite\ field}\ -q\ 32 \end 
\text{define}\ P -\text{projective\ space}\ -n\ 2 -\text{field}\ F -v\ 0 \end 
\text{with}\ P \text{-do} \ 
\text{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 \ 
\text{define}\ F -\text{finite\ field}\ -q\ 64 \end 
\text{define}\ P -\text{projective\ space}\ -n\ 2 -\text{field}\ F -v\ 0 \end 
\text{with}\ P \text{-do} \ 
\text{projective_space_activity} \ 
-control_six_arcs -problem_label sixarcs_q64 -end \ 
six_arcs_not_on_conic -filter_by_nb_Eckardt_points 13 -end

six_arcs_128_nbE13:
$\text{(ORBITER)} -v 3 \ 
\text{define}\ F -\text{finite\ field}\ -q\ 128 \end 
\text{define}\ P -\text{projective\ space}\ -n\ 2 -\text{field}\ F -v\ 0 \end 
\text{with}\ P \text{-do} \ 
\text{projective_space_activity} \ 
-control_six_arcs -problem_label sixarcs_q128 -end \ 
six_arcs_not_on_conic -filter_by_nb_Eckardt_points 13 -end

# 9 arcs: ago: 4, 8, 24^5, 48^2

six_arcs_16_nbE13:
$\text{(ORBITER)} -v 3 \ 
\text{define}\ F -\text{finite\ field}\ -q\ 16 \end 
\text{define}\ P -\text{projective\ space}\ -n\ 2 -\text{field}\ F -v\ 0 \end 
\text{with}\ P \text{-do} \ 
\text{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 \ 
\text{define}\ F -\text{finite\ field}\ -q\ 32 \end 
\text{define}\ P -\text{projective\ space}\ -n\ 2 -\text{field}\ F -v\ 0 \end 
\text{with}\ P \text{-do} \ 
\text{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 \ 
\text{define}\ F -\text{finite\ field}\ -q\ 64 \end 
\text{define}\ P -\text{projective\ space}\ -n\ 2 -\text{field}\ F -v\ 0 \end 
\text{with}\ P \text{-do} \ 
\text{projective_space_activity} \ 
-control_six_arcs -problem_label sixarcs_q64 -end \ 
six_arcs_not_on_conic -filter_by_nb_Eckardt_points 13 -end

six_arcs_128_nbE13:
$\text{(ORBITER)} -v 3 \ 
\text{define}\ F -\text{finite\ field}\ -q\ 128 \end 
\text{define}\ P -\text{projective\ space}\ -n\ 2 -\text{field}\ F -v\ 0 \end 
\text{with}\ P \text{-do} \ 
\text{projective_space_activity} \ 
-control_six_arcs -problem_label sixarcs_q128 -end \ 
six_arcs_not_on_conic -filter_by_nb_Eckardt_points 13 -end

# 9 arcs: ago: 4, 8, 24^5, 48^2

623
five_arcs_q13:

pdflatex five_arcs_q13_poset.tex
open five_arcs_q13_poset.pdf

# Section 6.7: Cubic Curves

SECTION_CUBIC_CURVES:
cubic_curves_PG_2.4:

$\text{define } F \text{-finite_field -q 3 -end}$
$\text{define } P \text{-projective_space -n 2 -field F -v 0 -end}$
$\text{with } P \text{-do}$
$\text{-projective_space_activity}$
$\text{-classify cubic_curves -q 4}$
$\text{-target_size 9 -n 3 -d 3}$
$\text{-poset_classification_control}$
$\text{-problem_label cc_4 -W -depth 9}$
$\text{-draw_poset}$
$\text{-draw_options -radius 200 -embedded -end}$
$\text{-report -end}$
$\text{-end}$
$\text{pdflatex cc_4_poset.tex}$
$\text{open cc_4_poset.pdf}$
$\text{pdflatex cc_4_poset_lvl_9.tex}$
$\text{open cc_4_poset_lvl_9.pdf}$
$\text{pdflatex Cubic_curves_q4.tex}$
$\text{open Cubic_curves_q4.pdf}$

$\text{cubic_curves_PG_2.4.draw:}$
$\text{define } G$
$\text{-define } F \text{-finite_field -q 8 -end}$
$\text{-define } P \text{-projective_space -n 2 -field F -v 0 -end}$
$\text{with } P \text{-do}$
$\text{-projective_space_activity}$
$\text{-classify cubic_curves -q 8 -target_size 9 -n 3 -d 3}$
$\text{-poset_classification_control}$
$\text{-problem_label cc_8 -W -depth 9}$
$\text{-draw_options -radius 200 -embedded -end}$
$\text{-recognize "0,1,2,3,35,28"}$
$\text{-recognize "1,2,3,51,28,61,46,71,40"}$
$\text{-draw_poset}$

$\text{pdflatex cc_4_poset_lvl_9.draw.tex}$
$\text{open cc_4_poset_lvl_9.draw.pdf}$

$\text{cubic_curves_PG_2.8:}$
$\text{define } G$
$\text{-define } F \text{-finite_field -q 8 -end}$
$\text{-define } P \text{-projective_space -n 2 -field F -v 0 -end}$
$\text{with } P \text{-do}$
$\text{-projective_space_activity}$
$\text{-classify cubic_curves -q 8 -target_size 9 -n 3 -d 3}$
$\text{-poset_classification_control}$
$\text{-problem_label cc_8 -W -depth 9}$
$\text{-draw_options -radius 200 -embedded -end}$
$\text{-recognize "0,1,2,3,35,28"}$
$\text{-recognize "1,2,3,51,28,61,46,71,40"}$
$\text{-draw_poset}$
$\text{cubic\_curves\_PG\_2\_8\_draw}: $

\begin{verbatim}
$\text{ $(ORBITER) -v 3 }$
$\text{ -draw\_layered\_graph \ }
\text{ cc\_8\_poset\_ lvl\_9\_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\_poset\_ lvl\_9\_draw\_tex }
\text{ open cc\_8\_poset\_ lvl\_9\_draw\_pdf }
\end{verbatim}

#cc\_8\_poset\_ lvl\_9\_layered\_graph
#cc\_8\_poset\_ detailed\_lvl\_9\_layered\_graph

#cc\_8\_poset\_ lvl\_9.layered\_graph
#cc\_8\_poset\_ detailed\_lvl\_9.layered\_graph

# Chapter 7 - Cubic Surfaces
# Section 7.1: Cubic Surfaces Creation

SECTION_CUBIC_SURFACES_CREATION:
surface_4.0:
  $(ORBITER) -v 3 \
  -define F -finite_field -q 4 -end \
  -define P -projective_space -n 3 -field F -v 0 -end \
  -with P -do \n  -projective_space_activity \n  -define_surface S -q 4 -catalogue 0 -end \n  -end \n  -with S -do \n  -cubic_surface_activity \n  -report \n  -report_with_group \n  -end

pdflatex surface_catalogue_q4_iso0_report.tex
open surface_catalogue_q4_iso0_report.pdf
pdflatex surface_catalogue_q4_iso0_with_group.tex
open surface_catalogue_q4_iso0_with_group.pdf

surface_7.0:
  $(ORBITER) -v 3 \
  -define F -finite_field -q 7 -end \
  -define P -projective_space -n 3 -field F -v 0 -end \
  -with P -do \n  -projective_space_activity \n  -define_surface S7_0 -q 7 -catalogue 0 -end \n  -end \n  -with S7_0 -do \n  -cubic_surface_activity \n  -report \n  -report_with_group \n  -all_quartic_curves \n  -end

pdflatex surface_catalogue_q7_iso0_report.tex
open surface_catalogue_q7_iso0_report.pdf
pdflatex surface_catalogue_q7_iso0_with_group.tex
open surface_catalogue_q7_iso0_with_group.pdf

Family_general_F7:
  $(ORBITER) -v 3 \
  -define F -finite_field -q 7 -end \n
# Fermat with 18 Eckardt points
# no automorphism group, so no -report_with_group and no -all_quartic_curves

# Joel:

eckardt_13_4_12:

$\text{(ORBTER)} \text{-v 6}$

$\text{-define } F \text{-finite_field -q 13 -end}$

$\text{-define } P \text{-projective_space -n 3 -field } F \text{-v 0 -end}$

$\text{-with } P\text{-do}$

$\text{-projective_space_activity}$

$\text{-define_surface } S_{2.1} \text{-q 13}$

$\text{-family_Eckardt 4 12 -end}$

$\text{-with } S_{2.1} \text{-do}$

$\text{-cubic_surface_activity}$

$\text{-report}$

$\text{-report_with_group}$

$\text{-end}$

# Joel:

surface_8.0.catalogue:

$\text{(ORBTER)} \text{-v 3}$

$\text{-define } F \text{-finite_field -q 8 -end}$

$\text{-define } P \text{-projective_space -n 3 -field } F \text{-v 0 -end}$

$\text{-with } P\text{-do}$

$\text{-projective_space_activity}$

$\text{-define_surface } S_{8.0} \text{-q 8 -catalogue 0 -end}$

$\text{pdflatex surface_family_general_abcd_q7_a2_b3_c3_d4_report.tex}$

$\text{open surface_family_general_abcd_q7_a2_b3_c3_d4_report.pdf}$

$\text{open surface_family_general_abcd_q7_a2_b3_c3_d4_report.pdf}$

$\text{(ORBTER)} \text{-v 6}$

$\text{-define } F \text{-finite_field -q 13 -end}$

$\text{-define } P \text{-projective_space -n 3 -field } F \text{-v 0 -end}$

$\text{-with } P\text{-do}$

$\text{-projective_space_activity}$

$\text{-family_general_abcd 2 3 3 4 -end}$

$\text{-with } S_{7,abcd}_2 3 3 4 \text{-do}$

$\text{-cubic_surface_activity}$

$\text{-report}$

$\text{-end}$

# Fermat with 18 Eckardt points
# no automorphism group, so no -report_with_group and no -all_quartic_curves

# Joel:
6185 ▶ ▶ -end 
6186 ▶ ▶ -with S8.0 -do 
6187 ▶ ▶ -cubic_surface_activity 
6188 ▶ ▶ ▶ -report 
6189 ▶ ▶ ▶ ▶ -report_with_group 
6190 ▶ ▶ -end 
6191 ▶ pdflatex surface_catalogue_q8_iso0_report.tex 
6192 ▶ open surface_catalogue_q8_iso0_report.pdf 
6193 ▶ pdflatex surface_catalogue_q8_iso0_with_group.tex 
6194 ▶ open surface_catalogue_q8_iso0_with_group.pdf 
6195 
6196 
6197 
6198 surface.8.0_clean: 
6199 ▶ $(ORBITER) -v 3 
6200 ▶ ▶ -define F -finite_field -q 8 -end 
6201 ▶ ▶ -define P -projective_space -n 3 -field F -v 0 -end 
6202 ▶ ▶ -with P -do 
6203 ▶ ▶ -projective_space_activity 
6204 ▶ ▶ ▶ -define_surface S8.0 -q 8 -catalogue 0 \ 
6205 ▶ ▶ ▶ ▶ -select_double_six "15,11,22,19,24,5,16,10,23,20,25,4" \ 
6206 ▶ ▶ ▶ ▶ -select_double_six "3,2,1,0,5,4,9,8,7,6,11,10" \ 
6207 ▶ ▶ ▶ ▶ -transform_inverse "1,4,4,0,6,0,0,0,6,2,0,1,7,0,4,0,0" \ 
6208 ▶ ▶ ▶ ▶ -transform "4,4,0,0, 0,0,1,0, 1,0,0,0, 0,0,0,1, 0" \ 
6209 ▶ ▶ ▶ ▶ -transform_inverse "2,0,0,0,0,2,0,0,0,0,2,0,1,1,2,3,0" \ 
6210 ▶ ▶ ▶ -end -end \ 
6211 ▶ ▶ ▶ -with S8.0 -do \ 
6212 ▶ ▶ ▶ -cubic_surface_activity \ 
6213 ▶ ▶ ▶ ▶ -report 
6214 ▶ ▶ ▶ ▶ ▶ -report_with_group \ 
6215 ▶ ▶ ▶ ▶ -end 
6216 ▶ pdflatex surface_catalogue_q8_iso0_report.tex 
6217 ▶ open surface_catalogue_q8_iso0_report.pdf 
6218 
6219 
6220 
6221 # clean equation for Tekirdag-1: 
6222 
6223 surface.8.0b: 
6224 ▶ $(ORBITER) -v 3 
6225 ▶ ▶ -define F -finite_field -q 8 -end 
6226 ▶ ▶ -define P -projective_space -n 3 -field F -v 0 -end 
6227 ▶ ▶ -with P -do 
6228 ▶ ▶ -projective_space_activity \ 
6229 ▶ ▶ ▶ -define_surface S8.0 -q 8 -catalogue 0 \ 
6230 ▶ ▶ ▶ ▶ -select_double_six "15,11,22,19,24,5,16,10,23,20,25,4" \ 
6231 ▶ ▶ ▶ ▶ -select_double_six "3,2,1,0,5,4,9,8,7,6,11,10" \
-transform "1,0,0,0,1,0,6,0,0,1,6,0,0,0,1,0"
-transform_inverse "3,1,1,0,0,1,0,0,0,1,0,0,0,0,1,0"
-transform_inverse "2,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0"
-end
-with S8_0 -do
-cubic_surface_activity
-report
-report_with_group
-end
pdflatex surface_catalogue_q8_iso0_with_group.tex
open surface_catalogue_q8_iso0_with_group.pdf

# writes tangents.txt

# 13.0 has 4 Eckardt points
# 13.1 has 6 Eckardt points
# 13.2 has 9 Eckardt points
# 13.3 has 18 Eckardt points

Eckardt_13:
$(ORBITER) -v 3
(define F -finite_field -q 13 -end
(define P -projective_space -n 3 -field F -v 0 -end
-with P -do
-projective_space_activity
(define_surface S_q13 -q 13
-family_Eckardt 3 1 -end
-end
-with S_q13 -do
-cubic_surface_activity
-report
-report_with_group
-end
pdflatex surface_family_Eckardt_q13_a3_b1_with_group.tex
open surface_family_Eckardt_q13_a3_b1_with_group.pdf

surface_13_0:
\begin{verbatim}
\verb|$\text{ORBITER}$ -v 3 \ 
\verb|$\text{-define F finite_field -q 13 -end}$ \ 
\verb|$\text{-define P projective_space -n 3 -field F -v 0 -end}$ \ 
\verb|$\text{-with P -do}$ \ 
\verb|$\text{-projective_space_activity}$ \ 
\verb|$\text{-define_surface S13_0 -q 13 -catalogue 0 -end}$ \ 
\verb|$\text{-end}$ \ 
\verb|$\text{-with S13_0 -do}$ \ 
\verb|$\text{-cubic_surface_activity}$ \ 
\verb|$\text{-report}$ \ 
\verb|$\text{-report_with_group}$ \ 
\verb|$\text{-end}$ \ 
\verb|$\text{pdflatex surface_catalogue_q13_iso0_report.tex}$ \ 
\verb|$\text{open surface_catalogue_q13_iso0_report.pdf}$ \ 
\verb|$\text{# clean equation for Tekirdag-2:}$ \ 
\verb|$\text{surface_16_0:}$ \ 
\verb|$\text{pdflatex surface_catalogue_q16_iso0_with_group.tex}$ \ 
\verb|$\text{open surface_catalogue_q16_iso0_with_group.pdf}$ \ 
\verb|$\text{# rank of lines ( 66591, 26737, 4093, 69855, 26208, 5}$ \ 
\verb|$\text{847, 369, 32230, 529, 30293, 70068, 2178, 261, 28666, 8575, 105, 31694, 0, 51784,}$ \ 
\verb|$\text{25209, 22193, 49862, 274 )}$ \ 
\end{verbatim}
# 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 -with P -do -projective_space_activity -define_surface T1 -family_G13 2 -q 8 -end -end -with T1 -do -cubic_surface_activity -define_surface T1 -family_G13 2 -q 8 -end -end -with T1 -do -cubic_surface_activity -define_surface T1 -family_G13 2 -q 8 -end -end

pdflatex surface_family_G13_q8_a2_with_group.tex
open surface_family_G13_q8_a2_with_group.pdf

F13 8:

$(ORBITER) -v 3 -define F -finite_field -q 8 -end -define P -projective_space -n 3 -field F -v 0 -end -with P -do -projective_space_activity -define_surface T1 -family_F13 2 -q 8 -end -end -with T1 -do -cubic_surface_activity -define_surface T1 -family_F13 2 -q 8 -end -end -with T1 -do -cubic_surface_activity -define_surface T1 -family_F13 2 -q 8 -end -end

pdflatex surface_family_F13_q8_a2_with_group.tex
open surface_family_F13_q8_a2_with_group.pdf

# Tekirdag-2:
F13_16:

$\text{ORBITER} -v 3$

```bash
$\text{define F -finite_field -q 16 -end} \$
```

```bash
$\text{define P -projective_space -n 3 -field F -v 0 -end} \$
```

```bash
$\text{-with P -do} \$
```

```bash
$\text{-projective_space_activity} \$
```

```bash
$\text{-define_surface T2 -family_F13 2 -q 16 -end} \$
```

```bash
$\text{-end} \$
```

```bash
$\text{-with T2 -do} \$
```

```bash
$\text{-cubic_surface_activity} \$
```

```bash
$\text{-report} \$
```

```bash
$\text{-report_with_group} \$
```

```bash
$\text{-end} \$
```

```bash
$\text{pdflatex surface_family_F13_q16_a2_with_group.tex} \$
```

```bash
$\text{open_surface_family_F13_q16_a2_with_group.pdf} \$
```

# Tekirdag-3:

F13_32:

$\text{ORBITER} -v 3$

```bash
$\text{define F -finite_field -q 32 -end} \$
```

```bash
$\text{define P -projective_space -n 3 -field F -v 0 -end} \$
```

```bash
$\text{-with P -do} \$
```

```bash
$\text{-projective_space_activity} \$
```

```bash
$\text{-define_surface T2 -family_F13 2 -q 16 -end} \$
```

```bash
$\text{-end} \$
```

```bash
$\text{-with T2 -do} \$
```

```bash
$\text{-cubic_surface_activity} \$
```

```bash
$\text{-report} \$
```

```bash
$\text{-report_with_group} \$
```

```bash
$\text{-end} \$
```

```bash
$\text{pdflatex surface_family_F13_q32_a2_with_group.tex} \$
```

```bash
$\text{open_surface_family_F13_q32_a2_with_group.pdf} \$
```

# Kapadokya-1:

F13_64a:

$\text{ORBITER} -v 3$

```bash
$\text{define F -finite_field -q 64 -end} \$
```

```bash
$\text{define P -projective_space -n 3 -field F -v 0 -end} \$
```

```bash
$\text{-with P -do} \$
```

```bash
$\text{-projective_space_activity} \$
```

```bash
$\text{-define_surface K1 -family_F13 2 -q 64 -end} \$
```

```bash
$\text{-end} \$
```

```bash
$\text{pdflatex surface_family_F13_q64_a2_with_group.tex} \$
```

```bash
$\text{open_surface_family_F13_q64_a2_with_group.pdf} \$
```
6416  ▷ ▷ with K1 -do \n6417  ▷ ▷ cubic_surface_activity \n6418  ▷ ▷ ▷ report \n6419  ▷ ▷ ▷ report_with_group \n6420  ▷ ▷ ▷ end \n6421  
6422  
6423  # Kapadokya-2: \n6424  
6425  F13_64b: \n6426  ▷ $(ORBITER) -v 3 \n6427  ▷ ▷ define F -finite_field -q 64 -end \n6428  ▷ ▷ define P -projective_space -n 3 -field F -v 0 -end \n6429  ▷ ▷ with P -do \n6430  ▷ ▷ projective_space_activity \n6431  ▷ ▷ ▷ define_surface K2 -family_F13 18 -q 64 -end \n6432  ▷ ▷ ▷ end \n6433  ▷ ▷ with K2 -do \n6434  ▷ ▷ cubic_surface_activity \n6435  ▷ ▷ ▷ report \n6436  ▷ ▷ ▷ report_with_group \n6437  ▷ ▷ ▷ end \n6438  
6439  
6440  Colorado1: \n6441  ▷ $(ORBITER) -v 3 \n6442  ▷ ▷ define F -finite_field -q 128 -end \n6443  ▷ ▷ define P -projective_space -n 3 -field F -v 0 -end \n6444  ▷ ▷ with P -do \n6445  ▷ ▷ projective_space_activity \n6446  ▷ ▷ ▷ define_surface CO-1 -q 128 -catalogue 0 \n6447  ▷ ▷ ▷ transform_inverse "1,0,0,0,0,1,0,96,0,0,1,96,0,0,0,1,0" \n6448  ▷ ▷ ▷ end \n6449  ▷ ▷ ▷ end \n6450  ▷ ▷ with CO-1 -do \n6451  ▷ ▷ cubic_surface_activity \n6452  ▷ ▷ ▷ report \n6453  ▷ ▷ ▷ report_with_group \n6454  ▷ ▷ ▷ end \n6455  
6456  
6457  # recognize the arcs from Colorado-1,2,3: \n6458  
6459  
6460  
6461  Colorado2: \n6462  ▷ $(ORBITER) -v 3 \n
634
define F -finite_field -q 128 -end \ndefine P -projective_space -n 3 -field F -v 0 -end \nwith P -do 
-define_surface CO-2 -q 128 -catalogue 926 \ntransform_inverse "1,0,0,0,0,1,0,32,0,0,1,32,0,0,0,1,0" \n-end \nw ith CO-2 -do 
cubic_surface_activity \n-report \n-report_with_group \n-end 

Colorado3: 
$\text{(ORBITER)} -v 3 \ndefine F -finite_field -q 128 -end \ndefine P -projective_space -n 3 -field F -v 0 -end \nwith P -do 
-define_surface CO-3 -q 128 -catalogue 928 \ntransform_inverse "1,0,0,0,0,1,0,59,0,0,1,59,0,0,0,1,0" \n-end \w ith CO-3 -do 
cubic_surface_activity \n-report \n-report_with_group \n-end 

# Colorado-1: 

F13.128a: 
$\text{(ORBITER)} -v 3 \ndefine F -finite_field -q 128 -end \ndefine P -projective_space -n 3 -field F -v 0 -end \nwith P -do 
-define_surface CO-1 -family_F13 2 -q 128 -end \n-end \w ith CO-1 -do 
cubic_surface_activity \n-report \n-report_with_group \n-end
# Colorado-2:

F13_128b:

-define F -finite_field -q 128 -end \n-define P -projective_space -n 3 -field F -v 0 -end \n-with P -do \n-projective_space_activity \n-define_surface CO-2 -family_F13 6 -q 128 -end \n-with CO-2 -do \n-cubic_surface_activity \n-report \n-report_with_group \n-end

# Colorado-3:

F13_128c:

-define F -finite_field -q 128 -end \n-define P -projective_space -n 3 -field F -v 0 -end \n-with P -do \n-projective_space_activity \n-define_surface CO-3 -family_F13 14 -q 128 -end \n-with CO-3 -do \n-cubic_surface_activity \n-report \n-report_with_group \n-end

move_two_lines:

$\$(ORBITER) -v 5 \n-define F -finite_field -q 8 -end \n-with F -do -finite_field_activity \n-move_two_lines_in_hyperplane_stabilizer 65 4680 72 657 -end

F_alpha_beta_gamma_delta:

$\$(ORBITER) -v 3 \n-define F -finite_field -q 7 -end \n-with F -do -finite_field_activity \n-parse_and_evaluate \n
"F.alpha_beta_gamma_delta" "x0,x1,x2,x3"

\$(F_ALPHA_BETA_GAMMA_DELTA) \n
"alpha=2,beta=3,\gamma=4,\delta=5" \n
end

dot -Tpng F_alpha_beta_gamma_delta.gv >F_alpha_beta_gamma_delta.png

F_abcd_Eckardt_q31:

\$(ORBITER) -v 3 \n
-define F -finite_field -q 31 -end \n
 DEFINE P -projective_space -n 3 -field F -v 0 -end \n
 -with P -do \n
-projective_space_activity \n
-define_surface F_abcd -q 31 \n
-by_equation "F_abcd" \n
"DF\{a,b,c,d\}D" "X0,X1,X2,X3" \n
\$(F_abcd_eqn) \n
"a=2,b=30,c=30,d=2" \n
"Da=2,b=30,c=30,d=2D" \n
-end \n
-end \n
with F_abcd -do \n
-cubic_surface_activity \n
-report \n
-end

pdflatex surface_equation_F_abcd_q31_report.tex

open surface_equation_F_abcd_q31_report.pdf

surface_F_abcd:

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

#dot -Tpng F_alpha_beta_gamma_delta.gv >F_alpha_beta_gamma_delta.png

F_abcd_sweep_4.27_q7:
$(ORBITER) -v 3 \
-define F -finite_field -q 7 -end \
-define P -projective_space -n 3 -field F -v 0 -end \
-with P -do \n-projective_space_activity \n-sweep 4.27 sweep 4.27_q7 -q 7 -by_equation "F_abcd" \n"\DF{a,b,c,d}\D" "X0,X1,X2,X3" \n"(F_abcd_eqn) \n"a=2,b=3,c=4,d=5" \n"(a=2,b=3,c=4,d=5)" \n-end \n-end \n-with F -do \n-projective_space_activity \n-define_surface F_2345 -q 7 \n-by_equation "F_alpha_beta_gamma_delta" \n"\DF{\alpha,\beta,\gamma,\delta}\D" "x0,x1,x2,x3" \n"(F_ALPHA_BETA_GAMMA_DELTA) \n"alpha=2,beta=3,gamma=4,delta=5" \n"(\alpha=2,\beta=3,\gamma=4,\delta=5)" \n-override_group 6 2 \n"1,5,0,0,3,6,0,0,1,1,3,0,5,5,0,3, \n1,0,2,5,0,1,6,1,0,0,3,5,0,0,4,4" \n-end \n-end \n-with F_2345 -do \n-cubic_surface_activity \n-report \n-report_with_group \n-end 
\pdflatex surface_equation_F_alpha_beta_gamma_delta_q7_report.tex 
\open surface_equation_F_alpha_beta_gamma_delta_q7_report.pdf 
\pdflatex surface_equation_F_alpha_beta_gamma_delta_q7_with_group.tex 
\open surface_equation_F_alpha_beta_gamma_delta_q7_with_group.pdf 
# cubic surfaces with 15 lines: 
\F_alpha_beta_gamma_delta_sweep_4_27: 

F_alpha_beta_gamma_delta_sweep_4_27:
6651 $\text{(ORBITER)} -v 3 \$
6652 \text{-define F -finite_field -q 3 -end} \$
6653 \text{-define P -projective_space -n 3 -field F -v 0 -end} \$
6654 \text{-with P -do} \$
6655 \text{-projective_space_activity} \$
6656 \text{-sweep}_4.15\_lines \text{sweep}_4.15\_lines\_q3 -q 3 \$
6657 \text{-by_equation } "\text{F}_alpha\_beta\_gamma\_delta" \$
6658 \"DF\{\alpha,\beta,\gamma,\delta}\" \$
6659 \"x0,x1,x2,x3" \$
6660 \$\{\text{F_ALPHA_BETA_GAMMA_DELTA}\} \$
6661 \"alpha=2,\beta=3,\gamma=4,\delta=5}\" \$
6662 \"D\alpha=2,\beta=3,\gamma=4,\delta=5\" \$
6663 \text{-end} \$
6664 \text{-end} \$
6665
6666 # cubic surfaces with 15 lines:
6667
6668 \text{surface}_15\_lines\_q7.1:}
6669 \$(\text{ORBITER}) -v 3 \$
6670 \text{-define F -finite_field -q 7 -end} \$
6671 \text{-define P -projective_space -n 3 -field F -v 0 -end} \$
6672 \text{-with P -do} \$
6673 \text{-projective_space_activity} \$
6674 \text{-control_six_arcs -end} \$
6675 \text{-define_surface S -q 7} \$
6676 \text{-by_equation } "\text{F}_alpha\_beta\_gamma\_delta" \$
6677 \"DF\{\alpha,\beta,\gamma,\delta}\" \"x0,x1,x2,x3" \$
6678 \$\{\text{F_ALPHA_BETA_GAMMA_DELTA}\} \$
6679 \"alpha=6,\beta=4,\gamma=2,\delta=2}\" \$
6680 \"D\alpha=6,\beta=4,\gamma=2,\delta=2\" \$
6681 \text{-end} \$
6682 \text{-end} \$
6683 \text{-with S -do} \$
6684 \text{-cubic_surface_activity} \$
6685 \text{-report} \$
6686 \text{-end} \$
6687 \text{pdflatex surface_equation_F_alpha_beta_gamma_delta_q7_report.tex}
6688 \text{open surface_equation_F_alpha_beta_gamma_delta_q7_report.pdf}
6689 
6690 
6691 
6692 
6693 
6694 \text{F_alpha_beta_gamma_delta_sweep}_4.15\_lines\_q7:}
6695 \$(\text{ORBITER}) -v 3 \$
6696 \text{-define F -finite_field -q 7 -end} \$
6697 \text{-define P -projective_space -n 3 -field F -v 0 -end} \$
6698
with P -do \  
-define F -finite_field -q 49 -end \  
-define P -projective_space -n 3 -field F -v 0 -end \  
with P -do \  
-projective_space_activity \  
-classify_surfaces_with_double_sixes Surf -W -end \  
-end \  
-with Surf -do \  
-classification_of_cubic_surfaces_with_double_sixes_activity \  
-recognize \  
-q 49 \  
-by_equation "F_alpha_beta_gamma_delta" \  
"\DF\{\alpha,\beta,\gamma,\delta\}\D" "x0,x1,x2,x3" \  
$(F\{\alpha,\beta,\gamma,\delta\}) \  
$\alpha=2,\beta=1,\gamma=1,\delta=2$ \  
"\D\alpha=2,\beta=1,\gamma=1,\delta=2\D" \  
-end \  
-end \  
-end \  
-end \  
#User time: 0:30  
# 348 parameter sets  
#F_alpha_beta_gamma_delta_q7_points.txt  
#F_alpha_beta_gamma_delta_q7_sweep.csv  
#F_alpha_beta_gamma_delta_q7_sweep4_15_data.csv  
F_alpha_beta_gamma_delta_q7_recognize:  
$\$(ORBITER) -v 2 \  
-define F -finite_field -q 49 -end \  
-define P -projective_space -n 3 -field F -v 0 -end \  
with P -do \  
-projective_space_activity \  
-classify_surfaces_with_double_sixes Surf -W -end \  
-end \  
-with Surf -do \  
-classification_of_cubic_surfaces_with_double_sixes_activity \  
-recognize \  
-q 49 \  
-by_equation "F_alpha_beta_gamma_delta" \  
"\DF\{\alpha,\beta,\gamma,\delta\}\D" "x0,x1,x2,x3" \  
$(F\{\alpha,\beta,\gamma,\delta\}) \  
$\alpha=2,\beta=1,\gamma=1,\delta=2$ \  
"\D\alpha=2,\beta=1,\gamma=1,\delta=2\D" \  
-end \  
-end \  
-end \  
-end \  
surf49_recognize:  
$\$(ORBITER) -v 3 \  
-define F -finite_field -q 49 -end \
define P -projective_space -n 3 -field F -v 0 -end \
-define P -do \n-projective_space_activity \n-classify_surfaces_with_double_sixes Surf27 -W -end \n-classification_of_cubic_surfaces_with_double_sixes_activity \n-recognize 
-q 49 \n-by_coefficients "2,5,1,16,4,10,1,18,4,8,4,9,3,11,4,14" \n-end \n-end \n-end \n-print_symbols 

McKean.15lines_q7: 
$\text{ORBITER} -v 3 \n\text{-define F -finite_field -q 7 -end \n\text{-define P -projective_space -n 3 -field F -v 0 -end \n\text{-with P -do \n\text{-projective_space_activity \n\text{-define_surface S \n\text{-by_coefficients $(SURFACE\_MCKEAN\_15\_LINES) -q 7 \n\text{-end \n\text{-end \n\text{-print_symbols \n
\text{# pdflatex surface_by_coefficients_q7_report.tex} 
\text{# open surface_by_coefficients_q7_report.pdf} 

# 2 Eckardt points 

F.4.4.3.3.q7: 
$\text{ORBITER} -v 3 \n\text{-define F -finite_field -q 7 -end \n\text{-define P -projective_space -n 3 -field F -v 0 -end \n\text{-with P -do \n\text{-projective_space_activity \n\text{-define_surface -q 7 -by_equation \n\"F\_alpha\_beta\_gamma\_delta" \n\"x0, x1, x2, x3" \n\text{-by_coefficients $(F\_\alpha\_beta\_gamma\_delta) \n\"alpha=4, beta=4, gamma=3, delta=3" \n\text{-by_coefficients \"D\alpha=4, beta=4, gamma=3, delta=3\"} \n\text{-end \n
641
section 7.2: cubic surfaces and quartic curves:

quartic_curve_9_0_report:

# has 4 eckardt points

F_alpha_beta_gamma_delta_points.txt:

# open surface_equation_F_alpha_beta_gamma_delta_q7_report.pdf
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 
-with P -do 
-projective_space_activity 
-define_quartic_curve C -q 13 
-catalogue 0 -end 
-with C -do 
-quartic_curve_activity 
-report 
-end 
-with C -do 
-quartic_curve_activity 
-report 
-end 
-with C -do 
-quartic_curve_activity 
-report 
-end 
-with C -do 
-quartic_curve_activity 
-report 
-end 
-with C -do 
-quartic_curve_activity 
-report 
-end 
-with C -do 
-quartic_curve_activity 
-report 
-end 
-with C -do 
-quartic_curve_activity 
-report 
-end 
-with C -do 
-quartic_curve_activity 
-report 
-end 

quartic_curve_13_1_report:

$(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 
-define_quartic_curve C -q 13 
-catalogue 1 -end 
-with C -do 
-quartic_curve_activity 
-report 
-end 
-with C -do 
-quartic_curve_activity 
-report 
-end 
-with C -do 
-quartic_curve_activity 
-report 
-end 
-with C -do 
-quartic_curve_activity 
-report 
-end 

surface_4_0_quartic_curves:

$(ORBITER) -v 3 
-define F -finite_field -q 4 -end 
-define P -projective_space -n 3 -field F -v 0 -end 
-with P -do 

-projective_space_activity \n-define_surface S4_0 -q 4 -catalogue 0 -end \n-end \n-with S4_0 -do \n-cubic_surface_activity \n-report \n-report_with_group \n-all_quartic_curves \n-end \n-pdflatex surface_catalogue_q4_iso0_report.tex \n-open surface_catalogue_q4_iso0_report.pdf \n-pdflatex surface_catalogue_q4_iso0_with_group.tex \n-open surface_catalogue_q4_iso0_with_group.pdf \n-pdflatex surface_catalogue_q4_iso0_quartics.tex \n-open surface_catalogue_q4_iso0_quartics.pdf

# full del Pezzo surfaces:

quartic_curves_q7:

$q(ORBITER) \ -v 3 \n-list_arguments \n-define F -finite_field -q 7 -end \n-define P -projective_space -n 3 -field F -v 0 -end \n-loop L 0 $(NB_CUBIC_SURFACES_Q7) 1 \n-with P -do \n-projective_space_activity \n-define_surface S_%L -q 7 -catalogue %L -end \n-end_loop \n-print_symbols \n-loop L 0 $(NB_CUBIC_SURFACES_Q7) 1 \n-with S_%L -do \n-cubic_surface_activity \n-export_all_quartic_curves \n-end \n-end_loop \n-print_symbols

quartic_curves_q7_classify:

$q(ORBITER) \ -v 3 \n-list_arguments \n
define F -finite_field -q 7 -end \ndefine P -projective_space -n 2 -field F -v 0 -end \nwith P -do \n-projective_space_activity \nclassify_quartic_curves_with_substructure \nsurface_catalogue_q7_iso%d_quartics.csv \n$NB_CUBIC_SURFACES_Q7\ 3 4 quartic_curves_q7 \nend \nprint_symbols
#pdflatex quartic_curves_q7_canonical.tex
#open quartic_curves_q7_canonical.pdf

# no quartic curve

NB_CUBIC_SURFACES_Q13=4

quartic_curves_q13:
$\text{ORBITER}\ -v 3 \n-list_arguments \ndefine F -finite_field -q 13 -end \ndefine P -projective_space -n 3 -field F -v 0 -end \nloop L 0 $(NB_CUBIC_SURFACES_Q13) 1 \n-with P -do \n-projective_space_activity \nDEFINE_surface S13_%L -q 13 -catalogue %L -end \nend_loop \nprint_symbols
loop L 0 $(NB_CUBIC_SURFACES_Q13) 1 \n-with S13_%L -do \ncubic_surface_activity \n-export_all_quartic_curves \nend \nend_loop \nprint_symbols

#surface_catalogue_q13_iso0_quartics.csv

quartic_curves_q13classify:
$\text{ORBITER}\ -v 3 \n-list_arguments \ndefine F -finite_field -q 13 -end \ndefine P -projective_space -n 2 -field F -v 0 -end \n-with P -do \n-projective_space_activity \n
# no quartic curve

NB_CUBIC_SURFACES_Q13=4

quartic_curves_q13:
$\text{ORBITER}\ -v 3 \n-list_arguments \ndefine F -finite_field -q 13 -end \ndefine P -projective_space -n 3 -field F -v 0 -end \nloop L 0 $(NB_CUBIC_SURFACES_Q13) 1 \n-with P -do \n-projective_space_activity \nDEFINE_surface S13_%L -q 13 -catalogue %L -end \nend_loop \nprint_symbols
loop L 0 $(NB_CUBIC_SURFACES_Q13) 1 \n-with S13_%L -do \ncubic_surface_activity \n-export_all_quartic_curves \nend \nend_loop \nprint_symbols
# quartic curves q13
# The number of types of quartic curves is 2
# idx : ago
#0 : 24
#1 : 48

NB_CUBIC_SURFACES_Q13=7

quartic_curves_q17:
$(ORBITER) -v 3 \
-define F -finite_field -q 17 -end \
-define P -projective_space -n 3 -field F -v 0 -end \
-loop L 0 $(NB_CUBIC_SURFACES_Q17) 1 \
-with P -do \
-projective_space_activity \
-define_surface S17,%L -q 17 -catalogue %L -end \
-end \
-end_loop \
-print_symbols \
-loop L 0 $(NB_CUBIC_SURFACES_Q17) 1 \
-with S17,%L -do \
-cubic_surface_activity \
-report \
-export_all_quartic_curves \
-end \
-end_loop \
-print_symbols \
#pdflatex surface_catalogue_q17_iso%d_quartics.csv \
#open surface_catalogue_q17_iso%d_report.tex 

quartic_curves_q17_classify:
$(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

classify_quartic_curves_with_substructure:

defined_surface_catalogue_q17_iso%d_quartics.csv

$(NB_CUBIC_SURFACES_Q17) 3 4 quartic_curves_q17

-end

-print_symbols

#User time: 2:33

#q17
The number of isomorphism types of quartic curves is 7

idx : ago

#0 : 24

#1 : 24

#2 : 4

#3 : 96

#4 : 6

#5 : 8

#6 : 2

NB_CUBIC_SURFACES_Q19=10

quartic_curves_q19:

$($ORBITER) -v 3 

-list_arguments

-define F -finite_field -q 19 -end

-define P -projective_space -n 3 -field F -v 0 -end

-loop L 0 $(NB_CUBIC_SURFACES_Q19) 1

-with P -do

-projective_space_activity

-define_surface S19_%L -q 19 -catalogue %L -end

-end

-end_loop

-print_symbols

-loop L 0 $(NB_CUBIC_SURFACES_Q19) 1

-with S19_%L -do

-cubic_surface_activity

-report

-export_all_quartic_curves

-end

-end_loop

-print_symbols

pdf_latex surface_catalogue_q19_iso0_report.tex

open surface_catalogue_q19_iso0_report.pdf
quartic_curves_q19.classify:

```bash
$\text{ORBITER} -v 3 \\
-list_arguments \\
-define F -finite_field -q 19 -end \\
-define P -projective_space -n 2 -field F -v 0 -end \\
-with P -do \\
-projective_space_activity \\
-classify_quartic_curves_with_substructure \\
-surface_catalogue_q19_iso%d_quartics.csv \\
-define $(NB_CUBIC_SURFACES_Q19) 4 4 quartic_curves_q19 \\
-end \\
-print_symbols
```

# writes:
quartic_curves_q19_canonical_data.csv
quartic_curves_q19_canonical.tex

# 14 isomorphism types:
ago dist: 4^1, 9^1, 2^4, 6^2, 8^3, 24^3

quartic_curves_q19_set_stabilizer:

```bash
$\text{ORBITER} -v 3 \\
-list_arguments \\
-define F -finite_field -q 19 -end \\
-define P -projective_space -n 2 -field F -v 0 -end \\
-with P -do \\
-projective_space_activity \\
-set_stabilizer 4 \\
-surface_catalogue_q19_iso%d_quartics.csv \\
-define $(NB_CUBIC_SURFACES_Q19) "pts_on_curve" \\
-end \\
-print_symbols
```

surface_13_0_quartics:

```bash
$\text{ORBITER} -v 3 \\
-define F -finite_field -q 13 -end \\
-define P -projective_space -n 3 -field F -v 0 -end \\
-with P -do \\
-projective_space_activity \\
-define_surface S13_0 -q 13 -catalogue 0 -end \\
-end \\
-with S13_0 -do \\
-cubic_surface_activity \\
```

648
surface_13_1_quartics:

$\texttt{(ORBITER)} -v 3 \$

-define F -finite_field -q 13 -end \\
-define P -projective_space -n 3 -field F -v 0 -end \\
-with P -do \\
-projective_space_activity \\
-define_surface S13_1 -q 13 -catalogue 1 -end \\
-with S13_1 -do \\
cubic_surface_activity \\
-define_surface_activity \\
-export_all_quartic_curves \\
-end \\
-pdflatex surface_catalogue_q13_iso0_report.tex \\
-open surface_catalogue_q13_iso0_report.pdf \\

quartic_curve_13_2_group:

$\texttt{(ORBITER)} -v 3 \$

-define G -linear_group -PGL 3 13 \\
-subgroup_by_generators "quartic_13_2" 24336 5 \\
-define "1,0,0,0,1,0,1,2,12, \$

1,0,0,1,7,0,11,0,7, \\
1,0,0,8,12,1,4,11,1, \\
1,0,0,11,4,1,2,10,0, \\
0,1,0,11,3,0,4,9,1" \\
-end \\
-define Gr -modified_group -from G \\
-restricted_action "29,86,97,154" \\
-end \\
-with Gr -do \\
-group_theoretic_activity \\
-report \\
-end \\
-pdflatex PGL_3_13_Subgroup_quartic_13_2_24336_report.tex \\
-open PGL_3_13_Subgroup_quartic_13_2_24336_report.pdf
surface_25.0:

$\$(ORBITER) -v 3 \\
$\$ -define F -finite_field -q 25 -end \\
$\$ -define P -projective_space -n 3 -field F -v 0 -end \\
$\$ -with P -do \\
$\$ -projective_space_activity \\
$\$ -define_surface S25_0 -q 25 -catalogue 0 -end \\
$\$ -end \\
$\$ -with S25_0 -do \\
$\$ -cubic_surface_activity \\
$\$ -report \\
$\$ -export_all_quartic_curves \\
$\$ -end \\
$\$ pdflatex surface_catalogue_q25_iso0_quartics.tex \\
$\$ open surface_catalogue_q25_iso0_quartics.pdf \\

quartic_curve_25_report:

$\$(ORBITER) -v 3 \\
$\$ -define F -finite_field -q 25 -end \\
$\$ -define P -projective_space -n 2 -field F -v 0 -end \\
$\$ -loop L 0 18 1 \\
$\$ -with P -do \\
$\$ -projective_space_activity \\
$\$ -define_quartic_curve QC25_%L \\
$\$ -q 25 -catalogue %L -end \\
$\$ -end_loop \\
$\$ -print_symbols \\
$\$ -loop L 0 18 1 \\
$\$ -with QC25_%L -do \\
$\$ -quartic_curve_activity \\
$\$ -report \\
$\$ -end_loop \\
$\$ -print_symbols 

$\$ pdflatex quartic_curve_catalogue_q25_iso0.report.tex \\
$\$ pdflatex quartic_curve_catalogue_q25_iso1.report.tex \\
$\$ pdflatex quartic_curve_catalogue_q25_iso2.report.tex \\
$\$ pdflatex quartic_curve_catalogue_q25_iso3.report.tex \\
$\$ pdflatex quartic_curve_catalogue_q25_iso4.report.tex \\
$\$ pdflatex quartic_curve_catalogue_q25_iso5.report.tex \\
$\$ pdflatex quartic_curve_catalogue_q25_iso6.report.tex
7215 ▶ pdflatex quartic_curve_catalogue_q25_iso7_report.tex
7216 ▶ pdflatex quartic_curve_catalogue_q25_iso8_report.tex
7217 ▶ pdflatex quartic_curve_catalogue_q25_iso9_report.tex
7218 ▶ pdflatex quartic_curve_catalogue_q25_iso10_report.tex
7219 ▶ pdflatex quartic_curve_catalogue_q25_iso11_report.tex
7220 ▶ pdflatex quartic_curve_catalogue_q25_iso12_report.tex
7221 ▶ pdflatex quartic_curve_catalogue_q25_iso13_report.tex
7222 ▶ pdflatex quartic_curve_catalogue_q25_iso14_report.tex
7223 ▶ pdflatex quartic_curve_catalogue_q25_iso15_report.tex
7224 ▶ pdflatex quartic_curve_catalogue_q25_iso16_report.tex
7225 ▶ pdflatex quartic_curve_catalogue_q25_iso17_report.tex
7226 ▶ gs -sDEVICE=pdfwrite -r120 -o quartic_curve_catalogue_q25.pdf \
7227 ▶ quartic_curve_catalogue_q25_iso0_report.pdf \
7228 ▶ quartic_curve_catalogue_q25_iso1_report.pdf \
7229 ▶ quartic_curve_catalogue_q25_iso2_report.pdf \
7230 ▶ quartic_curve_catalogue_q25_iso3_report.pdf \
7231 ▶ quartic_curve_catalogue_q25_iso4_report.pdf \
7232 ▶ quartic_curve_catalogue_q25_iso5_report.pdf \
7233 ▶ quartic_curve_catalogue_q25_iso6_report.pdf \
7234 ▶ quartic_curve_catalogue_q25_iso7_report.pdf \
7235 ▶ quartic_curve_catalogue_q25_iso8_report.pdf \
7236 ▶ quartic_curve_catalogue_q25_iso9_report.pdf \
7237 ▶ quartic_curve_catalogue_q25_iso10_report.pdf \
7238 ▶ quartic_curve_catalogue_q25_iso11_report.pdf \
7239 ▶ quartic_curve_catalogue_q25_iso12_report.pdf \
7240 ▶ quartic_curve_catalogue_q25_iso13_report.pdf \
7241 ▶ quartic_curve_catalogue_q25_iso14_report.pdf \
7242 ▶ quartic_curve_catalogue_q25_iso15_report.pdf \
7243 ▶ quartic_curve_catalogue_q25_iso16_report.pdf \
7244 ▶ quartic_curve_catalogue_q25_iso17_report.pdf
7245 ▶ # quartic_curve_catalogue_q25_iso0_report.pdf
7246 ▶ # quartic_curve_catalogue_q25_iso0_report.pdf
7247
7248 quartic_curve_13_table:
7249   $(ORBITER) -v 3 \
7250   -define F -finite_field -q 13 -end \
7251   -define P -projective_space -n 2 -field F -v 0 -end \
7252   -with P -do \
7253   -projective_space_activity \
7254   -table_of_quartic_curves \
7255   -end
7256
7257
7258
7259 # quartic_curves_q13_info.csv
7260
7261
quartic_curve_19_table:
    $(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_19_table_latex:
    $(ORBITER) -v 3 \
    -csv_file_latex 1 quartic_curves_q19_info.csv 
    ~/bin/tth quartic_curves_q19_info.tex

quartic_curve_25_table:
    $(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

quartic_curve_27_table:
    $(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

quartic_curve_29_table:
    $(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 \

# quartic_curves_q25_info.csv

# quartic_curves_q27_info.csv

# quartic_curves_q29_info.csv

# quartic_curves_q29_info.csv

quartic_curve_31_table:

$\text{(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_q31_info.csv

# quartic_curves_q25_info.csv

surface_25.12:

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

-define F -finite_field -q 25 -end \\
-define P -projective_space -n 3 -field F -v 0 -end \\
-with P -do \\
-projective_space_activity \\
-define_surface S25.12 -q 25 -catalogue 12 -end \\
-end \\
-with S25.12 -do \\
-cubic_surface_activity \\
-report \\
-report_with_group \\
-end

pdflatex surface_catalogue_q25.iso12.with_group.tex

open surface_catalogue_q25.iso12.with_group.pdf

surface_25.12_t1:

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

-define F -finite_field -q 25 -end \\
-define P -projective_space -n 3 -field F -v 0 -end \\
-with P -do \\
-projective_space_activity \\
-define_surface S25.12 -q 25 -catalogue 12 \\
-transform "1,0,0,16, 0,1,0,18, 0,0,1,8, 0,0,1,1, 0" \\
-end \\
-end \\

653
with S25,12
-cubic_surface_activity

define F : finite_field -q 25
define P : projective_space -n 3 -field F -v 0
with P
-projective_space_activity

define_surface S25,12 -q 25 -catalogue 12
-transform "1,0,0,16, 0,1,0,18, 0,0,1,8, 0,0,1,1, 0"
-transform_inverse "16,0,1,0, 3,5,1,0, 0,0,1,0, 0,0,0,1, 0"
-transform "3,0,0,0, 0,1,0,0, 0,0,1,0, 0,0,0,1, 0"
-end
-with S25,12
-cubic_surface_activity

define_surface S25,12 -q 25 -catalogue 12
-transform "1,0,0,16, 0,1,0,18, 0,0,1,8, 0,0,1,1, 0"
-transform_inverse "16,0,1,0, 3,5,1,0, 0,0,1,0, 0,0,0,1, 0"
-transform "3,0,0,0, 0,1,0,0, 0,0,1,0, 0,0,0,1, 0"
-end
-with S25,12
-cubic_surface_activity

$\{(ORBITER) -v 3 \$

$\{\text{pdflatex surface\_catalogue\_q25\_iso12\_with\_group.tex}$

$\{\text{pdflatex surface\_catalogue\_q25\_iso12\_with\_group.pdf}$

$\{\text{pdflatex surface\_catalogue\_q25\_iso12\_with\_group.tex}$

$\{\text{pdflatex surface\_catalogue\_q25\_iso12\_with\_group.pdf}$
open surface_catalogue_q25_iso12_with_group.pdf

surface_25_12_t4:

$\$(ORBITER) -v 3 \\n
-define F -finite_field -q 25 -end \\n
-define P -projective_space -n 3 -field F -v 0 -end \\n
-with P -do \\n
-projective_space_activity \\n
-define_surface S25_12 -q 25 -catalogue 12 \\n
-transform "1,0,0,16, 0,1,0,18, 0,0,1,8, 0,0,1,1, 0" \\n
-transform_inverse "16,0,1,0, 3,5,1,0, 0,0,1,0, 0,0,0,1, 0" \\n
-transform "3,0,0,0, 0,1,0,0, 0,0,1,0, 0,0,0,1, 0" \\n
-transform_inverse "1,0,0,0, 0,1,0,0, 0,0,1,0, 13,2,2,1, 0" \\n
-end \\n
-with S25_12 -do \\n
-cubic_surface_activity \\n
-report \\n
-report_with_group \\n
-end \\n
pdflatex surface_catalogue_q25_iso12_with_group.tex \\n
open surface_catalogue_q25_iso12_with_group.pdf

surface_25_12_t5:

$\$(ORBITER) -v 3 \\n
-define F -finite_field -q 25 -end \\n
-define P -projective_space -n 3 -field F -v 0 -end \\n
-with P -do \\n
-projective_space_activity \\n
-define_surface S25_12 -q 25 -catalogue 12 \\n
-transform "1,0,0,16, 0,1,0,18, 0,0,1,8, 0,0,1,1, 0" \\n
-transform_inverse "16,0,1,0, 3,5,1,0, 0,0,1,0, 0,0,0,1, 0" \\n
-transform "3,0,0,0, 0,1,0,0, 0,0,1,0, 0,0,0,1, 0" \\n
-transform_inverse "1,0,0,0, 0,1,0,0, 0,0,1,0, 13,2,2,1, 0" \\n
-transform_inverse "1,0,0,0, 0,1,0,0, 13,1,1,0, 0,0,0,1, 0" \\n
-end \\n
-end \\n
-with S25_12 -do \\n
-cubic_surface_activity \\n
-report \\n
-report_with_group \\n
-end \\n
pdflatex surface_catalogue_q25_iso12_with_group.tex \\n
open surface_catalogue_q25_iso12_with_group.pdf

655
PG_2.25:

```latex
\$\text{\textbackslash (ORBITER)} \ \ \ \end$
```

```
\textend{\textbackslash -define P -projective_space -n 2 -field F -v 0 -end \}
```

```
\textend{\textbackslash -with P -do -projective_space_activity -cheat_sheet -end}
```

```
pdflatex PG_2.25.tex
```

```
open PG_2.25.pdf
```

PG_2.25_lines:

```
\$\text{\textbackslash (ORBITER)} -v 5 \end$
```

```
\textend{\textbackslash -orbiter_path \$(ORBITER_PATH) \end$
```

```
\textend{\textbackslash -define G -linear_group -PGGL 3 25 -end \}
```

```
\textend{\textbackslash -define G_on_lines -modified_group -from G \}
```

```
\textend{\textbackslash -on_k_subspaces 2 \end$
```

```
\textend{\textbackslash -with G_on_lines -do \end$
```

```
\textend{\textbackslash -group_theoretic_activity \end$
```

```
\textend{\textbackslash -poset_classification_control \end$
```

```
\textend{\textbackslash -problem_label PGGL_3.25 \end$
```

```
\textend{\textbackslash -depth 3 -draw_poset -draw_options -radius 200 -end -report -end \end$
```

```
\textend{\textbackslash -recognize "0,25,650" \end$
```

```
\textend{\textbackslash -recognize "430,16,364" \end$
```

```
\textend{\textbackslash -orbits_on_subsets 3 \end$
```

```
\textend{\textbackslash -report \end$
```

```
\textend{\textbackslash -end
```

```
pdflatex PGGL_3.25_poset.tex
```

```
open PGGL_3.25_poset.pdf
```

surface_25_12_t6:

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

```
\textend{\textbackslash -define F -finite_field -q 25 -end \end$
```

```
\textend{\textbackslash -define P -projective_space -n 3 -field F -v 0 -end \end$
```

```
\textend{\textbackslash -with P -do \end$
```

```
\textend{\textbackslash -projective_space_activity \end$
```

```
\textend{\textbackslash -define_surface S25_12 -q 25 -catalogue 12 \end$
```

```
\textend{\textbackslash -transform "1,0,0,16, 0,1,0,18, 0,0,1,8, 0,0,1,1, 0" \end$
```

```
\textend{\textbackslash -transform_inverse "16,0,1,0, 3,5,1,0, 0,0,1,0, 0,0,0,1, 0" \end$
```

```
\textend{\textbackslash -transform "3,0,0,0, 0,1,0,0, 0,0,1,0, 0,0,0,1, 0" \end$
```

```
\textend{\textbackslash -transform_inverse "1,0,0,0, 0,1,0,0, 13,2,2,1, 0" \end$
```

```
\textend{\textbackslash -transform_inverse "1,0,0,0, 0,1,0,0, 13,1,1,0, 0,0,0,1, 0" \end$
```

```
\textend{\textbackslash -transform "3,8,8,0, 22,13,2,0, 14,19,15,0, 0,0,0,1, 1" \end$
```

```
\textend{\textbackslash -transform_inverse "16,0,0,0, 0,16,0,0, 21,21,21,0, 0,0,0,1, 0" \end$
```

656
7497 \* -end \ 
7498 \* -end \ 
7499 \* with S25_12 -do \ 
7500 \* -cubic_surface_activity \ 
7501 \* -report \ 
7502 \* -report_with_group \ 
7503 \* -end \ 
7504 \* pdflatex surface_catalogue_q25_iso12_with_group.tex \ 
7505 \* open surface_catalogue_q25_iso12_with_group.pdf \ 
7506 \* \ 
7507 \* \ 
7508 \* PG_2.25_stab_of_triangle:\ 
7509 \* $(ORBITER) -v 5 \ 
7510 \* -orbiter_path $(ORBITER_PATH) \ 
7511 \* -define G -linear_group -PGGL 3 25 \ 
7512 \* -subgroup_by_generators "triangle_stab" 6912 7 \ 
7513 \* "1,0,0,0,1,0,0,0,1,1, \ 
7514 \* 1,0,0,0,13,0,0,0,13,1, \ 
7515 \* 1,0,0,0,4,0,0,0,6,1, \ 
7516 \* 1,0,0,0,17,0,0,0,13,0, \ 
7517 \* 1,0,0,0,18,0,0,0,4,1, \ 
7518 \* 1,0,0,0,0,11,0,1,0,0, \ 
7519 \* 0,1,0,0,0,20,14,0,0,0" \ 
7520 \* -end \ 
7521 \* -do \ 
7522 \* -poset_classification_control \ 
7523 \* -problem_label PGGL.3.25 \ 
7524 \* -depth 3 -draw_poset -draw_options -radius 200 -end \ 
7525 \* -recognize "8,44,226" \ 
7526 \* -end \ 
7527 \* -orbits_on_subsets 3 \ 
7528 \* -end \ 
7529 \* \ 
7530 \* \ 
7531 \* \ 
7532 \* \ 
7533 \* surface_25.12_t7:\ 
7534 \* $(ORBITER) -v 3 \ 
7535 \* -finite_field -q 25 -end \ 
7536 \* -define P -projective_space -n 3 -field F -v 0 -end \ 
7537 \* -with P -do \ 
7538 \* -projective_space_activity \ 
7539 \* -define_surface S25_12 -q 25 -catalogue 12 \ 
7540 \* -transform "1,0,0,16, 0,1,0,18, 0,0,1,8, 0,0,1,1, 0" \ 
7541 \* -transform_inverse "16,0,1,0, 3,5,1,0, 0,0,1,0, 0,0,1,0, 0" \ 
7542 \* -transform "3,0,0,0, 0,1,0,0, 0,0,1,0, 0,0,0,1, 0" \ 
7543 \* -end
\begin{verbatim}
7544 ➤ ➤ ➤ -transform_inverse "1,0,0,0, 0,1,0,0, 0,0,1,0, 13,2,2,1, 0" \ 7545 ➤ ➤ ➤ -transform_inverse "1,0,0,0, 0,1,0,0, 13,1,1,0, 0,0,0,1, 0" \ 7546 ➤ ➤ ➤ -transform "3,8,8,0, 22,13,22,0, 14,19,15,0, 0,0,0,1, 1" \ 7547 ➤ ➤ ➤ -transform_inverse "16,0,0,0, 0,16,0,0, 21,21,21,0, 0,0,0,1, 0" \ 7548 ➤ ➤ ➤ -transform "1,0,0,0, 0,5,0,0, 0,0,17,0, 0,0,0,1, 1" \ 7549 ➤ ➤ ➤ -end \ 7550 ➤ ➤ ➤ -end \ 7551 ➤ ➤ ➤ -with S25_12 -do \ 7552 ➤ ➤ ➤ -cubic_surface_activity \ 7553 ➤ ➤ ➤ -report \ 7554 ➤ ➤ ➤ -report_with_group \ 7555 ➤ ➤ ➤ -end
7556 ➤ pdflatex surface_catalogue_q25_iso12_with_group.tex
7557 ➤ open surface_catalogue_q25_iso12_with_group.pdf
7558
7559
7560
7561 surface_25_12_t8:
7562 ➤ $(ORBITER) -v 3 \n7563 ➤ ➤ -define F -finite_field -q 25 -end \n7564 ➤ ➤ -define P -projective_space -n 3 -field F -v 0 -end \n7565 ➤ ➤ -with P -do \n7566 ➤ ➤ -projective_space_activity \n7567 ➤ ➤ ➤ -define_surface S25_12 -q 25 -catalogue 12 \n7568 ➤ ➤ ➤ -transform "1,0,0,16, 0,1,0,18, 0,0,1,8, 0,0,1,1, 0" \n7569 ➤ ➤ ➤ -transform_inverse "16,0,1,0, 3,5,1,0, 0,0,1,0, 0,0,0,1, 0" \n7570 ➤ ➤ ➤ -transform "3,0,0,0, 0,1,0,0, 0,0,1,0, 0,0,0,1, 0" \n7571 ➤ ➤ ➤ -transform_inverse "1,0,0,0, 0,1,0,0, 0,0,1,0, 13,2,2,1, 0" \n7572 ➤ ➤ ➤ -transform_inverse "1,0,0,0, 0,1,0,0, 13,1,1,0, 0,0,0,1, 0" \n7573 ➤ ➤ ➤ -transform "3,8,8,0, 22,13,22,0, 14,19,15,0, 0,0,0,1, 1" \n7574 ➤ ➤ ➤ -transform_inverse "16,0,0,0, 0,16,0,0, 21,21,21,0, 0,0,0,1, 0" \n7575 ➤ ➤ ➤ -transform "1,0,0,0, 0,5,0,0, 0,0,17,0, 0,0,0,1, 1" \n7576 ➤ ➤ ➤ -transform "1,0,0,0, 0,1,0,0, 0,0,1,0, 0,0,0,24, 0" \n7577 ➤ ➤ ➤ -end \n7578 ➤ ➤ ➤ -end \n7579 ➤ ➤ ➤ -with S25_12 -do \n7580 ➤ ➤ ➤ -cubic_surface_activity \n7581 ➤ ➤ ➤ -report \n7582 ➤ ➤ ➤ -report_with_group \n7583 ➤ ➤ ➤ -end
7584 ➤ pdflatex surface_catalogue_q25_iso12_with_group.tex
7585 ➤ open surface_catalogue_q25_iso12_with_group.pdf
7586
7587
7588 surface_25_12_t9:
7589 ➤ $(ORBITER) -v 3 \n7590 ➤ ➤ -define F -finite_field -q 25 -end \n\end{verbatim}
\begin{verbatim}
defined P -projective_space -n 3 -field F -v 0 -end \nwith P -do 
-define P -projective_space_activity 
-define surface S25.12 -q 25 -catalogue 12 
-define P -projective_space -n 3 -field F -v 0 -end 
-with P -do 
-projective_space_activity 
-define surface S25.12 -q 25 -catalogue 12 
-transform "1,0,0,16, 0,1,0,18, 0,0,1,8, 0,0,1,1, 0" 
-transform_inverse "16,0,1,0, 3,5,1,0, 0,0,1,0, 0,0,0,1, 0" 
-transform "3,0,0,0, 0,1,0,0, 0,0,1,0, 0,0,0,1, 0" 
-transform_inverse "1,0,0,0, 0,1,0,0, 0,0,1,0, 13,2,2,1, 0" 
-transform_inverse "1,0,0,0, 0,1,0,0, 13,1,1,0, 0,0,0,1, 0" 
-transform "3,8,8,0, 22,13,22,0, 14,19,15,0, 0,0,0,1, 1" 
-transform_inverse "1,0,0,0, 0,1,0,0, 0,0,17,0, 0,0,0,1, 0" 
-transform "1,0,0,0, 0,1,0,0, 0,0,1,0, 0,0,0,24, 0" 
-transform "1,0,0,0, 0,1,0,0, 0,0,1,0, 1,0,0,0, 0" 
-end 
end 
with S25.12 -do 
cubic_surface_activity 
-report 
-report_with_group 
-end 
pdflatex surface_catalogue_q25.iso12_with_group.tex 
open surface_catalogue_q25.iso12_with_group.pdf 
end 
end 
with S25.12 -do 
cubic_surface_activity 
-report 
-report_with_group 
-end 
pdflatex surface_catalogue_q25.iso12_with_group.tex 
open surface_catalogue_q25.iso12_with_group.pdf 

\end{verbatim}
7638 ▶ ▶ -cubic_surface_activity \n7639 ▶ ▶ ▶ -all_quartic_curves \n7640 ▶ ▶ -end \n7641 ▶ ▶ -with S25_12 -do \n7642 ▶ ▶ -cubic_surface_activity \n7643 ▶ ▶ ▶ -export_all_quartic_curves \n7644 ▶ ▶ -end \n7645 ▶ pdflatex surface_catalogue_q25_iso12_quartics.tex \n7646 ▶ open surface_catalogue_q25_iso12_quartics.pdf \n7647 \n7648 \n7649 quartic_curve_13_0_surface: \n7650 ▶ $(ORBITER) -v 3 \n7651 ▶ ▶ -define F -finite_field -q 13 -end \n7652 ▶ ▶ -define P -projective_space -n 2 -field F -v 0 -end \n7653 ▶ ▶ -with P -do \n7654 ▶ ▶ ▶ -projective_space_activity \n7655 ▶ ▶ ▶ ▶ -define_quartic_curve QC13_0 -q 13 -catalogue 0 -end \n7656 ▶ ▶ ▶ -end \n7657 ▶ ▶ -with QC13_0 -do \n7658 ▶ ▶ ▶ -quartic_curve_activity \n7659 ▶ ▶ ▶ ▶ -create_surface \n7660 ▶ ▶ ▶ -end \n7661 \n7662 #surface equation: 0, 0, 9, 0, 3, 0, 1, 0, 12, 7, 0, 4, 4, 0, 7, 12, 6, 2, 5, 10 \n7663 #9,2,3,4,1,6,12,8,7,9,4,11,4,12,7,14,12,15,6,16,2,17,5,18,10,19 \n7664 \n7665 quartic_curve_13_0_surface_create: \n7666 ▶ $(ORBITER) -v 3 \n7667 ▶ ▶ -define F -finite_field -q 13 -end \n7668 ▶ ▶ -define P -projective_space -n 3 -field F -v 0 -end \n7669 ▶ ▶ -with P -do \n7670 ▶ ▶ -projective_space_activity \n7671 ▶ ▶ ▶ -define_surface S -by_coefficients \n7672 ▶ ▶ ▶ ▶ "9,2,3,4,1,6,12,8,7,9,4,11,4,12,7,14,12,15,6,16,2,17,5,18,10,19" \n7673 ▶ ▶ ▶ -q 13 -end \n7674 ▶ ▶ ▶ -end \n7675 ▶ ▶ -with S -do \n7676 ▶ ▶ -cubic_surface_activity \n7677 ▶ ▶ -report \n7678 ▶ ▶ -end \n7679 ▶ pdflatex surface_by_coefficients_q13_report.tex \n7680 ▶ open surface_by_coefficients_q13_report.pdf \n7681 \n7682 #▷ ▶ -report_with_group \n7683 \n7684 ###############################################################################
# Section 7.3: Classification of Cubic Surfaces with 27 lines

## Section Classification of Cubic Surfaces With 27 Lines:

Surface classify q4:

```latex
\texttt{\$ (ORBITER) -v 5 \$
\texttt{\$ -define F -finite_field -q 4 -end \$
\texttt{\$ -define P -projective_space -n 3 -field F -v 0 -end \$
\texttt{\$ -with P -do \$
\texttt{\$ -projective_space_activity \$
\texttt{\$ -classify_surfaces_with_double_sixes Surf27 -W -end \$
\texttt{\$ -end \$
\texttt{\$ -with Surf27 -do \$
\texttt{\$ -classification_of_cubic_surfaces_with_double_sixes \$
\texttt{\$ -report -end \$
\texttt{\$ -print_symbols \$
\texttt{\$ pdflatex Surfaces_q4.tex \$
\texttt{\$ open Surfaces_q4.pdf \$
```

# time: 0:00

Surface classify q7:

```latex
\texttt{\$ (ORBITER) -v 5 \$
\texttt{\$ -define F -finite_field -q 7 -end \$
\texttt{\$ -define P -projective_space -n 3 -field F -v 0 -end \$
```

661
-with P -do \
-projective_space_activity \
-classify_surfaces_with_double_sixes Surf27 -W -end \
-end \
-with Surf27 -do \
-classification_of_cubic_surfaces_with_double_sixes_activity \
-report -end \
-end \
-with Surf27 -do \
-classification_of_cubic_surfaces_with_double_sixes_activity \
-report -end \
-end \
-print_symbols 
open Surfaces_q7.pdf 

surface_classify_q9: 
$\text{(ORBITER) -v 5} \\
\text{-define } F \text{ -finite_field -q 9 -end} \\
\text{-define } P \text{ -projective_space -n 3 -field } F \text{ -v 0 -end} \\
\text{-with } P \text{ -do} \\
\text{-projective_space_activity} \\
\text{-classify_surfaces_with_double_sixes Surf27 -W -end} \\
-end \\
-with Surf27 -do \
-classification_of_cubic_surfaces_with_double_sixes_activity \
-report -end \
-end \
-print_symbols 
open Surfaces_q7.pdf 

surface_classify_q13: 
$\text{(ORBITER) -v 5} \\
\text{-define } F \text{ -finite_field -q 13 -end} \\
\text{-define } P \text{ -projective_space -n 3 -field } F \text{ -v 0 -end} \\
\text{-with } P \text{ -do} \\
\text{-projective_space_activity} \\
\text{-classify_surfaces_with_double_sixes C -W -end} \\
-end \\
-with C -do \
-classification_of_cubic_surfaces_with_double_sixes_activity \
-report -end \
-end \
-print_symbols 
open Surfaces_q13.pdf
# Section 7.4: Cubic Surfaces - Isomorphism Testing and Recognition

## SECTION_CUBIC_SURFACES_ISOMORPHISM_TESTING_AND_RECOGNITION:

```bash
$ (ORBITER) -v 3

$ (ORBITER) -v 3

define F -finite_field -q 7 -end

define P -projective_space -n 3 -field F -v 0 -end

with P -do

-projective_space_activity

> -classify_surfaces_with_double_sixes Surf -W -end

> -end

> -with Surf -do

> -classification_of_cubic_surfaces_with_double_sixes_activity

> -recognize

> -q 7

> -family_general_abcd 2 3 3 4

> -end

> -end

> -end

> $ (ORBITER) -v 3

define F -finite_field -q 16 -end

define P -projective_space -n 3 -field F -v 0 -end

with P -do

-projective_space_activity

> -classify_surfaces_with_double_sixes Surf27 -W -end

> -end

> -with Surf27 -do

> -classification_of_cubic_surfaces_with_double_sixes_activity

> -isomorphism_testing

> -q 16 -by.coefficients

"1,5,1,8,1,9,1,10,1,11,1,12,6,14,6,15,7,18,7,19" -end

"13,6,3,8,3,11,13,13,1,19" -end

> -end

> -end

> -print_symbols
```

663
# 1 min 8 sec on Mac from scratch (with all data files removed)

```
surface_recognize_8:
    $(ORBITER) -v 3 \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    -classify_surfaces_with_double_sixes Surf27 -W -end \n    -end \n    -with Surf27 -do \n    -classification_of_cubic_surfaces_with_double_sixes.activity \n    -recognize \n    -q 8 \n    -by_coefficients "1,6,1,8,1,11,1,13,1,19" \n    -end \n    -end \n    -end \n    -print_symbols
```

```
surface_recognize_F13_q4:
    $(ORBITER) -v 3 \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    -classify_surfaces_with_double_sixes Surf27 -W -end \n    -end \n    -with Surf27 -do \n    -classification_of_cubic_surfaces_with_double_sixes.activity \n    -identify_F13 \n    -end \n    -print_symbols
```

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

F_sweep_15_q7: 
$(ORBITER) -v 20 
define F -finite_field -q 7 -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_q7 -q 7 
-by_equation "F_alpha_beta_gamma_delta" 
"DF_{\alpha,\beta,\gamma,\delta}\ D" "x0,x1,x2,x3" 
$(F_ALPHA_BETA_GAMMA_DELTA) 
"alpha=2,\beta=1,\gamma=2,\delta=3" 
"D\alpha=2,\beta=1,\gamma=2,\delta=3\ D" 
-end 
-end 

#0:29 
#F_alpha_beta_gamma_delta_q7_sweep4_15_data.csv 
#F_alpha_beta_gamma_delta_q7_sweep.csv 
#F_alpha_beta_gamma_delta_q7_points.txt 

SECTION_CUBIC_SURFACES_DICKSON: 

D6_q2: 
$(ORBITER) -v 3 
define F -finite_field -q 2 -end 
define P -projective_space -n 3 -field F -v 0 -end 
-with P -do 
-projective_space_activity 

# Section 7.5: Cubic Surfaces of Dickson type
\begin{verbatim}
 # 1 line over GF(2)

D3_q4:
$(ORBITER) -v 3 \\
$ -define F -finite_field -q 4 -end \\
$ -define P -projective_space -n 3 -field F -v 0 -end \\
$ -with P -do \\
$ -projective_space_activity \\
$ -define_surface S_D3_q4 -q 4 -by_coefficients $(D3) -end \\
$ -end \\
$ -with S_D3_q4 -do \\
$ -cubic_surface_activity \\
$ -report \\
$ -end \\
pdflatex surface_by_coefficients_q4_report.tex \\
open surface_by_coefficients_q4_report.pdf \\
mv surface_by_coefficients_q4_points.txt surface_by_coefficients_q4_D3_points.txt \\

#surface_by_coefficients_q4_points.txt

D4_q8:
$(ORBITER) -v 3 \\
$ -define F -finite_field -q 8 -end \\
$ -define P -projective_space -n 3 -field F -v 0 -end \\
$ -with P -do \\
$ -projective_space_activity \\
$ -define_surface S_D4_q8 -q 8 -by_coefficients $(D4) -end \\
$ -end \\
$ -with S_D4_q8 -do \\
\end{verbatim}
\texttt{D6.q4:}
\begin{verbatim}
$\texttt{(ORBITER) -v 3 \ }
$\texttt{-define F -finite_field -q 4 -end \ }
$\texttt{-define P -projective_space -n 3 -field F -v 0 -end \ }
$\texttt{-with P -do \ }
$\texttt{-projective_space_activity \ }
$\texttt{-define surface S.D6.q4 -q 4 -by_coefficients $(D6) -end \ }
$\texttt{-end \ }
$\texttt{-with S.D6.q4 -do \ }
$\texttt{-cubic_surface_activity \ }
$\texttt{-report \ }
$\texttt{-end \ }
\end{verbatim}

\texttt{D8.q4:}
\begin{verbatim}
$\texttt{(ORBITER) -v 3 \ }
$\texttt{-define F -finite_field -q 4 -end \ }
$\texttt{-define P -projective_space -n 3 -field F -v 0 -end \ }
$\texttt{-with P -do \ }
$\texttt{-projective_space_activity \ }
$\texttt{-define_surface S.D8.q4 -q 4 -by_coefficients $(D8) -end \ }
$\texttt{-end \ }
$\texttt{-with S.D8.q4 -do \ }
$\texttt{-cubic_surface_activity \ }
$\texttt{-report \ }
$\texttt{-end \ }
\end{verbatim}

# D6 has 7 lines over GF(4)
D1_q8:

```bash
$\text{ORBITER}$ -v 3 \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-define_surface S_{D1_q8} -q 8 -by_coefficients $(D1) -end \n-end \n-with S_{D1_q8} -do \n-cubic_surface_activity \n-report \n-end
```

```bash
pdflatex surface_by_coefficients_q4_report.tex
open surface_by_coefficients_q4_report.pdf
```

# surface_by_coefficients_q8_points.txt

```
## cleaning D1 with 15 lines over F2 and 27 lines over F4:
```

D1_q4_with_select_double_six:

```bash
$\text{ORBITER}$ -v 3 \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-define_surface S_{D1_q4} -q 4 -by_coefficients $(D1) \n-select_double_six "3,9,15,19,22,26,4,10,14,18,21,25" \n-end \n-end \n-with S_{D1_q4} -do \n-cubic_surface_activity \n-report \n-end
```

```bash
mv surface_by_coefficients_q4_report.tex D1_q4.tex
pdflatex D1_q4.tex
open D1_q4.pdf
```

668
D1\_q4\_with\_select\_double\_six\_b:\n
\$(\text{ORBITER}) -v 3 \$
\n-define F -finite\_field -q 4 -end \\
-define P -projective\_space -n 3 -field F -v 0 -end \\
-with P -do \\
-projective\_space\_activity \\
\define\_surface S\_D1\_q4 -q 4 -by\_coefficients $(D1) \\
\select\_double\_six "3,9,15,19,22,26,4,10,14,18,21,25" \\
\select\_double\_six "1,2,3,4,5,0,7,8,9,10,11,6" \\
-end \\
-with S\_D1\_q4 -do \\
-cubic\_surface\_activity \\
-report \\
-end \\

mv surface\_by\_coefficients\_q4\_report.tex D1\_q4.tex \\
pdflatex D1\_q4.tex \\
open D1\_q4.pdf

\# ToDo: now projective\_space\_activity:

D1\_q4\_trans:

\$(\text{ORBITER}) -v 5 -define F -finite\_field -q 4 -end \\
-with F -do -finite\_field\_activity \\
-move\_two\_lines\_in\_hyperplane\_stabilizer\_text \\
"1,0,0,0, 0,1,0,0, 0,0,1,0, 0,0,1,1, 0" \\
"1,0,0,0, 0,0,0,1, 1,0,1,0, 1,0,1,0, 0" \\
-end \\

D1\_q4\_with\_select\_double\_six\_c:\n
\$(\text{ORBITER}) -v 3 \$
\n-define F -finite\_field -q 4 -end \\
-define P -projective\_space -n 3 -field F -v 0 -end \\
-with P -do \\
-projective\_space\_activity \\
\define\_surface S\_D1\_q4 -q 4 -by\_coefficients $(D1) \\
\select\_double\_six "3,9,15,19,22,26,4,10,14,18,21,25" \\
\select\_double\_six "1,2,3,4,5,0,7,8,9,10,11,6" \\
-transform "1,0,0,0, 0,1,0,0, 0,0,1,0, 0,0,1,1, 0" \\
-end \\
-with S\_D1\_q4 -do \\
-cubic\_surface\_activity \\
-report \\
-end \\

mv surface\_by\_coefficients\_q4\_report.tex D1\_q4.tex
orbits_cubic_surfaces_q3:

```
$(ORBITER) -v 4 \ndefine G -linear_group -PGL 4 3 -end \nwith G -do \ngroup.theoretic_activity \norbits_on_polynomials 3 \nend
```

pdflatex poly_orbits_d3_n3_q3.tex
open poly_orbits_d3_n3_q3.pdf

# this takes 3 days and about 150 GB memory on ripoff

orbits_cubic_curves_q2_again:

```
$(ORBITER) -v 4 \ndefine G \nlinear_group -PGL 3 2 \nend \nwith G -do \ngroup.theoretic_activity \norbits_on_polynomials 3 \nend
```

pdflatex poly_orbits_d3_n2_q2.tex
open poly_orbits_d3_n2_q2.pdf

orbits_cubic_surfaces_q3:

```
$(ORBITER) -v 4 \ndefine G \nlinear_group -PGL 3 3 \nend \nwith G -do \ngroup.theoretic_activity \norbits_on_polynomials 3 \nend
```

pdflatex poly_orbits_d3_n2_q3.tex
open poly_orbits_d3_n2_q3.pdf
# compute and analyze properties over F2

```bash
poly_orbits_d3_n3_q2_F2.csv: poly_orbits_d3_n3_q2.csv
```

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

```
Dickson_q2.analyze: poly_orbits_d3_n3_q2_F2.csv
```

```
$ (ORBITER) -v 4 \
define F -finite_field -q 2 -end \ndefine P -projective_space -n 3 -field F -v 0 -end \nwith P -do \
projective_space_activity \ncubic_surface_properties.analyze \n› poly_orbits_d3_n3_q2_F2.csv 2 \nend
```

```
pdflatex poly_orbits_d3_n3_q2_F2_report.tex
open poly_orbits_d3_n3_q2_F2_report.pdf
```

# compute and analyze properties over F4

```bash
poly_orbits_d3_n3_q2_F4.csv: poly_orbits_d3_n3_q2.csv
```

```bash
$ (ORBITER) -v 4 \
define F -finite_field -q 4 -end \ndefine P -projective_space -n 3 -field F -v 0 -end \nwith P -do \
projective_space_activity \ntable_of_cubic_surfaces_compute_properties \n› poly_orbits_d3_n3_q2_F4.csv 2 \nend
```

```
Dickson_q4.analyze: poly_orbits_d3_n3_q2_F4.csv
```

```
$ (ORBITER) -v 4 \
define F -finite_field -q 4 -end \ndefine P -projective_space -n 3 -field F -v 0 -end \nwith P -do \
projective_space_activity \ncubic_surface_properties.analyze \n› poly_orbits_d3_n3_q2_F4.csv 2 \nend
```

```
pdflatex poly_orbits_d3_n3_q2_F4_report.tex
```
8201 poly_orbits_d3_n3_q2_F8.csv: poly_orbits_d3_n3_q2.csv
8202 $(ORBITER) -v 4 \ $
8203 -define F -finite_field -q 8 -end \ $
8204 -define P -projective_space -n 3 -field F -v 0 -end \ $
8205 -with P -do \ $
8206 -projective_space_activity \ $
8207 -table_of_cubic_surfaces_compute_properties \ $
8208 -poly_orbits_d3_n3_q2.csv 2 0 \ $
8209 -end
8210
8211 Dickson_q8.analyze: poly_orbits_d3_n3_q2_F8.csv
8212 $(ORBITER) -v 4 \ $
8213 -define F -finite_field -q 8 -end \ $
8214 -define P -projective_space -n 3 -field F -v 0 -end \ $
8215 -with P -do \ $
8216 -projective_space_activity \ $
8217 -cubic_surface_properties.analyze \ $
8218 -poly_orbits_d3_n3_q2_F8.csv 2 \ $
8219 -end
8220 pdflatex poly_orbits_d3_n3_q2_F8_report.tex
8221 open poly_orbits_d3_n3_q2_F8_report.pdf
8222
8224 # compute and analyze properties over F16
8225
8226 poly_orbits_d3_n3_q2_F16.csv: poly_orbits_d3_n3_q2.csv
8227 $(ORBITE) -v 4 \ $
8228 -define F -finite_field -q 16 -end \ $
8229 -define P -projective_space -n 3 -field F -v 0 -end \ $
8230 -with P -do \ $
8231 -projective_space_activity \ $
8232 -table_of_cubic_surfaces_compute_properties \ $
8233 -poly_orbits_d3_n3_q2.csv 2 \ $
8234 -end
8235
8236 Dickson_q16.analyze: poly_orbits_d3_n3_q2_F16.csv
8237 $(ORBITE) -v 4 \ $
8238 -define F -finite_field -q 16 -end \ $
8239 -define P -projective_space -n 3 -field F -v 0 -end \ $
8240 -with P -do \ $
8241 -projective_space_activity \ $
8242 -cubic_surface_properties.analyze \
# Section 7.6: Cubic Surfaces - ATLAS and Tables

SECTION CUBIC SURFACES ATLAS AND TABLES:

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

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

cubic_surfaces_table_latex_17: 
$(ORBITER) -v 3 -csv_file_latex 1 \n$(ORBITER) -v 3 -csv_file_latex 1 \n$(ORBITER) -v 3 -csv_file_latex 1 \n$(ORBITER) -v 3 -csv_file_latex 1 \n$(ORBITER) -v 3 -csv_file_latex 1 \n$(ORBITER) -v 3 -csv_file_latex 1

cubic_surfaces_tables_up_to_17: 
$(ORBITER) -v 3 -define F -finite_field -q 4 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$(ORBITER) -v 3 -define F -finite_field -q 4 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$(ORBITER) -v 3 -define F -finite_field -q 4 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$(ORBITER) -v 3 -define F -finite_field -q 4 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$(ORBITER) -v 3 -define F -finite_field -q 4 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)}
cubic_surfaces_tables_19_37:

$(ORBITER) -v 3 -define F -finite_field -q 17 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$(ORBITER) -v 3 -define F -finite_field -q 19 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$(ORBITER) -v 3 -define F -finite_field -q 23 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$(ORBITER) -v 3 -define F -finite_field -q 25 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$(ORBITER) -v 3 -define F -finite_field -q 27 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$(ORBITER) -v 3 -define F -finite_field -q 29 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$(ORBITER) -v 3 -define F -finite_field -q 31 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$(ORBITER) -v 3 -define F -finite_field -q 32 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$(ORBITER) -v 3 -define F -finite_field -q 37 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$(ORBITER) -v 3 -define F -finite_field -q 41 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$(ORBITER) -v 3 -define F -finite_field -q 43 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$(ORBITER) -v 3 -define F -finite_field -q 47 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$(ORBITER) -v 3 -define F -finite_field -q 49 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$(ORBITER) -v 3 -define F -finite_field -q 53 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$(ORBITER) -v 3 -define F -finite_field -q 59 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$(ORBITER) -v 3 -define F -finite_field -q 61 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$(ORBITER) -v 3 -define F -finite_field -q 64 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$(ORBITER) -v 3 -define F -finite_field -q 67 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$(ORBITER) -v 3 -define F -finite_field -q 71 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$(ORBITER) -v 3 -define F -finite_field -q 73 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$(ORBITER) -v 3 -define F -finite_field -q 79 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
$(ORBITE) -v 3 -define F -finite_field -q 81 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$(ORBITE) -v 3 -define F -finite_field -q 83 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$(ORBITE) -v 3 -define F -finite_field -q 89 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$(ORBITE) -v 3 -define F -finite_field -q 97 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$(ORBITE) -v 3 -define F -finite_field -q 101 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$(ORBITE) -v 3 -define F -finite_field -q 103 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$(ORBITE) -v 3 -define F -finite_field -q 107 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$(ORBITE) -v 3 -define F -finite_field -q 109 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$(ORBITE) -v 3 -define F -finite_field -q 113 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$(ORBITE) -v 3 -define F -finite_field -q 121 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$(ORBITE) -v 3 -define F -finite_field -q 127 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$(ORBITE) -v 3 -define F -finite_field -q 128 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

cubic_surfaces_tables_latex:

$(ORBITE) -v 3 -csv_file_latex 1 test.csv
$(ORBITE) -v 3 -csv_file_latex 0 table_of_cubic_surfaces_q4_info.csv
$(ORBITE) -v 3 -csv_file_latex 0 table_of_cubic_surfaces_q7_info.csv
$(ORBITE) -v 3 -csv_file_latex 0 table_of_cubic_surfaces_q8_info.csv
$(ORBITE) -v 3 -csv_file_latex 0 table_of_cubic_surfaces_q9_info.csv
$(ORBITE) -v 3 -csv_file_latex 0 table_of_cubic_surfaces_q11_info.csv
$(ORBITE) -v 3 -csv_file_latex 0 table_of_cubic_surfaces_q13_info.csv
$(ORBITE) -v 3 -csv_file_latex 0 table_of_cubic_surfaces_q16_info.csv
$(ORBITE) -v 3 -csv_file_latex 0 table_of_cubic_surfaces_q17_info.csv
$(ORBITE) -v 3 -csv_file_latex 0 table_of_cubic_surfaces_q19_info.csv

$(ORBITE) -v 3 -csv_file_latex 0 table_of_cubic_surfaces_q23_info.csv
$(ORBITE) -v 3 -csv_file_latex 0 table_of_cubic_surfaces_q25_info.csv
$(ORBITE) -v 3 -csv_file_latex 0 table_of_cubic_surfaces_q27_info.csv
$(ORBITE) -v 3 -csv_file_latex 0 table_of_cubic_surfaces_q29_info.csv
$(ORBITE) -v 3 -csv_file_latex 0 table_of_cubic_surfaces_q31_info.csv
$(ORBITE) -v 3 -csv_file_latex 0 table_of_cubic_surfaces_q32_info.csv
8345  ▷ $(ORBITER) -v 3 -csv_file_latex 0 table_of_cubic_surfaces_q37_info.csv
8346  ▷ $(ORBITER) -v 3 -csv_file_latex 0 table_of_cubic_surfaces_q41_info.csv
8347  ▷ $(ORBITER) -v 3 -csv_file_latex 0 table_of_cubic_surfaces_q43_info.csv
8348  ▷ \#$(ORBITER) -v 3 -csv_file_latex 1 quartic_curves_q9_info.csv
8349  ▷ pdflatex quartic_curves_q13_info.tex
8350  ▷ \#open quartic_curves_q13_info.pdf
8351  ▷ ~/bin/tth quartic_curves_q13_info.tex
8352  ▷ \#open quartic_curves_q13_info.html
8353
8354
8355
8356
8357  surface_table:
8358  ▷ $(ORBITER) -v 3 -make_table_of_surfaces
8359  ▷ pdflatex surfaces_report.tex
8360  ▷ open surfaces_report.pdf
8361
8362
8363
8364
8365  surface_atlas:
8366  ▷ $(ORBITER) -v 3 -create_surface_atlas 97
8367  ▷ ~/bin/tth surface_atlas.tex
8368
8369
8370  surface_reports:
8371  ▷ $(ORBITER) -v 3 \$
8372  ▷ ▷ -orbiter_path $(ORBITER_PATH) -create_surface_reports 4,7,8,9,11
8373
8374
8375
8376
8377
8378
8379
8380
8381  quartic_curve_tables_23:
8382  ▷ $(ORBITER) -v 3 \$
8383  ▷ ▷ -define F -finite_field -q 23 -end \$
8384  ▷ ▷ -define P -projective_space -n 2 -field F -v 0 -end \$
8385  ▷ ▷ -with P -do \$
8386  ▷ ▷ ▷ -projective_space_activity \$
8387  ▷ ▷ ▷ ▷ -table_of_quartic_curves \$
8388  ▷ ▷ ▷ ▷ -end
8389
8390  quartic_curve_tables:
8391  ▷ $(ORBITER) -v 3 \$

676
-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 -csv_file_latex 1 test.csv \\
$(ORBITER) -v 3 -csv_file_latex 0 quartic_curves_q9_info.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_q19_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
# Section 8.1: Polynomials over Finite Fields

SECTION_POLYNOMIALS:

# check which polynomials are irreducible and which are primitive:

sift_polynomials_deg3_q2:

sift_polynomials_deg4_q2:

poly_division:

poly_division2:
poly_gcd:
  $(ORBITER) -v 2 
  -define F -finite_field -q 2 -end 
  -with F -do 
  -finite_field_activity 
  -extended_gcd_for_polynomials "1,0,0,0,0,0,0,0,0,0,1" "1,0,1,1" -end

poly_mult_mod1:
  $(ORBITER) -v 2 
  -define F -finite_field -q 7 -end 
  -with F -do 
  -finite_field_activity 
  -polynomial_mult_mod "1,2,3" "3,4,5" "6,0,0,1" -end

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

poly_mult_mod4:
  $(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

mult_polynomials_2.5.7:
  $(ORBITER) -v 2 

mult.polynomials_2.8_15:

mult.polynomials_2.7_7:
mult.polynomials_2.4.6:
$$(\text{ORBITER}) \ -v \ 2 \ \ \ \\
\text{define F -finite_field -q 2 -end} \ \\
\text{with F -do} \ \\
\text{finite_field_activity} \ \\
\text{mult.polynomials 4 6 -end} \ \\
pdflatex \ polynomial\_mult\_4.6.tex \ \\
open \ polynomial\_mult\_4.6.pdf

polynomial\_division\_ranked.2.24.13:
$$(\text{ORBITER}) \ -v \ 2 \ \ \ \\
\text{define F -finite_field -q 2 -end} \ \\
\text{with F -do} \ \\
\text{finite_field_activity} \ \\
\text{polynomial\_division\_ranked 24 13} \ \\
\text{end} \ \\
pdflatex \ polynomial\_division.24.13.tex \ \\
open \ polynomial\_division.24.13.pdf

mult.polynomials.1024.999.997:
$$(\text{ORBITER}) \ -v \ 2 \ \ \ \\
\text{define F -finite_field -q 2 -end} \ \\
\text{with F -do} \ \\
\text{finite_field_activity} \ \\
\text{mult.polynomials 999 997} \ \\
\text{end} \ \\
pdflatex \ polynomial\_mult.999.997.tex \ \\
open \ polynomial\_mult.999.997.pdf

polynomial\_division\_ranked.2.349147.1033:
$$(\text{ORBITER}) \ -v \ 2 \ \ \ \\
\text{define F -finite_field -q 2 -end} \ \\
\text{with F -do} \ \\
\text{finite_field_activity} \ \\
\text{polynomial\_division\_ranked 349147 1033} \ \\
\text{end} \ \\
pdflatex \ polynomial\_division.349147.1033.tex \ \\
open \ polynomial\_division.349147.1033.pdf
mult_polynomials_1024_999_997_check:
\begin{verbatim}
$ (ORBITER) -v 3 \\
-define F -finite_field -q 1024 -end \\
-with F -do \\
-finite_field_activity -parse_and_evaluate \\
"test" "a*b" "a=999,b=997" -end
\end{verbatim}
# evaluates to 61

mult_polynomials_17_12:
\begin{verbatim}
$ (ORBITER) -v 2 \\
-define F -finite_field -q 2 -end \\
-with F -do \\
-finite_field_activity \\
mult_polynomials 17 12 -end
\end{verbatim}
pdflatex polynomial_mult_17_12.tex
open polynomial_mult_17_12.pdf
# gives 204

polynomial_division_ranked_204_37:
\begin{verbatim}
$ (ORBITER) -v 2 \\
-define F -finite_field -q 2 -end \\
-with F -do \\
-finite_field_activity \\
polynomial_division_ranked 204 37 \\
-end
\end{verbatim}
pdflatex polynomial_division_204_37.tex
open polynomial_division_204_37.pdf
# answer is 18

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"
crc32 "123456789"

682
Berlekamp_matrix_crc32:

```bash
$ (ORBITER) -v 2 \n-define F -finite_field -q 2 -end \n-define v -vector -field F -sparse 33 $(CRC32_SPARSE) -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-polygonal_mult_mod A B M \n-polygonal_power_mod A $(TWO_TO_THE_32_MINUS_2) M \n-polygonal_power_mod B $(TWO_TO_THE_32_MINUS_2) M \n-polygonal_power_mod B $(TWO_TO_THE_32_MINUS_2) M \n-polygonal_power_mod B $(TWO_TO_THE_32_MINUS_2) M
```

# 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 \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-define B -vector -field F -sparse 33 $(INVERSE_SPARSE) -end \n-polygonal_power_mod A $(TWO_TO_THE_32_MINUS_2) M \n-polygonal_power_mod B $(TWO_TO_THE_32_MINUS_2) M
```

# 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 \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-define B -vector -field F -sparse 33 $(INVERSE_SPARSE) -end \n-define C -vector -field F -sparse 33 $(INVERSE_SPARSE) -end \n-polygonal_mult_mod A B M \n-polygonal_mult_mod A B M \n-polygonal_mult_mod A B M \n-polygonal_mult_mod A B M
```

#C(X)=1
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 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 4a:

```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 4b:

```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
```

# the polynomial $X^4+X+1$ is irreducible over GF(2) because the rank of the Berlekamp matrix is 3.
find roots a:
\[$(ORBITER) -v 2 \$
\[define F -finite_field -q 19 -end \]
\[define v -vector -field F -dense "18,1,1" -end \]
\[with F -do \]
\[finite_field_activity \]
\[polynomial_find_roots v -end \]

find roots b:
\[$(ORBITER) -v 2 \$
\[define F -finite_field -q 19 -end \]
\[define v -vector -field F -dense "1,3,1" -end \]
\[with F -do \]
\[finite_field_activity \]
\[polynomial_find_roots v -end \]

find roots c:
\[$(ORBITER) -v 2 \$
\[define F -finite_field -q 19 -end \]
\[define v -vector -field F -dense "1,16,1" -end \]
\[with F -do \]
\[finite_field_activity \]
\[polynomial_find_roots v -end \]

find roots d:
\[$(ORBITER) -v 2 \$
\[define F -finite_field -q 19 -end \]
\[define v -vector -field F -dense "1,18,1" -end \]
\[with F -do \]
\[finite_field_activity \]
\[polynomial_find_roots v -end \]

find roots e:
\[$(ORBITER) -v 2 \$
\[define F -finite_field -q 19 -end \]
\[define v -vector -field F -dense "1,16,3" -end \]
\[with F -do \]
\[finite_field_activity \]
\[polynomial_find_roots v -end \]

roots_over_F2:
\[$(ORBITER) -v 2 \$
\[define F -finite_field -q 2 -end \]
\[define v -vector -field F -dense "0,1,0,1,1,1" -end \]
roots_over_F8:

```
>>> $(ORBITER) -v 2 \
>>> -finite_field_activity \
>>> -polynomial_find_roots v -end
```

```
>>> define F -finite_field -q 8 -override_polynomial 11 -end \
>>> define v -vector -field F -dense "0,1,0,1,1,1" -end \
>>> -with F -do \
>>> -finite_field_activity \
>>> -polynomial_find_roots v -end
```

```
# degree and then order of the field of coefficients:

irred_3_2:
```
>>> $(ORBITER) -v 3 \
>>> -define F -finite_field -q 2 -end \
>>> -with F -do \
>>> -finite_field_activity \
>>> -make_table_of_irreducible_polynomials 3 -end
```

```
pdflatex Irred_q2_d3.tex
open Irred_q2_d3.pdf
```

```
irred_4_2:
```
>>> $(ORBITER) -v 3 \
>>> -define F -finite_field -q 2 -end \
>>> -with F -do \
>>> -finite_field_activity \
>>> -make_table_of_irreducible_polynomials 4 -end
```

```
pdflatex Irred_q2_d4.tex
open Irred_q2_d4.pdf
```

```
# 3 polys

irred_5_2:
```
>>> $(ORBITER) -v 3 \
>>> -define F -finite_field -q 2 -end \
>>> -with F -do \
>>> -finite_field_activity \
>>> -make_table_of_irreducible_polynomials 5 -end
```

```
pdflatex Irred_q2_d5.tex
```

686
8860 ▷ open Irred_q2_d5.pdf
8861
8862 # 6 polys
8863
8864  irred_6_2:
8865 ▷ $(ORBITER) -v 3 \
8866 ▷ ▷ -define F -finite_field -q 2 -end \
8867 ▷ ▷ -with F -do \
8868 ▷ ▷ -finite_field_activity \
8869 ▷ ▷ -make_table_of_irreducible_polynomials 6 -end
8870 ▷ pdflatex Irred_q2_d6.tex
8871 ▷ open Irred_q2_d6.pdf
8872
8873 # 9 polys
8874
8875  irred_7_2:
8876 ▷ $(ORBITER) -v 3 \
8877 ▷ ▷ -define F -finite_field -q 2 -end \
8878 ▷ ▷ -with F -do \
8879 ▷ ▷ -finite_field_activity \
8880 ▷ ▷ -make_table_of_irreducible_polynomials 7 -end
8881 ▷ pdflatex Irred_q2_d7.tex
8882 ▷ open Irred_q2_d7.pdf
8883
8884 # 18 polys
8885
8886  irred_8_2:
8887 ▷ $(ORBITER) -v 3 \
8888 ▷ ▷ -define F -finite_field -q 2 -end \
8889 ▷ ▷ -with F -do \
8890 ▷ ▷ -finite_field_activity \
8891 ▷ ▷ -make_table_of_irreducible_polynomials 8 -end
8892 ▷ pdflatex Irred_q2_d8.tex
8893 ▷ open Irred_q2_d8.pdf
8894
8895 # 30 polys
8896
8897  irred_9_2:
8898 ▷ $(ORBITER) -v 3 \
8899 ▷ ▷ -define F -finite_field -q 2 -end \
8900 ▷ ▷ -with F -do \
8901 ▷ ▷ -finite_field_activity \
8902 ▷ ▷ -make_table_of_irreducible_polynomials 9 -end
8903 ▷ pdflatex Irred_q2_d9.tex
8904 ▷ open Irred_q2_d9.pdf
8905
8906 # 56 polys
irred_10.2:
- $(ORBITER) -v 3 \n- define F -finite_field -q 2 -end \n- with F -do \n- finite_field_activity \n- make_table_of_irreducible_polynomials 10 -end
- pdflatex Irred_q2_d10.tex
- open Irred_q2_d10.pdf
- # 99 polys

irred_2.4:
- $(ORBITER) -v 3 \n- define F -finite_field -q 4 -end \n- with F -do \n- finite_field_activity \n- make_table_of_irreducible_polynomials 2 -end
- pdflatex Irred_q4_d2.tex
- open Irred_q4_d2.pdf
- # 6 polys

irred_3.4:
- $(ORBITER) -v 6 \n- define F -finite_field -q 4 -end \n- with F -do \n- finite_field_activity \n- make_table_of_irreducible_polynomials 3 -end
- pdflatex Irred_q4_d3.tex
- open Irred_q4_d3.pdf
- # 20 polys

search_primitive_poly.2:
- $(ORBITER) -v 3 \n- search_for_primitive_polynomial_in_range 2 2 2 10 #| grep //
- # stuck in factoring 2^61-1 (which is prime)

search_primitive_poly.3:
8954 ▷ $(ORBITER) -v 6 \\
8955 ▷ ▷ -search_for_primitive_polynomial_in_range 3 3 2 60 
8956 ▷ 8957 
8958 search_primitive_poly_4: 8959 ▷ $(ORBITER) -v 6 \\
8960 ▷ ▷ -search_for_primitive_polynomial_in_range 4 4 2 30 
8961 ▷ 8962 search_primitive_poly_5: 8963 ▷ $(ORBITER) -v 6 \\
8964 ▷ ▷ -search_for_primitive_polynomial_in_range 5 5 2 30 
8965 ▷ 8966 search_primitive_poly_7: 8967 ▷ $(ORBITER) -v 6 \\
8968 ▷ ▷ -search_for_primitive_polynomial_in_range 7 7 2 20 
8969 ▷ 8970 search_primitive_poly_8: 8971 ▷ $(ORBITER) -v 6 \\
8972 ▷ ▷ -search_for_primitive_polynomial_in_range 8 8 2 20 
8973 ▷ 8974 search_primitive_poly_9: 8975 ▷ $(ORBITER) -v 6 \\
8976 ▷ ▷ -search_for_primitive_polynomial_in_range 9 9 2 15 
8977 ▷ 8978 search_primitive_poly_11: 8979 ▷ $(ORBITER) -v 6 \\
8980 ▷ ▷ -search_for_primitive_polynomial_in_range 11 11 2 15 
8981 ▷ 8982 search_primitive_poly_13: 8983 ▷ $(ORBITER) -v 6 \\
8984 ▷ ▷ -search_for_primitive_polynomial_in_range 13 13 2 15 
8985 ▷ 8986 search_primitive_poly_degree_16: 8987 ▷ $(ORBITER) -v 6 \\
8988 ▷ ▷ -search_for_primitive_polynomial_in_range 2 2 16 16 
8989 ▷ 8990 search_primitive_poly_32: 8991 ▷ $(ORBITER) -v 6 \\
8992 ▷ ▷ -search_for_primitive_polynomial_in_range 32 32 2 10 
8993 ▷ 8994 8995 8996 8997 8998 8999 9000
# Section 8.2: Multivariate Polynomials

SECTION_MULTIVARIATE_POLYNOMIALS:

CREMONA_MAP_Y0="3*y0*y0*y0*y0*y2+4*y0*y0*y1*y1*y2\n  +6*y0*y1*y2*y2+9*y0*y2*y2*y2*y2"

CREMONA_MAP_Y1="y0*y0*y0*y0*y0*y0+5*y0*y0*y1*y1*y1\n  +12*y0*y0*y1*y2*y2+3*y0*y1*y1*y1*y1*y1\n  +5*y0*y1*y2*y2*y2+y0*y1*y2*y2*y2*y2"

CREMONA_MAP_Y2="10*y0*y0*y0*y0*y0*y0+11*y0*y0*y0*y0*y0*y1*y1\n  +11*y0*y0*y0*y0*y0*y2*y2+4*y0*y0*y1*y1*y1*y1\n  +9*y0*y0*y0*y1*y2*y2+4*y0*y0*y2*y2*y2*y2"

CREMONA_MAP_Y3="0"

Cremona_map:

$\text{(ORBITER)} -v 3 \n$ -define F -finite_field -q 13 -end \n$ -define P -projective_space -n 2 -field F -v 0 -end \n$ -define R -polynomial_ring \n$ -field F \n$ -number_of_variables 3 \n$ -homogeneous_of_degree 6 \n$ -monomial_ordering_lex \n$ -variables "y0,y1,y2" "y_0,y_1,y_2" \n$ -end \n$ -define Y0 -formula \n$ "y0" "y_0" "y0,y1,y2" \n$ -define Y1 -formula \n$ "y1" "y_1" "y0,y1,y2" \n$ -define Y2 -formula \n$ "y2" "y_2" "y0,y1,y2" \n$ -define Cremona -collection "Y0,Y1,Y2" \n$ -with P -do \n$ -projective_space_activity \n
690
\begin{verbatim}
9048 \$\texttt{map R Cremona "\"}\ \$
9049 \$\texttt{-end}\ $
9050 $
9051 $
9052 $
9053 arcs_5_2_q11:
9054 $\texttt{$\texttt{\$(ORBITER) -v 4 }$}\$
9055 \$\texttt{-define F -finite_field -q 11 -end }$
9056 \$\texttt{-define P -projective_space -n 2 -field F -v 0 -end }$
9057 \$\texttt{-with P -do }$
9058 \$\texttt{-projective_space_activity }$
9059 \$\texttt{-classify arcs }$
9060 \$\texttt{-poset.classification_control }$
9061 \$\texttt{-problem_label arcs_5_2_q11 }$
9062 \$\texttt{-W -depth 5 }$
9063 \$\texttt{-report -end }$
9064 \$\texttt{-end }$
9065 \$\texttt{-target_size 5 }$
9066 \$\texttt{-d 2 }$
9067 \$\texttt{-end }$
9068 \$\texttt{-end }$
9069 \$\texttt{-end }$
9070 \$\texttt{pdflatex arcs_5_2_q11_poset.tex}$
9071 \$\texttt{open arcs_5_2_q11_poset.pdf}$
9072 $
9073 $
9074 # 2 orbits:
9075 # 0 1 2 3 37
9076 # 0 1 2 3 49
9077 $
9078 $
9079 arcs_5_2_q11_ideal:
9080 \$\texttt{$\texttt{\$(ORBITER) -v 2 }$}\$
9081 \$\texttt{-define F -finite_field -q 11 -end }$
9082 \$\texttt{-define R -polynomial_ring }$
9083 \$\texttt{-field F }$
9084 \$\texttt{-number_of_variables 3 }$
9085 \$\texttt{-homogeneous_of_degree 2 }$
9086 \$\texttt{-monomial_ordering_lex }$
9087 \$\texttt{-variables "x0,x1,x2" "x_0,x_1,x_2" }$
9088 \$\texttt{-end }$
9089 \$\texttt{-define C -combinatorial_objects }$
9090 \$\texttt{-file_of_points arcs_5_2_q11_lvl5 }$
9091 \$\texttt{-end }$
9092 \$\texttt{-with C -do }$
9093 \$\texttt{-combinatorial_object_activity }$
9094 \$\texttt{-ideal R }$
\end{verbatim}

691
We found 12 points on the generator of the ideal
They are: (0, 1, 2, 3, 37, 54, 74, 80, 93, 105, 121, 128)

We found 12 points on the generator of the ideal
They are: (0, 1, 2, 3, 49)

looping over all generators of the ideal:

We found 12 points on the generator of the ideal
They are: (0, 1, 2, 3, 41, 49, 58, 77, 83, 95, 109, 130)

surface_9lines_4E_ideal:
$\langle \text{ORBITER} \rangle -v 2$
$\langle \text{PTS OF SURFACE ORBIT211 Q3 L9 E4} \rangle$
$\langle \text{PTS OF SURFACE ORBIT211 Q3 L9 E4} \rangle$

The ideal has dimension 2
generators for the ideal:

# The ideal has dimension 2
generators for the ideal:
SURFACE_F_9="x0*x0*x1 - x0*x1*x1 -x0*x1*x3 -x2*x2*x3 - x2*x3*x3"

F_9.q7:

\begin{verbatim}
\verb|\$(ORBITER) -v 3 \|
\verb|\$define F -finite_field -q 7 -end \|
\verb|\$define P -projective_space -n 3 -field F -v 0 -end \|
\verb|\$with P -do \|
\verb|\$projective_space_activity \|
\verb|\$define_surface F_9 -q 7 \|
\verb|\$by_equation "F_9" \|
\verb|"\DF_9\D" "x0,x1,x2,x3" \|
\verb|\$\$(SURFACE_F_9) \|
\verb|"\Dno parameters\D" \|
\verb|\$end \|
\verb|\$end \|
\verb|\$with F_9 -do \|
\verb|\$cubic_surface_activity \|
\verb|\$report \|
\verb|\$end \|
\verb|pdflatex surface_equation_F_9_q7_report.tex
\verb|open surface_equation_F_9_q7_report.pdf
\end{verbatim}

# 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:

\begin{verbatim}
\verb|\$(ORBITER) -v 4 \|
\verb|\$create_random_k_subsets 133 5 20
\end{verbatim}

# We compute the line intersections:

line_type_in_PG_2_11:

\begin{verbatim}
\verb|\$(ORBITER) -v 3 \|
\verb|\$orbiter_path $(ORBITER_PATH) \|
\verb|\$define F -finite_field -q 11 -end \|
\verb|\$define P -projective_space -n 2 -field F -v 0 -end \|
\verb|\$define C -combinatorial_objects \|
\verb|\$file_of_points random_k_subsets_n133_k5_nb20.csv \|
\end{verbatim}

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

creating the ideal:

creating the ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:

random_arc_5_2_q11_ideal:
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 -square_root_mod 2221 7817

# Section 9.2: Representation Theory
SECTION_REPRESENTATION THEORY:
representation_on_polynomials_of_degree_3:
$(ORBITER) -v 4 
(define G -linear_group -PGL 4 3 -end 
(with G -do 

representation_tetrahedral_group_on_polynomials_of_degree_3:
$(ORBITER) -v 4 \\
$ORBITER -v 2 -loop L 0 2 1 -draw_matrix \\
-input_csv_file GL_3_3_Group_tetra12_rep_3_%L.csv \\
-box_width 40 -bit_depth 24 -partition 3 10 10 -end -end_loop \\
open GL_3_3_Group_tetra12_rep_3_0_draw.bmp \\
open GL_3_3_Group_tetra12_rep_3_1_draw.bmp \\
write GL_3_3_Group_tetra12_rep_3_0.csv \\
# Section 9.3: Cryptography \\
SECTION_CRYPTOGRAPHY: \\
EC_add: \\
$(ORBITER) -v 2 \\
-definition F -finite_field -q 11 -end \\
-with F -do \\
-finite_field_activity \\
-EC_add 1 3 "1,4" "1,4" -end \\
EC_cyclic_subgroup:
-define F -finite_field -q 11 -end \n-with F -do \n-finite_field_activity \n-EC_cyclic_subgroup 1 3 "1,4" -end

EC_points_13:
$\mathrm{ORBITER}$ -v 2 \n-def F -finite_field -q 13 -end \n-with F -do \n-finite_field_activity \n-EC_points "EC_2_5.q13" 2 5 -end

$\mathrm{ORBITER}$ -v 2 -draw_matrix \n-input_csv_file EC_2_5.q13.points_xy.csv 
-box_width 20 -bit_depth 24 \n-partition 2 "1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1" "1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1" -end

EC_points_199:
$\mathrm{ORBITER}$ -v 2 \n-def F -finite_field -q 199 -end \n-with F -do \n-finite_field_activity \n-EC_points "EC_5_7.q199" 5 7 -end

$\mathrm{ORBITER}$ -v 2 \n-draw_matrix -input_csv_file EC_5_7.q199.points_xy.csv 
-box_width 10 -bit_depth 24 \n-partition 2 199 199 -end

EC_Koblitz_encoding:
$\mathrm{ORBITER}$ -v 6 -seed 17 \n-def F -finite_field -q 199 -end \n-with F -do \n-finite_field_activity \n-EC_Koblitz_encoding 5 7 67 "147,164" "DEADBEEF" \n-end

EC_bsgs:
$\mathrm{ORBITER}$ -v 2 \n-def F -finite_field -q 199 -end \n-with F -do \n-finite_field_activity \n-EC_bgs 5 7 "147,164" 212 \n-"172,158,45,195,50,22,10,103,55,33,50,22,145,105,31,74,73,155,67,60,25,6" \n-end
EC_bsgs_decode:
	$\text{ORBITER} -v 2 \$
	define F -finite_field -q 199 -end 
	with F -do 
	finite_field_activity 
	-EC_bsgs_decode 5 7 "129,176" 212 
	"127,188,51,141,85,29,106,90,41,105,179,71,171,2,16,197,183,72,27,129,37,10" 
	"50,179,169,13,153,169,115,116,188,110,176" 
	xend

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"
ALICE_PRIVATE_FQ="37,2,40,21,31,26,8"

$\text{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) 
-xend

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"
Alice:

```
#F_p(x) = X^6 + 2X^5 + X^3 + X^2 + X + 1
ALICE_PRIVATE_FP="1,1,1,1,0,2,1"
```

Bob's message

```
BOB_MESSAGE="1,-1,1,0,-1"
```

Bob's one-time key

```
BOB_ONE_TIME_KEY="-1,1,0,0,0,-1,1"
```

NTRU encrypt:

```
#E(X) = 31X^6 + 19X^5 + 4X^4 + 2X^3 + 40X^2 + 3X + 25
BOB_ENCRYPT= "25,3,40,2,4,19,31"
```

NTRU decrypt1:

```
```

\[ C(X) = X^6 + 10X^5 + 33X^4 + 40X^3 + 40X^2 + X + 40 \]

ALICE \( C_1 = "40,1,40,40,33,10,1" \)

NTRU decrypt2:
\[
\text{ORBITER} -v 2 -define F -finite_field -q $(NTRU \_Q) -end -with F -do -finite_field_activity -polynomial_center_lift $(ALICE \_C1) -end
\]

\[ A(X) = X^6 + 10X^5 - 8X^4 - X^3 - X^2 + X - 1 \]

ALICE \( C_2 = "-1,1,-1,-1,-8,10,1" \)

NTRU decrypt3:
\[
\text{ORBITER} -v 2 -define F -finite_field -q $(NTRU \_P) -end -with F -do -finite_field_activity -polynomial_reduce_mod_p $(ALICE \_C2) -end
\]

\[ A(X) = X^6 + X^5 + X^4 + 2X^3 + 2X^2 + X + 2 \]

ALICE \( C_3 = "2,1,2,2,1,1,1" \)

NTRU decrypt4:
\[
\text{ORBITER} -v 2 -define F -finite_field -q $(NTRU \_Q) -end -with F -do -finite_field_activity -polynomial_mult_mod $(ALICE \_PRIVATE \_FP) -polynomial_center_lift $(ALICE \_C3) $(NTRUE \_XN1) -end
\]

\[ C(X) = 2X^5 + X^3 + X^2 + 2X + 1 \]

ALICE \( C_4 = "1,2,1,1,0,2" \)

NTRU decrypt5:
\[
\text{ORBITER} -v 2 -define F -finite_field -q $(NTRU \_P) -end -with F -do -finite_field_activity -polynomial_center_lift $(ALICE \_C4) -end
\]

\[ A(X) = - X^5 + X^3 + X^2 - X + 1 \]
#plaintext BOB_MESSAGE

# 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
▷ # the inverse of 1061 mod 25320204 is 2076209

RSA_e2:
▷ $(ORBITER) -v 2 \\
▷ ▷ -RSA_encrypt_text 2076209 25330309 3 creamcheese
▷ #-RSA_encrypt_text 386093 1846303 creamcheese
▷ #408918,1735142,239809,654636

RSA_d2:
▷ $(ORBITER) -v 2 \\
▷ ▷ -RSA 1061 25330309 3 "19019931,1619805,740498,2671344"

im3:
▷ $(ORBITER) -v 2 \\
▷ ▷ -inverse_mod 2909 59248840
▷ #the inverse of 2909 mod 59248840 is 4358629

RSA_e3:
▷ $(ORBITER) -v 2 \\
▷ ▷ -RSA_encrypt_text 2909 59264263 3 encrypted

RSA_d3:
▷ $(ORBITER) -v 2 \\
▷ ▷ -RSA 4358629 59264263 3 "35270141,9642524,49091707"
# e =

im4:

$(ORBITER) -v 2 -inverse_mod 583 62236200

# the inverse of 583 mod 62236200 is 32559247

RSA_e4:

$(ORBITER) -v 2 \n
-RSA_encrypt_text 583 62251979 3 venividivici

#-RSA_encrypt_text 583 62251979 venividivici

#40513610,53979973,56449676,35068535

RSA_d4:

$(ORBITER) -v 2 \n
-RSA 32559247 62251979 "40513610,53979973,56449676,35068535"

# 7369 * 7127 = 52518863

# 7368 * 7126 = 52504368

im5:

$(ORBITER) -v 2 -inverse_mod 173 52504368

# the inverse of 173 mod 52504368 is 38543669

RSA_e5:

$(ORBITER) -v 2 \n
-RSA_encrypt_text 38543669 52518863 3 fascinating

#-RSA_encrypt_text 38543669 52518863 fascinating

#31526751,8962078,51045732,51894467

RSA_d5:

$(ORBITER) -v 2 \n
-RSA 173 52518863 "31526751,8962078,51045732,51894467"

RSA_d6:

$(ORBITER) -v 2 \n
-RSA 47177497 55040413 "28702119,48926559"
smooth:

```
$ (ORBITER) -v 2 -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

im8:

```
$ (ORBITER) -v 2 -inverse_mod 9017 60240544
```

# the inverse of 9017 mod 60240544 is 14430473

RSA_e7:

```
$ (ORBITER) -v 2 -RSA_encrypt_text 10048553 15806093 3 beachandfun
```

# 7853 * 7673 = 60256069
# 7852 * 7672 = 60240544

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:
 $\diamond$ $(ORBITER)$ -v 5 \\ 
 $\diamond$ $\diamond$ -seed 2531011 -find_pseudoprime 3 5 0 0
 $\diamond$ pdflatex pseudoprime_3.tex
 $\diamond$ open pseudoprime_3.pdf

pseudoprime10:
 $\diamond$ $(ORBITER)$ -v 5 \\ 
 $\diamond$ $\diamond$ -seed 2531011 -find_pseudoprime 10 5 5 5
 $\diamond$ pdflatex pseudoprime_10.tex
 $\diamond$ open pseudoprime_10.pdf

# 4460190157

pseudoprime11:
 $\diamond$ $(ORBITER)$ -v 5 \\ 
 $\diamond$ $\diamond$ -seed 2531011 -find_pseudoprime 11 5 5 5
 $\diamond$ pdflatex pseudoprime_11.tex
 $\diamond$ open pseudoprime_11.pdf

# 63814633367

# product is 284625399616057168619

pseudoprime11:
 $\diamond$ $(ORBITER)$ -v 5 \\ 
 $\diamond$ $\diamond$ -seed 2531011 -find_pseudoprime 11 5 5 5
 $\diamond$ pdflatex pseudoprime_11.tex
 $\diamond$ open pseudoprime_11.pdf

# 63814633367

# product is 284625399616057168619

pseudoprime20:
 $\diamond$ $(ORBITER)$ -v 5 \\ 
 $\diamond$ $\diamond$ -seed 2531011 -find_pseudoprime 20 5 5 5
 $\diamond$ pdflatex pseudoprime_20.tex
 $\diamond$ open pseudoprime_20.pdf

PR10:

705
9751 \> $(ORBITER) -v 5 -primitive_root 4460190157
9752
9753
9754 # mistake! long integer overflow
9755 # a primitive root modulo 165222861 is 1293
9756
9757
9758
9759
9760 pseudoprime50:
9761 \> $(ORBITER) -v 5 \
9762 \> -seed 2531011 -find_pseudoprime 50 5 0 0
9763 \> pdflatex pseudoprime_50.tex
9764 \> open pseudoprime_50.pdf
9765
9766
9767 #91322792878581218181431392170986926262336688354473
9768
9769 pseudoprime51:
9770 \> $(ORBITER) -v 5 \
9771 \> -seed 2531011 -find_pseudoprime 51 5 5 5
9772 \> pdflatex pseudoprime_51.tex
9773 \> open pseudoprime_51.pdf
9774
9775 #75460072774683470214089702490004944659715367045417
9776
9777 # product 68912245966050819606199994423264315732335295324400658436661744403244049
9778 572914094379904326661586100241
9779
9780 pseudoprime30:
9781 \> $(ORBITER) -v 5 \
9782 \> -seed 2531011 -find_pseudoprime 30 5 5 5
9783 \> pdflatex pseudoprime_30.tex
9784 \> open pseudoprime_30.pdf
9785
9786
9787 pseudoprime31:
9788 \> $(ORBITER) -v 5 \
9789 \> -seed 2531011 -find_pseudoprime 31 5 5 5
9790 \> pdflatex pseudoprime_31.tex
9791 \> open pseudoprime_31.pdf
9792 #877726676542264552372412985331
9793
9794 # 25149113232832986988837184692002835573476743643265896783515097
9795
9796 # maybe 2 seconds
pseudoprime33:

$(ORBITER) -v 5 \ -seed 2531011 -find_pseudoprime 33 5 5 5

pdflatex pseudoprime_33.tex
open pseudoprime_33.pdf

#371674199498295345543363004459891

pseudoprime34:

$(ORBITER) -v 5 \ -seed 2531011 -find_pseudoprime 34 5 5 5

pdflatex pseudoprime_34.tex
open pseudoprime_34.pdf

#9309708224110488378214945245346817

# 3460178351758962531912872979731874528849142238619677890786061016947
# 18 sec

pseudoprime35:

$(ORBITER) -v 5 \ -seed 2531011 -find_pseudoprime 35 5 5 5

pdflatex pseudoprime_35.tex
open pseudoprime_35.pdf

#81329557792505271120435930267680203

pseudoprime36:

$(ORBITER) -v 5 \ -seed 2531011 -find_pseudoprime 36 5 5 5

pdflatex pseudoprime_36.tex
open pseudoprime_36.pdf

#162624680891993404333363207561599139

# 13226193383093105242537919350220354135219441323641636665484262532145217
# factoring takes 46 seconds

MATH360_hw2:

$(ORBITER) -v 3 \
```bash
9844 ▷ ▷ -define F -finite_field -q 16 -end \n9845 ▷ ▷ -with F -do -finite_field_activity \n9846 ▷ ▷ -parse_and_evaluate "test" "" "a+b" "a=8,b=14" -end
9847 ▷ $(ORBITER) -v 3 \n9848 ▷ ▷ -define F -finite_field -q 16 -end \n9849 ▷ ▷ -with F -do -finite_field_activity \n9850 ▷ ▷ -parse_and_evaluate "test" "" "a*b" "a=9,b=13" -end
9851 ▷ $(ORBITER) -v 3 \n9852 ▷ ▷ -define F -finite_field -q 16 -end \n9853 ▷ ▷ -with F -do -finite_field_activity \n9854 ▷ ▷ -parse_and_evaluate "test" "" "a*a*a*a*a" "a=9" -end
9855 ▷ $(ORBITER) -v 3 \n9856 ▷ ▷ -define F -finite_field -q 16 -end \n9857 ▷ ▷ -with F -do -finite_field_activity \n9858 ▷ ▷ -parse_and_evaluate "test" "" "(a+b)*(a+b)" "a=5,b=7" -end
9859 ▷ $(ORBITER) -v 3 \n9860 ▷ ▷ -define F -finite_field -q 16 -end \n9861 ▷ ▷ -with F -do -finite_field_activity \n9862 ▷ ▷ -parse_and_evaluate "test" "" "a*a+b*b" "a=5,b=7" -end
9863
9864
9865 F_256_Rijndahl:
9866 ▷ $(ORBITER) -v 3 \n9867 ▷ ▷ -define F -finite_field -q 256 -override_polynomial 283 -end \n9868 ▷ ▷ -with F -do -finite_field_activity -cheat_sheet_GF -end
9869
9870
9871
9872
9873
9874
9875
9876 all_square_roots_mod_n_1549411:
9877 ▷ $(ORBITER) -v 3 -all_square_roots_mod_n 1075922 1549411
9878 ▷
9879
9880
9881 power_mod_211:
9882 ▷ $(ORBITER) -v 3 -power_mod_n 2 211
9883 ▷ $(ORBITER) -v 3 \n9884 ▷ ▷ -plot.function power_mod_n_a2_n211.csv
9885 ▷ $(ORBITER) -v 2 -draw_matrix \n9886 ▷ ▷ -input_csv_file power_mod_n_a2_n211_graph.csv \n9887 ▷ ▷ -box_width 10 -bit_depth 8 -partition 3 211 211 -end
9888
9889 power_mod_2_31:
9890 ▷ $(ORBITER) -v 3 -power_mod_n 2 31
```
# Chapter 10 - Coding Theory

## Section 10.1: Coding Theory

**SECTION CODING THEORY INTRODUCTION:**

Allen Gates noise 1 percent:

```bash
$ (ORBITER) -v 3
$ (ORBITER) -v 3
$ (ORBITER) -v 3

``` -random noise in bitmap file

```bash
open allen Gates.bmp
```

Hamming space 4 2 distance matrix:

```bash
$ (ORBITER) -Hamming_space_distance_matrix 4 2
```

Hamming space 4 2 distance matrix draw:

```bash
$ (ORBITER) -v 2 -draw_matrix
```

```bash
open Hamming_n4_q2_draw.bmp
```
Hamming code:

```bash
$ (ORBITER) -v 2 \
-define_macwilliams_system 7 4 2
pdflatex MacWilliams_n7_k4_q2.tex
open MacWilliams_n7_k4_q2.pdf
```

code.5.2.3.diagram:

```bash
$ (ORBITER) -v 2 \
-define F -finite_field -q 2 -end \
-with F -do -coding_theoretic_activity \
-code_diagram "code.5.2.3" \
-code_diagram $(CODE.5.2.3 CODEWORDS) 5 \
-metric_balls 1 \
-end
$ (ORBITER) -v 2 \
-draw_matrix \
-input_csv_file code.5.2.3_diagram_01_5_4.csv \
-box_width 25 -bit_depth 24 \
-partition 4 8 4 \
-end
```

Hamming.5.2_graph:

```bash
$ (ORBITER) -v 2 \
-define G -graph -Hamming 5 2 -end \
-with G -do \
-graph_theoretic_activity -export_csv -end \
-with G -do \
-graph_theoretic_activity -export_graphviz -end \
-with G -do \
-graph_theoretic_activity -save -end
$ (ORBITER) -v 2 -draw_matrix \
-input_csv_file Hamming.5.2.csv \
-box_width 8 -bit_depth 24 -partition 4 32 32 -end
```

```
```
```
```
```
```

Hamming.5.2_with.5.2.3.code:

```bash
$ (ORBITER) -v 2 \
```

Hamming.5.2_with.5.2.3.graph:
-define G -graph -Hamming 5 2\n-define G -graph -Hamming 5 2\n\n-graph_theoretic_activity -export_csv -end\n-graph_theoretic_activity -export_graphviz -end\n-graph_theoretic_activity -save -end\n-graph_theoretic_activity -automorphism_group -end
\n-pdflatex Hamming_5_2_code_5_2_3_report.tex
open Hamming_5_2_code_5_2_3_report.pdf

# group has order 32

\n
\n
\n
\n
code_6:
$(ORBITER) -v 2\n-define F -finite_field -q 2 -end\n-with F -do -coding_theoretic_activity\n-general_code_binary 6 "0,60,50,41,14,21,27,39"\n-end\n$(ORBITER) -v 2 -draw_matrix\n-input_csv_file code_matrix_8_8.csv\n-box_width 20 -bit_depth 24\n-partition 2 "1,1,1,1,1,1,1,1" "1,1,1,1,1,1,1,1"\n-end\npdflatex code_6_8.tex
open code_6_8.pdf
open code_matrix_8_8_draw.bmp

# linear code with generator matrix
\n
# Section 10.2: Linear codes

SECTION CODING THEORY LINEAR CODES:
10032
10033
10034 RM_3_1:
10035 \(\texttt{(ORBITER) -v 2 \}\)
10036 \(\texttt{\quad -define F -finite_field -q 2 -end \}\)
10037 \(\texttt{\quad -define C -code -field F \}\)
10038 \(\texttt{\quad \quad -first_order_Reed_Muller 3 \}\)
10039 \(\texttt{\quad -end \}\)
10040 \(\texttt{\quad -with C -and F -do -coding_theoretic_activity \}\)
10041 \(\texttt{\quad \quad -export_magma RM_3_1.magma \}\)
10042 \(\texttt{\quad -end \}\)
10043
10044
10045
10046
10047
10048 simplex_code:
10049 \(\texttt{(ORBITER) -v 2 \}\)
10050 \(\texttt{\quad -define F -finite_field -q 2 -end \}\)
10051 \(\texttt{\quad -define v -vector -field F -format 3 \}\)
10052 \(\texttt{\quad \quad -dense $(SIMPLEX_CODE_GENERATOR) \}\)
10053 \(\texttt{\quad \quad -end \}\)
10054 \(\texttt{\quad -define C -code -field F \}\)
10055 \(\texttt{\quad \quad -linear_code_through_generator_matrix v \}\)
10056 \(\texttt{\quad -end \}\)
10057
10058
10059
10060 Hamming_generator:
10061 \(\texttt{(ORBITER) -v 2 \}\)
10062 \(\texttt{\quad -define F -finite_field -q 2 -end \}\)
10063 \(\texttt{\quad -define v -vector -field F -format 3 \}\)
10064 \(\texttt{\quad \quad -dense $(SIMPLEX_CODE_GENERATOR) \}\)
10065 \(\texttt{\quad \quad -end \}\)
10066 \(\texttt{\quad -with F -do \}\)
10067 \(\texttt{\quad -finite_field_activity \}\)
10068 \(\texttt{\quad \quad -nullspace v \}\)
10069 \(\texttt{\quad -end \}\)
10070 pdflatex nullspace_3_7.tex
10071 open nullspace_3_7.pdf
10072
10073 # basis in binary:
10074 # 67,37,22,15
10075 #-normalize_from_the_right \)
10076
10077
10078
Hamming code:

```plaintext
$(ORBITER) -v 2 
(define F -finite_field -q 2 -end \n(define v -vector -field F -format 3 \n(dense $(SIMPLEX_CODE_GENERATOR) 
(end \n(define C -code -field F \n(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:

$ (ORBITER) -v 2 \n(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: (0,255,170,85,204,51,102,153,240,15,90,165,60,195,150,105)

```
RM_3_1_from_generator_matrix:

$ (ORBITER) -v 2 \n(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

```
RM_4_1 and codewords:

```plaintext
$(ORBITER) -v 2 \n-def F -finite_field -q 2 -end \n-def 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-end \n-with C -and F -do -coding_theoretic_activity \n-export_codewords RM_4_1_codewords.csv \n-end \n-with C -and F -do -coding_theoretic_activity \n-export_genma RM_4_1_genma.csv \n-end
```

RM_5_1 and codewords:

```plaintext
$(ORBITER) -v 2 \n-def F -finite_field -q 2 -end \n-def C -code -field F -first_order_Reed_Muller 5 -end \n-with C -and F -do -coding_theoretic_activity \n-export_magma RM_5_1.magma \n-end \n-with C -and F -do -coding_theoretic_activity \n-export_codewords RM_5_1_codewords.csv \n-end \n-with C -and F -do -coding_theoretic_activity \n-export_genma RM_5_1_genma.csv \n-end
```

Hamming code words old:

```plaintext
$(ORBITER) -v 2 \n-def v -vector -dense $(HAMMING_CODE_ROWS_IN_BINARY_RANKS) -end \n-linear_code_through_bases 7 v
```

Hamming weight enumerator:

```plaintext
$(ORBITER) -v 2 \n-def F -finite_field -q 2 -end \n-def v -vector -field F -format 4 
```
Hamming minimum distance:

```plaintext
Hamming_minimum_distance:

Hamming code diagram:
```

Golay23 minimum distance:

```
Golay23_minimum_distance:
```
-draw_matrix

-input_csv_file Hamming_7_4_diagram_7_16.csv

-box_width 25 -bit_depth 14

-partition 4 16 8

-end

$(ORBITER) -v 2 -define F -finite_field -q 2

-coding_theoretic_activity -code_diagram "Hamming_7_4_word_0" "0" 7
-metric_balls 1

-end

$(ORBITER) -v 2 -define F -finite_field -q 2

-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

-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

-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

-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

-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

-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

-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

-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

-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

-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

-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

-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

-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

-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

-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

-input_csv_file Hamming_7_4_word_%L_diagram_7_1.csv

-box_width 25 -bit_depth 8 -partition 4 16 8 -end

-end_loop

code_Hamming_systematic:

$(ORBITER) -v 2
-define v -vector -dense $(HAMMING_CODE_ROWS_IN_BINARY_RANKS) -end \
-linear_code_through_basis 7 v
-define v -vector -format 4 -field F -dense $(HAMMING_CODE_GENERATOR) -end \
-with F -do \
-finite_field_activity \
-RREF v -end
pdflatex RREF_example_q2_4_7.tex
gs -sDEVICE=png16 -dFIXEDMEDIA \
-dDEVICEWIDTHPOINTS=500 -dDEVICEHEIGHTPOINTS=450 \
-r240 -oHamming_dual_page%02d.png \
RREF_example_q2_4_7.pdf

Hamming_nullspace:
-define v -vector -format 4 -field F2 -dense $(HAMMING_CODE_GENERATOR) -end \
-with F2 -do \
-finite_field_activity \
-nullspace v \
-normalize_from_the_right \
-end
10298 \> pdflatex nullspace_4_7.tex
10299 \> open nullspace_4_7.pdf
10300
10301
10302 \#check equations of the Hamming code:
10303 \> a4+a5+a6+a7 =1+0+1+0=0 mod2 OK.
10304 \> a2+a3+a6+a7 =0+1+1+0=0 mod2 OK.
10305 \> a1+a3+a5+a7 =1+1+0+0=0 mod2 OK.
10306
10307 \#1010101
10308 \#0110011
10309 \#0001111
10310
10311
10312
10313
10314
10315 Hamming_long:
10316 \> $(ORBITER) -v 2 -long_code 7 4 \n10317 \> "0,5,6" \n10318 \> "1,4,6" \n10319 \> "2,4,5" \n10320 \> "3,4,5,6"
10321 \> $(ORBITER) -v 2 -loop L 0 16 1 -draw_matrix \n10322 \> -input_csv_file long_code_genma_n7_k4_codeword_%L.csv \n10323 \> -box_width 25 -bit_depth 8 -partition 3 4 2 -end \n10324 \> -end_loop
10325
10326
10327 \#long_code_genma_n7_k4_codeword_0.csv
10328 \#long_code_genma_n7_k4_codeword_15.csv
10329 \#Weight distribution:( 0, 3^7, 4^7, 7 )
10330
10331
10332 Hamming_singer:
10333 \> $(ORBITER) -v 3 \n10334 \> -define G -linear_group -PGL 3 2 -singer 1 -end \n10335 \> -with G -do \n10336 \> -group_theoretic_activity \n10337 \> -report \n10338 \> -orbits.on.points \n10339 \> -end
10340 \> pdflatex PGL_3_2_Singer_3_2_1_report.tex
10341 \> open PGL_3_2_Singer_3_2_1_report.pdf
10342
10343 \# cycle is 0,1,2,5,3,4,6
10344
718
Hamming_cyclic_generator:

```
$\text{ORBITER} -v 2 \
\text{define } F \text{ -finite_field -q 2 -end} \\
\text{define } v \text{ -vector -format 3 -field } F \\
\text{dense } $(\text{SIMPLEX_CODE_GENMA_CYCLIC}) \\
\text{end} \\
\text{with } F \text{ -do -finite_field_activity} \\
\text{nullspace } v \\
\text{end} \\
pdflatex nullspace_{3,7}.tex \\
open nullspace_{3,7}.pdf
```

Hamming_cyclic_long:

```
$\text{ORBITER} -v 2 -long \_code 7 4 \\
"0,4,6" \\
"1,4,5,6" \\
"2,4,5" \\
"3,5,6" \\
$\text{ORBITER} -v 2 -loop L 0 16 1 -draw_matrix \\
\text{input_csv_file long_code_genma_n7_k4_codeword_{L}.csv} \\
\text{box_width 25 -bit_depth 8 -partition 3 4 2 -end} \\
\text{end_loop}
```

Hamming_cyclic:

```
$\text{ORBITER} -v 2 \\
\text{define } v \text{ -vector -dense } "69,39,22,11" \text{ -end} \\
\text{linear_code_through_basis 7 } v \\
$\text{ORBITER} -v 2 -draw_matrix \\
\text{input_csv_file code_matrix_{16,8}.csv} \\
\text{box_width 25 -bit_depth 8 -partition 2 16 8 -end} \\
\text{end} \\
open code_matrix_{16,8}.draw.bmp \\
pdflatex code_{7,16}.tex \\
open code_{7,16}.pdf
```

Hamming_cyclic_clean_ns:

```
$\text{ORBITER} -v 2 \\
\text{define } F \text{ -finite_field -q 2 -end} \\
\text{define } v \text{ -vector -format 3 -field } F \\
```
nullspace_4_7.tex
open nullspace_4_7.pdf

Hamming_cyclic_clean:
$(ORBITER) -v 2 
  -define v -vector -dense "88,44,22,11" -end 
  -linear_code_through_basis 7 v 
$(ORBITER) -v 2 -draw_matrix 
-input_csv_file code_matrix_16_8.csv 
-box_width 25 -bit_depth 8 -partition 2 16 8 -end 
open code_matrix_16_8.draw.bmp 
pdflatex code_7_16.tex 
open code_7_16.pdf

Hamming_cyclic_clean_long:
$(ORBITER) -v 2 -long_code 7 4 
"0,2,3" 
"1,3,4" 
"2,4,5" 
"3,5,6" 
$(ORBITER) -v 2 
-loop L 0 16 1 -draw_matrix 
-input_csv_file 
-long_code_genma_n7_k4_codeword_%L.csv 
-box_width 25 -bit_depth 8 
-partition 3 4 2 -end 
-end_loop

#11111111 = 255
#01010101 = 170
#00110011 = 204
#00001111 = 240
Golay23 code words:
 Golay23_code_words:
 Golay23_code_diagram:
 Golay23_code_diagram_draw:
10486 $\text{(ORBITER)} -v 2$
10487 $\text{-draw_matrix}$
10488 $\text{-input_csv_file Golay_23_diagram_01_23_4096.csv}$
10489 $\text{-box_width 4 -bit_depth 8}$
10490 $\text{-partition 20 4096 2048}$
10491 $\text{-end}$
10492
10493
10494
10495
10496
10497 # Section 10.4: Coding Theory - CRC codes
10498
10500 SECTION CODING_THEORY_CRC_CODES:
10501 10502 10503 10504 10505 10506 encode_text_5bits:
10507 $\text{(ORBITER)} -encode_text_5bits$
10508 $\text{"Hithere" "text.csv"}$
10509 $\text{(ORBITER)} -v 2$
10510 $\text{-define F -finite_field -q 2 -end}$
10511 $\text{-with F -do}$
10512 $\text{-coding_theoretic_activity}$
10513 $\text{-polynomial_division_from_file}$
10514 $\text{text.csv 13 -end}$
10515 $\text{pdflatex polynomial_division_file_13.tex}$
10516 $\text{open polynomial_division_file_13.pdf}$
10517 10518 10519 encode_text_5bits_check:
10520 $\text{(ORBITER)} -v 2$
10521 $\text{-define F -finite_field -q 2 -end}$
10522 $\text{-with F -do}$
10523 $\text{-coding_theoretic_activity}$
10524 $\text{-polynomial_division_from_file}$
10525 $\text{text_with_1error.csv 13}$
10526 $\text{-end}$
10527 $\text{pdflatex polynomial_division_file_13.tex}$
10528 $\text{open polynomial_division_file_13.pdf}$
10529 10530 10531 10532 encode_text_5bits_1error:
10533 ❯ $(ORBITER) -encode_text_5bits \n10534 ❯ "Hithere" "text.csv"
10535 ❯ $(ORBITER) -v 2 \n10536 ❯ -define F -finite_field -q 2 -end \n10537 ❯ -with F -do \n10538 ❯ -coding_theoretic_activity \n10539 ❯ -polynomial_division_from_file_all_k_bit_error_patterns \n10540 ❯ -polynomial_division_from_file_all_k_bit_error_patterns \n10541 ❯ -end
10542 ❯ pdflatex polynomial_division_file_all_1_error_patterns_13.tex
10543 ❯ open polynomial_division_file_all_1_error_patterns_13.pdf
10544
10545
10546
10547 CRC_3_128:10:
10548 ❯ $(ORBITER) -v 1 \n10549 ❯ -define F -finite_field -q 2 -end \n10550 ❯ -with F -do -coding_theoretic_activity \n10551 ❯ -find_CRC_polynomials 3 128 10 \n10552 ❯ -end
10553
10554
10555
10556
crc32_test:
10557 ❯ $(ORBITER) -v 3 \n10558 ❯ -define F -finite_field -q 2 -end \n10559 ❯ -with F -do -coding_theoretic_activity \n10560 ❯ -crc32 "123456789" \n10561 ❯ -end
10562
crc32_test_hexdata:
10563 ❯ $(ORBITER) -v 3 \n10564 ❯ -define F -finite_field -q 2 -end \n10565 ❯ -with F -do -coding_theoretic_activity \n10566 ❯ -crc32_hexdata "7BD11C4010" \n10567 ❯ -end
10568
crc32_Berlekamp_matrix:
10569 ❯ $(ORBITER) -v 2 \n10570 ❯ -define F -finite_field -q 2 -end \n10571 ❯ -define v -vector -finite_field F -sparse 33 $(CRC32_ETH) -end \n10572 ❯ -with F -do \n10573 ❯ -finite_field_activity \n10574 ❯ -Berlekamp_matrix v \n10575 ❯ -end
10576
10577
10578
10579

723
10580
10581 CRC_F256_roots_771:
10582 >> $(ORBITER) -v 3 \n10583 >> >> -define F -finite_field -q 256 -end \n10584 >> >> -with F -do -coding_theoretic_activity \n10585 >> >> >> -nth_roots 771 \n10586 >> >> >> -end
10587
10588
10589
10590
10591 CRC_F256_BCH_code_d2:
10592 >> $(ORBITER) -v 3 \n10593 >> >> -define F -finite_field -q 256 -end \n10594 >> >> -with F -do -coding_theoretic_activity \n10595 >> >> >> -make_BCH_code 771 2 \n10596 >> >> >> -end
10597 >> pdflatex BCH_codes_q256_n771_d2.tex
10598 >> open BCH_codes_q256_n771_d2.pdf
10599
10600
10601 CRC_POLY_Q256_DEG2_DENSE="214,167,1"
10602
10603
10604 CRC_F256_BCH_write_code_for_division_d2:
10605 >> $(ORBITER) -v 2 \n10606 >> >> -define F -finite_field -q 256 -end \n10607 >> >> -define A -vector -field F -sparse 772 "1,771,1,0" -end \n10608 >> >> -define B -vector -field F -dense $(CRC_POLY_Q256_DEG2_DENSE) -end \n10609 >> >> -with F -do \n10610 >> >> -coding_theoretic_activity \n10611 >> >> >> -write_code_for_division \n10612 >> >> >> -check_q256_n771_r2.cpp A B \n10613 >> >> >> -end
10614 >> g++ check_q256_n771_r2.cpp -o check_q256_n771_r2.out
10615 >> ./check_q256_n771_r2.out
10616
10617
10618
10619
10620
10621
10622
10623 F256_BCH_code_d16:
10624 >> $(ORBITER) -v 3 \n10625 >> >> -define F -finite_field -q 256 -end \n10626 >> >> -with F -do -coding_theoretic_activity \n
#generator polynomial is $X^{30} + 253X^{29} + 174X^{28} + 109X^{27} + 97X^{26} + 144X^{25} + 112X^{24} + 212X^{23} + 192X^{22} + 169X^{21} + 24X^{20} + 150X^{19} + 110X^{18} + 248X^{17} + 3X^{16} + 193X^{15} + 194X^{14} + 205X^{13} + 9X^{12} + 56X^{11} + 95X^{10} + 199X^{9} + 108X^{8} + 58X^{7} + 160X^{6} + 148X^{5} + 138X^{4} + 24X^{3} + 210X^{2} + 26X + 1$

POLY_Q256
DEG30
SPARSE="1,0,26,1,210,2,24,3,"

POLY_Q256
DEG30
DENSE="1,26,210,24,138,148,160,58,7,108,8,199,9,95,10,56,11,9,12,205,13,194,14,193,15,3,16,248,17,110,19,150,24,169,192,212,23,112,24,144,25,97,26,109,27,174,28,253,29,1,30"

POLY_Q256
DEG30
DENSE="1,26,210,24,138,148,160,58,7,108,8,199,9,95,10,56,11,9,12,205,13,194,14,193,15,3,16,248,17,110,19,150,24,169,192,212,23,112,24,144,25,97,26,109,27,174,28,253,29,1,30"

F256_BCH
write
code
for
division
d16:

$\text{ORBITER} -v 2$

-define F -finite_field -q 256 -end 
-define A -vector -field F -sparse 772 "1,771,1,0" -end 
-define B -vector -field F -dense $(POLY_Q256\_DEG30\_DENSE) -end 
-with F -do 
-coding.theoretic_activity 
-write_code_for_division 
-check_q256_n771_r30.cpp A B
-end

\texttt{g++ check\_q256\_n771\_r30.cpp -o check\_q256\_n771\_r30.out}

./check\_q256\_n771\_r30.out

F256_BCH_code\_d16\_division:

$\text{ORBITER} -v 2$

-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 

725
10670 ▷ ▷ -finite_field_activity \\  
10671 ▷ ▷ -polynomial_division A B -end  
10672  
10673  
10674  
10675 F256_BCH_code_d16_error:  
10676 ▷ $(ORBITER) -v 2 \  
10677 ▷ ▷ -define F -finite_field -q 256 -end \  
10678 ▷ ▷ -define A -vector -field F -sparse 771 "2,30,3,31,55,770" -end \  
10679 ▷ ▷ -define B -vector -field F -dense $(POLY_Q256_DEG30_DENSE) -end \  
10680 ▷ ▷ -with F -do \  
10681 ▷ ▷ -finite_field_activity \  
10682 ▷ ▷ -polynomial_division A B -end  
10683  
10684  
10685  
10686  
10687 #CRC_FILE=allen_Gates  
10688 CRC_FILE=javad-allahyari-Fs1E2JXM3Gc-unsplash  
10689  
10690 CRC_FILE_EXTENSION=bmp  
10691  
10692  
10693 crc_encode_16:  
10694 ▷ $(ORBITER) -v 3 \  
10695 ▷ ▷ -define F -finite_field -q 2 -end \  
10696 ▷ ▷ -with F -do \  
10697 ▷ ▷ ▷ -coding_theoretic_activity \  
10698 ▷ ▷ ▷ ▷ -crc_encode_file_based $(CRC_FILE).$(CRC_FILE_EXTENSION) $(CRC_FILE)_crc1 6.bin crc16 771 \  
10699 ▷ ▷ ▷ -end  
10700  
10701 #-rw-r--r-- 1 betten staff  646576 Aug 24 14:35 allen_Gates_crc16.bin  
10702 #-rw-r--r-- 1 betten staff  21656232 Aug 24 15:35 javad-allahyari-Fs1E2JXM3Gc-unsplash_crc16.bin  
10703  
10704  
10705  
10706 crc_encode_32:  
10707 ▷ $(ORBITER) -v 3 \  
10708 ▷ ▷ -define F -finite_field -q 2 -end \  
10709 ▷ ▷ -with F -do \  
10710 ▷ ▷ ▷ -coding_theoretic_activity \  
10711 ▷ ▷ ▷ ▷ -crc_encode_file_based $(CRC_FILE).$(CRC_FILE_EXTENSION) $(CRC_FILE)_crc3 2.bin crc32 771 \  
10712 ▷ ▷ -end  
10713
10714
10715 # rw-r--r-- 1 betten staff 648262 Aug 24 14:34 allen_Gates_crc32.bin
10716
10717
10718
10719 # crc_encode_new:
10720 # $(ORBITER) -v 3 \
10721 # $define F -finite_field -q 256 -end \n10722 # $with F -do \n10723 # $coding_theoretic_activity \n10724 # $crc_new_file_based $(CRC_FILE).$(CRC_FILE_EXTENSION) \n10725 # $end
10726
10727
10728 introduce_errors_16_500000:
10729 $(ORBITER) -v 3 \n10730 $introduce_errors \n10731 $input $(CRC_FILE)_crc16.bin \n10732 $output $(CRC_FILE)_crc16.e.bin \n10733 $block_based_error_generator \n10734 $block_length 771 \n10735 $threshold 500000 \n10736 $file_based_error_generator 500000 \n10737 $nb_repeats 30 \n10738 $end
10739
10740
10741 introduce_errors_32_100000:
10742 $(ORBITER) -v 3 \n10743 $introduce_errors \n10744 $input $(CRC_FILE)_crc32.bin \n10745 $output $(CRC_FILE)_crc32.e.bin \n10746 $block_based_error_generator \n10747 $block_length 771 \n10748 $threshold 100000 \n10749 $file_based_error_generator 100000 \n10750 $nb_repeats 30 \n10751 $end
10752
10753
10754 check_errors_16:
10755 $(ORBITER) -v 3 \n10756 $check_errors \n10757 $input $(CRC_FILE)_crc16.e.bin \n10758 $output $(CRC_FILE)_recovered.$(CRC_FILE_EXTENSION) \n10759 $crc_type crc16 \n10760 $error_log $(CRC_FILE)_crc16_e_pattern.csv \n
727
check_errors_32:

extract_block:

# Section 10.5: Coding Theory - Reed-Muller codes

SECTION CODING THEORY REED MULLER CODES:

RM_3_1_Hamming_space_diagram:
$\text{ORBITER} -v 2 \$

```
$\text{ORBITER} -v 2\
$\text{ORBITER} -draw_matrix\
$\text{ORBITER} -input_csv_file RM_3_1_holes_8_16.csv\n$\text{ORBITER} -box_width 25 -bit_depth 8\n$\text{ORBITER} -partition 4 16 16\n$\text{ORBITER} -end
```

```
$\text{ORBITER} -v 2\
$\text{ORBITER} -draw_matrix\
$\text{ORBITER} -input_csv_file RM_3_1_diagram_01_8_16.csv\
$\text{ORBITER} -box_width 25 -bit_depth 8\n$\text{ORBITER} -partition 4 16 16\n$\text{ORBITER} -end
```

```
$\text{ORBITER} -v 2\
$\text{ORBITER} -draw_matrix\
$\text{ORBITER} -input_csv_file RM_3_1_diagram_01_8_16.csv\
$\text{ORBITER} -box_width 25 -bit_depth 8\n$\text{ORBITER} -partition 4 16 16\n$\text{ORBITER} -end
```

```
open RM_3_1_holes_8_16_draw.bmp
```

```
RM_3_1_split:
$\text{ORBITER} -split_by_values RM_3_1_holes_8_16.csv
```

```
RM_3_1_holes_draw:
$\text{ORBITER} -v 2\
$\text{ORBITER} -loop L 0 3 1\
$\text{ORBITER} -draw_matrix\
$\text{ORBITER} -input_csv_file RM_3_1_holes_8_16_value%L.csv\
$\text{ORBITER} -box_width 25 -bit_depth 8 -partition 5 16 16\
$\text{ORBITER} -end\
$\text{ORBITER} -end_loop
```

```
RM_3_1_hole0:
$\text{ORBITER} -v 3\
$\text{ORBITER} -define F -finite_field -q 2 -end\
$\text{ORBITER} -with F -do -finite_field_activity\
$\text{ORBITER} -algebraic_normal_form\
$\text{ORBITER} -RM_3_1_holes_8_16_value0.csv 8\
$\text{ORBITER} -end
```

```
E_0 + E_1 + E_2 + E_3 + E_4
```

```
RM_3_1_hole1:
$\text{ORBITER} -v 3\
$\text{ORBITER} -define F -finite_field -q 2 -end\
```
10855  
10856  
10857  
10858  
10859  
10860  
10861  
10862  
10863  
10864  
10865  
10866  
10867  
10868  
10869  
10870  
10871  
10872  

# 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} 

RM_3_1_holes_8_16_value1.csv 8 

RM_3_1_holes_8_16_value2.csv 8 

E_2 + E_3 + E_4
10873
10874
10875
10876 RM_4_1:
10877 \$ \$(ORBITER) -v 2 \$
10878 \$ define F -finite_field -q 2 -end \$
10879 \$ define C -code -field F -first_order_Reed_Muller 4 -end \$
10880 \$ with C -and F -do -coding_theoretic_activity \$
10881 \$ export_wmagma RM_4_1.magma \$
10882 \$ end \$
10883 \$ with C -and F -do -coding_theoretic_activity \$
10884 \$ export_codewords RM_4_1_codewords.csv \$
10885 \$ end \$
10886
10887
10888
10889 RM_4_1_old:
10890 \$ \$(ORBITER) -v 2 \$
10891 \$ define F -finite_field -q 2 -end \$
10892 \$ with F -do \$
10893 \$ coding_theoretic_activity \$
10894 \$ linear_code_through_columns_of_parity_check 5 \$
10895 \$ $(REED_MULLER_4_1_COLUMNS_OF_PARITY_CHECK) \$
10896 \$ end \$
10897 pdflatex code_n16_k5_q2.tex
10898 open code_n16_k5_q2.pdf
10899
10900
10901 # codewords_n16_k5_q2.csv
10902
10903
10904
10905 RM_4_1_diagram:
10906 \$ \$(ORBITER) -v 2 \$
10907 \$ define F -finite_field -q 2 -end \$
10908 \$ with F -do \$
10909 \$ coding_theoretic_activity \$
10910 \$ code_diagram_from_file "RM_4_1" \$
10911 \$ codewords_n16_k5_q2.csv 16 \$
10912 \$ end \$
10913 \$ #-enhance 4
10914 \$ #-metric_balls 3
10915
10916 RM_4_1_diagram_draw:
10917 \$ \$(ORBITER) -v 2 \$
10918 \$ draw_matrix \$
10919 \$ input_csv_file RM_4_1_diagram_01_16_32.csv \$

731
RM_4_1_split:
$(ORBITER) -split_by_values RM_4_1_holes_16_32.csv
RM_4_1_diagram_draw_holes:
$(ORBITER) -v 2
$(ORBITER) -draw_matrix
$(ORBITER) -input_csv_file RM_4_1_holes_16_32.csv
$(ORBITER) -box_width 5 -bit_depth 8
$(ORBITER) -partition 10 256 256
$(ORBITER) -end
$(ORBITER) -v 2
$(ORBITER) -loop L 0 7 1
$(ORBITER) -draw_matrix
$(ORBITER) -input_csv_file RM_4_1_holes_16_32_value%L.csv
$(ORBITER) -box_width 5 -bit_depth 8
$(ORBITER) -partition 10 256 256
$(ORBITER) -end
$(ORBITER) -end_loop
RM_4_1_diagram_metric_balls:
$(ORBITER) -v 2
$(ORBITER) -define F -finite_field -q 2 -end
$(ORBITER) -with F -do
$(ORBITER) -coding_theoretic_activity
$(ORBITER) -code_diagram_from_file "RM_4_1"
$(ORBITER) -codewords n16_k5_q2.csv 16
$(ORBITER) -metric_balls 3
$(ORBITER) -end
$(ORBITER) -v 2
$(ORBITER) -draw_matrix
$(ORBITER) -input_csv_file RM_4_1_diagram_16_32.csv
$(ORBITER) -box_width 25 -bit_depth 8
$(ORBITER) -partition 10 256 256
$(ORBITER) -end
RM_4_1_hole0:
$(ORBITER) -v 3
\texttt{define F} -\texttt{finite_field} -q 2 -\texttt{end} \\
\texttt{with F} -\texttt{do} -\texttt{finite_field} -\texttt{activity} \\
\texttt{\ldots} -\texttt{algebraic}\texttt{\_normal_form} \\
\texttt{\ldots} -\texttt{RM}\texttt{\_4\_1}\texttt{\_holes}\texttt{\_16\_32}\texttt{\_value0}\texttt{\_csv} 16 \\
\texttt{end} \\
2 \texttt{-draw_options} \\
2 \texttt{-\_radius 100} \\
2 \texttt{-\_line_width 1.0 -\_embedded} \\
2 \texttt{-end} \\
2 \texttt{-\_mod_n} -n 21 -\texttt{\_file mod}\texttt{\_21}\texttt{\_cyclo}\texttt{\_tomic} \\
2 \texttt{-\_cyclo}\texttt{\_sets 8 }^\texttt{\_1,2,4,5,7,10,13}\texttt{\_end} \\
2 \texttt{pdflatex mod}\texttt{\_21}\texttt{\_cyclo}\texttt{\_tomic}\texttt{\_draw.tex} \\
2 \texttt{open mod}\texttt{\_21}\texttt{\_cyclo}\texttt{\_tomic}\texttt{\_draw.pdf} \\
2 \texttt{\ldots} -\texttt{\_generator polynomial is } X^{4} + 4X^{3} + 4X^{2} + 3X + 4 \\
2 \texttt{\ldots} -\texttt{\_BCH}\texttt{\_code 21}\texttt{\_3} \\
2 \texttt{\ldots} -\texttt{\_end} \\
2 \texttt{pdflatex BCH}\texttt{\_codes}\texttt{q8}\texttt{\_n21}\texttt{\_d3.tex} \\
2 \texttt{open BCH}\texttt{\_codes}\texttt{q8}\texttt{\_n21}\texttt{\_d3.pdf} \\
2 \texttt{\ldots} -\texttt{\_generator polynomial is } X^{4} + 4X^{3} + 4X^{2} + 3X + 4 \\
2 \texttt{\ldots} -\texttt{\_BCH}\texttt{\_code 21}\texttt{\_4} \\
2 \texttt{\ldots} -\texttt{\_end}
11014
11015
11016
11017
11018
11019
11020
11021
11022
11023
11024
11025
11026
11027
11028
11029
11030
11031
11032
11033
11034
11035
11036
11037
11038
11039
11040
11041
11042
11043
11044
11045
11046
11047
11048
11049
11050
11051
11052
11053
11054
11055
11056
11057
11058
11059

#generator polynomial is X^{5} + 6X^{4} + 7X^{3} + 2X + 3

F 8 BCH code d5:
. $(ORBITER) -v 3 \
. . -define F -finite field -q 8 -override polynomial 11 -end \
. . -with F -do -coding theoretic activity \
. . . -make BCH code 21 5 \
. . -end
. 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="\
2,1,2,1,2,3,3,1,0,0,0,0,0,0,0,0,0,0,0,0,0,\
0,2,1,2,1,2,3,3,1,0,0,0,0,0,0,0,0,0,0,0,0,\
0,0,2,1,2,1,2,3,3,1,0,0,0,0,0,0,0,0,0,0,0,\
0,0,0,2,1,2,1,2,3,3,1,0,0,0,0,0,0,0,0,0,0,\
0,0,0,0,2,1,2,1,2,3,3,1,0,0,0,0,0,0,0,0,0,\
0,0,0,0,0,2,1,2,1,2,3,3,1,0,0,0,0,0,0,0,0,\
0,0,0,0,0,0,2,1,2,1,2,3,3,1,0,0,0,0,0,0,0,\
0,0,0,0,0,0,0,2,1,2,1,2,3,3,1,0,0,0,0,0,0,\
0,0,0,0,0,0,0,0,2,1,2,1,2,3,3,1,0,0,0,0,0,\
0,0,0,0,0,0,0,0,0,2,1,2,1,2,3,3,1,0,0,0,0,\
0,0,0,0,0,0,0,0,0,0,2,1,2,1,2,3,3,1,0,0,0,\
0,0,0,0,0,0,0,0,0,0,0,2,1,2,1,2,3,3,1,0,0,\
0,0,0,0,0,0,0,0,0,0,0,0,2,1,2,1,2,3,3,1,0,\
0,0,0,0,0,0,0,0,0,0,0,0,0,2,1,2,1,2,3,3,1"
CODE BCH F8 N21 K14 D5 GENMA="\
6,1,5,5,4,4,3,1,0,0,0,0,0,0,0,0,0,0,0,0,0,\
0,6,1,5,5,4,4,3,1,0,0,0,0,0,0,0,0,0,0,0,0,\
0,0,6,1,5,5,4,4,3,1,0,0,0,0,0,0,0,0,0,0,0,\
0,0,0,6,1,5,5,4,4,3,1,0,0,0,0,0,0,0,0,0,0,\
0,0,0,0,6,1,5,5,4,4,3,1,0,0,0,0,0,0,0,0,0,\
0,0,0,0,0,6,1,5,5,4,4,3,1,0,0,0,0,0,0,0,0,\
0,0,0,0,0,0,6,1,5,5,4,4,3,1,0,0,0,0,0,0,0,\
0,0,0,0,0,0,0,6,1,5,5,4,4,3,1,0,0,0,0,0,0,\
0,0,0,0,0,0,0,0,6,1,5,5,4,4,3,1,0,0,0,0,0,\
0,0,0,0,0,0,0,0,0,6,1,5,5,4,4,3,1,0,0,0,0,\

734


F$_8$BCH code $d=5$

$\text{(ORBITER)} -v 2$

-define F -finite_field -q 8 -override_polynomial 11 -end

-define v -vector -format 14 -field F

-compact $(\text{CODE}_BCH_F8_N21_D5\_GENMA\_OVERRIDE\_POLYNOMIAL11)$

-end

-with F -do

-coding.theoretic_activity

-minimum_distance v

-end

# important: use the same polynomial as when creating the code.

d=5

F$_8$BCH code $n=63$ $d=43$

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

-define F -finite_field -q 8 -override_polynomial 11 -end

-with F -do -coding.theoretic_activity

-make BCH code 63 43

-end

pdflatex BCH_codes_q8_n63_d43.tex

open BCH_codes_q8_n63_d43.pdf
F8_BCH_code_n63_d43_minimum_distance:

$\text{ORBITER} -v 2$

-define F -finite_field -q 8 -override_polynomial 11 -end

-define v -vector -format 9 -field F

-compact $(\text{CODE_BCH_F8_N63_K9_D43_GENMA})$

-end

-with F -do

-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
F_64.again:

```
$\text{\texttt{ORBITER}} -v 3 \ $
$\text{\texttt{-define F -finite_field -q 64 -end \}}$
$\text{\texttt{-with F -do -finite_field_activity \}}$
$\text{\texttt{-cheat_sheet_GF \}}$
$\text{\texttt{-end}}$
```

```
pdflatex GF_64.tex
open GF_64.pdf
```

```
BCH

\texttt{BCH}_255_5\_evaluate\_elementary\_symmetric\_functions\_1:

```
$\text{\texttt{ORBITER}} -v 3 \text{\texttt{-define F -finite_field -q 256 -end \}}$
```

```
BCH

\texttt{BCH}_255_5\_evaluate\_elementary\_symmetric\_functions\_2:

```
$\text{\texttt{ORBITER}} -v 3 \text{\texttt{-define F -finite_field -q 256 -end \}}$
```

```
BCH15:

```
$\text{\texttt{ORBITER}} -BCH 15 2 3$
```

```
#$(\text{\texttt{ORBITER}}) \text{\texttt{-BCH 15 2 3}}$
```

```
BCH15:
```

```
$\text{\texttt{-define F -finite_field -q 2 -end \}}$
```

```
$\text{\texttt{-with F -do -coding_theoretic_activity \}}$
```

```
$\text{\texttt{-BCH 15 2 5 \}}$
```

737
draw_mod_31:
$\text{\texttt{\$\texttt{(ORBITER) -v 2}}}$ \\
$\text{\texttt{\$\texttt{(ORBITER) -draw\_options -embedded -end}}}$ \\
$\text{\texttt{-draw\_mod\_n 31}}$ \\
$\text{\texttt{-file mod.31}}$ \\
$\text{\texttt{-draw\_mod\_n\_power\_cycle 2}}$ \\
$\text{\texttt{-end}}$

pdflatex mod.31.draw.tex

open mod.31.draw.pdf

draw_mod_127_power:
$\text{\texttt{\$\texttt{(ORBITER) -v 2}}}$ \\
$\text{\texttt{\$\texttt{(ORBITER) -draw\_options -scale 0.4 -embedded -end}}}$ \\
$\text{\texttt{-draw\_mod\_n 127 mod.127 -draw\_mod\_n\_power\_cycle 3}}$

pdflatex mod.127.draw.tex

open mod.127.draw.pdf

draw_mod_251:
$\text{\texttt{\$\texttt{(ORBITER) -v 2}}}$ \\
$\text{\texttt{\$\texttt{(ORBITER) -draw\_options -nodes\_empty -radius 10 -embedded -end}}}$ \\
$\text{\texttt{-draw\_mod\_n 251 mod.251}}$

pdflatex mod.251.draw.tex

open mod.251.draw.pdf

#-draw_mod_n_inverse

draw_mod_255_cyclotomic_1:
$\text{\texttt{\$\texttt{(ORBITER) -v 2}}}$ \\
$\text{\texttt{\$\texttt{(ORBITER) -draw\_options -nodes\_empty -radius 10}}}$ \\
$\text{\texttt{-line\_width 0.4 -embedded -end}}$ \\
$\text{\texttt{-draw\_mod\_n -n 255 -file mod.255\_cyclotomic.1}}$

$\text{\texttt{-cyclotomic\_sets 2 "1" -end}}$
\begin{verbatim}
11238 \> pdflatex mod_255_cyclotomic_1_draw.tex
11239 \> open mod_255_cyclotomic_1_draw.pdf
11240
11241 draw_mod_255_cyclotomic_3:
11242 \> \$\textsc{Orbiter} -v 2 \\
11243 \> \> -draw_options -nodes_empty -radius 10 \\
11244 \> \> \> -line_width 0.4 -embedded -end \\
11245 \> \> -draw_mod_n -n 255 -file mod_255_cyclotomic_3 \\
11246 \> \> -cyclotomic_sets 2 "3" -end
11247 \> pdflatex mod_255_cyclotomic_3_draw.tex
11248 \> open mod_255_cyclotomic_3_draw.pdf
11249
11250 draw_mod_255_cyclotomic_1_and_3:
11251 \> \$\textsc{Orbiter} -v 2 \\
11252 \> \> -draw_options -nodes_empty -radius 10 \\
11253 \> \> \> -line_width 0.4 -embedded -end \\
11254 \> \> -draw_mod_n -n 255 -file mod_255_cyclotomic_1_and_3 \\
11255 \> \> -cyclotomic_sets 2 "1,3" -end
11256 \> pdflatex mod_255_cyclotomic_1_and_3_draw.tex
11257 \> open mod_255_cyclotomic_1_and_3_draw.pdf
11258
11259 draw_mod_63_4_cyclotomic_3_6:
11260 \> \$\textsc{Orbiter} -v 2 \\
11261 \> \> -draw_options -radius 20 \\
11262 \> \> \> -line_width 0.1 -embedded -end \\
11263 \> \> -draw_mod_n -n 63 -file mod_63_4_cyclotomic_3_6 \\
11264 \> \> -cyclotomic_sets 4 "3,6" \\
11265 \> \> -cyclotomic_sets_thickness 50 \\
11266 \> \> -end
11267 \> pdflatex mod_63_4_cyclotomic_3_6_draw.tex
11268 \> open mod_63_4_cyclotomic_3_6_draw.pdf
11269
11270 BCH_F_64:
11271 \> \$\textsc{Orbiter} -v 3 \\
11272 \> \> -define F -finite_field -q 64 -end \\
11273 \> \> -with F -do -finite_field_activity \\
11274 \> \> \> -cheat_sheet_GF \\
11275 \> \> -end
11276 \> pdflatex GF_64.tex
11277
11278
11279 BCH_elementary_symmetric_functions_3:
11280 \> \$\textsc{Orbiter} -make_elementary_symmetric_functions 3 3
11281
11282
11283 BCH_63_4_evaluate_elementary_symmetric_functions_1:
11284 \> \$\textsc{Orbiter} -v 3 -define F -finite_field -q 64 -end \\
\end{verbatim}
define e1 -formula "e1" "e1" "" $(ELEMENTARY_SYMMETRIC_3_1) \\
define e2 -formula "e2" "e2" "" $(ELEMENTARY_SYMMETRIC_3_2) \\
define e3 -formula "e3" "e3" "" $(ELEMENTARY_SYMMETRIC_3_3) \\
define E3 -collection "e1,e2,e3" \\
with F -do -finite -field_activity \\
evaluate E3 "x0=8,x1=62,x2=15" -end

#The values of the formulae are:
#0 : 57
#1 : 0
#2 : 1

evaluate elementary symmetric functions 2:

define e1 -formula "e1" "e1" "" $(ELEMENTARY_SYMMETRIC_3_1) \\
define e2 -formula "e2" "e2" "" $(ELEMENTARY_SYMMETRIC_3_2) \\
define e3 -formula "e3" "e3" "" $(ELEMENTARY_SYMMETRIC_3_3) \\
define E3 -collection "e1,e2,e3" \\
with F -do -finite -field_activity \\
evaluate E3 "x0=33,x1=45,x2=52" -end

#The values of the formulae are:
#0 : 56
#1 : 0
#2 : 1

# poly: 1,0,2,1

BCH_63_4_evaluate_elementary_symmetric_functions_2:
define F -finite -field -q 64 -end \\
define e1 -formula "e1" "e1" "" $(ELEMENTARY_SYMMETRIC_3_1) \\
define e2 -formula "e2" "e2" "" $(ELEMENTARY_SYMMETRIC_3_2) \\
define e3 -formula "e3" "e3" "" $(ELEMENTARY_SYMMETRIC_3_3) \\
define E3 -collection "e1,e2,e3" \\
with F -do -finite -field_activity \\
evaluate E3 "x0=33,x1=45,x2=52" -end

#The values of the formulae are:
#0 : 56
#1 : 0
#2 : 1

# poly: 1,0,3,1

BCH_21_poly_mult_mod_F4:
define F -finite -field -q 4 -end \\
with F -do \\
finite_field_activity \\
polynomial_mult_mod "1,0,2,1" "1,0,3,1" \\
polynomial_mult_mod "1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1" \\
end

c(X)=X^{6} + X^{5} + X^{4} + X^{2} + 1

poly 1,0,1,0,1,1,1
BCH\_21\_poly\_division\_a:

$\text{(ORBITER)} -v 2 \$

-define F -finite_field -q 4 -end \
-with F -do \
-finite_field_activity \
-polynomial\_division \
- "1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1" \
- "1,0,2,1" \
-end

BCH\_21\_poly\_division\_b:

$\text{(ORBITER)} -v 2 \$

-define F -finite_field -q 4 -end \
-with F -do \
-finite_field_activity \
-polynomial\_division \
- "1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1" \
- "1,0,3,1" \
-end

BCH\_21\_poly\_division\_ab:

$\text{(ORBITER)} -v 2 \$

-define F -finite_field -q 4 -end \
-with F -do \
-finite_field_activity \
-polynomial\_division \
- "1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1" \
- "1,0,1,0,1,1,1" \
-end

BCH\_21\_generator\_matrix:

$\text{(ORBITER)} -v 2 \$

-define F -finite_field -q 4 -end \
-with F -do \
-coding\_theoretic\_activity \
- -generator\_matrix\_cyclic\_code \
-21 "1,0,1,0,1,1,1" \
-end

BCH\_21\_15\_weight\_enumerator:

$\text{(ORBITER)} -v 2 \$
11379 ▶ ▶ -define F -finite_field -q 4 -end \n11380 ▶ ▶ -define v -vector -field F -format 15 \n11381 ▶ ▶ ▶ -dense $(BCH\_21\_15\_GENERATOR\_MATRIX) \n11382 ▶ ▶ -end \n11383 ▶ ▶ -with F -do \n11384 ▶ ▶ -coding_theoretic_activity \n11385 ▶ ▶ ▶ -weight Enumerator v \n11386 ▶ ▶ -end \n11387
11388 # too slow!
11389
11390 BCH\_21\_15\_dual:
11391 ▶ $(ORBITER) -v 2 \n11392 ▶ ▶ -define F -finite_field -q 4 -end \n11393 ▶ ▶ -define v -vector -field F -format 15 \n11394 ▶ ▶ ▶ -dense $(BCH\_21\_15\_GENERATOR\_MATRIX) -end \n11395 ▶ ▶ -with F -do -finite_field_activity \n11396 ▶ ▶ ▶ -nullspace v \n11397 ▶ ▶ -normalize_from_the_right \n11398 ▶ ▶ -end
11399
11400
11401 BCH\_21\_6\_weight\_enumerator:
11402 ▶ $(ORBITER) -v 2 \n11403 ▶ ▶ -define F -finite_field -q 4 -end \n11404 ▶ ▶ -define v -vector -format 6 -field F \n11405 ▶ ▶ ▶ -dense $(BCH\_21\_6\_GENERATOR\_MATRIX) \n11406 ▶ ▶ -end \n11407 ▶ ▶ -with F -do \n11408 ▶ ▶ -coding_theoretic_activity -weight Enumerator v -end
11409
11410 # 1y^{21} + 63x^8y^{13} + 294x^{12}y^9 + 756x^{14}y^7 + 1890x^{16}y^5 + 1092x^{18}y^3
11411
11412 #( 1, 0, 0, 0, 0, 0, 0, 0, 63, 0, 0, 294, 0, 756, 0, 1890, 0, 1092, 0, 0, 0 )
11413
11414
11415
11416 BCH\_21\_6\_4\_macwilliams:
11417 ▶ $(ORBITER) -v 2 \n11418 ▶ ▶ -make_macwilliams_system 21 6 4
11419 ▶ pdflatex MacWilliams\_n21\_k6\_q4.tex
11420 ▶ open MacWilliams\_n21\_k6\_q4.pdf
11421
11422
11423
11424 #ww := [1, 0, 0, 84, 252, 1575, 10080, 58032, 319662, 1411116, 5133744, 15282792,
BCH_21.15.4_field_reduction:
$\text{ORBITER} -v 2$
$\text{-define}\ F \text{-finite_field } -q 4 \text{-end}$
$\text{-with}\ F \text{-do}$
$\text{-finite_field}\ \text{activity}$
$\text{-field_reduction } "BCH_21.15.4"\ 2\ 15\ 21\ $(BCH_21.15)\$
$\text{-end}$
$\text{-draw_matrix}$
$\text{-input_csv_file}\ BCH_21.15.4.csv$
$\text{-box_width}\ 20\ \text{-bit_depth}\ 24$
$\text{-partition}\ 4\ "30"\ "42$\$
\text{-end}$
\text{pdflatex field_reduction_Q4_q2.15_21.tex}$
\text{open}\ BCH_21.15.4.draw.bmp$
\text{#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:
$\text{ORBITER} -v 2$
$\text{-define}\ F \text{-finite_field } -q 2 \text{-end}$
$\text{-with}\ F \text{-do}$
$\text{-finite_field}\ \text{activity}$
$\text{-polynomial_division}$
$\text{-polynomial}\ \text{division}\ "1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1"$
$\text{-end}$

F16.roots_5:
$\text{ORBITER} -v 3$
$\text{-define}\ F \text{-finite_field } -q 2 \text{-end}$
$\text{-with}\ F \text{-do } \text{-coding_theoretic_activity}$
$\text{-nth_roots}\ 5$
$\text{-end}$
\text{pdflatex Nth_roots_q2_n5.tex}$
\text{open}\ Nth_roots_q2_n5.pdf$

F64.roots_21:
\begin{verbatim}
\$(ORBITER) -v 3 \\
\define F -finite_field -q 2 -end \\
\with F -do -coding_theoretic_activity \\
\nth Roots 21 \\
\end

df\latex Nth Roots q2 n21.tex
open Nth Roots q2 n21.pdf

BCH_F256_roots_771:
\$(ORBITER) -v 3 \\
\define F -finite_field -q 256 -end \\
\with F -do -coding_theoretic_activity \\
\nth Roots 771 \\
\end

df\latex BCH_F256_BCH_code_d16.tex
open BCH_F256_BCH_code_d16.pdf

BCH_F256_BCH_code_d16:
\$(ORBITER) -v 3 \\
\define F -finite_field -q 256 -end \\
\with F -do -coding_theoretic_activity \\
\make BCH code 771 16 \\
\end

df\latex BCH_codes_q256_n771_d16.tex
open BCH_codes_q256_n771_d16.pdf

#generator polynomial is X^{30} + 253X^{29} + 174X^{28} + 109X^{27} + 97X^{26} + 144X^{25} + 112X^{24} + 212X^{23} + 192X^{22} + 169X^{21} + 24X^{20} + 150X^{19} + 110X^{18} + 248X^{17} + 3X^{16} + 193X^{15} + 194X^{14} + 205X^{13} + 9X^{12} + 56X^{11} + 95X^{10} + 199X^{9} + 108X^{8} + 58X^{7} + 160X^{6} + 148X^{5} + 138X^{4} + 24X^{3} + 210X^{2} + 26X + 1
\end{verbatim}
11512 # ToDo:
11513
11514 F_7_BCH_code_n6:
11515 ▶ $(ORBITER) -v 3 \\
11516 ▶ ▶ -define F -finite_field -q 7 -end \\
11517 ▶ ▶ -with F -do -finite_field_activity \\
11518 ▶ ▶ ▶ -coding_theoretic_activity 7 3 \\
11519 ▶ ▶ -end
11520
11521
11522
11523 RREF_RS_6_4_7_weight_enumerator:
11524 ▶ $(ORBITER) -v 2 \\
11525 ▶ ▶ -define F -finite_field -q 7 -end \\
11526 ▶ ▶ -define v -vector -format 4 -field F \\
11527 ▶ ▶ ▶ -compact $(CODE_RS_6_4_7) \\
11528 ▶ ▶ -end \\
11529 ▶ ▶ -with F -do \\
11530 ▶ ▶ -coding_theoretic_activity \\
11531 ▶ ▶ ▶ -weight enumerator v \\
11532 ▶ ▶ -end
11533
11534
11535 #y^6 + 120x^3y^3 + 360x^4y^2 + 972x^5y + 948x^6
11536 #weight enumerator:
11537 #( 1, 0, 0, 120, 360, 972, 948 )
11538
11539
11540
11541
11542
11543 Code_RS_11:
11544 ▶ $(ORBITER) -v 2 \\
11545 ▶ ▶ -define F -finite_field -q 11 -end \\
11546 ▶ ▶ -define v -vector -format 8 -field F \\
11547 ▶ ▶ ▶ -compact $(CODE_RS_10_8_11) \\
11548 ▶ ▶ -end \\
11549 ▶ ▶ -with F -do \\
11550 ▶ ▶ -finite_field_activity -RREF v -end
11551 ▶ pdflatex RREF_example_q11_8_10.tex
11552 ▶ #gs -sDEVICE=png16 -dFIXEDMEDIA -dDEVICEWIDTHPOINTS=500 -dDEVICEHEIGHTPOINTS=450 \\
11553 ▶ ▶ -r240 -oRREF_example_q11_8_10_page%02d.png \\
11554 ▶ ▶ RREF_example_q11_8_10.pdf
11555 ▶ open RREF_example_q11_8_10.pdf
11556
11557

745
11558 Code_RS_11_weightenumerator:
11559 > $(ORBITER) -v 2 \n11560 > > -define F -finite_field -q 11 -end \n11561 > > -define v -vector -format 8 -field F \n11562 > > > -compact $(CODE_RS_11_RREF) \n11563 > > > -end \n11564 > > > -with F -do \n11565 > > > -coding_theoretic_activity \n11566 > > > > -weightenumerator v \n11567 > > > > -end
11568
11569
11570 #1*y^(10) + 1200*x^3*y^7 + 16800*x^4*y^6 + 209160*x^5*y^5 + 1734600*x^6*y^4 + 991 8000*x^7*y^3 + 37189800*x^8*y^2 + 82644700*x^9*y + 82644620*x^(10)
11571
11572
11573 RREF_RS_8_weightenumerator:
11574 > $(ORBITER) -v 2 \n11575 > > -define F -finite_field -q 8 -end \n11576 > > -define v -vector -format 5 -field F \n11577 > > > -compact $(CODE_RS_8) \n11578 > > > -end \n11579 > > > -with F -do \n11580 > > > -coding_theoretic_activity \n11581 > > > > -weightenumerator v \n11582 > > > > -end
11583
11584
11585 # the group cannot be computed
11586
11587 RS_8_field_reduction:
11588 > $(ORBITER) -v 2 \n11589 > > -define F -finite_field -q 8 -end \n11590 > > -with F -do \n11591 > > -finite_field_activity \n11592 > > -field_reduction "RS_8_red_2" \n11593 > > > 2 5 7 $(CODE_RS_8) \n11594 > > > -end
11595 > > $(ORBITER) -v 2 \n11596 > > > -draw_matrix -input_csv_file RS_8_red_2.csv \n11597 > > > -box.width 40 -bit.depth 24 \n11598 > > > -partition 4 "3,3,3,3,3" "3,3,3,3,3,3,3" -end
11599 > pdflatex field_reduction_Q8_q2.5.7.tex
11600 > open field_reduction_Q8_q2.5.7.pdf
11601
11602
11603 RREF_RS_8_reduced_weightenumerator:
11604 \>$\text{ORBITER}$ -v 2 \>
11605 \>$\text{define F -finite_field -q 2 -end}$ \>
11606 \>$\text{define v -vector -format 15 -field F}$ \>
11607 \>$\text{compact $(\text{RS}_8\text{.reduced})$}$ \>
11608 \>$\text{end}$ \>
11609 \>$\text{with F -do}$ \>
11610 \>$\text{coding_theoretic_activity}$ \>
11611 \>$\text{weight_enumerator v}$ \>
11612 \>$\text{end}$ \>
11613
11614
11615
11616 \texttt{CODE\textunderscore 21\textunderscore 15\textunderscore 4}\texttt{.store}$:
11617 \>$\text{ORBITER}$ -v 2 \>
11618 \>$\text{store_as_csv_file "code\textunderscore 21\textunderscore 15\textunderscore 4.csv"}$ \>
11619 \>$\text{15 21 (CODE\textunderscore 21\textunderscore 15\textunderscore 4)}$ \>
11620 \>$\text{draw_matrix}$ \>
11621 \>$\text{input_csv_file code\textunderscore 21\textunderscore 15\textunderscore 4.csv}$ \>
11622 \>$\text{box_width 40 -bit_depth 24}$ \>
11623 \>$\text{partition 4 "15" "21"}$ \>
11624 \>$\text{end}$ \>
11625
11626 \texttt{CODE\textunderscore 21\textunderscore 15\textunderscore 4}\texttt{.weight_enumerator}$:
11627 \>$\text{ORBITER}$ -v 2 \>
11628 \>$\text{define F -finite_field -q 2 -end}$ \>
11629 \>$\text{define v -vector -format 15 -field F}$ \>
11630 \>$\text{compact $(\text{CODE\textunderscore 21\textunderscore 15\textunderscore 4})$}$ \>
11631 \>$\text{end}$ \>
11632 \>$\text{with F -do}$ \>
11633 \>$\text{coding_theoretic_activity}$ \>
11634 \>$\text{weight_enumerator v}$ \>
11635 \>$\text{end}$ \>
11636
11637 \texttt{CODE\textunderscore 21\textunderscore 15\textunderscore 4}\texttt{.minimum_distance}$:
11638 \>$\text{ORBITER}$ -v 2 \>
11639 \>$\text{define F -finite_field -q 2 -end}$ \>
11640 \>$\text{define v -vector -format 15 -field F}$ \>
11641 \>$\text{compact $(\text{CODE\textunderscore 21\textunderscore 15\textunderscore 4})$}$ \>
11642 \>$\text{end}$ \>
11643 \>$\text{with F -do}$ \>
11644 \>$\text{coding_theoretic_activity}$ \>
11645 \>$\text{minimum_distance v}$ \>
11646 \>$\text{end}$ \>
11647
11648 \#d=4
11649
11650 Reed\textunderscore solomon\textunderscore F8\textunderscore work:
SECTION CODING THEORY BOUNDS:

bounds for d given n6 k4 q7:
> $(ORBITER) -v 2 \\n> -make bounds for d given n and k and q 6 4 7

bounds for d given n15 k6 q2:
> $(ORBITER) -v 2 \\n> -make bounds for d given n and k and q 15 6 2

# n = 15 k=6 q=2
# dGV = 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

GV_n15_k6_d5:
> $(ORBITER) -v 2 \\n> -define F -finite_field -q 2 -end \\n> -with F -do \\n> -coding_theoretic_activity \\n> -make_gilbert_varshamov_code 15 6 5 \\n> -end

# [15,6] code created
bounds_for_d_given_n12_k4_q13:

- $(ORBITER) -v 2 \n- make_bounds_for_d_given_n_and_k_and_q 12 4 13

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- with F -do \n- coding_theoretic_activity \n- weight Enumerator v \n- 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 \n- define F -finite_field -q 2 -end \n- define v -vector -format 6 -field F \n- compact $(CODE_15_6_6_A) \n- end \n- with F -do \n- coding_theoretic_activity \n- weight Enumerator v \n- end

#1y^{15} + 27x^6y^9 + 24x^8y^7 + 9x^{10}y^5 + 3x^{12}y^3

# weight Enumerator

code_n15_k6_d6_RREF:

- $(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
# weight Enumerator

#1y^{15} + 28x^6y^9 + 21x^8y^7 + 12x^{10}y^5 + 2x^{12}y^3
11745 ▶ ▶ -end \ 
11746 ▶ ▶ -with F -do -finite_field_activity \ 
11747 ▶ ▶ -RREF v -normalize_from_the_right \ 
11748 ▶ ▶ -end 
11749 ▶ pdflatex RREF_example_q2_6_15.tex 
11750 ▶ open RREF_example_q2_6_15.pdf 
11751 code_n15_k6_d6_check_RREF: 
11753 ▶ $(ORBITER) -v 2 \ 
11754 ▶ ▶ -define F -finite_field -q 2 -end \ 
11755 ▶ ▶ -define v -vector -format 9 -field F \ 
11756 ▶ ▶ ▶ -compact $(CODE_GV_N15_K6_CHECK) \ 
11757 ▶ ▶ -end \ 
11758 ▶ ▶ -with F -do -finite_field_activity \ 
11759 ▶ ▶ -RREF v -normalize_from_the_right \ 
11760 ▶ ▶ -end 
11761 ▶ pdflatex RREF_example_q2_9_15.tex 
11762 ▶ open RREF_example_q2_9_15.pdf 
11763 
11764 
11765 
11766 
11767 
11768 ##################################################################################################################
11769 # Section 10.9: Coding Theory - Classification
11770 
11771 SECTION_CODING_THEORY_CLASSIFICATION:
11772 
11773 
11774 
11775 # code classification:
11776 
11777 codes_8_4_4: 
11778 ▶ $(ORBITER) -v 6 \ 
11779 ▶ ▶ -orbiter_path $(ORBITER_PATH) \ 
11780 ▶ ▶ -define G \ 
11781 ▶ ▶ -linear_group -PGL 4 2 -end \ 
11782 ▶ ▶ -with G -do \ 
11783 ▶ ▶ -group_theoretic_activity \ 
11784 ▶ ▶ -poset_classification_control \ 
11785 ▶ ▶ ▶ -problem_label codes_8_4_4 \ 
11786 ▶ ▶ ▶ -draw_poset \ 
11787 ▶ ▶ ▶ -draw_options -embedded -radius 250 \ 
11788 ▶ ▶ ▶ -line_width 1.0 -spanning_tree -end \ 
11789 ▶ ▶ ▶ -report -end \ 
11790 ▶ ▶ -end \ 
11791 ▶ ▶ -linear_codes 3 8 \ 

750
codes_8_4_4_draw:

.codes_8_4_4_draw:

$$\text{(ORBITER)} -v 3 \ \end{code}

\text{-draw_layered_graph}

\text{codes_8_4_4_poset_lvl_8.layered_graph}

\text{-radius 250 -embedded -line_width 1.0}

\text{-y_stretch 1.0 -scale 0.5}

\text{-end}

\text{pdflatex codes_8_4_4_poset_lvl_8.draw.tex}

\text{open codes_8_4_4_poset_lvl_8.draw.pdf}

\text{-end}

\text{-linear codes 9 14}

\text{-end}

\text{pdflatex codes_14_4_9_3_poset_lvl_13.tex}

\text{open codes_14_4_9_3_poset_lvl_13.pdf}

\text{-end}

\text{-linear codes 9 14}

\text{-end}

\text{pdflatex codes_15_6_6_2.tex}

\text{open codes_15_6_6_2.pdf}

\text{-end}

\text{-draw_options}

751
# ToDo

codes_16.5.9.3:

$(ORBITER) -v 6 \
-codes_classify -n 16 -k 5 -q 3 -d 9 -w -W -lex \
-draw_poset \
-end

# 5/31/2020: 28 min 22 sec on Mac

0 : 1 orbits
1 : 1 orbits
2 : 1 orbits
3 : 1 orbits
4 : 1 orbits
5 : 1 orbits
6 : 1 orbits
7 : 1 orbits
8 : 1 orbits
9 : 2 orbits
10 : 3 orbits
11 : 4 orbits
12 : 5 orbits
13 : 5 orbits
14 : 4 orbits
15 : 3 orbits
16 : 1 orbits

#total: 36

codes_d4:

$(ORBITER) -v 3 \
-define G -linear_group -PGL 4 2 -end \
-with G -do \
-group_theoretic_activity \
-poset_classification_control -W \

codes_24_12_8:

$(ORBITER) -v 6 \\
-orbiter_path $(ORBITER_PATH) \\
-define G \\
-linear_group -PGL 12 2 -end \\
-with G -do \\
-group_theoretic_activity \\
-poset.classification_control \\
-problem_label codes_24_12_8 \\
-draw_graph \\
-draw_options -embedded -radius 250 \\
-line_width 1.0 -spanning_tree -end \\
-report -end \\
-end \\
-linear_codes 8 24 \\
-end \\
-pdflatex codes_24_12_8_poset.tex

open codes_24_12_8_poset.pdf

#codes_24_12_8_poset_lvl24.layered_graph

codes_24_12_8.draw:

$(ORBITER) -v 3 \\
-draw_layered_graph \\
codes_24_12_8_poset_lvl24.layered_graph \\
-radius 100 -spanning_tree -embedded \\
-line_width 0.5 -x_stretch 1.4 \\
-scale 0.25 -nodes_empty \\
-end \\
pdflatex codes_24_12_8_poset_lvl24_draw.tex

open codes_24_12_8_poset_lvl24_draw.pdf

glynn.arc:

$(ORBITER) -v 5 \\
-orbiter_path $(ORBITER_PATH) \\
-define G \\
-linear_group -PGGL 5 9 -end \\
-with G -do \\
-group_theoretic_activity \\
-poset.classification_control \\
-problem_label glynn.arc \\

five_points_in_general:

codes_q13_12_4:
# Chapter 11 - Combinatorics

# Section 11.1: Combinatorics

## SECTION_COMBINATORICS:

12000 Sym\_4\_conj\_classes:
12001 $(ORBITER) -v 2 -conjugacy\_classes\_Sym\_n 4

12002 Sym\_10\_conj\_classes:
12003 $(ORBITER) -v 2 -conjugacy\_classes\_Sym\_n 10

12004 open\_classes\_Sym\_10.csv

12005 Sym\_15\_conj\_classes:
12006 $(ORBITER) -v 2 -conjugacy\_classes\_Sym\_n 15

12007 Char\_Sym\_4:
12008 $(ORBITER) -v 2 -character\_table\_symmetric\_group 4

12009 Char\_Sym\_5:
12010 $(ORBITER) -v 2 -character\_table\_symmetric\_group 5

12011 Char\_Sym\_6:
12012 $(ORBITER) -v 2 -character\_table\_symmetric\_group 6

12013 all\_subsets\_10\_3:
12014 $(ORBITER) -v 2 -tree\_of\_all\_k\_subsets 10 3

12015 random\_k\_subsets:
12016 $(ORBITER) -v 4 \\n
12017 $(ORBITER) -v 4 \\n
12018 -create\_random\_k\_subsets 10 5 20
rank_k_subsets_test:

- $(ORBITER) -v 2 \

- -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

Walsh_matrix_4:

- $(ORBITER) -v 3 \ 

- -define F -finite_field -q 2 -end \ 

- -with F -do -finite_field_activity \ 

- -Walsh_matrix 4 -end \ 

- $(ORBITER) -v 2 -draw_matrix \ 

- -input_csv_file Walsh_01_4.csv \ 

- -box_width 10 -bit_depth 24 -partition 3 16 16 -end \ 

- #pdflatex GF_2.tex \ 

- #open GF_2.pdf

Dedekind_10_10:

- $(ORBITER) -v 3 -Dedekind_numbers 2 10 2 10

Dedekind_30_2:

- $(ORBITER) -v 3 -Dedekind_numbers 2 30 2 2

Dedekind_100_2:

- $(ORBITER) -v 3 -Dedekind_numbers 2 100 2 2

elementary_symmetric_functions_4:

- $(ORBITER) -make_elementary_symmetric_functions 4 4

elementary_symmetric_functions_8:

- $(ORBITER) -make_elementary_symmetric_functions 8 8

LARGE_SET_S0="0,1,2,3,4,5,6,7,8,9,10,11,12"

# identity

LARGE_SET_S1="6,8,9,2,7,10,1,11,0,3,5,4,12"
LARGE_SET_S2="2,0,1,6,3,4,11,5,7,8,10,9,12"
LARGE_SET_S3="12,5,6,11,3,7,10,8,9,1,4,2,0"
LARGE_SET_S4="5,8,10,3,11,0,2,1,12,4,6,7,9"
LARGE_SET_S5="10,11,0,7,12,2,3,1,4,5,8,6,9"
LARGE_SET_S6="3,4,1,9,5,6,8,2,7,11,12,10,0"
LARGE_SET_S7="9,11,0,6,1,3,5,4,2,12,8,7,10"
LARGE_SET_S8="10,2,12,8,0,3,4,1,5,6,9,7,11"
LARGE_SET_S9="1,3,4,10,5,6,9,7,8,11,0,12,2"
LARGE_SET_S10="7,12,1,6,0,4,5,2,3,10,9,8,11"
LARGE_SET_S11="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)"
END"
" >S.csv

Large_set_H5:
Large set $C_{13}$:

```
$\textit{(ORBITER)} -v 10 \$
$\textit{-define G -permutation_group -symmetric_group 13 \$
$\textit{-subgroup_by_generators H5 5 1 $(GENERATORS_H5) -end \$
$\textit{-with G -do \$
$\textit{-group_theoretic_activity \$
$\textit{-report \$
$\textit{-end \$
$\textit{-with G -do \$
$\textit{-group_theoretic_activity \$
$\textit{-save_elements_csv "H5_elts.csv" \$
$\textit{-end

dflatex Perm13_Subgroup_H5_5_report.tex
open Perm13_Subgroup_H5_5_report.pdf
```

```
## the following lines were created using -export_orbiter:
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
Large_set_mult_C13xS:

$\texttt{(ORBITER)} -v 10 \ \\
-\texttt{define \ G \ -permutation\_group \ -symmetric\_group \ 13 \ -end} \ \\
-\texttt{with \ G \ -do} \ \\
-\texttt{group\_theoretic\_activity} \ \\
-\texttt{multiply\_elements\_csv\_column\_major\_ordering} \ \\
-\texttt{C13\_elts.csv \ S.csv \ C13xS.csv} \ \\
-\texttt{-end}$

Large_set_mult_C13xSxH5:

$\texttt{(ORBITER)} -v 10 \ \\
-\texttt{define \ G \ -permutation\_group \ -symmetric\_group \ 13 \ -end} \ \\
-\texttt{with \ G \ -do} \ \\
-\texttt{group\_theoretic\_activity} \ \\
-\texttt{multiply\_elements\_csv\_column\_major\_ordering} \ \\
-\texttt{C13xS.csv \ H5\_elts.csv \ C13xSxH5.csv} \ \\
-\texttt{-end}$

Large_set_mult_C13xSxH5\_apply:

$\texttt{(ORBITER)} -v 10 \ \\
-\texttt{define \ G \ -permutation\_group \ -symmetric\_group \ 13 \ -end} \ \\
-\texttt{with \ G \ -do} \ \\
-\texttt{group\_theoretic\_activity} \ \\
-\texttt{apply\_elements\_csv\_to\_set} \ \\
-\texttt{C13xSxH5.csv \ C13xSxH5\_images.csv \ "0,1,2,3"} \ \\
-\texttt{-end}$

domino\_portrait:

$\texttt{(ORBITER)} -v 3 \ -domino\_portrait \ 7 \ 4 \ anton\_28x32 \ -end$

domino\_portrait\_input:

$\texttt{(ORBITER)} -v 2 \ \\
-\texttt{define \ all\_one\_r \ -vector \ -repeat \ 1 \ 28 \ -end} \ \\
-\texttt{define \ all\_one\_c \ -vector \ -repeat \ 1 \ 32 \ -end} \ \\
-\texttt{draw\_matrix} \ \\
-\texttt{grayscale} \ \\
-\texttt{invert\_colors} \ \\
-\texttt{input\_csv\_file \ anton\_28x32\_m.csv} \ \\
-\texttt{box\_width \ 20 \ -bit\_depth \ 8} \ \\
-\texttt{partition \ 3} \ \\
-\texttt{all\_one\_c \ all\_one\_r} \ \\
-\texttt{-end}$

open anton\_28x32\_m\_draw.bmp
# Section 11.2: Diophantine Systems

```
part10: $(ORBITER) -v 4 \\
   -define A -vector -dense "10,9,8,7,6,5,4,3,2,1" -end \\
   -define D -diophant \\
   -label part10 \\
   -coefficient_matrix A \\
   -RHS "10,10,1" \\
   -x_min_global 0 -x_max_global 10 \\
   -end \\
   -with D -do \\
   -diophant_activity -solve_mckay \\
   -end 

do $(ORBITER) -v 4 \\
   -define A -vector -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 
```

# Finds 42 solutions with 67 backtrack steps

# octic_monomials:

```
do $(ORBITER) -v 4 \\
   -define A -vector -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 
```

# Found 165 solutions with 210 backtrack steps

# solve_test_system:
 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 \ 

  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 \ 

#DLX_test.sol
# 1 solution in 6 backtrack steps

# Section 11.3: Combinatorial Linear Spaces
SECTION_COMBINATORIAL_LINEAR_SPACES:
12306 12307 linsp6:
12308 ▷ $(ORBITER) -v 4 \ 12309 ▷ ▷ -define A -vector -format 1 -dense "15,10,6,3,1" -end \ 12310 ▷ ▷ -define D -diophant -label linsp6 \ 12311 ▷ ▷ -coefficient_matrix A \ 12312 ▷ ▷ -RHS "15,15,1" \ 12313 ▷ ▷ -x_min_global 0 \ 12314 ▷ ▷ -x_max_global 15 \ 12315 ▷ ▷ -end \ 12316 ▷ ▷ -with D -do \ 12317 ▷ ▷ ▷ -diophant_activity -solve_mckay \ 12318 ▷ ▷ ▷ -end \ 12319 ▷
12320 # Found 15 solutions with 22 backtrack steps
12321
12322
12323
12324
12325 linsp7:
12326 ▷ $(ORBITER) -v 4 \ 12327 ▷ ▷ -define A -vector -format 1 -dense "21,15,10,6,3,1" -end \ 12328 ▷ ▷ -define D -diophant -label linsp7 \ 12329 ▷ ▷ -coefficient_matrix A \ 12330 ▷ ▷ -RHS "21,21,1" \ 12331 ▷ ▷ -x_min_global 0 \ 12332 ▷ ▷ -x_max_global 21 \ 12333 ▷ ▷ -end \ 12334 ▷ ▷ -with D -do \ 12335 ▷ ▷ ▷ -diophant_activity -solve_mckay \ 12336 ▷ ▷ ▷ -end \ 12337 ▷
12338
12339 # 32 solutions in 45 backtrack steps
12340
12341
12342
12343
12344
12345
12346
12347 linsp30_pt_types:
12348 ▷ $(ORBITER) -v 4 \ 12349 ▷ ▷ -define A -vector -format 1 -dense "6,4,3" -end \ 12350 ▷ ▷ -define D -diophant \ 12351 ▷ ▷ ▷ -label linsp30_pt_types \ 12352 ▷ ▷ ▷ -coefficient_matrix A \ 12353

762
# Section 11.4: Combinatorial Linear Spaces

### geometry builder:

```plaintext
geo

#### pasch:
```
```
12400 \x20-define Geo -geometry_builder \n12401 \x20-define Geo -geometry_builder \n12402 \x20-V 10 -B 15 -TDO 3 -fuse 1 \n12403 \x20-file_GEO petersen -girth 5 \n12404 \x20-search_tree \n12405 \x20-test Test_lines \n12406 \x20-end

12407 geo_7_3:
12408 \x20-define Test_lines -set -loop 3 8 1 -end \n12409 \x20-define Geo -geometry_builder \n12410 \x20-V 7 -B 7 -TDO 3 \n12411 \x20-fname_GEO 7_3 \n12412 \x20-test Test_lines \n12413 \x20-end

12416 geo_7_3_no_square_test:
12417 \x20-define Test_lines -set -loop 3 8 1 -end \n12418 \x20-define Geo -geometry_builder \n12419 \x20-V 7 -B 7 -TDO 3 \n12420 \x20-fname_GEO 7_3_nst \n12421 \x20-test Test_lines \n12422 \x20-no_square_test \n12423 \x20-end

12426 geo_7_3_no_square_test_draw:
12427 \x20-define Test_lines -set -loop 3 8 1 -end \n12428 \x20-draw_incidence_structure_description \n12429 \x20-width 60 -with_10 6 -end \n12430 \x20-file_of_incidence_geometries 7_3_nst.inc 7 7 21 \n12431 \x20-end \n12432 \x20-with C -do \n12433 \x20-draw_incidence_matrices \n12434 \x20-7_3_nst \n12435 \x20-end
12436 \x20-pdflatex 7_3_nst_incma.tex
12437 \x20-open 7_3_nst_incma.pdf

12440
12441
12442
12443 geo_7_3_orderly:
12444 \x20-define Test_lines -set -loop 3 8 1 -end \n12445 \x20-define Geo -geometry_builder 

764


12447 \[ \triangledown \triangledown \triangledown -V \ 7 \ -B \ 7 \ -TDO \ 3 \ \backslash \]
12448 \[ \triangledown \triangledown \triangledown -fuse \ 1 \ -fname\_GEO \ 7\_3 \ \backslash \]
12449 \[ \triangledown \triangledown \triangledown -test \ Test\_lines \ \backslash \]
12450 \[ \triangledown \triangledown \triangledown -search\_tree \ \backslash \]
12451 \[ \triangledown \triangledown \triangledown -orderly \ \backslash \]
12452 \[ \triangledown \triangledown \triangledown -end \]
12453
12454 geo\_7\_3\_orderly\_draw:
12455 \[ \$(ORBITER) -v \ 20 \ \backslash \]
12456 12457 \[ \triangledown \triangledown \triangledown -draw\_options \ -embedded \ -radius \ 50 \ \backslash \]
12458 12459 \[ \triangledown \triangledown \triangledown -xin \ 10000 \ -yin \ 10000 \ \backslash \]
12460 12461 \[ \triangledown \triangledown \triangledown -xout \ 1000000 \ -yout \ 1000000 \ \backslash \]
12462 12463 \[ \triangledown \triangledown \triangledown -nodes\_empty \ \backslash \]
12464 12465 \[ \triangledown \triangledown \triangledown -scale \ 0.5 \ -line\_width \ 0.3 \ \backslash \]
12466 12467 \[ \triangledown \triangledown \triangledown -end \ \backslash \]
12468 12469 \[ \triangledown \triangledown \triangledown -tree\_draw \ -file \ 7\_3\_tree\_txt \ -end \backslash \]
12470 12471 \[ pdflatex \ 7\_3\_tree\_draw\_txt \backslash \]
12472 12473 \[ open \ 7\_3\_tree\_draw\_pdf \backslash \]
12474
12475 geo\_7\_3\_orderly\_mem\_debug:
12476 \[ $(ORBITER) -v \ 20 \ \backslash \]
12477 12478 \[ \triangledown \triangledown \triangledown -memory\_debug \ 2 \ \backslash \]
12479 12480 \[ \triangledown \triangledown \triangledown -define \ Test\_lines \ -set \ -loop \ 3 \ 8 \ 1 \ -end \ \backslash \]
12481 12482 \[ \triangledown \triangledown \triangledown -define \ Geo \ -geometry\_builder \ \backslash \]
12483 12484 \[ \triangledown \triangledown \triangledown -V \ 7 \ -B \ 7 \ -TDO \ 3 \ \backslash \]
12485 12486 \[ \triangledown \triangledown \triangledown -fuse \ 1 \ -fname\_GEO \ 7\_3 \ \backslash \]
12487 12488 \[ \triangledown \triangledown \triangledown -test \ Test\_lines \ \backslash \]
12489 12490 \[ \triangledown \triangledown \triangledown -search\_tree \ \backslash \]
12491 12492 \[ \triangledown \triangledown \triangledown -orderly \ \backslash \]
12493 12494 geo\_8\_3:
12495 \[ $(ORBITER) -v \ 2 \ \backslash \]
12496 12497 \[ \triangledown \triangledown \triangledown -define \ Test\_lines \ -set \ -loop \ 3 \ 9 \ 1 \ -end \ \backslash \]
12498 12499 \[ \triangledown \triangledown \triangledown -define \ Geo \ -geometry\_builder \ \backslash \]
12500 12501 \[ \triangledown \triangledown \triangledown -V \ 8 \ -B \ 8 \ -TDO \ 3 \ \backslash \]
12502 12503 \[ \triangledown \triangledown \triangledown -fuse \ 1 \ -fname\_GEO \ 8\_3 \ \backslash \]
12504 12505 \[ \triangledown \triangledown \triangledown -test \ Test\_lines \ \backslash \]
12506 12507 \[ \triangledown \triangledown \triangledown -end \ \backslash \]
12508
12509 #\-print\_at\_line \ 4
12510 # \ 1 \ geo: \ 0 \ 11 \ 18 \ 29 \ 30 \ 38 \ 44 \ 54
12511 # ago=48
12512
12513
12514
12515

765
geo_9_3:
  $(ORBITER) -v 2 \n  -define Test_lines -set -loop 3 10 1 -end \n  -define Geo -geometry_builder \n  -V 9 -B 9 -TDO 3 \n  -fuse 1 -fname_GEO 9_3 \n  -test Test_lines \n  -end

geo_10_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  -test Test_lines \n  -end
  # 10 geos
  # 8/26/2021: 0 sec on Mac

geo_10_3_inc_draw:
  $(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  -draw_incidence_matrices \n  -end \n  -with C -do \n  -combinatorial_object_activity \n  -draw_incidence_matrices \n  -end
  pdflatex 10_3_inc_incma.tex
  open 10_3_inc_incma.pdf

geo_10_3_orderly:
  $(ORBITER) -v 20 \n  -define Test_lines -set -loop 4 11 1 -end \n  -define Geo -geometry_builder \n
12541 ▶ ▶ ▶ -V 10 -B 10 -TDO 3 -fuse 1 \n12542 ▶ ▶ ▶ -fname_GEO 10_3 \n12543 ▶ ▶ ▶ -test Test_lines \n12544 ▶ ▶ ▶ -orderly \n12545 ▶ ▶ -end
12546
12547 geo_10_3_orderly_mem_debug:
12548 ▶ $(ORBITER) -v 2 \n12549 ▶ ▶ -memory_debug 2 \n12550 ▶ ▶ -define Test_lines -set -loop 4 11 1 -end \n12551 ▶ ▶ -define Geo -geometry_builder \n12552 ▶ ▶ ▶ -V 10 -B 10 -TDO 3 -fuse 1 \n12553 ▶ ▶ ▶ -fname_GEO 10_3 \n12554 ▶ ▶ ▶ -test Test_lines \n12555 ▶ ▶ ▶ -orderly \n12556 ▶ ▶ -end
12557
12558
12559 geo_10_3_tree:
12560 ▶ $(ORBITER) -v 20 \n12561 ▶ ▶ -define Test_lines -set -loop 0 11 1 -end \n12562 ▶ ▶ -define GEO -geometry_builder \n12563 ▶ ▶ ▶ -V 10 -B 10 -TDO 3 -fuse 1 \n12564 ▶ ▶ ▶ -fname_GEO 10_3 \n12565 ▶ ▶ ▶ -search_tree \n12566 ▶ ▶ ▶ -test Test_lines \n12567 ▶ ▶ ▶ -end
12568 ▶ $(ORBITER) -v 20 \n12569 ▶ ▶ -draw_options -embedded -radius 20 \n12570 ▶ ▶ ▶ -paperheight 220 \n12571 ▶ ▶ ▶ -paperwidth 330 \n12572 ▶ ▶ ▶ -xin 10000 -yin 10000 \n12573 ▶ ▶ ▶ -xout 1000000 -yout 500000 \n12574 ▶ ▶ ▶ -scale 2 -line_width 0.3 \n12575 ▶ ▶ ▶ -nodes_empty \n12576 ▶ ▶ ▶ -end \n12577 ▶ ▶ -tree_draw \n12578 ▶ ▶ ▶ -file 10_3_tree.txt \n12579 ▶ ▶ ▶ -end
12580 ▶ pdflatex 10_3_tree_draw.tex
12581 ▶ open 10_3_tree_draw.pdf
12582
12583
12584
12585
12586 geo_10_3_tree_path:
12587 ▶ $(ORBITER) -v 20 \n
767
12588  ▷  ▷  -define Test_lines -set -loop 0 11 1 -end \ 
12589  ▷  ▷  -define GEO -geometry builder \ 
12590  ▷  ▷  ▷  -V 10 -B 10 -TDO 3 -fuse 1 \ 
12591  ▷  ▷  ▷  -fname GEO 10_3 \ 
12592  ▷  ▷  ▷  -search_tree \ 
12593  ▷  ▷  ▷  -test Test_lines \ 
12594  ▷  ▷  -end \ 
12595  ▷  ▷  $(ORBITER) -v 20 \ 
12596  ▷  ▷  ▷  -draw_options -embedded -radius 20 \ 
12597  ▷  ▷  ▷  ▷  -paperheight 220 \ 
12598  ▷  ▷  ▷  ▷  ▷  -paperwidth 330 \ 
12599  ▷  ▷  ▷  ▷  ▷  ▷  -xin 10000 -yin 10000 \ 
12600  ▷  ▷  ▷  ▷  ▷  ▷  ▷  -xout 1000000 -yout 500000 \ 
12601  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  -scale 2 -line_width 0.3 \ 
12602  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  -end \ 
12603  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  -tree_draw \ 
12604  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  -restrict 2 \ 
12605  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  -file 10_3.tree.txt \ 
12606  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  -select_path "0,0,15,26,46,56,72,80,93,106,119" \ 
12607  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  -end \ 
12608  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  pdflatex 10_3.tree_draw.tex \ 
12609  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  open 10_3.tree_draw.pdf \ 
12610 \ 
12611  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  -nodes_empty \ 
12612  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  #-sideways \ 
12613 \ 
12614 \ 
12615  Desargues_path_lex_least.draw: \ 
12616  ▷  ▷  echo $(DESARGUES_PATH_LEX_LEAST) >Desargues_path_lex_least.inc \ 
12617  ▷  ▷  $(ORBITER) -v 10 \ 
12618  ▷  ▷  ▷  -draw_incidence_structure_description \ 
12619  ▷  ▷  ▷  ▷  -width 60 -with 10 6 -end \ 
12620  ▷  ▷  ▷  ▷  ▷  -define C -combinatorial_objects \ 
12621  ▷  ▷  ▷  ▷  ▷  ▷  -file_of_incidence_geometries_by_row_ranks \ 
12622  ▷  ▷  ▷  ▷  ▷  ▷  ▷  -Desargues_path_lex_least.inc 10 10 3 \ 
12623  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  -end \ 
12624  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  -with C -do \ 
12625  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  -combinatorial_object_activity \ 
12626  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  -draw_incidence_matrices \ 
12627  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  Desargues_path_lex_least \ 
12628  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  -end \ 
12629  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  pdflatex Desargues_path_lex_least_incma.tex \ 
12630  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  ▷  open Desargues_path_lex_least_incma.pdf \ 
12631
12632
12633
12634
12635
12636
12637
12638
12639
12640
12641
12642
12643
12644
12645
12646
Desargues_path_can_anc_draw:
  echo "$(DESARGUES_PATH_CANONICAL_ANCESTOR) >Desargues_path_can_anc_inc"
  $(ORBITER) -v 10
  -draw.incidence.structure.description
  -width 60 -with 10 6
  -define C -combinatorial_objects
  -file_of.incidence.geometries.by.row.ranks Desargues_path_can_anc_inc 10 10
  3
  -end
  -with C -do
  -combinatorial_object.activity
  -draw.incidence.matrices
  Desargues_path_can_anc
  -end
  pdflatex Desargues_path_can_anc_incma.tex
  open Desargues_path_can_anc_incma.pdf

geo_11.3:
  $(ORBITER) -v 2
  -define Test_lines -set -loop 4 12 1 -end
  -define Geo -geometry.builder
  -V 11 -B 11 -TDO 3
  -fuse 1 -fname GEO 11 3
  -test Test_lines
  -end

# 31 geos
# 8/26/2021: 0 sec on Mac

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
  -test Test_lines
  -end

# 229 geos
# User time: 0.45 of a second, dt=45 tps = 100
# nb_calls_to_densenauty=24586
12678 geo_12_3_orderly:
12679  $(ORBITER) -v 2 \
12680  -define Test_lines -set -loop 4 13 1 -end \
12681  -define Geo -geometry_builder \
12682  -V 12 -B 12 -TDO 3 \
12683  -fuse 1 -fname_GEO 12_3 \
12684  -test Test_lines \
12685  -f_orderly \
12686  -end
12687
12688
12689
12690 geo_13_3:
12691  $(ORBITER) -v 2 \
12692  -define Test_lines -set -loop 4 14 1 -end \
12693  -define Geo -geometry_builder \
12694  -V 13 -B 13 -TDO 3 \
12695  -fuse 1 -fname_GEO 13_3 \
12696  -test Test_lines \
12697  -end
12698
12699 # 2036 geos, 96, 39, 13, 12^4, 8^3, 6^16, 4^30, 3^20, 2^190, 1^1770
12700 #User time: 0:4
12701 #nb_calls_to_densenauty=216777
12702
12703 geo_13_3_orderly:
12704  $(ORBITER) -v 2 \
12705  -define Test_lines -set -loop 4 14 1 -end \
12706  -define Geo -geometry_builder \
12707  -V 13 -B 13 -TDO 3 \
12708  -fuse 1 -fname_GEO 13_3 \
12709  -test Test_lines \
12710  -f_orderly \
12711  -end
12712
12713
12714
12715 geo_14_3:
12716  $(ORBITER) -v 2 \
12717  -define Test_lines -set -loop 4 15 1 -end \
12718  -define Geo -geometry_builder \
12719  -V 14 -B 14 -TDO 3 \
12720  -fuse 1 -fname_GEO 14_3 \
12721  -test Test_lines \
12722  -end
12723
12724 # 21399 geos, 56448, 128, 24^2, 16^3, 14^3, 12^7, 8^15, 7, 6^12, 4^91, 3^19, 2^91
6, 1^20328
12725 #User time: 0:55
12726 #nb_calls_to_densnauty=2089344
12727
12728
12729 geo_14_3_orderly:
12730 > $(ORBITER) -v 2 \n12731 > > -define Test_lines -set -loop 4 15 1 -end \n12732 > > -define Geo -geometry_builder \n12733 > > > -V 14 -B 14 -TDO 3 \n12734 > > > -fuse 1 -fname_GEO 14_3 \n12735 > > > -test Test_lines \n12736 > > > -f_orderly \n12737 > > -end
12738
12739 #User time: 0:50
12740
12741
12742 15_3.inc:
12743 > $(ORBITER) -v 2 \n12744 > > -define Test_lines -set -loop 4 16 1 -end \n12745 > > -define Geo -geometry_builder \n12746 > > > -V 15 -B 15 -TDO 3 \n12747 > > > -fuse 1 -fname_GEO 15_3 \n12748 > > > -test Test_lines \n12749 > > -end
12750
12751 # 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
12752 # 8 min 11 sec on Mac
12753 # running out of memory
12754
12755
12756 geo_15_3_g4:
12757 > $(ORBITER) -v 2 \n12758 > > -define Test_lines -set -loop 4 16 1 -end \n12759 > > -define Geo -geometry_builder \n12760 > > > -V 15 -B 15 -TDO 3 \n12761 > > > -fuse 1 -fname_GEO 15_3_g4 \n12762 > > > -girth 4 \n12763 > > > -search_tree \n12764 > > > -test Test_lines \n12765 > > -end
12766 > $(ORBITER) -v 2 \n12767 > > -draw_options -embedded -radius 50 \n12768 > > > -nodes_empty \n12769 > > > -scale 0.5 -line_width 0.3 -end \n
771
12770  ▶  ▶  -tree_draw -file 15_3_g4_tree.txt -end
12771  ▶  pdflatex 15_3_g4_tree_draw.tex
12772  ▶  open 15_3_g4_tree_draw.pdf
12773
12774  # The unique Cremona Richmond configuration with group of order 720
12775  #User time: 0 of a second, dt=0 tps = 100
12776  #nb_calls_to_densenauty=23
12777
12778  #-sideways
12779
12780
12781
12782
12783  geo.17.3_g4.orderly:
12784  ▶  $(ORBITER) -v 2 \n12785  ▶  ▶  -memory_debug 2 \n12786  ▶  ▶  ▶  -define Test_lines -set -loop 4 18 1 -end \n12787  ▶  ▶  ▶  ▶  -define Geo -geometry_builder \n12788  ▶  ▶  ▶  ▶  ▶  -V 17 -B 17 -TDO 3 \n12789  ▶  ▶  ▶  ▶  ▶  ▶  -fuse 1 -fname_GEO 17.3_g4 \n12790  ▶  ▶  ▶  ▶  ▶  ▶  -girth 4 \n12791  ▶  ▶  ▶  ▶  ▶  ▶  ▶  -test Test_lines \n12792  ▶  ▶  ▶  ▶  ▶  ▶  ▶  -orderly \n12793  ▶  ▶  ▶  ▶  ▶  ▶  ▶  ▶  -end
12794
12795  # 1 sol
12796
12797  geo.18.3_g4:
12798  ▶  $(ORBITER) -v 2 \n12799  ▶  ▶  -define Test_lines -set -loop 4 19 1 -end \n12800  ▶  ▶  ▶  -define Geo -geometry_builder \n12801  ▶  ▶  ▶  ▶  -V 18 -B 18 -TDO 3 \n12802  ▶  ▶  ▶  ▶  ▶  -fuse 1 -fname_GEO 18.3_g4 \n12803  ▶  ▶  ▶  ▶  ▶  ▶  -girth 4 \n12804  ▶  ▶  ▶  ▶  ▶  ▶  ▶  -search_tree \n12805  ▶  ▶  ▶  ▶  ▶  ▶  ▶  ▶  -test Test_lines \n12806  ▶  ▶  ▶  ▶  ▶  ▶  ▶  ▶  ▶  -end
12807
12808  # 4 sol, 1 sec
12809
12810
12811  geo.19.3_g4:
12812  ▶  $(ORBITER) -v 2 \n12813  ▶  ▶  -define Test_lines -set -loop 4 20 1 -end \n12814  ▶  ▶  ▶  -define Geo -geometry_builder \n12815  ▶  ▶  ▶  ▶  -V 19 -B 19 -TDO 3 \n12816  ▶  ▶  ▶  ▶  ▶  -fuse 1 -fname_GEO 19.3_g4 \n
772
```
12817 ▷ ▷ ▷ -girth 4 \\
12818 ▷ ▷ ▷ -test Test_lines \\
12819 ▷ ▷ -end \\
12820 \\
12821 # 14 sol, 10 sec on Mac \\
12822 \\
12823 geo_20_3_g4: \\
12824 ▷ $(ORBITER) -v 2 \\
12825 ▷ ▷ -define Test_lines -set -loop 4 21 1 -end \\
12826 ▷ ▷ -define Geo -geometry_builder \\
12827 ▷ ▷ ▷ -V 20 -B 20 -TDO 3 \\
12828 ▷ ▷ ▷ -fuse 1 -fname_GEO 20_3_g4 \\
12829 ▷ ▷ ▷ -girth 4 \\
12830 ▷ ▷ ▷ -test Test_lines \\
12831 ▷ ▷ -end \\
12832 \\
12833 # 162 sol, User time: 2:5 on Mac \\
12834 \\
12835 geo_21_3_g4: \\
12836 ▷ $(ORBITER) -v 2 \\
12837 ▷ ▷ -define Test_lines -set -loop 4 22 1 -end \\
12838 ▷ ▷ -define Geo -geometry_builder \\
12839 ▷ ▷ ▷ -V 21 -B 21 -TDO 3 \\
12840 ▷ ▷ ▷ -fuse 1 -fname_GEO 21_3_g4 \\
12841 ▷ ▷ ▷ -girth 4 \\
12842 ▷ ▷ ▷ -test Test_lines \\
12843 ▷ ▷ -end \\
12844 \\
12845 \\
12846 geo_15_4: \\
12847 ▷ $(ORBITER) -v 2 \\
12848 ▷ ▷ -define Test_lines -set -loop 4 16 1 -end \\
12849 ▷ ▷ -define Geo -geometry_builder \\
12850 ▷ ▷ ▷ -V 15 -B 15 -TDO 4 \\
12851 ▷ ▷ ▷ -fuse 1 -fname_GEO 15_4 \\
12852 ▷ ▷ ▷ -search_tree \\
12853 ▷ ▷ ▷ -test Test_lines \\
12854 ▷ ▷ -end \\
12855 ▷ ▷ $(ORBITER) -v 2 \\
12856 ▷ ▷ ▷ -draw_options -embedded -radius 50 \\
12857 ▷ ▷ ▷ -nodes_empty \\
12858 ▷ ▷ ▷ -scale 0.5 -line_width 0.3 -end \\
12859 ▷ ▷ ▷ -tree_draw -file 15_4_tree.txt -end \\
12860 ▷ ▷ pdflatex 15_4_tree_draw.tex \\
12861 ▷ ▷ open 15_4_tree_draw.pdf \\
12862 \\
12863 # 4 objects
```
12864 # ago=360, 30, 24, 15
12865 #User time: 0.15 of a second, dt=15 tps = 100
12866 #nb_calls_to_densenauty=6767
12867
12868
12869
12870 geo_16_4_g4:
    $(ORBITER) -v 2 \
12871   -define Test_lines -set -loop 4 17 1 -end \
12872   -define Geo -geometry_builder \
12873   -V 16 -B 16 -TDO 4 \ 
12874   -fuse 1 -fname GEO 16_4_g4 \ 
12875   -girth 4 \ 
12876   -test Test_lines \ 
12877   -end
12878
12879
12880 # none
12881
12882
12883 40_4_g4.inc:
    $(ORBITER) -v 2 \
12884   -define Test_lines -set -loop 0 41 1 -end \
12885   -define Geo -geometry_builder \
12886   -V 40 -B 40 -TDO 4 \ 
12887   -fuse 1 -fname GEO 40_4_g4 \ 
12888   -girth 4 \ 
12889   -search_tree \ 
12890   -test Test_lines \ 
12891   -end
12892
12893 $(ORBITER) -v 2 \
12894   -draw_options -embedded -radius 50 \ 
12895   -xin 10000 -yin 10000 \ 
12896   -xout 1000000 -yout 1000000 \ 
12897   -nodes_empty \ 
12898   -scale 0.5 -line_width 0.3 -end \ 
12899   -tree_draw -file 40_4_g4_tree.txt -end
12900   pdflatex 40_4_g4_tree_draw.tex
12901   open 40_4_g4_tree_draw.pdf
12902
12903
12904 # 2 geos, ago=51840^2
12905 #User time: 0.18 of a second, dt=18 tps = 100
12906 #nb_calls_to_densenauty=1065
12907
12908
12909 geo_63_3_g6:
    $(ORBITER) -v 2 \
12910
define Test_lines -set -loop 4 64 1 -end \
define Geo -geometry_builder \
-define V 63 -B 63 -TDO 3 \n-fuse 1 -fname GEO 63 3g6 \ngirth 6 \ntest Test_lines \nend

define Test_lines -set -loop 1 19 1 -end \
define Geo -geometry_builder \
-V 6,6,6 -B 1,1,36 -TDO "1,0,0,6, 0,1,0,6, 0,0,1,6" \nfuse 3 -fname GEO LSQ6 \ntest Test_lines \nend

define Test_lines -set -loop 3 17 1 -end \
define Geo -geometry_builder \
-V 16 -B 20 -TDO 5 \nfuse 1 -fname GEO geo 16 \ntest Test_lines \nend

# Section 11.5: Design Theory

SECTION DESIGN THEORY:

define D -design -q 3 -family PG 2 -end 
-with D -do 
-design_activity \n-export_inc \n-end

# writes PG_2_3_inc.txt
design PG_2.4:
  $(ORBITER) -v 8 \
  -define D -design -q 4 -family PG_2.q -end \n  -with D -do \n  -design_activity \n  -export_inc \n  -end

design PG_2.3_table_create:
  $(ORBITER) -v 2 \
  -define D -design -q 3 -family PG_2.q -end \n  -define Sym13 -permutation_group -symmetric_group 13 -end \n  -define T -design_table D "PG_2.13" Sym13

# written file PG_2.13_design_table.csv
# 1108800 designs
#User time: 7:30

design PG_2.3_group_5:
  $(ORBITER) -v 2 \
  -define D -design -q 3 -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  -normalizer_on_orbits_of_a_given_length 5 \n  -end

#H5_N_orbit_reps.csv
# 678 orbits
#User time: 2:39

design PG_2.3_group_5_sol_0:
  $(ORBITER) -v 2 \
  -define D -design -q 3 -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
#User time: 776
wreath_product_designs_n4_k2_inc.txt:

wreath_product_designs_n8_k6_inc.txt:

# wreath_product_designs_n8_k6_inc.txt
# The design with have 16 points and 3920 blocks of size 6.

KM_cyclic_7:

# wreath_product_designs_n8_k6_inc.txt

13005 ▶ ▶ ▶ ▶ -prefix "H5" \n13006 ▶ ▶ ▶ ▶ -selected_orbit_length 5 \n13007 ▶ ▶ ▶ ▶ -end \n13008 ▶ ▶ -with LSW -do \n13009 ▶ ▶ ▶ ▶ -large_set_with_symmetry_assumption_activity \n13010 ▶ ▶ ▶ ▶ -read_solution_file 5 case_0_sol.txt \n13011 ▶ ▶ ▶ ▶ -end

13013 wreath_product_designs_n4_k2_inc.txt:

13014 ▶ $(ORBITER) -v 8 \n13015 ▶ ▶ -define D -design -wreath_product_designs 4 2 -end \n13016 ▶ ▶ -with D -do \n13017 ▶ ▶ ▶ ▶ -design_activity \n13018 ▶ ▶ ▶ ▶ ▶ -export_inc \n13019 ▶ ▶ ▶ ▶ ▶ -end

13020
13021
13022
13023 wreath_product_designs_n8_k6_inc.txt:

13024 ▶ $(ORBITER) -v 8 \n13025 ▶ ▶ -define D -design -wreath_product_designs 8 6 -end \n13026 ▶ ▶ -with D -do \n13027 ▶ ▶ ▶ ▶ -design_activity \n13028 ▶ ▶ ▶ ▶ ▶ -export_inc \n13029 ▶ ▶ ▶ ▶ ▶ -end

13030
13031
13032 # wreath_product_designs_n8_k6_inc.txt
13033 # The design with have 16 points and 3920 blocks of size 6.
13034
13035
13036 KM_cyclic_7:

13037 ▶ $(ORBITER) -v 3 \n13038 ▶ ▶ -define gens -vector -dense "1,2,3,4,5,6,0" -end \n13039 ▶ ▶ -define G -permutation_group -symmetric_group 7 \n13040 ▶ ▶ ▶ ▶ -subgroup_by_generators "C7" 7 1 gens \n13041 ▶ ▶ ▶ ▶ -end \n13042 ▶ ▶ -with G -do \n13043 ▶ ▶ ▶ -group_theoretic_activity \n13044 ▶ ▶ ▶ ▶ -poset_classification_control \n13045 ▶ ▶ ▶ ▶ ▶ -problem.label C7 -W -depth 3 \n13046 ▶ ▶ ▶ ▶ ▶ ▶ -Kramer_Mesner_matrix 2 3 \n13047 ▶ ▶ ▶ ▶ ▶ ▶ ▶ -draw_poset \n13048 ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ -draw_options -embedded -sideways -radius 50 \n13049 ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ -scale 0.5 -line_width 0.3 -end \n13050 ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ -end \n13051 ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ -orbits_on_subsets 3 \n
777
13052 \> \> -end
13053 \> \> $(ORBITER) -v 4 \$
13054 \> \> -define A -vector -file C7_KM_2_3.csv -end \$
13055 \> \> -define D -diophant \$
13056 \> \> -label "C7_KM_2_3_system" \$
13057 \> \> -coefficient_matrix A \$
13058 \> \> -RHS_constant "1,1,1" \$
13059 \> \> -x_min_global 0 -x_max_global 1 \$
13060 \> \> -end \$
13061 \> \> -with D -do \$
13062 \> \> \> -diophant_activity -solve_mckay \$
13063 \> \> \> -end
13064
13065
13066
13067 \# to create simple 7-designs on 33 points with block size 8 and lambda = 10 invariant under PGGL(2,32):
13068
13069 KM_PGGL_2_32:
13070 \> $(ORBITER) -v 3 \$
13071 \> \> -define G -linear_group -PGGL 2 32 -end \$
13072 \> \> -with G -do \$
13073 \> \> -group_theoretic_activity \$
13074 \> \> \> -poset_classification_control \$
13075 \> \> \> \> -problem_label KM_PGGL_2_32 -W -depth 8 \$
13076 \> \> \> \> -Kramer_Mesner_matrix 7 8 \$
13077 \> \> \> \> -draw_poset \$
13078 \> \> \> \> \> -draw_options -embedded -sideways -radius 50 \$
13079 \> \> \> \> \> \> -scale 0.5 -line_width 0.3 -end \$
13080 \> \> \> \> -end \$
13081 \> \> \> -orbits_on_subsets 8 \$
13082 \> \> -end
13083 \> $(ORBITER) -v 2 -draw_matrix \$
13084 \> \> -input_csv_file KM_PGGL_2_32_KM_7_8.csv \$
13085 \> \> -box_width 20 -bit_depth 24 \$
13086 \> \> -partition 3 32 97 -end
13087 \> pdflatex KM_PGGL_2_32_poset_lvl_8.tex
13088 \> open KM_PGGL_2_32_poset_lvl_8.pdf
13089 \> open KM_PGGL_2_32_KM_7_8_draw.bmp
13090 \> $(ORBITER) -v 4 \$
13091 \> \> -define A -vector -file KM_PGGL_2_32_KM_7_8.csv -end \$
13092 \> \> -define D -diophant \$
13093 \> \> -label "KM_PGGL_2_32_KM_7_8_system" \$
13094 \> \> -coefficient_matrix A \$
13095 \> \> -RHS_constant "10,10,1" \$
13096 \> \> -x_min_global 0 -x_max_global 1 \$
13097 \> \> -end \$

778
with D -do \  
diophant_activity -solve_mckay \  
end  
end  
end  
end  

## SECTION DESIGN THEORY LARGE SETS:

# Section 11.6: Design Theory - Large Sets

### KM_PSL_3.5:

```bash
$ORBITER -v 3 
define G -linear_group -PSL 3 5 -end 
with G -do 
group_theoretic_activity 
poset_classification_control 
-problem.label KM_PSL_3.5 -W -depth 10 
-Kramer_Mesner_matrix 8 10 
-draw_poset 
draw_options -embedded -sideways 
-radius 50 -scale 0.5 -line_width 0.3 -end 
end 
orbits_on_subsets 10 
end 
$ORBITER -v 2 -draw_matrix 
in_csv_file KM_PSL_3.5_KM_8.10.csv 
-box_width 10 -bit_depth 8 -partition 3 42 174 -end
$ORBITER -v 4 
define A -vector -file KM_PSL_3.5_KM_8.10.csv -end 
define D -diophant 
-label "KM_PSL_3.5_KM_8.10_system" 
-coefficient_matrix A 
-RHS_constant "93,93,1" 
-x_min_global 0 -x_max_global 1 
end 
with D -do 
diophant_activity -solve_mckay 
end
```
13145 AG_2_3.inc:
13146 $(ORBITER) -v 2 \
13147 -define Geo -geometry_builder 
13148 -V 9 -B 12 \
13149 -TDO 4 -fuse 1 \
13150 -fname GEO AG_2_3 \
13151 -test 3,4,5,6,7,8,9 \
13152 -end
13153
13154 #9 12 3
13155 #0 13 22 27 35 41 47 53 55 59 71 76
13156 # -1 1
13157 #432
13158
13159
13160
13161
13162
13163
13164 LS_AG_2_3_design_table_create:
13165 $(ORBITER) -v 20 \
13166 -define D -design -list_of_blocks 
13167 9 3 $(AG_2_3_BLOCKS) -end \
13168 -define Sym9 -permutation_group -symmetric_group 9 -end \
13169 -define T -design_table D "AG_2_3" Sym9
13170
13171 # creates AG_2_3_design_table.csv
13172 # and AG_2_3.makefile
13173
13174 #0,0,13,22,27,35,41,47,53,55,59,71,76
13175 # is the first design in AG_2_3_design_table.csv
13176
13177 #poset_orbit_node::init_root_node storing strong generators for a group of order 362880
13178 # stabilizer order 432
13179 # 840 designs
13180
13181 #User time: 0.13 of a second, dt=13 tps = 100
13182
13183
13184 AG_2_3.on.designs:
13185 $(ORBITER) -v 2 \
13186 -define gens -vector -file AG_2_3_gens.csv -end \
13187 -define G -permutation_group \
13188 -bsgs AG_2_3 "AG_2_3" 840 362880 "0,1,2,3,4,5,6,7" 8 gens -end \
13189 -with G -do \
13190 -group_theoretic_activity \

780
orbits on points

stabilizer of orbit rep 0

-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

-define gens -vector -file AG_2_3_stab_orb_0_gens.csv -end

-define G -permutation_group

-bsgs AG_2_3_stab_orb_0 "AG_2_3_stab_orb_0"

-define G -permutation_group

-bsgs AG_2_3_stab_orb_0 "AG_2_3_stab_orb_0"


-define Gr -modified_group -from G

-restricted_action $(LARGE_SET_AG_2_3_NEIGHBOR_SET)

-end

-with G -do

-group_theoretic_activity

-export_orbiter

-end

AG_2_3_stab_orb_0_Perm840_res192:

$(ORBITER) -v 2

-define gens -vector -file Perm840_res192_gens.csv -end

-define G -permutation_group

-bsgs Perm840_res192 "Perm840 {\rm res192}"

-define G -permutation_group

-bsgs Perm840_res192 "Perm840 {\rm res192}"

-define G -permutation_group

-bsgs Perm840_res192 "Perm840 {\rm res192}"

192 432 "0,1,2,3,4,5,6,7,8" 4 gens

-end

-with G -do

-group_theoretic_activity

-report

-end

pdfflatex Perm840_res192_report.tex

open Perm840_res192_report.pdf

LS_AG_2_3_disjoint_sets_graph_and_cliques:

$(ORBITER) -v 2

-define Gamma -graph

-disjoint_sets_graph

-AG_2_3_design_table.csv
13238  ▶  ▶  -end \n13239  ▶  ▶  -with Gamma -do \n13240  ▶  ▶  -graph_theoretic_activity \n13241  ▶  ▶  ▶  -save \n13242  ▶  ▶  -end \n13243  ▶  ▶  -with Gamma -do \n13244  ▶  ▶  -graph_theoretic_activity \n13245  ▶  ▶  ▶  -find_cliques -target_size 7 -end \n13246  ▶  ▶  -end \n13247  ▶  ▶  -print_symbols
13248
13249
13250  #AG_2_3_design_table_disjoint_sets.colored_graph
13251  #User time: 0.66 of a second, dt=66 tps = 100
13252  #AG_2_3_design_table_disjoint_sets.sol.txt
13253  #AG_2_3_design_table_disjoint_sets.sol.csv
13254
13255  #15360 solutions
13256
13257  LS_AG_2_3_disjoint_sets_split:
13258  ▶  $(ORBITER) -v 4 \n13259  ▶  ▶  -define Gamma -graph -load \n13260  ▶  ▶  ▶  AG_2_3_design_table_disjoint_sets.colored_graph \n13261  ▶  ▶  -end \n13262  ▶  ▶  -with Gamma -do \n13263  ▶  ▶  -graph_theoretic_activity \n13264  ▶  ▶  ▶  -split_by_clique "0" "0" \n13265  ▶  ▶  -end
13266
13267
13268  #AG_2_3_design_table_disjoint_sets_0.graph
13269  #AG_2_3_design_table_disjoint_sets_0_subset.txt
13270
13271
13272
13273  LS_AG_2_3_export_solutions:
13274  ▶  $(ORBITER) -v 20 \n13275  ▶  ▶  -define D -design -list_of_blocks 9 3 \n13276  ▶  ▶  ▶  $(AG_2_3_BLOCKS) -end \n13277  ▶  ▶  -define Sym9 -permutation_group -symmetric_group 9 -end \n13278  ▶  ▶  -define T -design_table D "AG_2_3" Sym9 \n13279  ▶  ▶  -with D -do \n13280  ▶  ▶  -design_activity \n13281  ▶  ▶  ▶  -extract_solutions_by_index "AG_2_3" Sym9 \n13282  ▶  ▶  ▶  ▶  AG_2_3_design_table_disjoint_sets_sol.csv \n13283  ▶  ▶  ▶  ▶  ▶  solutions.csv \n13284  ▶  ▶  ▶  ▶  ▶  "" \n
782
Section 11.7: Design Theory - Delandtsheer Doyen

DD_PP4:
 $(ORBITER) -v 6 \n- Delandtsheer_Doyen $(PP4) $(PP4_GROUP1) $(PP4_MASK1) \n- end \n
DD_PP4_system:
 $(ORBITER) -v 4 \n-define D -diophant -label PP4 \n-problem_of_Steiner_type 10 PP4_pair_covering.csv \n-has_sum 1 \n-end \n-with D -do \n-diophant_activity -solve_mckay \n-end

DD_CC:
 $(ORBITER) -v 6 \n-Delandtsheer_Doyen -search_wrt_subgroup \n-Delandtsheer_Doyen_PROBLEM_COLBOURN_COLBOURN_7_13 \n-Delandtsheer_Doyen_PROBLEM_COLBOURN_COLBOURN_7_13_GROUP1 \n-Delandtsheer_Doyen_PROBLEM_COLBOURN_COLBOURN_7_13_MASK1 \n-end

#target level: 6
#k2: 15
#number of k-orbits at target level: 1774964

# creates DD_CC_7_13_pair_covering.csv
$$\text{ORBITER} -v 4 \ \ -\text{define D} \ -\text{diophant} \ -\text{label DD}\_CC\_7\_13 \ \ -\text{problem of Steiner type 45} \ -\text{DD}\_CC\_7\_13\_pair\_covering.csv \ -\text{has sum 3} \ \ -\text{end} \ \ -\text{with D} \ -\text{do} \ \ -\text{diophant_activity} \ -\text{solve_mckay} \ \ -\text{end} \ $$

# no solution

# 18603 = 27 * 53 * 13

$$(\text{ORBITER}) -v 4 \ \ -\text{Delandtsheer Doyen} \ \ -\text{DELANDTSHEER DOYEN PROBLEM 27 53} \ \ -\text{DELANDTSHEER DOYEN PROBLEM 27 53 GROUP1} \ \ -\text{DELANDTSHEER DOYEN PROBLEM 27 53 MASK1} \ \ -\text{singletons} \ \ -\text{end} \ $$

$$(\text{ORBITER}) -v 4 \ \ -\text{Delandtsheer Doyen} \ \ -\text{DELANDTSHEER DOYEN PROBLEM 3 7} \ \ -\text{DELANDTSHEER DOYEN PROBLEM 3 7 GROUP1} \ \ -\text{DELANDTSHEER DOYEN PROBLEM 3 7 MASK1} \ \ -\text{end} \ $$

PG_27_special:
$(ORBITER) -v 2 \$
> -define F -finite_field -q 27 -override_polynomial 46 -end \$
> -define P -projective_space -n 2 -field F -v 0 -end \$
> -with P -do -projective_space_activity \$
> -cheat_sheet \$
> -end
pdflatex PG_2.27.tex
open PG_2.27.pdf

# Section 11.8: Tactical Decompositions

max_arc_16_4_start:
$(ORBITER) -v 4 -maximal_arc_parameters 16 4

max_arc_16_4_convert_stack_tdo:
$(ORBITER) -v 4 -convert_stack_to_tdo max_arc_q16_r4.stack

max_arc_16_4_refine:
$(ORBITER) -v 4 -tdo_refinement \$
-convert_stack_to_tdo max_arc_q16_r4.tdo -dual_is_linear_space -end

max_arc_16_4r_print:
$(ORBITER) -v 4 -tdo_print max_arc_q16_r4r.tdo

# Chapter 12 - Finite Geometry

# Section 12.1: Spreads
SECTION_SPREADS:

create_spread_9a:
$\text{(ORBITER)} -v 3 \ 
\text{-define } F \text{-finite_field } -q 3 -\text{end } \ 
\text{-define } G \text{-linear_group } -\text{PGL 4 } F -\text{end } \ 
\text{-define } S \text{-spread } -\text{kernel_field } F \ 
\text{-group } G \text{-k 2 -catalogue 0 } \ 
\text{-end} \ 

# desarguesian spread, ago = 5760

create_spread_9b:
$\text{(ORBITER)} -v 3 \ 
\text{-define } F \text{-finite_field } -q 3 -\text{end } \ 
\text{-define } G \text{-linear_group } -\text{PGL 4 } F -\text{end } \ 
\text{-define } S \text{-spread } -\text{kernel_field } F \ 
\text{-group } G \text{-k 2 -catalogue 1 } \ 
\text{-end} \ 

# Hall spread, ago = 1920

create_spread_25_7:
$\text{(ORBITER)} -v 3 \ 
\text{-define } F \text{-finite_field } -q 5 -\text{end } \ 
\text{-define } G \text{-linear_group } -\text{PGL 4 } F -\text{end } \ 
\text{-define } S \text{-spread } -\text{kernel_field } F \ 
\text{-group } G \text{-k 2 -catalogue 7 } \ 
\text{-end} \ 

SPREAD_SET_27_RAO_RAO="\ 
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, \ 
1,1,0,2,1,1,0,0,2, \ 
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, \ 
2,1,2,0,2,0,2,0,1, \ 
786
create_spread_Rao_Rao_27:
$\text{(ORBITER) -v 3} \$
\begin{verbatim}
\text{-define F -finite_field -q 3 -end} \ \\
\text{-define SS -vector -dense $(SPREAD_SET_27_RA0_RA0) -end} \ \\
\text{-define G -linear_group -PGL 6 F -end} \ \\
\text{-define S -spread -kernel_field F} \ \\
\text{-group G -k 3 -spread_set SS} \ \\
\text{-end}
\end{verbatim}

desarguesian_spread_in_PG_3_2:
$\text{(ORBITER) -v 3} \$
\begin{verbatim}
\text{-define FQ -finite_field -q 4 -end} \ \\
\text{-define Fq -finite_field -q 2 -end} \ \\
\text{-with FQ -and Fq -do -finite_field_activity} \ \\
\text{-cheat_sheet_desarguesian_spread 2 -end}
\end{verbatim}
\text{pdflatex Desarguesian_Spread_3_2.tex}
\text{open Desarguesian_Spread_3_2.pdf}

desarguesian_spread_in_PG_5_2:
$\text{(ORBITER) -v 3} \$
\begin{verbatim}
\text{-define FQ -finite_field -q 8 -end} \ \\
\text{-define Fq -finite_field -q 2 -end} \ \\
\text{-with FQ -and Fq -do -finite_field_activity} \ \\
\text{-cheat_sheet_desarguesian_spread 2 -end}
\end{verbatim}
\text{pdflatex Desarguesian_Spread_5_2.tex}
\text{open Desarguesian_Spread_5_2.pdf}
desarguesian spread in PG 3:

```
$\text{(ORBITER)} -v 3 \ \
\text{-define FQ -finite_field -q 16 -end} \ 
\text{-define Fq -finite_field -q 4 -end} \ 
\text{-with FQ -and Fq -do -finite_field_activity} \ 
\text{-cheat_sheet_desarguesian_spread 2 -end}
```

```
pdflatex Desarguesian_Spread_3_4.tex
open Desarguesian_Spread_3_4.pdf
```

desarguesian spread in PG 3:

```
$\text{(ORBITER)} -v 3 \ 
\text{-define FQ -finite_field -q 25 -end} \ 
\text{-define Fq -finite_field -q 5 -end} \ 
\text{-with FQ -and Fq -do -finite_field_activity} \ 
\text{-cheat_sheet_desarguesian_spread 2 -end}
```

```
pdflatex Desarguesian_Spread_3_5.tex
open Desarguesian_Spread_3_5.pdf
```

classify spreads 4:

```
$\text{(ORBITER)} -v 10 \ 
\text{-define F -finite_field -q 2 -end} \ 
\text{-define P -projective_space -n 3 -field F -v 0 -end} \ 
\text{-define C -spread_classifier} \ 
\text{-projective_space P} \ 
\text{-k 2} \ 
\text{-starter_size 5} \ 
\text{-poset_classification_control} \ 
\text{-draw_options} \ 
\text{-embedded} \ 
\text{-end} \ 
\text{-draw_poset} \ 
\text{-problem_label spreads_2_2} \ 
\text{-end} \ 
\text{-output_prefix "."} \ 
\text{-end} \ 
\text{-with C -do -spread_classifier_activity} \ 
\text{-compute_starter} \ 
\text{-problem_label spreads_2_2} \ 
\text{-W -depth 5} \ 
\text{-report -end} \ 
\text{-end} \ 
\text{-end}
```

```
pdflatex spreads_2_2_poset_lvl_5.tex
open spreads_2_2_poset_lvl_5.pdf
```

788
13567 classify_spreads_16_4:
13568 \$\text{(ORBITER)} \ -v \ 4 \ \$
13569 \text{define F } \ -\text{finite_field } -q \ 4 \ -\text{end} \$
13570 \text{define P } \ -\text{projective_space } -n \ 3 \ -\text{field F } -v \ 0 \ -\text{end} \$
13571 \text{define C } \ -\text{spread_classifier} \$
13572 \text{define P } \ -\text{projective_space P} \$
13573 \ \text{k 2} \$
13574 \text{starter_size 17} \$
13575 \text{poset_classification_control} \$
13576 \text{draw_options} \$
13577 \text{radius 20} \$
13578 \text{nodes_empty} \$
13579 \text{line_width 0.2} \$
13580 \text{embedded} \$
13581 \text{end} \$
13582 \text{draw_poset} \$
13583 \text{problem_label spreads_16_4} \$
13584 \text{end} \$
13585 \text{output_prefix "."} \$
13586 \text{end} \$
13587 \text{with C} \ -\text{do} \ -\text{spread_classify_activity} \$
13588 \text{compute_starter} \$
13589 \text{problem_label spreads_16_4} \$
13590 \text{W} \ -\text{depth 17} \$
13591 \text{report} \ -\text{end} \$
13592 \text{end} \$
13593 \text{end} \$
13594 \text{pdflatex spreads_16_4.poset_lvl_17.tex} \$
13595 \text{open spreads_16_4.poset_lvl_17.pdf} \$
13596 \$
13597 \$
13598 \$
13599 \$
13600 classify_spreads_25_starter_lift_case_0:
13601 \$\text{(ORBITER)} \ -v \ 3 \ \$
13602 \text{define F } \ -\text{finite_field } -q \ 5 \ -\text{end} \$
13603 \text{define P } \ -\text{projective_space } -n \ 3 \ -\text{field F } -v \ 0 \ -\text{end} \$
13604 \text{define C } \ -\text{spread_classifier} \$
13605 \text{define P } \ -\text{projective_space P} \$
13606 \text{k 2} \$
13607 \text{starter_size 5} \$
13608 \text{recoordinatize} \$
13609 \text{poset_classification_control} \$
13610 \text{draw_options} \$
13611 \text{radius 20} \$
13612 \text{nodes_empty} \$
13613 \text{end} \$

789
#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

spreads_25_starter_0_cliques:

graph_theoretic_activity::perform_activity Gr->label=spreads_25_graph_0 nb_sol = 7680

classify_spreads_25_starter_lift_all_cases:

classify_spreads_25_starter_lift_all_cases:

classify_spreads_25_starter_lift_all_cases:

classify_spreads_25_starter_lift_all_cases:
-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 

-draw_poset 
-problem_label spreads_25 
-end 
-output_prefix "" 

-with C -do -spread_classify_activity 
-compute_starter 
-problem_label spreads_25 
-W -depth 5 
-report -end 
-end 

-prepare_lifting_all_cases 

-loop L 0 29 1 

-define G -graph -load spreads_25_graph_%L.bin -end 
-with G -do 
-graph_theoretic_activity 
-find_cliques -rainbow -target_size 21 -end 
-end 

-end_loop

spreads_25_starter_cliques:

-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 
-starter_size 5 

classify_spreads_25_isomorph:
-recoordinatize \
-poset_classification_control \
-draw_options \
-radius 20 \n-nodes_empty \
-line_width 0.2 \n-embedded \n-end \n-draw_poset \n-problem_label_spreads_25 \n-end \n-output_prefix "" 
-end 
-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-isomorph \
."/" \
."/" \
-use_database_for_starter \n-build_db \n-solution_prefix "" 
-base_fname "" 
-end \n-end \n-with C -do -spread_classify_activity \n-isomorph \
."/" \
."/" \
-use_database_for_starter \n-read_solutions \n-solution_prefix "" 
-base_fname "spreads_25_graph" \n-end \n-end \n-with C -do -spread_classify_activity \n-isomorph \
."/" \
."/" \
-use_database_for_starter \n-compute_orbits \

-solution_prefix "" \
-base_f_name "spreads_25_graph" \
-end \ 
-with C -do -spread_classify_activity \ 
isomorph \ 
"./" \ 
isomorph_testing \ 
"./" \ 
-use_database_for_starter \ 
-solution_prefix "" \
-base_f_name "spreads_25_graph" \
-end \ 
-end 

#We found 21 isomorphism types
1:33

dont:

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 \ 
-draw_poset \ 
-problem_label spreads_27_3 \ 
-end \ 
-output_prefix "." \ 
-end \ 
-with C -do -spread_classify_activity \ 
-compute_starter \ 
-problem_label spreads_27_3 \ 
-W -depth 5 \ 
-report -end \ 

793
# 50 orbits at level 5:
#5 : 50 orbits
#total: 60
#(39^2, 26^2, 10, 6^2, 5, 3^9, 2^9, 1^{24}) average is 4 + 26 / 50

# time 4:31

classify_spreads_32.starter:
$ (ORBITER) -v 10 \n-define F -finite.field -q 2 -end \n-define P -projective.space -n 9 -field F -v 0 -end \n-define C -spread.classifier \n-define -projective.space P \n-k 5 \n-starter.size 5 \n-poset.classification.control \n-draw.options \n-radius 20 \n-nodes.empty \n-line_width 0.2 \n-embedded \n-end \n-W -depth 5 \n-draw_poset \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-W -depth 5 \n-report -end \n-end \n-with C -do -spread.classify.activity \n-prepare_lifting.single_case 0 \n
# Section 12.2: Translation planes

**SECTION TRANSLATION PLANES:**

create_translation_plane_9b:

```
$ (ORBITER) -v 3 \
-define F -finite_field -q 3 -end \
-define G -linear_group -PGL 4 F -end \
-define G1 -linear_group -PGL 5 F -end \
-define S -spread -kernel_field F \
-group G -k 2 -catalogue 1 \
-end \
-define T -translation_plane S G G1 -end
```

```
$ (ORBITER) -v 2 \
-draw_matrix \
-input_csv_file plane_catalogue_q3_k2_1_incma.csv \
-box_width 6 -bit_depth 8 \
-partition 2 91 91 \
-end
```

```
open plane_catalogue_q3_k2_1_incma_draw.bmp
```

```
# creates plane_catalogue_q3_k2_1_incma.csv
```

TP_9.0:

```
$ (ORBITER) -v 3 \
-define F -finite_field -q 3 -end \
-define PGL4 -linear_group -PGL 4 F -end \
-define PGL5 -linear_group -PGL 5 F -end \
-with PGL4 -and PGL5 -do \
-group.theoretic.activity \
-Andre_Bruck_Bose_construction 0 "TP9-0" \
-end
```

```
$ (ORBITER) -v 2 \
-draw_matrix \
-input_csv_file TP9-0_incma.csv \
-box_width 6 -bit_depth 8 \
```
\begin{verbatim}
13895  ▶ ▶ ▶ -partition 6 91 91 \13896  ▶ -end
13897  ▶ open TP9-0_incma_draw.bmp
13898  ▶ pdflatex TP9-0_report.tex
13899  ▶ open TP9-0_report.pdf
13900
13901  TP_9.1:
13902  ▶ $(ORBITER) -v 3 \13903  ▶ ▶ -define F -finite_field -q 3 -end \13904  ▶ ▶ -define PGL4 -linear_group -PGL 4 F -end \13905  ▶ ▶ -define PGL5 -linear_group -PGL 5 F -end \13906  ▶ ▶ -with PGL4 -and PGL5 -do \13907  ▶ ▶ -group_theoretic_activity \13908  ▶ ▶ ▶ -Andre_Bruck_Bose.construction 1 "TP9-1" \13909  ▶ ▶ -end
13910  ▶ $(ORBITER) -v 2 \13911  ▶ ▶ -draw_matrix \13912  ▶ ▶ ▶ -input_csv_file TP9-1_incma.csv \13913  ▶ ▶ ▶ -box_width 6 -bit_depth 8 \13914  ▶ ▶ ▶ -partition 6 91 91 \13915  ▶ ▶ -end
13916  ▶ open TP9-1_incma_draw.bmp
13917  ▶ pdflatex TP9-1_report.tex
13918  ▶ open TP9-1_report.pdf
13919
13920
13921
13922  TP_16.4:
13923  ▶ $(ORBITER) -v 3 \13924  ▶ ▶ -define F -finite_field -q 4 -end \13925  ▶ ▶ -define PGL4 -linear_group -PGGL 4 F -end \13926  ▶ ▶ -define PGGL5 -linear_group -PGGL 5 F -end \13927  ▶ ▶ -with PGGL4 -and PGGL5 -do \13928  ▶ ▶ -group_theoretic_activity \13929  ▶ ▶ ▶ -Andre_Bruck_Bose.construction 0 "TP16-4-HALL" \13930  ▶ ▶ -end
13931  ▶ $(ORBITER) -v 2 \13932  ▶ ▶ -draw_matrix \13933  ▶ ▶ ▶ -input_csv_file TP16-4-HALL_incma.csv \13934  ▶ ▶ ▶ -box_width 6 -bit_depth 8 \13935  ▶ ▶ ▶ -partition 6 273 273 \13936  ▶ ▶ -end
13937  ▶ open TP16-4-HALL_incma_draw.bmp
13938  ▶ pdflatex TP16-4-HALL_report.tex
13939  ▶ open TP16-4-HALL_report.pdf
13940
13941
\end{verbatim}
# Section 12.3: Packings

SECTION PACKINGS:

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
# Section 12.4: BLT-sets

SECTION_BLT_SETS:

BLT_5_1:
$\text{ORBITER} -v 2$
\begin{verbatim}
define F -finite_field -q 5 -end
define O -orthogonal_space 0 5 F -end
with O do orthogonal_space_activity
create_BLT_set -catalogue 1 -end
end
\end{verbatim}

pdflatex catalogue_q5_iso1.tex
open catalogue_q5_iso1.pdf

BLT_5_Linear:
$\text{ORBITER} -v 2$
\begin{verbatim}
define F -finite_field -q 5 -end
define O -orthogonal_space 0 5 F -end
with O do orthogonal_space_activity
create_BLT_set -family "Linear" -end
end
\end{verbatim}
pdflatex BLT_Linear_q5.tex
open BLT_Linear_q5.pdf

BLT_9_K1:
$\text{ORBITER} -v 2$
\begin{verbatim}
define F -finite_field -q 9 -end
define O -orthogonal_space 0 5 F -end
with O do orthogonal_space_activity
create_BLT_set -family "K1" -end
end
\end{verbatim}
pdflatex BLT_K1_q9.tex
open BLT_K1_q9.pdf

BLT_11_0:
$\text{ORBITER} -v 2$
\begin{verbatim}
define F -finite_field -q 11 -end
define O -orthogonal_space 0 5 F -end
with O do orthogonal_space_activity
\end{verbatim}
#pdflatex O 1 6 2 report.tex
#open O 1 6 2 report.pdf

BLT
Fisher:

$$(ORBITER) -v 2 \ 
-define F -finite_field -q 11 -end \ 
-define O -orthogonal_space 0 5 F -end \ 
-with 0 -do -orthogonal_space_activity \ 
-do -orthogonal_space_activity \ 
-do -create_BLT_set -family "Fisher" -end \ 
-end

pdflatex BLT_Fisher_q11.tex
open BLT_Fisher_q11.pdf

BLT
Mondello:

$$(ORBITER) -v 2 \ 
-define F -finite_field -q 11 -end \ 
-define O -orthogonal_space 0 5 F -end \ 
-with 0 -do -orthogonal_space_activity \ 
-do -orthogonal_space_activity \ 
-do -create_BLT_set -family "Mondello" -end \ 
-end

pdflatex BLT_Mondello_q11.tex
open BLT_Mondello_q11.pdf

BLT
FTWKB:

$$(ORBITER) -v 2 \ 
-define F -finite_field -q 11 -end \ 
-define O -orthogonal_space 0 5 F -end \ 
-with 0 -do -orthogonal_space_activity \ 
-do -orthogonal_space_activity \ 
-do -create_BLT_set -family "FTWKB" -end \ 
-end

pdflatex BLT_FTWKB_q11.tex
open BLT_FTWKB_q11.pdf

# for K2, q must be congruent to 2 or 3 mod 5

BLT
K2:

$$(ORBITER) -v 2 \ 
-define F -finite_field -q 13 -end \ 
-define O -orthogonal_space 0 5 F -end \ 
-with 0 -do -orthogonal_space_activity \ 
-do -orthogonal_space_activity \ 
-do -create_BLT_set -family "Kantor2" -end \ 
-end

pdflatex BLT_K2_q13.tex
BLT_13_deep_14:
$\text{(ORBITER)} -v 2 \$
\begin{verbatim}
define F -finite_field -q 13 -end
define 0 -orthogonal_space 0 5 F -end
with 0 -do -orthogonal_space_activity 
BLT_set_starter 14
\end{verbatim}
\begin{verbatim}
problem_label BLT_q13 -W -depth 14 -end
\end{verbatim}
\begin{verbatim}
end
\end{verbatim}

BLT_11_deep_search:
$\text{(ORBITER)} -v 2 \$
\begin{verbatim}
define F -finite_field -q 11 -end
define 0 -orthogonal_space 0 5 F -end
define C -BLT_set_classifier 0 -starter_size 12 -end
with C -do -BLT_set_classify_activity 
compute_starter 
problem_label BLT_q11
\end{verbatim}
\begin{verbatim}
W -depth 12
report -end
end
\end{verbatim}
\begin{verbatim}
end
\end{verbatim}
pdflatex BLT_q11_poset.tex

BLT_13_deep_search:
$\text{(ORBITER)} -v 2 \$
\begin{verbatim}
define F -finite_field -q 13 -end
define 0 -orthogonal_space 0 5 F -end
define C -BLT_set_classifier 0 -starter_size 14 -end
with C -do -BLT_set_classify_activity 
compute_starter 
problem_label BLT_q13
\end{verbatim}
\begin{verbatim}
W -depth 14
report -end
end
\end{verbatim}
pdflatex BLT_q13_poset.tex
open BLT_q13_poset.pdf

BLT_13.classify_starter:

$\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 \\
> -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 \\
> -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 \\

# 3 solutions:
BlT 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 
-define O -orthogonal_space 0 5 F -end 
-define C -BLT_set_classifier 0 -starter_size 5 -end 
compute_starter 
-problem_label BLT_q13 
-W -depth 5 
-end 
-with C -do -BLT_set_classify_activity 
-isomorph 
"./" 
"./" 
-use_database_for_starter 
-build_db 
-solution_prefix "" 
-base_fname "" 
-end 
-end 
-end 
-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 
```

802
Compute starter

-problem_label BLT_q13
-W -depth 5
-end
-end
-with C -do -BLT_set_classify_activity

-isomorph

-use_database_for_starter
-compute_orbits
-list_of_cases BLT_q13_list_of_cases_5_0_1.csv
-solution_prefix ""
-base_fname "BLT_q13_graph"
-end
-end
-end
-with C -do -BLT_set_classify_activity

-compute_starter

-def F -finite_field -q 13 -end
-def 0 -orthogonal_space 0 5 F -end
-def C -BLT_set_classifier 0 -starter_size 5 -end
-with C -do -BLT_set_classify_activity

-compute_starter

-def F -finite_field -q 13 -end
-def 0 -orthogonal_space 0 5 F -end
-def C -BLT_set_classifier 0 -starter_size 5 -end

-isomorph

-use_database_for_starter
-isomorph_testing
-isomorph_testing
-solution_prefix ""
-base_fname "BLT_q13_graph"
-end
-end

BLT.13_isomorph_testing:

 $(ORBTER) -v 4

-def F -finite_field -q 13 -end
-def 0 -orthogonal_space 0 5 F -end
-def C -BLT_set_classifier 0 -starter_size 5 -end
-with C -do -BLT_set_classify_activity

-compute_starter

-def F -finite_field -q 13 -end
-def 0 -orthogonal_space 0 5 F -end
-def C -BLT_set_classifier 0 -starter_size 5 -end

-isomorph

-use_database_for_starter
-isomorph_testing
-solution_prefix ""
-base_fname "BLT_q13_graph"
-end
-end

###############################################################################
Section 13.1: Creating Graphs:

```bash
make_triangle_graph:
  echo $(TRIANGLE_GRAPH) >triangle_graph.csv

Chain_232:
  $(ORBITER) -v 2 \
  define P1 -vector -dense 2,3,2 -end \
  define P2 -vector -dense 2,3,2 -end \
  define Gamma -graph \
  chain_graph P1 P2 \
  -end

Paley_13_graph:
  $(ORBITER) -v 2 \
  define Gamma -graph -Paley 13 -end \
```

804
triheiral_pair_graph:
  $(ORBITER) -v 2 \
  $define Gamma \\
  $graph -triheiral_pair_disjointness_graph \\
  $end

small_graph:
  $(ORBITER) -v 2 \
  $define G -graph -edges_as_pairs \\
  5 "0,1,0,2,0,3,0,4,1,3,1,4,2,4" \\
  $end

petersen:
  $(ORBITER) -v 2 \
  $define G -graph -Johnson 5 2 0 -end

Johnson_6_2_0:
  $(ORBITER) -v 2 \
  $define G -graph -Johnson 6 2 0 -end

Hamming_graph_3:
  $(ORBITER) -v 2 \
  $define G -graph -Hamming 3 2 -end

Hamming_graph_7:
  $(ORBITER) -v 2 \
  $define G -graph -Hamming 7 2 -end

# needs halljanko315.csv
# from https://www.win.tue.nl/~aeb/drg/graphs/HJ315.html
#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).
define G -graph
load csv no border halljanko315.csv
-end

HJ315_orbital_graph_3:
$(ORBITER) -v 2 
-define gens -vector -file halljanko315_gens.csv -end
-define G -permutation_group
-bsgs halljanko315 "File\halljanko315" 
315 1209600 "0,1,2" 6 gens
-end
-define Gamma -graph
-orbital_graph G 3
-end

HJ_d2_graph:
$(ORBITER) -v 6 
-define G -graph
-load_csv_no_border halljanko315.csv 
-distance 2
-end

Cayley_Z11_1mod3:
$(ORBITER) -v 2 
-define F -finite_field -q 11 -end
-define S -vector -dense
"1,1, 1,4, 1,7, 1,10" -end
-define G -linear_group -AGL 1 F
-subgroup_by_generators "Z11" 11 1 "1,1" 
-end
-define Gamma -graph
-Cayley_graph G S
-end

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

SECTION_GRAPH_THEORETIC_ACTIVITIES:

triangle_graph_properties:

echo $(TRIANGLE_GRAPH) >triangle_graph.csv
$ORBITER -v 6

define G -graph

dump_csv_no_border

triangle_graph.csv

dump -with G -do

dump_theoretic_activity -properties

dump -end

table

Cycle_13_draw:

dump -v 2

dump -graph -cycle 13 -end

dump -with Gamma -do

dump_theoretic_activity -export_csv -end

dump -with Gamma -do

dump -export_graphviz -end

dump -v 2 -draw_matrix

dump -input_csv_file Cycle_13.csv

dump -box_width 20 -bit_depth 8 -partition 4 13 13 -end

dump -Tpng Cycle_13.gv >Cycle_13.png

dump -#twopi -Tpng Cycle_13.gv >Cycle_13.png

Section 13.2: Graphs Theoretic Activities

Cayley_Sym4 Star:

Define S = vector dense "1,0,2,3, 2,1,0,3, 3,1,2,0" -end

Define G = permutation group = symmetric group 4

Define Gamma = graph

With Gamma -do

Graph theoretic activity -export_csv -end

With Gamma -do

Graph theoretic activity -export_graphviz -end

Table -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
Cycle 9 eigenvalues:

```bash
$ (ORBITER) -v 2 \n (define Gamma -graph \n  -cycle 9 \n  -end \n  -with Gamma -do \n  -graph_theoretic_activity -eigenvalues -end

pdflatex Cycle_9_eigenvalues.tex
open Cycle_9.eigenvalues.pdf
```

Paley 13 draw:

```bash
$ (ORBITER) -v 2 \n  (define Gamma -graph -Paley 13 -end \n  -with Gamma -do \n  -graph_theoretic_activity -export.csv -end \n  -with Gamma -do \n  -graph_theoretic_activity -export.graphviz -end

$ (ORBITER) -v 2 -draw_matrix \n  -input_csv_file Paley_13.csv \n  -box_width 20 -bit_depth 8 -partition 4 13 13 -end

dot -Tpng Paley_13.gv >Paley_13.png
open Paley_13.draw.bmp
```

Paley 13 eigenvalues:

```bash
$ (ORBITER) -v 2 \n  (define Gamma -graph \n  -Paley 13 \n  -end \n  -with Gamma -do \n  -graph_theoretic_activity -eigenvalues -end

pdflatex Paley_13.eigenvalues.tex
open Paley_13.eigenvalues.pdf
```

Cayley \text{Z}_{11, \text{lmod3}} eigenvalues and draw:

```bash
$ (ORBITER) -v 2 \n  (define Gamma -graph \n  -Paley 13 \n  -end \n  -with Gamma -do \n  -graph_theoretic_activity -eigenvalues -end

pdflatex Paley_13.eigenvalues.tex
open Paley_13.eigenvalues.pdf
```

```
808
```
Define G as a linear group AGL(1, F).

Define Gamma as a graph by subgroup generators Z11.

Define Gamma as a Cayley graph G, S.

With Gamma, do graph theoretic activity such as drawing eigenvalues.

pdflatex Cayley_graph_AGL_1_11_draw.tex

open Cayley_graph_AGL_1_11_draw.pdf

CayleySym4.coxeter_draw:

$(\text{ORBITER}) -v 2 \$

Define S as a vector-

Define G as a permutation group symmetric group 4.

With Gamma, do graph theoretic activity such as drawing.

pdflatex Cayley_graph_Perm4_draw.tex

open Cayley_graph_Perm4_draw.pdf

CayleySym5.coxeter_draw:

$(\text{ORBITER}) -v 2 \$

Define S as a vector-

Define G as a permutation group symmetric group 5.

With Gamma, do graph theoretic activity such as drawing.

pdflatex Cayley_graph_Perm5_draw.tex

open Cayley_graph_Perm5_draw.pdf
Cayley_Sym4_star_eigenvalues_and_draw:

```bash
$(ORBITER) -v 2 \n  -draw_options -xin 1000000 -yin 1000000 -embedded -end \n  -define S -vector -dense "1,0,2,3, 2,1,0,3, 3,1,2,0" -end \n  -define G -permutation_group -symmetric_group 4 \n  -end \n  -define Gamma -graph \n  -Cayley_graph G S \n  -end \n  -with Gamma -do \n  -graph_theoretic_activity -eigenvalues -end \n  -with Gamma -do \n  -graph_theoretic_activity -draw -end
```

```
pdflatex Cayley_graph_Perm4_draw.tex
open Cayley_graph_Perm4_draw.pdf
```

```
pdflatex Cayley_graph_Perm4_eigenvalues.tex
open Cayley_graph_Perm4_eigenvalues.pdf
```

```
graph_v5_e7.colored_graph:
$(ORBITER) -v 2 \n  -define G -graph -edges_as_pairs 5 \n  -"0,1,0,2,0,3,0,4,1,3,1,4,2,4" \n  -end \n  -with G -do \n  -graph_theoretic_activity -save -end
```

```
small_graph_draw:
$(ORBITER) -v 2 \n  -define G -graph -edges_as_pairs 5 \n  -"0,1,0,2,0,3,0,4,1,3,1,4,2,4" \n  -end \n  -with G -do \n  -graph_theoretic_activity -export_csv -end \n  -with G -do \n  -graph_theoretic_activity -export_graphviz -end \n  -with G -do \n  -graph_theoretic_activity -save -end
```

```
$(ORBITER) -v 2 -draw_matrix \n  -input_csv_file graph_v5_e7.csv \n  -box_width 40 -bit_depth 24 \n  -partition 4 "1,1,1,1," "1,1,1,1" -end
```

810
14598  \texttt{\$\{ORBITER\} -v 2 \$
14599  \texttt{-draw_matrix \$
14600  \texttt{-input_csv_file Johnson_5.2.0.csv \$
14601  \texttt{-box_width 40 -bit_depth 24 -partition 4 \"10\" \"10\" -end \$
14602  \texttt{\$\{ORBITER\} -v 2 -draw_matrix \$
14603  \texttt{-input_csv_file Johnson_6.2.0.csv \$
14604  \texttt{-box_width 40 -bit_depth 24 -partition 4 \"10\" \"10\" -end \$
14605  \texttt{\$\{ORBITER\} -v 2 -draw_matrix \$
14606  \texttt{-input_csv_file Johnson_6.2.0.csv \$
14607  \texttt{-box_width 40 -bit_depth 24 -partition 4 \"10\" \"10\" -end \$

Hamming graph 7, draw:
$\text{ORBITER} -v 2$
\begin{verbatim}
$\text{-define G -graph -Hamming 7 2 -end}$
\end{verbatim}
\begin{verbatim}
$\text{-define P1 -vector -dense 2,3,2 -end}$
\end{verbatim}
\begin{verbatim}
$\text{-define P2 -vector -dense 2,3,2 -end}$
\end{verbatim}
\begin{verbatim}
$\text{-define Gamma -graph -chain_graph P1 P2 -end}$
\end{verbatim}
\begin{verbatim}
$\text{-export -csv}$
\end{verbatim}
\begin{verbatim}
$\text{-with Gamma -do}$
\end{verbatim}
\begin{verbatim}
$\text{-properties}$
\end{verbatim}
\begin{verbatim}
$\text{-save -end}$
\end{verbatim}

Chain 232, properties:
$\text{ORBITER} -v 2$
\begin{verbatim}
$\text{-define P1 -vector -dense 2,3,2 -end}$
\end{verbatim}
\begin{verbatim}
$\text{-define P2 -vector -dense 2,3,2 -end}$
\end{verbatim}
\begin{verbatim}
$\text{-define Gamma -graph -chain_graph P1 P2 -end}$
\end{verbatim}
\begin{verbatim}
$\text{-export -csv}$
\end{verbatim}
\begin{verbatim}
$\text{-with Gamma -do}$
\end{verbatim}
\begin{verbatim}
$\text{-properties}$
\end{verbatim}
\begin{verbatim}
$\text{-save -end}$
\end{verbatim}

Chain 232, eigen:
$\text{ORBITER} -v 2$
\begin{verbatim}
$\text{-define P1 -vector -dense 2,3,2 -end}$
\end{verbatim}
\begin{verbatim}
$\text{-define P2 -vector -dense 2,3,2 -end}$
\end{verbatim}
\begin{verbatim}
$\text{-define Gamma -graph -chain_graph P1 P2 -end}$
\end{verbatim}
\begin{verbatim}
$\text{-with Gamma -do}$
\end{verbatim}
\begin{verbatim}
$\text{-properties}$
\end{verbatim}
\begin{verbatim}
$\text{-save -end}$
\end{verbatim}
14692 \> \> \> -graph_theoretic_activity \\
14693 \> \> \> -eigenvalues \\
14694 \> \> -end \\
14695 \> pdflatex chain_graph_eigenvalues.tex \\
14696 \> open chain_graph_eigenvalues.pdf \\
14697 \\
14698 \\
14699 \\
14700 # need the file halljanko315.csv \\
14701 \\
14702 HJ_properties: \\
14703 \> $(ORBITER) -v 6 \\
14704 \> \> -define G -graph \\
14705 \> \> \> -load_csv_no_border \\
14706 \> \> \> halljanko315.csv \\
14707 \> \> -end \\
14708 \> \> -with G -do \\
14709 \> \> \> -graph_theoretic_activity -properties \\
14710 \> \> -end \\
14711 \\
14712 #Degree type: (10^{315} ) \\
14713 \\
14714 \\
14715 \\
14716 HJ_d2_properties: \\
14717 \> $(ORBITER) -v 6 \\
14718 \> \> -define G -graph \\
14719 \> \> \> -load_csv_no_border \\
14720 \> \> \> halljanko315.csv \\
14721 \> \> \> -distance_2 \\
14722 \> \> -end \\
14723 \> \> -with G -do \\
14724 \> \> \> -graph_theoretic_activity \\
14725 \> \> \> -properties \\
14726 \> \> -end \\
14727 \\
14728 \\
14729 #Degree type: (80^{315} ) \\
14730 \\
14731 \\
14732 \\
14733 \\
14734 PGO_5_2_collinearity_graph: 0_5_2_incidence_matrix.csv \\
14735 \> $(ORBITER) -v 3 \\
14736 \> \> -define Inc -vector -file 0_5_2_incidence_matrix.csv -end \\
14737 \> \> -define Gamma -graph -collinearity_graph Inc -end \\
14738 \> \> -with Gamma -do \\

813
triangular pair graph draw:
$$(\text{ORBITER}) -v 2 -define \Gamma \$$
$$-\text{graph } -\text{triangular_pair_disjointness_graph } -\text{end } \$$
$$-\text{with } \Gamma -\text{do } \$$
$$-\text{graph_theoretic_activity } -\text{export } \text{csv } -\text{end } \$$
$$(\text{ORBITER}) -v 2 -\text{draw_matrix } \$$
$$-\text{input } \text{csv file } \text{triangular_pair_disjointness.csv } \$$
$$-\text{box_width } 20 -\text{depth } 8 -\text{end } \$$
$$\text{open } \text{triangular_pair_disjointness.draw.bmp } \$$

# Section 13.3: Graph Theory: Classification

SECTION GRAPH THEORY CLASSIFICATION:

graph classify 5:
$$(\text{ORBITER}) -v 2 \$$
$$-\text{orbiter_path } (\text{ORBITER_PATH}) \$$
$$-\text{define } \Gamma -\text{graph_classification } \$$
$$-\text{n } 5 \$$
$$-\text{poset_classification_control } \$$
$$-\text{problem_label } \text{graphs.v5 } \$$
$$-\text{depth } 10 -\text{draw_poset } \$$
$$-\text{draw_options } -\text{radius } 250 \$$
$$-\text{embedded } -\text{end } \$$
$$-\text{report } -\text{end } \$$
$$-\text{end } \$$
$$-\text{with } \Gamma -\text{do } \$$
$$-\text{graph_classification_activity } \$$
$$-\text{list_graphs_at.level } 5 5 \$$
$$-\text{end } \$$
$$-\text{with } \Gamma -\text{do } \$$
$$-\text{graph_classification_activity } \$$
$$-\text{draw_options } \$$
$$-\text{radius } 300 -\text{nodes_empty } \$$
$$-\text{line_width } 1.5 \$$
14797 tournament_classify_4:
14798 \$\texttt{(ORBITER)} \ -v \ 2 \ \\
14799 \texttt{-define GC -graph.classification} \\
14800 \texttt{-n 4 -tournament} \\
14801 \texttt{-poset.classification.control} \\
14802 \texttt{-problem.label tournament_4} \\
14803 \texttt{-depth 6 -draw.poset} \\
14804 \texttt{-draw.options} \\
14805 \texttt{-radius 250 -embedded} \\
14806 \texttt{-end} \\
14807 \texttt{-end} \\
14808 \texttt{-end} \\
14809 \texttt{-with GC -do} \\
14810 \texttt{-graph.classification.activity} \\
14811 \texttt{-draw.options} \\
14812 \texttt{-radius 400} \\
14813 \texttt{-line.width 2 -scale 0.1} \\
14814 \texttt{-end} \\
14815 \texttt{-drawgraphs.at.level 6} \\
14816 \texttt{-end} \\
14817 \texttt{-print.symbols} \\
14818 \texttt{pdflatex tournament_4.level_6.reps.tex} \\
14819 \texttt{open tournament_4.level_6.reps.pdf} \\
14820 \texttt{\ldots}
14821
14822
14823
14824
14825 graphclassify_8.r3:
14826 \$\texttt{(ORBITER)} \ -v \ 3 \ \\
14827 \texttt{-define GC -graph.classification} \\
14828 \texttt{-n 8 -regular 3} \\
14829 \texttt{-poset.classification.control} \\
14830 \texttt{-problem.label graphs.v8.r3} \\
14831 \texttt{-depth 12 -draw.poset} \\
14832 \texttt{-draw.options -radius 250} \\
14833 \texttt{-end} \\
14834 \texttt{-print.symbols} \\
14835 \texttt{pdflatex graphclassify_8.r3}
Symmetric 4 inversion graph recognize:

$(ORBITER) -v 10$
$define G - permutation_group - symmetric_group 4 - end$
$with G - do$
$group_theoretic_activity$
$export_inversion_graphs "Symmetric4_inversion_graphs.csv"$
$end$
$(ORBITER) -v 2$
$define GC - graph_classification$
$n 4$
$poset_classification_control$
$problem_label graphs_v4 - depth 6 - draw_poset$
$draw_options - radius 250 - embedded - end$
$end$
$with GC - do$
$graph_classification_activity$
$recognize_graphs_from_adjacency_matrix_csv$
$Symmetric4_inversion_graphs.csv$
$end$

Symmetric 5 inversion graph recognize:

$(ORBITER) -v 10$
$define G - permutation_group - symmetric_group 5 - end$
$with G - do$
$group_theoretic_activity$
SECTION GRAPH THEORY CLIQUE FINDING:

small_graph_cliques: graph_v5_e7.colored_graph

# nb_sol = 3
# all_cliques_black_and_white

BLT_q13_graph_5_0_cliques_rainbow:

# all_rainbow_cliques

small_graph_cliques_Sajeeb:

# nb_sol = 3

Paley_13_aut:

# writes Paley_13_group.makefile

# User time: 0 of a second, dt=0 tps = 100

# nb_calls_to_densenauty=1

Paley_13:
14973 ▶ ▶ -define G -permutation_group \\
14974 ▶ ▶ -bsgs Paley_13 "Paley\_13" 13 78 "0,1" 3 gens -end \\
14975 14976 14977 Paley_13.clique classify:
14978 ▶ $(ORBITER) -v 4 \\
14979 ▶ ▶ -define gens -vector -file Paley_13_gens.csv -end \\
14980 ▶ ▶ -define G -permutation_group \\
14981 ▶ ▶ -bsgs Paley_13 "Paley\_13" 13 78 "0,1" 3 gens -end \\
14982 ▶ ▶ -define Gamma -graph -Paley 13 -end \\
14983 ▶ ▶ -with G -do \\
14984 ▶ ▶ -group.theoretic_activity \\
14985 ▶ ▶ ▶ -poset.classification.control \\
14986 ▶ ▶ ▶ ▶ -W \\
14987 ▶ ▶ ▶ ▶ -problem.label Paley13.clique \\
14988 ▶ ▶ ▶ ▶ -clique.test Gamma \\
14989 ▶ ▶ ▶ ▶ -depth 5 \\
14990 ▶ ▶ ▶ -end \\
14991 ▶ ▶ ▶ -orbits.on.subsets 5 \\
14992 ▶ ▶ ▶ -report \\
14993 ▶ ▶ -end \\
14994 #User time: 0.01 of a second, dt=1 tps = 100
14996 14997 14998 Paley_13.clique.all:
14999 ▶ $(ORBITER) -v 10 \\
15000 ▶ ▶ -define Gamma -graph -Paley 13 -end \\
15001 ▶ ▶ -with Gamma -do \\
15002 ▶ ▶ -graph.theoretic.activity \\
15003 ▶ ▶ ▶ -findCliques -target.size 3 \\
15004 ▶ ▶ -end \\
15005 15006 15007 15008 15009 15010 15011 PG0_5_2.cliques: 0_5_2.incidence_matrix.csv
15012 ▶ $(ORBITER) -v 3 \\
15013 ▶ ▶ -define Inc -vector -file 0_5_2.incidence_matrix.csv -end \\
15014 ▶ ▶ -define Gamma -graph -collinearity_graph Inc -end \\
15015 ▶ ▶ -with Gamma -do \\
15016 ▶ ▶ -graph.theoretic.activity \\
15017 ▶ ▶ ▶ -findCliques -target.size 3 -end \\
15018 ▶ ▶ -end \\
15019
HJ_d2_c5:
$($ORBITER) -v 6 \
-define G -graph \n-define Gamma -graph \
-define gens -vector \
-define G -permutation group \
-bsgs halljanko315 "File\halljanko315" \

HJ64_cliques5:
$($ORBITER) -v 6 \
-define Gamma -graph \
-define Gamma -graph \
-define gens -vector \
-define G -permutation_group \

HJ64_cliques5_classify:
$($ORBITER) -v 6 \
-define Gamma -graph \
-define gens -vector \
-define G -permutation_group \

#graph_theoretic_activity::perform_activity Gr->label=halljanko315 nb_sol = 26208 0

#graph_theoretic_activity::perform_activity Gr->label=Group_Perms15_Orbital_3 nb_sol = 1008

#Group_Perms15_Orbital_3.sol.csv

HJ64_cliques5_classify:
15065 ▷▷▷ 315 1209600 "0,1,42,95" 6 gens -end \n15066 ▷▷▷ -with G -do \n15067 ▷▷▷ -group_theoretic_activity \n15068 ▷▷▷ ▷▷▷ -poset_classification_control \n15069 ▷▷▷ ▷▷▷ ▷▷▷ -W \n15070 ▷▷▷ ▷▷▷ ▷▷▷ -problem_label HJ64_cliques \n15071 ▷▷▷ ▷▷▷ ▷▷▷ -clique_test Gamma \n15072 ▷▷▷ ▷▷▷ ▷▷▷ -depth 5 \n15073 ▷▷▷ ▷▷▷ ▷▷▷ -end \n15074 ▷▷▷ ▷▷▷ ▷▷▷ -orbits_on_subsets 5 \n15075 ▷▷▷ ▷▷▷ ▷▷▷ -report \n15076 ▷▷▷ ▷▷▷ -end \n15077 \n15078 #HJ64_cliques.reps_lvl_5.csv \n15079 \n15080 # 1 orbit \n15081 #ROW,REP,AGO,OL \n15082 #0,"0,8,31,110,283",1200,1008 \n15083 #END \n15084 \n15085 \n15086 \n15087 \n15088 \n15089 \n15090 #**************************************************************************** \n15091 # Chapter 14 - Combinatorial Objects \n15092 #**************************************************************************** \n15093 \n15094 \n15095 #**************************************************************************** \n15096 # Section 14.1: Finite Projective Spaces \n15097 \n15098 SECTION_COMBINATORIALOBJECTS: \n15099 \n15100 \n15101 \n15102 \n15103 Hirschfeld_q4_from_set: \n15104 ▷▷▷ $(ORBITER) -v 4 \n15105 ▷▷▷ ▷▷▷ -define H -set -here \n15106 ▷▷▷ ▷▷▷ ▷▷▷ $(HIRSCHFELD_SURFACE_Q4_SET_OF_POINTS) \n15107 ▷▷▷ ▷▷▷ ▷▷▷ -end \n15108 ▷▷▷ ▷▷▷ ▷▷▷ -define C -combinatorial_objects \n15109 ▷▷▷ ▷▷▷ ▷▷▷ ▷▷▷ -set_of_points H \n15110 ▷▷▷ ▷▷▷ ▷▷▷ ▷▷▷ -end\n15111
15112
15113
15114 hyperoval_16_create:
15115 \$ (ORBITER) -v 2 \\n15116 \> \> -define C -combinatorial_objects \\
15117 \> \> \> -set_of_points $(HYPEROVAL_16_16320) \\
15118 \> \> \> -set_of_points $(HYPEROVAL_16_144) \\
15119 \> \> -end \\
15120
15121
15122 EC_read: elliptic_curve_b1_c3_q11.txt
15123 \$ (ORBITER) -v 4 \\n15124 \> \> -define C -combinatorial_objects \\
15125 \> \> \> -file_of_points elliptic_curve_b1_c3_q11.txt \\
15126 \> \> -end
15127
15128
15129
15130 PG_3.5_assume_31_read:
15131 \$ (ORBITER) -v 2 \\n15132 \> \> -define C -combinatorial_objects \\
15133 \> \> \> -file_of_packings_through_spread_table \\
15134 \> \> \> \> H31_packings.csv \\
15135 \> \> \> \> SPREAD_TABLES_5_REG/spread_25_spreads.csv \\
15136 \> \> \> \> 5 \\
15137 \> \> \> -end
15138
15139
15140
15141 LS_AG_2.3_read:
15142 \$ (ORBITER) -v 2 \\n15143 \> \> -define C -combinatorial_objects \\
15144 \> \> \> -file_of_designs \\
15145 \> \> \> solutions.csv 9 84 3 12 \\
15146 \> \> -end
15147
15148
15149
15150 geo_7.3_read:
15151 \$ (ORBITER) -v 10 \\n15152 \> \> -draw_incidence_structure.description \\
15153 \> \> \> -width 60 -with_10 6 -end \\
15154 \> \> -define C -combinatorial_objects \\
15155 \> \> \> -file_of_incidence_geometries \\
15156 \> \> \> \> 7.3.inc 7 7 21 \\
15157 \> \> \> -end
15158
822
Desargues_path_lex_least_read:

```bash
echo $(DESARGUES_PATH_LEX_LEAST) >Desargues_path_lex_least.inc
```

```bash
$(ORBITER) -v 10 \
  -draw.incidence_structure_description \
  -width 60 -with_10 6 -end \
  -define C -combinatorial_objects \
  -file_of_incidence_geometries_by_row_ranks \
  Desargues_path_lex_least.inc 10 10 3 \
  -end
```

# Section 14.2: File Formats

geo_pasch_read:

```bash
$(ORBITER) -v 10 \
  -define C -combinatorial_objects \
  -file_of_incidence_geometries \
  pasch.inc 6 4 12 \
  -end
```

geo_pasch_given:

```bash
$(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

# Section 15.1: Overview of Canonical Forms

SECTION_OVERVIEW_CANONICAL_FORMS:
15211 # Section 15.2: Objects in projective Space
15212
15214 SECTION_OBJECTS_IN_PROJECTIVE_SPACE:
15215
15216 15217 EC_canon: elliptic_curve_b1_c3_q11.txt
15218 15219 15220 15221 15222 15223 15224 15225 15226 15227 15228 15229 15230 15231 15232 15233 15234 15235 15236 15237 15238 15239 15240 15241 15242 15243 15244 15245 15246 15247 15248 15249 15250 15251 15252
15206
15207
15208
15209
15210
15211
15212
15213
15214
15215
15216
15217
15218
15219
15220
15221
15222
15223
15224
15225
15226
15227
15228
15229
15230
15231
15232
15233
15234
15235
15236
15237
15238
15239
15240
15241
15242
15243
15244
15245
15246
15247
15248
15249
15250
15251
15252
824
Hirschfeld_q4_c: Hirschfeld_surface_q4.txt

$(ORBITER) -v 6 \
-define C -combinatorial_objects \
-file_of_points Hirschfeld_surface_q4.txt \
-end \
-define F -finite_field -q 4 -end \
-define P -projective_space -n 3 -field F -v 0 -end \
-with C -do \
-define C -combinatorial_objects \
-with C -do \
-define F -finite_field -q 4 -end \
-define P -projective_space -n 3 -field F -v 0 -end \
-with C -do \
-define C -combinatorial_object_activity \
-canonical_form_PG P \
-classification_prefix Hirschfeld_surface_q4 \
-save_ago \
-max_TDO_depth 10 \
-end \
-report \
-prefix Hirschfeld_surface_q4 \
-export_flag_orbits \
-show_TDO \
-show_TDA \
-dont_show_incidence_matrices \
-export_group \
-end \
-pdflatex Hirschfeld_surface_q4_classification.tex 

$pdflatex Hirschfeld_surface_q4_classification.tex 

$pdflatex Hirschfeld_surface_q4_classification.tex

Hirschfeld_q4_classification.pdf

# group order is 51840

Hirschfeld_q4_set_c:

$(ORBITER) -v 4 \
-define H -set -here \
-SHIELD_SURFACE_Q4_SET_OF_POINTS \
-end \
-define C -combinatorial_objects \
-set_of_points H \
-end \
-define F -finite_field -q 4 -end \
-define P -projective_space -n 3 -field F -v 0 -end \
-with C -do \
-combinatorial_object_activity \
-canonical_form_PG P \
-classification_prefix Hirschfeld_surface_q4 \
-save_ago \
-end \
-pdflatex Hirschfeld_surface_q4_classification.tex

825
Dickson_sets_stabilizer:

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

\$\text{-define } C\ \text{-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 } "0,1,2,3,5,6,10" \$

\$\text{-set_of_points } "0,1,2,3,5,6,4" \$

\$\text{-set_of_points } "0,1,2,3,5,6,7,11,13" \$

\$\text{-set_of_points } "3,6,9,7,10,12,8,11,13,14,4" \$

\$\text{-set_of_points } "3,5,6,9,7,10,12,11,13,14,4" \$

\$\text{-set_of_points } "0,1,2,3,5,6,9,7,10,12,4" \$

\$\text{-end} \$

\$\text{-define } F\ \text{-finite_field -q 2} \text{-end} \$

\$\text{-define } P\ \text{-projective_space -n 3 -field } F\ \text{-v 0} \text{-end} \$

\$\text{-with } C\ \text{-do} \$

\$\text{-combinatorial_object_activity} \$

\$\text{-canonical_form_PG} P \$

\$\text{-classification_prefix }\text{Dickson_sets} \$

\$\text{-save}_\text{ago} \$

\$\text{-report} \$

\$\text{-end} \$

\$\text{pdflatex }\text{Dickson_sets_classification.tex} \$

\$\text{open }\text{Dickson_sets_classification.pdf} \$

Endrass_7c: Endrass_F7.txt

\$\text{ORBITER} -v 2 \$

\$\text{-define } C\ \text{-combinatorial_objects} \$

\$\text{-file_of_points }\text{Endrass_F7.txt} \$

\$\text{-end} \$

\$\text{-define } F\ \text{-finite_field -q 7} \text{-end} \$

\$\text{-define } P\ \text{-projective_space -n 3 -field } F\ \text{-v 0} \text{-end} \$

\$\text{-with } C\ \text{-do} \$

\$\text{-combinatorial_object_activity} \$

\$\text{-canonical_form_PG} P \$

\$\text{-classification_prefix }\text{Endrass_F7} \$

826
# group order is 32

```latex
hyperoval_{16} canonical_form:

\$\text{ORBITER} -v 2 \$
\$-\text{define C -combinatorial_objects} \$
\$-\text{set_of_points }\$\text{HYPEROVAL}_{16320}\$
\$-\text{set_of_points }\$\text{HYPEROVAL}_{164}\$
\$-\text{end} \$
\$-\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}\text{PG}P \$
\$-\text{classification_prefix hyperoval_q16} \$
\$-\text{label hyperoval_q16} \$
\$-\text{save}_\text{ago} \$
\$-\text{save}_\text{transversal} \$
\$-\text{max}_\text{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} \$
\$-\text{end} \$
```

15384 \$\text{ORBITER} -v 2 -draw_matrix \$
15385 \$\text{open hyperoval_q16_classification.pdf} \$
15386 \$\text{pdflatex hyperoval_q16_classification.tex} \$
15387 \$\text{open hyperoval_q16_classification.pdf} \$
15388 \$\text{pdflatex Endrass_F7_classification.tex} \$
15389 \$\text{open Endrass_F7_classification.pdf} \$
```
15394 -secondary_input.csv_file hyperoval_q16_object1_TDA.csv \
15395 -box_width 4 -bit_depth 24 \
15396 -end 
15397 open hyperoval_q16_object1_TDA_flag_orbits_draw.bmp 
15398 
15399 
15400 
15401 
15402 
15403 cubic_curves_PG_2_8.canon: 
15404 $(ORBITER) -v 6 \
15405 -define C -combinatorial_objects \ 
15406 -set_of_points "2,3,28,46,51,61,40,71" \ 
15407 -end \ 
15408 -define F -finite_field -q 8 -end \ 
15409 -define P -projective_space -n 2 -field F -v 0 -end \ 
15410 -with C -do \ 
15411 "-combinatorial_object_activity " \
15412 "-canonical_form_PG P " \ 
15413 "-classification_prefix cc_8 " \ 
15414 "-save_ago " \ 
15415 "-max_TDO_depth 10 " \ 
15416 "-end " \ 
15417 "-report " \ 
15418 -end 
15419 pdflatex cc_8_classification.tex 
15420 open cc_8_classification.pdf 
15421 
15422 
15423 F_alpha_beta_gamma_delta_classify_q7_nauty: F_alpha_beta_gamma_delta_q7_points.txt 
15424 $(ORBITER) -v 6 \
15425 -define C -combinatorial_objects \ 
15426 -file_of_points \ 
15427 F_alpha_beta_gamma_delta_q7_points.txt \ 
15428 -end \ 
15429 -define F -finite_field -q 7 -end \ 
15430 -define P -projective_space -n 3 -field F -v 0 -end \ 
15431 -with C -do \ 
15432 -combinatorial_object_activity \ 
15433 "-canonical_form_PG P " \ 
15434 "-classification_prefix surface_15_lines_q7 " \ 
15435 "-save_ago " \ 
15436 "-save_transversal " \ 
15437 "-end " \ 
15438 -end 
15439 #pdflatex surface_15_lines_q7_classification.tex
#open surface_15_lines_q7_classification.pdf

# User time: 4:12 on Mac

# 6 orbits

ovoid_q8_canon: ovoid_q8.txt

$\texttt{(ORBITER) -v 6 \ }
\texttt{\ -define C -combinatorial_objects \ }
\texttt{\ -file_of_points ovoid_q8.txt \ }
\texttt{\ -end \ }
\texttt{\ -define F -finite_field -q 8 -end \ }
\texttt{\ -define P -projective_space -n 3 -field F -v 0 -end \ }
\texttt{\ -with C -do \ }
\texttt{\ -combinatorial_object_activity \ }
\texttt{\ -canonical_form_PG P \ }
\texttt{\ -classification_prefix ovoid \ }
\texttt{\ -label ovoid \ }
\texttt{\ -save_ago \ }
\texttt{\ -max_TDO_depth 4 \ }
\texttt{\ -end \ }
\texttt{\ -report \ }
\texttt{\ -prefix ovoid \ }
\texttt{\ -show_TDO \ }
\texttt{\ -show_TDA \ }
\texttt{\ -dont_show_incidence_matrices \ }
\texttt{\ -export_group \ }
\texttt{\ -end \ }
\texttt{\ -end}

pdflatex ovoid_classification.tex

open ovoid_classification.pdf
$\texttt{(ORBITER) -v 6 \ \}
\texttt{\-define C \-combinatorial_objects \}
\texttt{\-define F \-finite_field \-q 8 \-end \}
\texttt{\-define P \-projective_space \-n 3 \-field F \-v 0 \-end \}
\texttt{\-with C \-do \}
\texttt{\-combinatorial_object_activity \}
\texttt{\-canonical_form_PG P \}
\texttt{\-classification_prefix ovoid_ST \}
\texttt{\-label ovoid_ST \}
\texttt{\-save Ago \}
\texttt{-max_TDO_depth 4 \}
\texttt{-end \}
\texttt{-report \}
\texttt{-prefix ovoid_ST \}
\texttt{-show_TDO \}
\texttt{-show_TDA \}
\texttt{-don't_show_incidence_matrices \}
\texttt{-export_group \}
\texttt{-end \}
\texttt{pdflatex ovoid_ST_classification.tex}
\texttt{open ovoid_ST_classification.pdf}
\texttt{# group order 87360 = 3 \times 29120}
\texttt{SUZUKI.8_GENERATORS="\}
\texttt{1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1, \}
\texttt{1,0,0,0,0,6,0,0,0,0,2,0,0,0,0,3,0, \}
\texttt{1,0,0,0,1,1,1,0,0,0,1,0,1,0,1,0,1, \}
\texttt{1,0,0,0,3,6,2,2,5,0,2,0,3,0,6,3,2, \}
\texttt{0,1,0,0,1,0,0,0,0,0,0,1,0,0,1,0,2"}
\texttt{Suzuki.8:}
\texttt{$\texttt{(ORBITER) -v 6 \ \}
\texttt{\-define F \-finite_field \-q 8 \-end \}
\texttt{\-define gens \-vector \-field F \}
\texttt{\-compact $\langle\text{SUZUKI.8_GENERATORS}\rangle \-end \}
\texttt{\-define G \-linear_group \-PGGL 4 8 \}
\texttt{\-subgroup_by_generators "Sz8" "87360" 5 gens \}
\texttt{\-end \}
\texttt{-with G \-do \}
\texttt{-group_theoretic_activity \}
\texttt{-report \}
\texttt{-end \}
\texttt{pdflatex PGGL_4_8_Subgroup_Sz8_87360_report.tex}
# Section 15.3: Incidence Geometries

SECTION_Incidence_Geometries:

geo_7.3_c:

$(ORBITER) -v 10 \
-define C -combinatorial_objects \
-file_of_incidence_geometries 7.3.inc 7 7 21 \
-define C -combinatorial_object_activity \
-canonial_form \
-classification_prefix 7.3 \
-label 7.3 \
-save_age \
-save_transversal \
-end \
-with C -do \
-combinatorial_object_activity \
-canonial_form \
-classification_prefix 7.3 \
-label 7.3 \
-save_age \
-save_transversal \
-end \
-report \
-prefix 7.3 \
-export_flag_orbits \
-show_flag_orbits \
-export_group \
-end \
-end

Open 7.3_classification.pdf

$(ORBITER) -v 2 -draw_matrix \
-input_csv_file 7.3_object0_TDA_flag_orbits.csv \
-secondary_input_csv_file 7.3_object0_TDA.csv \
-box_width 32 -bit_depth 24 \
-end

$(ORBITER) -v 2 -draw_matrix \
-input_csv_file 7.3_object0_INP_flag_orbits.csv \
-secondary_input_csv_file 7.3_object0_INP.csv \
-box_width 32 -bit_depth 24 \
-end
open 7_3_object0_INP_flag_orbits_draw.bmp

geo_10_3_c:

$\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 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} \\
-\text{-end} \\
-\text{-end}

\text{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} \
geo_14_3_c:
$ (ORBITER) -v 2 \ 
$draw.incidence_structure_description \ -width 60 -with_10 6 -end \ 
$define Test_lines -set -loop 4 15 1 -end \ 
$define C -combinatorial_objects \ 
$file_of_incidence_geometries 14_3.inc 14 14 42 \ 
$end \ 
-with C -do \ 
-combinatorial_object_activity \ 
$canonical_form \ 
-classification_prefix 14_3 \ 
-label 14_3 \ 
-save_ago \ 
-save_transversal \ 
-end \ 
-end
report \ 
-prefix 14_3 \ 
-export_flag_orbits \ 
-show_incidence_matrices \ 
-export_group \ 
-end \ 
geo_15_3_c:
$ (ORBITER) -v 2 \ 
$draw.incidence_structure_description \ 
-width 50 -with_10 5 -end \ 
$define C -combinatorial_objects \ 
$file_of_incidence_geometries 15_3.inc 15 15 45 \ 
$end \ 
-with C -do \ 
-combinatorial_object_activity \ 
$canonical_form \ 
-classification_prefix 10_3 \ 
-label 10_3 \ 
-save_ago \ 
-end
pdflatex 15_3_classification.tex
open 15_3_classification.pdf
TFC_24_3_c:
echo $(FILE_24_3_TFC_INC) >24_3_TFC.inc
$ (ORBITER) -v 6 \
define C -combinatorial_objects \
file_of_incidence_geometries 24_3_TFC.inc 24 24 72 \ne -end \nwith C -do \ncombinatorial_object_activity \ncanonical_form \nclassification_prefix 24_3_TFC \nlabel 24_3_TFC \nsave ago \nend \nreport \nprefix 24_3_TFC \nexport_flag_orbits \nshow_TDO \nshow_TDA \nshow_incidence_matrices \nend \npdflatex 24_3_TFC_classification.tex \nopen 24_3_TFC_classification.pdf \n$(ORBITER) -v 2 -draw_matrix \n-input_csv_file 24_3_TFC_object2_TDA_flag_orbits.csv \n-secondary_input_csv_file 24_3_TFC_object2_TDA.csv \n-box_width 40 -bit_depth 24 \nend \nopen 24_3_TFC_object2_TDA_flag_orbits_draw.bmp
geo_40_4_g4_c:
$ (ORBITER) -v 2 \ndraw_incidence_structure_description \n-width 50 -with_10 5 -end \nwith C -do \ncombinatorial_object_activity \ncanonical_form \nclassification_prefix 40_4_g4 \nlabel 40_4_g4 \nsave ago \nend \nreport \nprefix 40_4_g4 \nexport_flag_orbits \nshow_TDO \nshow_TDA \n
geo_17_3_g4.c:
$\$(ORBITER) -v 2 \\
-draw_incidence_structure_description \\
-width 50 -with_10 5 -end \\
-define C -combinatorial_objects \\
-file_of_incidence_geometries 17_3_g4.inc 17 17 51 \\
-end \\
-with C -do \\
-combinatorial_object_activity \\
-canoncial_form \\
-classification_prefix 17_3_g4 \\
-label 17_3_g4 \\
-save_ago \\
-end \\
-report \\
-prefix 17_3_g4 \\
-export_flag_orbits \\
-show_TDO \\
-show_TDA \\
-show_incidence_matrices \\
-end \\
-end

geo_2_3.c: AG_2_3.inc
$\$(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 \\
-canoncial_form \\
-classification_prefix AG_2_3 \\
-label AG_2_3 \\
-save_ago \\
-max_TDO_depth 10 \\
-end \\
-report \\

836
-prefix AG_2_3 \\
-export_flag_orbits \\
-show_TDO \\
-show_TDA \\
-show_incidence_matrices \\
-end \\
-end \\
pdflatex AG_2_3.classification.tex \\
open AG_2_3.classification.pdf 
$(ORBITER) -v 2 -draw_matrix \\
-input Csv_file AG_2_3_object0_INP_flag_orbits.csv \\
-secondary_input_csv_file AG_2_3_object0_INP.csv \\
-box_width 40 -bit_depth 24 \\
-end \\
open AG_2_3_object0_INP_flag_orbits_draw.bmp 

geo_LSQ6.c: 
$(ORBITE) -v 10 \\
-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 \\
-cannonical_form \\
-classification_prefix LSQ6 \\
-label LSQ6 \\
-save_ago \\
-save_transversal \\
-end \\
-report \\
-prefix LSQ6 \\
-export_flag_orbits \\
-show_incidence_matrices \\
-export_group \\
-end \\
-end \\
pdflatex LSQ6.classification.tex 
#open LSQ6.classification.pdf 
$(ORBITE) -v 2 -draw_matrix \\
-input_csv_file LSQ6_object0_TDA_flag_orbits.csv \\
-secondary_input_csv_file LSQ6_object0_TDA_flag_orbits.csv
15863 \> & $\text{-box_width 32 -bit_depth 24}$ \\
15864 \> & $\text{-end}$ \\
15865 \> & $\text{$\$(ORBITER) -v 2 -draw_matrix \}$}$ \\
15866 \> & $\text{$\text{-input_csv_file LSQ6.object0.INP_flag_orbits.csv}$}$ \\
15867 \> & $\text{$\text{-secondary_input_csv_file LSQ6.object0.INP_flag_orbits.csv}$}$ \\
15868 \> & $\text{$\text{-box_width 32 -bit_depth 24}$}$ \\
15869 \> & $\text{-end}$ \\
15870 \> & $\text{open LSQ6.object0.INP_flag_orbits_draw.bmp}$ \\
15871 \\
15872 \\
15873 \\
15874 \\
15875 \\
15876 \\
15877 \\
15878 \\
15879 \# To Do: \\
15880 \\
15881 \quad \text{quartic_curve.25.0.0.canonical:}$ \\
15882 \> & $\text{$\$(ORBITER) -v 3 \}$}$ \\
15883 \> & $\text{$\text{-define F -finite_field -q 25 -end}$}$ \\
15884 \> & $\text{$\text{-define P -projective_space -n 2 -field F -v 0 -end}$}$ \\
15885 \> & $\text{$\text{-with P -do}$}$ \\
15886 \> & $\text{$\text{-projective_space_activity}$}$ \\
15887 \> & $\text{$\text{-canonical_form_PG}$}$ \\
15888 \> & $\text{$\text{-input}$}$ \\
15889 \> & $\text{$\text{-set_of_points "10,11,59,63,124,135,136,170,206,257,275,284,285,367,378,393,433,619,641,644"}$}$ \\
15890 \> & $\text{$\text{-set_of_points "9, 24, 62, 67, 77, 84, 87, 89, 125, 130, 158, 172, 197, 219, 266, 271, 325, 356, 391, 392, 400, 429, 454, 458, 470, 503, 531, 553, 605, 625, 627, 646"}$}$ \\
15891 \> & $\text{$\text{-set_of_points "9, 24, 55, 75, 79, 88, 93, 94, 112, 142, 186, 213, 230, 249, 316, 329, 337, 371, 381, 383, 392, 405, 417, 439, 464, 466, 470, 524, 530, 567, 583, 594"}$}$ \\
15892 \> & $\text{$\text{-set_of_points "10, 23, 34, 51, 65, 83, 117, 126, 144, 146, 147, 159, 172, 181, 185, 197, 198, 207, 240, 281, 283, 293, 300, 301, 305, 346, 357, 384, 409, 419, 444, 459, 463, 465, 468, 514, 533, 543, 547, 549, 586, 615, 618, 628"}$}$ \\
15893 \> & $\text{$\text{-set_of_points "2, 12, 48, 65, 87, 120, 189, 246, 305, 323, 354, 375, 434, 435, 455, 482, 496, 557, 586, 595"}$}$ \\
15894 \> & $\text{-end}$ \\
15895 \> & $\text{-classification.prefix quartic.25.0.0}$ \\
15896 \> & $\text{-report}$ \\
15897 \> & $\text{-end}$ \\
15898 \> & $\text{-end}$ \\
15899 \> & $\text{pdflatex quartic.25.0.0_classification.tex}$ \\
15900 \> & $\text{open quartic.25.0.0_classification.pdf}$ \\
15901
Section 15.4: Objects from Design Theory

SECTION OBJECTS FROM DESIGN THEORY:
design 27c:

```bash
$(ORBITER) -v 4
  -define C -combinatorial_objects
  -set_points "2,56,30,112,253,90,440,508"
  -end
```

15949 LS_AG_2_3_solutions.classify:

```bash
$(ORBITER) -v 2
  -draw.incidence_structure.description
  -width 20 -width.10 2 -end
  -define C -combinatorial_objects
  -file.of.designs
  solutions.csv 9 84 3 12
  -end
  -with C -do
  -combinatorial_object.activity
  -canonical_form
  -save ago
  -save_transversal
  -classification.prefix LS_AG_2_3
  -label LS_AG_2_3
  -max_TDO.depth 10
  -end
  -report
  -prefix LS_AG_2_3
  -export_flag.orbits
  -show_TDO
  -end
  -end
```

15972 pdflatex LS_AG_2_3.classification.tex
15973 open LS_AG_2_3.classification.pdf
15974 $(ORBITER) -v 2 -draw.matrix
15975 -input_csv_file LS_AG_2_3.object0.INP_flag.orbits.csv
15976 -secondary_input_csv_file LS_AG_2_3.object0.INP.csv
15977 -box_width 12 -bit_depth 24
15978 -end
15979 open LS_AG_2_3.object0.INP_flag.orbits.draw.bmp
15980 $(ORBITER) -v 2 -draw.matrix
15981 -input_csv_file LS_AG_2_3.object1.INP_flag.orbits.csv
15982 -secondary_input_csv_file LS_AG_2_3.object1.INP.csv
15983 -box_width 12 -bit_depth 24
15984 -end
15985 open LS_AG_2_3.object1.INP_flag.orbits.draw.bmp
15986
15987
15988
15989
15990
15991 design_27c:
15992 $(ORBITER) -v 4
15993 -define C -combinatorial_objects
15994 -set_of_points "2,56,30,112,253,90,440,508"
15995 -end
```
15996 \> \> -define F -finite_field -q 27 -override_polynomial 46 -end \ 
15997 \> \> -define P -projective_space -n 2 -field F -v 0 -end \ 
15998 \> \> -with C -do \ 
15999 \> \> -combinatorial_object_activity \ 
16000 \> \> \> -canonical_form PG P \ 
16001 \> \> \> -classification_prefix design \ 
16002 \> \> \> -end \ 
16003 \> \> \> -report \ 
16004 \> \> -end \ 
16005 \> pdflatex design_classification.tex \ 
16006 \> open design_classification.pdf \ 
16007 \ 
16008 \ 
16009 \ 
16010 design_PG_2_3_canonical: \ 
16011 \> $(ORBITER) -v 3 \ 
16012 \> \> -define D -design -q 3 -family PG_2_q -end \ 
16013 \> \> -with D -do \ 
16014 \> \> \> -design_activity \ 
16015 \> \> \> \> -export_inc \ 
16016 \> \> \> \> -end \ 
16017 \> \> \> -end \ 
16018 \> $(ORBITER) -v 3 \ 
16019 \> \> -draw_incidence_structure_description \ 
16020 \> \> \> -width 60 -with_10 6 -end \ 
16021 \> \> -define C -combinatorial_objects \ 
16022 \> \> \> -file_of_incidence_geometries PG_2_3_inc.txt 13 13 52 \ 
16023 \> \> -end \ 
16024 \> \> -with C -do \ 
16025 \> \> -combinatorial_object_activity \ 
16026 \> \> \> -canonical_form \ 
16027 \> \> \> \> -classification_prefix PG_2_3 \ 
16028 \> \> \> \> -label PG_2_3 \ 
16029 \> \> \> \> -save_ago \ 
16030 \> \> \> \> -save_transversal \ 
16031 \> \> \> \> -end \ 
16032 \> \> \> \> -report \ 
16033 \> \> \> \> \> -prefix PG_2_3 \ 
16034 \> \> \> \> \> -export_flag_orbits \ 
16035 \> \> \> \> \> -show_incidence_matrices \ 
16036 \> \> \> \> \> -export_group \ 
16037 \> \> \> \> \> -end \ 
16038 \> \> \> -end \ 
16039 \> pdflatex PG_2_3_classification.tex \ 
16040 \> open PG_2_3_classification.pdf \ 
16041 \> $(ORBITER) -v 2 -draw_matrix \ 
16042 \> \> -input_csv_file PG_2_3_object0_TDA_flag_orbits.csv \ 

841
wreath_product_designs_n4_k2_c: wreath_product_designs_n4_k2_inc.txt
$(ORBITER) -v 10 \
-draw_incidence_structure_description \
-width 60 -with_10 6 -end \
-define C -combinatorial_objects \
-file_of_incidence_geometries \
-wreath_product_designs_n4_k2_inc.txt \
-8 12 24 \
-end \
-with C -do \
-combinatorial_object_activity \
-canonical_form \
-classification_prefix wreath_4_2 \
-label wreath_4_2 \
-save Ago \
-save_transversal \
-end \
-report \
-prefix wreath_4_2 \
-export_flag_orbits \
-show_incidence_matrices \
-export_group \
-end \
-end
pdflatex wreath_4_2_classification.tex
open wreath_4_2_classification.pdf

wreath_product_designs_n8_k6_c: wreath_product_designs_n8_k6_inc.txt
$(ORBITER) -v 10 \
-draw_incidence_structure_description \
-width 60 -with_10 6 -end \
-define C -combinatorial_objects \
-file_of_incidence_geometries \
-wreath_product_designs_n8_k6_inc.txt \
-16 3920 23520 \
-end \
-with C -do \
-combinatorial_object_activity \
-canonical_form \n
Section 15.5: Linear Codes

SECTION CANONICAL FORMS OF LINEAR CODES:

code_3.2.aut:

pdflatex 3.2_classification.tex
open 3.2_classification.pdf
16137 \> \> -end
16138 \> open 3_2_object0_TDA_flag_orbits_draw.bmp
16139
16140
16141
16142
16143 code_6_3_aut:
16144 \> $(ORBITER) -v 20 \\n16145 \> \> -define F -finite_field -q 2 -end \\n16146 \> \> -define genma -vector -field F -format 3 \\n16147 \> \> \> -compact $(CODE_N6_K3_Q2_GENMA) \\n16148 \> \> -end \\n16149 \> \> -define P -projective_space -n 2 -field F -v 0 -end \\n16150 \> \> -with P -do \\n16151 \> \> -projective_space_activity \\n16152 \> \> \> -canonical_form_of_code \\n16153 \> \> \> \> "6_3" genma -save_ago -label "6_3" \\n16154 \> \> \> \> -classification_prefix "6_3" \\n16155 \> \> \> \> -end \\n16156 \> \> -end \\n16157 \> pdflatex 6_3_classification.tex
16158 \> open 6_3_classification.pdf
16159 \> $(ORBITER) -v 2 -draw_matrix \\n16160 \> \> -input_csv_file 6_3_object0_TDA_flag_orbits.csv \\n16161 \> \> -secondary_input_csv_file 6_3_object0_TDA.csv \\n16162 \> \> -box_width 16 -bit_depth 24 \\n16163 \> \> -end
16164 \> open 6_3_object0_TDA_flag_orbits_draw.bmp
16165
16166 # group of order 24
16167
16168
16169 RM_3_1_group:
16170 \> $(ORBITER) -v 2 \\n16171 \> \> -define F -finite_field -q 2 -end \\n16172 \> \> -define genma -vector -field F -format 4 \\n16173 \> \> \> -compact $(CODE_RM_3_1_GENMA) \\n16174 \> \> -end \\n16175 \> \> -define P -projective_space -n 3 -field F -v 0 -end \\n16176 \> \> -with P -do \\n16177 \> \> -projective_space_activity \\n16178 \> \> \> -canonical_form_of_code \\n16179 \> \> \> \> "RM_3_1" genma -save_ago -label "RM_3_1" \\n16180 \> \> \> \> -classification_prefix "RM_3_1" \\n16181 \> \> \> \> -end \\n16182 \> \> \> -end
16183 \> pdflatex RM_3_1_classification.tex
16184▷open RM_3_1_classification.pdf
16185
16186# group order 1344
16187#RM_3_1_object0_INP_flag_orbits.csv
16188
16189RM_3_1_group_and_diagram:
16190▷$(ORBITER) -v 2 \n16191▷▷-define F -finite_field -q 2 -end \n16192▷▷-define genma -vector -field F -format 4 \n16193▷▷▷-compact $(CODE_RM_3_1_GENMA) \n16194▷▷-end \n16195▷▷-define P -projective_space -n 3 -field F -v 0 -end \n16196▷▷-with P -do \n16197▷▷-projective_space_activity \n16198▷▷▷-canonical_form_of_code \n16199▷▷▷▷"RM_3_1" genma -save_ago -label "RM_3_1" \n16200▷▷▷▷-classification_prefix "RM_3_1" \n16201▷▷▷-end \n16202▷▷-end
16203▷pdflatex RM_3_1_classification.tex
16204▷open RM_3_1_classification.pdf
16205▷$(ORBITER) -v 2 -draw_matrix \n16206▷▷-input_csv_file RM_3_1_object0_INP_flag_orbits.csv \n16207▷▷-secondary_input_csv_file RM_3_1_object0_INP.csv \n16208▷▷-box_width 16 -bit_depth 24 \n16209▷▷-end
16210▷$(ORBITER) -v 2 -draw_matrix \n16211▷▷-input_csv_file RM_3_1_object0_TDA_flag_orbits.csv \n16212▷▷-secondary_input_csv_file RM_3_1_object0_TDA.csv \n16213▷▷-box_width 16 -bit_depth 24 \n16214▷▷-end
16215▷open RM_3_1_object0_INP_flag_orbits_draw.bmp
16216▷open RM_3_1_object0_TDA_flag_orbits_draw.bmp
16217
16218
16219
16220RM_4_1_group:
16221▷$(ORBITER) -v 2 \n16222▷▷-define F -finite_field -q 2 -end \n16223▷▷-define genma -vector -field F -format 5 \n16224▷▷▷-compact $(CODE_RM_4_1_GENMA) \n16225▷▷-end \n16226▷▷-define P -projective_space -n 4 -field F -v 0 -end \n16227▷▷-with P -do \n16228▷▷-projective_space_activity \n16229▷▷▷-canonical_form_of_code \n16230▷▷▷▷"RM_4_1" genma -save_ago -label "RM_4_1" \n845
16231 \# -classification_prefix "RM_4_1" \\
16232 \# -end \\
16233 \# -end \\
16234 \# pdflatex RM_4_1.classification.tex \\
16235 \# open RM_4_1.classification.pdf \\
16236 $(ORBITER) -v 2 -draw_matrix \\
16237 \# -input_csv_file RM_4_1_object0_INP_flag_orbits.csv \\
16238 \# -secondary_input_csv_file RM_4_1_object0_INP.csv \\
16239 \# -box_width 16 -bit_depth 24 \\
16240 \# -end \\
16241 $(ORBITER) -v 2 -draw_matrix \\
16242 \# -input_csv_file RM_4_1_object0_TDA_flag_orbits.csv \\
16243 \# -secondary_input_csv_file RM_4_1_object0_TDA.csv \\
16244 \# -box_width 16 -bit_depth 24 \\
16245 \# -end \\
16246 \# open RM_4_1_object0_INP_flag_orbits_draw.bmp \\
16247 \# open RM_4_1_object0_TDA_flag_orbits_draw.bmp \\
16248 \\
16249 \# group order 322560 = 24*30*28*16 \\
16251 \\
16252 \# RS_6.4.7_group: \\
16253 $(ORBITER) -v 20 \\
16254 \# -define F -finite_field -q 7 -end \\
16255 \# -define genma -vector -field F -format 4 \\
16256 \# -compact $(CODE_RS_6.4.7) \\
16257 \# -end \\
16258 \# -define P -projective_space -n 3 -field F -v 0 -end \\
16259 \# -with P -do \\
16260 \# -projective_space_activity \\
16261 \# -canonical_form_of_code \\
16262 \# -classification_prefix "RS_6" \\
16263 \# -save_ago -label "RS_6" \\
16264 \# -end \\
16265 \# -end \\
16266 \# -end \\
16267 \\
16268 \# GV_n15_k6_d5_group: \\
16269 $(ORBITER) -v 20 \\
16270 \# -define F -finite_field -q 2 -end \\
16271 \# -define genma -vector -field F -format 6 \\
16272 \# -compact $(CODE_GV_n15_k6) \\
16273 \# -end \\
16274 \# -define P -projective_space -n 5 -field F -v 0 -end \\
16275 \# -with P -do \\
16276 \# -projective_space_activity \\
16277
code_n15_k6_d6_a_group:
\[$(\text{ORBITER})\ -v\ 20\\]
\>$\define F\ -\text{finite\ field}\ -q\ 2\ -\text{end}\\]
\>$\define genma\ -\text{vector}\ -\text{field}\ F\ -\text{format}\ 6\\]
\>$\compact $(\text{CODE}_{15\_6\_6\_A})\\]
\>$\end\\]
\>$\define P\ -\text{projective\ space}\ -n\ 5\ -\text{field}\ F\ -v\ 0\ -\text{end}\\]
\>$\with P\ -\text{do}\\]
\>$\text{projective\ space\ activity}\\]
\>$\compact $(\text{CODE}_{15\_6\_6\_B})\\]
\>$\end\\]
\>$\define P\ -\text{projective\ space}\ -n\ 5\ -\text{field}\ F\ -v\ 0\ -\text{end}\\]
\>$\with P\ -\text{do}\\]
\>$\text{projective\ space\ activity}\\]
\>$\compact $(\text{CODE}_{15\_6\_6\_B})\\]
\>$\end\\]
\>$\define P\ -\text{projective\ space}\ -n\ 5\ -\text{field}\ F\ -v\ 0\ -\text{end}\\]
\>$\with P\ -\text{do}\\]
\>$\text{canonical\ form\ of\ code}\\]
\>$\end\\]
\>$\define P\ -\text{projective\ space}\ -n\ 5\ -\text{field}\ F\ -v\ 0\ -\text{end}\\]
\>$\with P\ -\text{do}\\]
\>$\text{canonical\ form\ of\ code}\\]
\>$\end\\]
\>$\define P\ -\text{projective\ space}\ -n\ 5\ -\text{field}\ F\ -v\ 0\ -\text{end}\\]
\>$\with P\ -\text{do}\\]
\>$\text{canonical\ form\ of\ code}\\]
\>$\end\\]
\>$\define P\ -\text{projective\ space}\ -n\ 5\ -\text{field}\ F\ -v\ 0\ -\text{end}\\]
\>$\with P\ -\text{do}\\]
\>$\text{canonical\ form\ of\ code}\\]
\>$\end\\]
# Section 15.6: General Codes

SECTION_CANONICAL_FORMS_OF_GENERAL.CODES:

Hamming_graph_7_with_Hamming_code:

```bash
$(ORBITER) -v 2
define G -graph -Hamming 7 2
-subset "$\text{Hamming\_code}"
$(HAMMING\_CODE\_CODEWORDS) -end
-with G -do
-graph_theoretic_activity -export_csv -end
-with G -do
-graph_theoretic_activity -export_graphviz -end
-with G -do
-graph_theoretic_activity -save -end
-with G -do
-graph_theoretic_activity -automorphism_group -end
```

pdflatex Hamming_7_2_Hamming_code_report.tex
open Hamming_7_2_Hamming_code_report.pdf

# group of order 2688 = 16 * 168

# Section 15.7: Graphs

SECTION_CANONICAL_FORMS_OF_GRAPHS:

Cycle_13.aut:

```bash
$(ORBITER) -v 2
define Gamma -graph -cycle 13 -end
-with Gamma -do
-graph_theoretic_activity -automorphism_group -end
```

848
16372 inversion_graph:
16373 ▶ $(ORBITER) -v 6 \n16374 ▶ ▶ -define G -graph \n16375 ▶ ▶ ▶ -inversion_graph "1,0,2,3" \n16376 ▶ ▶ -end \n16377 ▶ ▶ -with G -do \n16378 ▶ ▶ ▶ -graph_theoretic_activity -properties \n16379 ▶ ▶ -end \n16380 ▶ ▶ -with G -do \n16381 ▶ ▶ ▶ -graph_theoretic_activity -automorphism_group \n16382 ▶ ▶ -end
16383
16384
16385
16386
16387 Chain_232_aut:
16388 ▶ $(ORBITER) -v 2 \n16389 ▶ ▶ -define P1 -vector -dense 2,3,2 -end \n16390 ▶ ▶ -define P2 -vector -dense 2,3,2 -end \n16391 ▶ ▶ -define Gamma -graph \n16392 ▶ ▶ ▶ -chain_graph P1 P2 \n16393 ▶ ▶ -end \n16394 ▶ ▶ -with Gamma -do \n16395 ▶ ▶ ▶ -graph_theoretic_activity -automorphism_group \n16396 ▶ ▶ -end
16397 ▶ pdflatex chain_graph_report.tex
16398 ▶ open chain_graph_report.pdf
16399
16400
16401
16402 JK_graph_pp16_1:
16403 ▶ $(ORBITER) -v 2 \n16404 ▶ ▶ -define Gamma -graph -load_dimacs \n16405 ▶ ▶ ../JUNTTILA_KASKI/benchmarks/pp/pp16-1 \n16406 ▶ ▶ -end \n16407 ▶ ▶ -with Gamma -do \n16408 ▶ ▶ ▶ -graph_theoretic_activity -save -end \n16409 ▶ ▶ ▶ -with Gamma -do \n16410 ▶ ▶ ▶ -graph_theoretic_activity -automorphism_group -end \n16411
16412 # go=34217164800
16413 #nauty_interface_with_group::create_automorphism_group_of_graph_with_partition_and_labeling: nb_backtrack1 = 6
16414 #nauty_interface_with_group::create_automorphism_group_of_graph_with_partition_and_labeling: nb_backtrack2 = 134
16415
16416 JK_graph_pp16_2:
16417 $\$(ORBITER) -v 2 \ 
16418 $\$ -define Gamma -graph -load_dimacs \ 
16419 $\$ -define Gamma -graph -load_dimacs \ 
16420 $\$ -end \ 
16421 $\$ -with Gamma -do \ 
16422 $\$ -with Gamma -do \ 
16423 $\$ -with Gamma -do \ 
16424 $\$ -with Gamma -do \ 
16425 # does not finish
16426 JK_graph_pp16_9:
16427 JK_graph_pp16_9:
16428 JK_graph_pp16_9:
16429 JK_graph_pp16_9:
16430 JK_graph_pp16_9:
16431 JK_graph_pp16_9:
16432 JK_graph_pp16_9:
16433 JK_graph_pp16_9:
16434 JK_graph_pp16_9:
16435 JK_graph_pp16_9:
16436 JK_graph_pp16_9:
16437 JK_graph_pp16_9:
16438 JK_graph_pp16_9:
16439 JK_graph_pp16_9:
16440 JK_graph_pp16_9:
16441 JK_graph_pp16_9:
16442 JK_graph_pp16_9:
16443 JK_graph_pp16_9:
16444 JK_graph_pp16_9:
16445 JK_graph_pp16_9:
16446 JK_graph_pp16_9:
16447 JK_graph_pp16_9:
16448 JK_graph_pp16_9:
16449 JK_graph_pp16_9:
16450 #nauty_interface_with_group::create_automorphism_group_of_graph_with_partition_an
16451 #nauty_interface_with_group::create_automorphism_group_of_graph_with_partition_an
16452 #Written file grid-w-3-3_group.makefile of size 579
16453 #User time: 0 of a second, dt=0 tps = 100
16454 #nb_calls_to_densenauty=1
16455
16456
16457 JK_graph_sts_13:
16458 JK_graph_sts_13:
16459 JK_graph_sts_13:
16460 JK_graph_sts_13:
16461 JK_graph_sts_13:
$ORBITER\ -v\ 6\ -define\ G\ -graph\ -load\ csv_no_border\ halljanko315.csv\ -end\ -with\ G\ -do\ -graph_theoretic_activity\ -automorphism_group\ -end\ $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\ HJ\ aut:\ $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\ -automorphism_group\ -end\ -with\ G\ -do\ -graph_theoretic_activity\ -automorphism_group\ -end\ -with\ G\ -do\ -group_theoretic_activity\ -poset_classification\ -depth\ 2\ -end\ or -orbits\ on\ subsets\ 2\ -report\ -end\ 851
HJ_orbital_graph_3:

\$ (ORBITER) -v 2 \\
\$ \$define gens -vector -file \\
\$ \$define G -permutation_group \\
\$ -bsgs halljanko315 "File\h halljanko315" \\
\$ \$define Gamma -graph \\
\$ \$with Gamma -do \\
\$ \$define Gamma -graph \\
\$ \$with Gamma -do \\
\$ \$with Gamma -do \\
\$ \$with Gamma -do \\
\$ pdflatex collinearity_graph
eigenvalues.tex

PGO_5.2_graph_group: 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 \\
\$ \$with Gamma -do \\
\$ \$with Gamma -do \\
\$ \$with Gamma -do \\
\$ \$with Gamma -do \\
\$ \$with Gamma -do \\
\$ pdflatex collinearity_graph_eigenvalues.tex
# Section 15.8: Quartic Curves

SECTION CANONICAL FORMS OF QUARTIC CURVES:

F_{17} edge:

Edge_curve_{17}_nauty:

Edge_curve_{17}_nauty:
16601 ▶ ▶ ▶ -report \ 
16602 ▶ ▶ ▶ ▶ -prefix Edge_curve_q17 \ 
16603 ▶ ▶ ▶ ▶ -export_flag_orbits \ 
16604 ▶ ▶ ▶ ▶ -show_TDO \ 
16605 ▶ ▶ ▶ ▶ -show_TDA \ 
16606 ▶ ▶ ▶ ▶ -dont_show_incidence_matrices \ 
16607 ▶ ▶ ▶ ▶ -export_group \ 
16608 ▶ ▶ ▶ -end \ 
16609 ▶ ▶ -end
16610 ▶ pdflatex Edge_curve_q17_classification.tex
16611 ▶ open Edge_curve_q17_classification.pdf
16612 ▶ $(ORBITER) -v 2 -draw_matrix \ 
16613 ▶ ▶ -input_csv_file Edge_curve_q17_object0_TDA_flag_orbits.csv \ 
16614 ▶ ▶ -secondary_input_csv_file Edge_curve_q17_object0_TDA.csv \ 
16615 ▶ ▶ -box_width 4 -bit_depth 24 \ 
16616 ▶ ▶ -end
16617 ▶ open Edge_curve_q17_object0_TDA_flag_orbits_draw.bmp
16618
16619 # 9 backtrack nodes total
16620
16621
16622 # aut = 24
16623 # User time: 0.04 of a second, dt=4 tps = 100
16624
16625
16626 # generators for a group of order 24:
16627 #1,0,0,0,13,0,0,0,4,
16628 #1,0,0,0,0,16,0,16,0,
16629 #0,1,16,2,4,4,15,4,4,
16630
16631
16632 Edge_curve_17_group:
16633 ▶ $(ORBITER) -v 3 \ 
16634 ▶ ▶ -define G -linear_group -PGL 3 17 \ 
16635 ▶ ▶ -subgroup_by_generators "Stab_Edge" "24" 3 \ 
16636 ▶ ▶ ▶ "1,0,0,0,13,0,0,0,4" \ 
16637 ▶ ▶ ▶ "1,0,0,0,0,16,0,16,0" \ 
16638 ▶ ▶ ▶ "0,1,16,2,4,4,15,4,4" \ 
16639 ▶ ▶ -end \ 
16640 ▶ ▶ -with G -do \ 
16641 ▶ ▶ -group_theoretic_activities \ 
16642 ▶ ▶ -print_elements_tex \ 
16643 ▶ ▶ -group_table \ 
16644 ▶ ▶ -report \ 
16645 ▶ ▶ -end
16646 ▶ pdflatex PGL_3_17_Subgroup_Stab_Edge_24_report.tex
16647 ▶ open PGL_3_17_Subgroup_Stab_Edge_24_report.pdf
# Chapter 16 - Interfaces

# Section 16.1: Graphical Output:

SECTION GRAPHICAL_OUTPUT:

$$(ORBITER) -v 3 \$

$$(ORBITER) -define F -finite_field -q 7 -end \$

$$(ORBITER) -with F -do -finite_field_activity \$

$$(ORBITER) -cheat_sheet_GF \$

$$(ORBITER) -v 2 \$

$$(ORBITER) -define F -finite_field -q 4 -end \$

$$(ORBITER) -define P -projective_space -n 2 -field F -v 0 -end \$

$$(ORBITER) -with P -do -projective_space_activity \$

$$(ORBITER) -cheat_sheet_for_decomposition_by_element_PG \$

$$(ORBITER) -list_arguments \$

$$(ORBITER) -define R -vector -repeat 1 21 -end \$

$$(ORBITER) -define C -vector -repeat 1 21 -end \$

$$(ORBITER) -draw_matrix \$

$$(ORBITER) -input_csv_file PG_2_4_singer_incma_cyclic.csv \$

open GF_q7_addition_table_draw.bmp
open PG_2_4_singer_incma_cyclic_draw.bmp

PGL_4_2_Wedge_4_0_graphical_output:

$(ORBITER) -v 4 \
define G -linear_group -PGL 4 2 \
end \
with G -do \
group_theoretic_activity \
ся report \
end \
pdflatex PGL_4_2_Wedge_4_2_detached_report.tex 
open PGL_4_2_Wedge_4_2_detached_report.pdf

schreier_tree_graphical_output:

$(ORBITER) -v 4 \
draw_options \
yout 500000 \
-radius 15 -nodes_empty \
-line_width 0.5 -y_stretch 0.25 \
end \
define G -linear_group -PGL 4 2 -end \
with G -do \
group_theoretic_activity \
orbits_on_polynomials 3 \
orbits_on_polynomials_draw_tree 6 \
end 
pdflatex poly_orbits_d3_n3_q2.tex 
open poly_orbits_d3_n3_q2.pdf

Queens_graph:

$(ORBITER) -v 2 \
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 \

856
The Povray Interface

SECTION_POVRAY:

cube:

$\text{ORBITER} -v 2 -povray \
-\text{round 0 -nb\_frames\_default 30} \ 
-\text{output\_mask cube\_\%d\_\%03d.pov} \ 
-\text{video\_options -W 1024 -H 768} \ 
-\text{global\_picture\_scale 0.5} \ 
-\text{default\_angle 75} \ 
-\text{clipping\_radius 2.7} \ 
-\text{end} \ 

-\text{scene\_objects} \ 
-\text{obj\_file cube\_centered.obj} \ 
-\text{edge "0, 1"} \ 
-\text{edge "0, 2"} \ 
-\text{edge "0, 4"} \ 
-\text{edge "1, 3"} \ 
-\text{edge "1, 5"} \ 
-\text{edge "2, 3"} \ 
-\text{edge "2, 6"} \ 
-\text{edge "3, 7"} \ 
-\text{edge "4, 5"} \ 
-\text{edge "4, 6"} \ 
-\text{edge "5, 7"} \ 
-\text{edge "6, 7"} \ 
-\text{group\_of\_things\_as\_interval 0 8} \ 
-\text{spheres 0 0.3 $(POLISHED\_CHROME\_WHITE)} \ 
-\text{group\_of\_things\_as\_interval 0 6} \ 
-\text{prisms 1 0.05 $(YELLOW\_TRANSPARENT)} \ 
-\text{group\_of\_things\_as\_interval 0 12} \ 
-\text{cylinders 2 0.15 $(COLOR\_RED)} \ 

math261_test:

$(ORBITER) -v 2 -povray -round 0 -nb_frames_default 30 -output_mask math261%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%d03d.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

-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" -planes 0 $(COLOR_BLUE_SEE 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.*.pov POV mv makefile_animation POV

plane2:

$(ORBITER) -v 2 -povray
round 0 -nb_frames_default 30 \
-output_mask plane2_%d_%03d.pov \n-video_options -W 2560 -H 1920 \n-global_picture_scale 0.40 \n-default_angle 75 \n-clipping_radius 5 -omit_bottom_plane \n-camera 0 "0,0,1" "6,6,2" "0,0,0" \n-rotate_about_z_axis \n-boundary_box \n-end \n-scene_objects \n-line_through_two_points_recentered_from_csv_file coordinate_grid.csv \n-plane_by_dual_coordinates "0,0,1,0" \n-plane_by_dual_coordinates "0,1,0,0" \n-plane_by_dual_coordinates "1,0,0,0" \n-plane_by_dual_coordinates "4,5,-1,9" \n-group_of_things "0" \n-group_of_things "1" \n-group_of_things "2" \n-group_of_things_as_interval 3 39 \n-lines 0 0.15 $(COLOR_RED_SHINY) \n-lines 1 0.15 $(COLOR_GREEN_SHINY) \n-lines 2 0.15 $(COLOR_BLUE_SHINY) \n-lines 3 0.05 $(COLOR_BLACK_SHINY) \n-group_of_things "0" \n-group_of_things "3" \n-scene_objects_end \n-povray_end \n-rm -rf POV \n-mkdir POV \n-mv plane2_0_*.pov POV \n-mv makefile_animation POV

 analytic_geo_1:
$(ORBITER) -v 2 -povray \n-round 0 -nb_frames_default 30 \n-output_mask analytic_geo_1_%d_%03d.pov \n-video_options -W 2560 -H 1920 \n-global_picture_scale 0.80 \n-default_angle 75 \n-clipping_radius 5 -omit_bottom_plane 

planes 5 "texture{ pigment{ color Yellow transmit 0.5 } finish { diffuse 0.9 phong 1}}" \n
analytic_geo_1:
-camera 0 "0,0,1" "6,6,2" "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"
-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"
-planes 4 $(COLOR_BLUESEE_THROUGH)
-planes 5 $(COLOR_BLUESEE_THROUGH)
-planes 6 $(COLOR_REDSSEE_THROUGH)
-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)
-spheres 8 0.10 $(COLOR_YELLOW_SHINY)
-group_of_things "87"
-group_of_things "0,1,2"
-group_of_things "3"
-cylinders 9 0.075 $(POLISHED_CHROME_WHITE)
-cylinders 10 0.075 $(COLOR_YELLOW_SHINY)
-scene_objects_end
-povray_end
analytic_geo_1_video:
- rm -r FRAMES
- mkdir FRAMES
- rm analytic_geo_1.mp4
- $(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 \\  
- -end \\  
- 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 
group_of_things_as_interval 6 6 
group_of_things_as_interval_with_exceptions 12 15 
> "14,19,23" 
-lines 1 0.02 $(COLOR_RED_SHINY) 
-lines 2 0.02 $(COLOR_BLUE_SHINY) 
-lines 3 0.02 $(COLOR_YELLOW_SHINY) 
-label 0 "a1" 
-label 2 "a2" 
-label 4 "a3" 
-label 6 "a4" 
-label 8 "a5" 
-label 10 "a6" 
-label 12 "b1" 
-label 14 "b2" 
-label 16 "b3" 
-label 18 "b4" 
-label 20 "b5" 
-label 22 "b6" 
-label 24 "c12" 
-label 26 "c13" 
-label 30 "c15" 
-label 32 "c16" 
-label 34 "c23" 
-label 36 "c24" 
-label 40 "c26" 
-label 42 "c34" 
-label 44 "c35" 
-label 48 "c45" 
-label 50 "c46" 
-label 52 "c56" 
group_of_things_as_interval 0 6 
-labels 4 0.2 0.15 $(COLOR_BLACK_NO_SHADOW) 
group_of_things_as_interval 6 6 
-labels 5 0.2 0.15 $(COLOR_BLACK_NO_SHADOW) 
group_of_things_as_interval 12 12 
-labels 6 0.2 0.15 $(COLOR_BLACK_NO_SHADOW) 
-scene_objects_end 
-povray_end
17070 ⊲ - rm -rf POV
17071 ⊲ mkdir POV
17072 ⊲ mv Eckardt_0*.pov POV
17073 ⊲ mv makefile_animation POV
17074
17075
17076 "-3,2.333,4" * 1.5 = "-4.5,3.5,6"
17077 #M := Matrix([[[-4.5, 3.5, 6], [1, 1, 1]]])
17078 #NullSpace(M)
17079 #=0.186080731891197,-0.781539073943026,0.595458342051830
17080 #> -rotate_about_custom_axis "0.186080731891197,-0.781539073943026,0.595458342051830" 51830 \ 
17081 #-W 1024 -H 768
17082 #-W 2560 -H 1920
17083 #-W 4096 -H 3072
17084
17085
17086
17087
17088 Eckardt_deform:
17089 ⊲ $(ORBITER) -v 2 -povray \ 
17090 ⊲ ⊲ -round 0 -nb_frames_default 93 \ 
17091 ⊲ ⊲ -output_mask Eckardt_deform_%d_%03d.pov \ 
17092 ⊲ ⊲ -video_options -W 1024 -H 768 \ 
17093 ⊲ ⊲ -global_picture_scale 0.9 \ 
17094 ⊲ ⊲ -default_angle 75 \ 
17095 ⊲ ⊲ -clipping_radius 2.4 \ 
17096 ⊲ ⊲ -camera 0 "1,1,1" "-3,1,3" "0.12,0.12,0.12" \ 
17097 ⊲ ⊲ -end \ 
17098 ⊲ ⊲ -scene_objects \ 
17099 ⊲ ⊲ ⊲ -Hilbert_Cohn_Vossen_surface \ 
17100 ⊲ ⊲ ⊲ -group_of_things "0" \ 
17101 ⊲ ⊲ ⊲ -deformation_of_cubic_lex 93 1.107148718 1.570796327 0 \ 
17102 ⊲ ⊲ ⊲ ⊲ $(ECKARDT_CUBIC_DEFORM1_LEX) \ 
17103 ⊲ ⊲ ⊲ ⊲ $(ECKARDT_CUBIC_DEFORM2_LEX) \ 
17104 ⊲ ⊲ ⊲ -group_of_things_as_interval 0 93 \ 
17105 ⊲ ⊲ ⊲ -group_is_animated 1 \ 
17106 ⊲ ⊲ ⊲ -cubics 1 $(SURFACE_COLOR_SEETHROUGH) \ 
17107 ⊲ ⊲ ⊲ -scene_objects_end \ 
17108 ⊲ ⊲ -povray_end
17109 ⊲ - rm -rf POV
17110 ⊲ mkdir POV
17111 ⊲ mv Eckardt_deform_0*.pov POV
17112 ⊲ mv makefile_animation POV
17113
17114
17115 864
Eckardt_deform_2:

$\text{(ORBITER) -v 2 -povray \}
\text{-round 0 -nb_frames_default 30 \}
\text{-output_mask Eckardt_deform\%d\%03d.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{-scene_objects \}
\text{-Hilbert_Cohn_Vossen_surface \}
\text{-group_of_things "0" \}
\text{-deformation_of_cubic_lex 93 1.107148718 1.570796327 0 \}
\text{-group_of_things as_interval 0 93 \}
\text{-group_is_animated 1 \}
\text{-group_of_things "0" \}
\text{-cubics 1 $\text{(SURFACE\_COLOR\_SEETHROUGH)} \}
\text{-group_of_things "24" \}
\text{-cubics 2 $(COLOR\_RED) \}
\text{-group_of_things "70" \}
\text{-cubics 3 $(COLOR\_BLUE) \}
\text{-scene_objects_end \}
\text{-povray_end \}
\text{-rm -rf POV \}
\text{mkdir POV \}
\text{mv Eckardt_deform_0\_*\_pov POV \}
\text{mv makefile_animation POV \}

Clebsch:

$\text{(ORBITER) -v 2 -povray \}
\text{-round 0 -nb_frames_default 30 \}
\text{-output_mask Clebsch\%d\%03d.pov \}
\text{-video_options -W 1024 -H 768 \}
\text{-global_picture_scale 0.9 \}
\text{-default_angle 80 \}
\text{-clipping_radius 2.4 \}
\text{-camera 0 "1,1,1" ":-4.5,3.5,6" ":0,0,0" \}
\text{-end \}
\text{-scene_objects \}
\text{-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
-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
-endrass8:
- $(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"
-oitics 4 $(SURFACE_COLOR_SEETHROUGH)
-planes 5 "texture{ pigment{ color Blue transmit 0.5 } \\ 17211 finish { diffuse 0.9 phong 1}}" \\
-scene_objects_end \\
-povray_end \\
- rm -rf POV \\
-mkdir POV \\
mv endrass_oectic_0.*.pov POV \\
mv makefile_animation POV \\

MESSAGE

# Section 16.3: Creating Animations

SECTION_ANIMATIONS:

dode:

$(ORBITER) -v 2 \\
-povray \\
-round 0 -nb_frames_default 30 \\
-output_mask dode_%d_%03d.pov \\
-video_options -W 1024 -H 768 \\
-global_picture_scale 0.50 \\
-default_angle 45 \\
-clipping_radius 5 \\
-camera 0 "1,1,1" "-2,2,4" "0,0,0" \\
-rotate_about_111 \\
-end \\
-scene_objects \\
-dodecahedron \\
-group_of_things_as_interval 0 20 \\
-spheres 0 0.075 $(POLISHED_CHROME_WHITE) \\
-group_of_things_as_interval 0 30\n
cylinders 1 0.05 $(COLOR_RED_SHINY) \\
-group_of_things_as_interval 0 12\n
cylinders 2 0.02 $(YELLOW_TRANSPARENT) \\
-scene_objects_end \\
povray_end \\
- rm -rf POV \\
mkdir POV \\
mv dode_0.*.pov POV \\
mv makefile_animation POV
17257
dode_video:
17259  ▷ - rm -r FRAMES
17260  ▷ - mkdir FRAMES
17261  ▷ - rm dode.mp4
17262  ▷ $(ORBITER) \ 
17263  ▷  ▷ -prepare_frames \ 
17264  ▷  ▷  ▷ -i 0 30 DODE/dode_0%03d.png \ 
17265  ▷  ▷  ▷ -output_starts_at 0 \ 
17266  ▷  ▷  ▷ -o FRAMES/frame%04d.png \ 
17267  ▷  ▷ -end
17268  ▷ ffmpeg -r 5 -f image2 -i FRAMES/frame%04d.png \ 
17269  ▷  ▷ -f mp4 -q:v 0 -vcodec mpeg4 dode.mp4
17270
17271
17272 monkey_video:
17273  ▷ - rm -r FRAMES
17274  ▷ - mkdir FRAMES
17275  ▷ - rm monkey.mp4
17276  ▷ $(ORBITER) \ 
17277  ▷  ▷ -prepare_frames \ 
17278  ▷  ▷  ▷ -i 0 30 monkey_0%03d.png \ 
17279  ▷  ▷  ▷ -output_starts_at 0 \ 
17280  ▷  ▷  ▷ -o FRAMES/frame%04d.png \ 
17281  ▷  ▷ -end
17282  ▷ ffmpeg -r 5 -f image2 -i FRAMES/frame%04d.png \ 
17283  ▷  ▷ -f mp4 -q:v 0 -vcodec mpeg4 monkey.mp4
17284
17285 Eckardt_deform_video:
17286  ▷ - rm -r FRAMES
17287  ▷ - mkdir FRAMES
17288  ▷ - rm Eckardt_deform.mp4
17289  ▷ $(ORBITER) \ 
17290  ▷  ▷ -prepare_frames \ 
17291  ▷  ▷  ▷ -i 0 93 Eckardt_deform_0/Eckardt_deform_0%03d.png \ 
17292  ▷  ▷  ▷ -output_starts_at 0 \ 
17293  ▷  ▷  ▷ -o FRAMES/frame%04d.png \ 
17294  ▷  ▷ -end
17295  ▷ ffmpeg -r 5 -f image2 -i FRAMES/frame%04d.png \ 
17296  ▷  ▷ -f mp4 -q:v 0 -vcodec mpeg4 Eckardt_deform.mp4
17297
17298
17299 Eckardt_surface:
17300  ▷ $(ORBITER) \ -v 2 -povray \ 
17301  ▷  ▷ -round 0 -nb_frames_default 30 \ 
17302  ▷  ▷ -output_mask Eckardt.%d%03d.pov \ 
17303  ▷  ▷ -video_options -W 1024 -H 768 \ 

868
17304  ▶  ▶  -global_picture_scale 0.9 \
17305  ▶  ▶  -default_angle 75 \
17306  ▶  ▶  -clipping_radius 2.4 \
17307  ▶  ▶  -camera 0 "1,1,1" "-3,1,3" "0.12,0.12,0.12" \
17308  ▶  ▶  -end \
17309  ▶  ▶  -scene_objects \
17310  ▶  ▶  ▶  -cubic_Goursat "6,3,-15" \
17311  ▶  ▶  ▶  -group_of_things "0" \
17312  ▶  ▶  ▶  -cubics 0 $(SURFACE_COLOR_SEETHROUGH) \
17313  ▶  ▶  ▶  -scene_objects_end \
17314  ▶  ▶  -povray_end 
17315  ▶  ▶  - rm -rf POV
17316  ▶  ▶  mkdir POV
17317  ▶  ▶  mv Eckardt_0_*.pov POV
17318  ▶  ▶  mv makefile_animation POV
17319
17320
17321
17322  Kummer_surface:
17323  ▶  ▶  $(ORBITER) -v 2 -povray \
17324  ▶  ▶  ▶  -round 0 -nb_frames_default 30 \
17325  ▶  ▶  ▶  -output_mask Kummer_%d_%03d.pov \
17326  ▶  ▶  ▶  -video_options -W 1024 -H 768 \
17327  ▶  ▶  ▶  -global_picture_scale 0.9 \
17328  ▶  ▶  ▶  -default_angle 75 \
17329  ▶  ▶  ▶  -clipping_radius 2.4 \
17330  ▶  ▶  ▶  -camera 0 "1,1,1" "-3,1,3" "0.12,0.12,0.12" \
17331  ▶  ▶  ▶  -end \
17332  ▶  ▶  -scene_objects \
17333  ▶  ▶  ▶  ▶  -quartic_lex_35 $(KUMMER_QUARTIC_LEX_35) \
17334  ▶  ▶  ▶  ▶  -group_of_things "0" \
17335  ▶  ▶  ▶  ▶  -quartics 0 $(SURFACE_COLOR_SEETHROUGH) \
17336  ▶  ▶  ▶  ▶  -scene_objects_end \
17337  ▶  ▶  ▶  -povray_end 
17338  ▶  ▶  - rm -rf POV
17339  ▶  ▶  mkdir POV
17340  ▶  ▶  mv Kummer_0_*.pov POV
17341  ▶  ▶  mv makefile_animation POV
17342
17343
17344  # Maple:
17345  #Kummer := expand((x0^2 + x1^2 + x2^2 + x3^2)^2 - 3*(x0^4 + x1^4 + x2^4 + x3^4))
17346
17347
17348  Kummer_video:
17349  ▶  ▶  - rm -r FRAMES
17350  ▶  ▶  - mkdir FRAMES

869
Beauville surface:

```
17363 Beauville_surface:
17364 $(ORBITER) -v 2 -povray \
17365 -round 0 -nb_frames_default 30 \n17366 -output_mask Beauville_%d_03d.pov \n17367 -video_options -W 1024 -H 768 \n17368 -global_picture_scale 0.3 \n17369 -default_angle 75 \n17370 -clipping_radius 2.4 \n17371 -camera 0 "1,1,1" "-3,1,3" "0.12,0.12,0.12" \n17372 -end \n17373 -scene_objects \n17374 -quintic_lex_56 $(BEAUVILLE_QUINTIC_LEX_56) \n17375 -group_of_things "0" \n17376 -quintics 0 $(SURFACE_COLOR_SEETHROUGH) \n17377 -scene_objects_end \n17378 -povray_end
17379 - rm -rf POV
17380 mkdir POV
17381 mv Beauville_0_*.pov POV
17382 mv makefile_animation POV
17383
17384
17385
17386
17387
17388
17389
17390 # Clebsch map up for surface created using arc lifting
17391 # We take a circle of radius r centered at the origin in the affine real plane
17392 # and map it up on the surface.
17393 # The Clebsch surface has
17394 # a = d = 2.618033988 = (3+sqrt(5))/2
17395 # b = c = 1.618033988 = (1+sqrt(5))/2
17396 #
17397 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, 0],
[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
T20=4.2360680044124255
T21=-4.2360680044124255
T22=-4.2360680044124255
T23=0.
T30=1.6180340022062127
17441
17442 T31=-2.6180340022062127
17443
17444 T32=-1.6180340022062127
17445
17446 T33=0.
17447
17448
17449 CLEBSCH_CUBICS=\n17450 △ △ △ △ push b push b push d push c push m push c push m add mult add mult \n17451 △ △ △ △ push b push c push d push d push m mult mult add mult add mult \n17452 △ △ △ △ push a push d push d push m add mult add mult add mult
17453 △ △ △ △ push a push c push m mult add mult \n17454 △ △ △ △ store c001 \n17455 △ △ △ △ push b push d mul \n17456 △ △ △ △ push b push 1 push m push c mult add mult \n17457 △ △ △ △ push d push a push 1 push m mult add mult add mult \n17458 △ △ △ △ push m push a mult add push c add \n17459 △ △ △ △ push c push m push a mult add \n17460 △ △ △ △ mult mult \n17461 △ △ △ △ store c002 \n17462 △ △ △ △ push b \n17463 △ △ △ △ push d push c push a push m mult add mult \n17464 △ △ △ △ push c push a push m push 1 mult add mult add mult \n17465 △ △ △ △ push a push d mult push c push 1 push m mult add mult \n17466 △ △ △ △ push m mult add \n17467 △ △ △ △ push a push c push m mult add mult \n17468 △ △ △ △ store c011 \n17469 △ △ △ △ push b push b push c mult mult \n17470 △ △ △ △ push 1 push d push m mult add mult \n17471 △ △ △ △ push a push b mult push c push d push m mult mult add mult \n17472 △ △ △ △ push m mult add \n17473 △ △ △ △ push a push d mult push c push d push m mult add mult add mult \n17474 △ △ △ △ push a push c push m mult add mult \n17475 △ △ △ △ store c012 \n17476 △ △ △ △ push m \n17477 △ △ △ △ push b push d push m mult add push c mult \n17478 △ △ △ △ push d push b push 1 push m mult add mult push a mult \n17479 △ △ △ △ push b push c mult push d push 1 push m mult add mult add mult \n17480 △ △ △ △ push b push d push m mult add mult \n17481 △ △ △ △ store d001 \n17482 △ △ △ △ push m \n17483 △ △ △ △ push d push c push m mult add push a push a mult mult \n17484 △ △ △ △ push c push c mult push d push m mult add push a mult add \n17485 △ △ △ △ push m push b push c mult mult push c push 1 push m mult add mult add mult \n17486 △ △ △ △ push b push d push m mult add mult \n
872
store d011
push m
push c push d mult push d push m mult add push a push a mult mult
push c push c 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 m mult mult add
push b push d push m mult add mult 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 d112
push m
push b push d push m mult add push c mult 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
push b push d push m mult add mult add
store m002
push m
push c push d push m mult add push a push a mult mult
push c 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
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 add
push m push b push d push c push m mult add push c 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
push a push c mult mult
push b push d push m mult add mult
store m122
push m push a mult push c add push d mult push c push a push 1 push m mult add mult add
push b mult
push m push a push d mult mult push c push 1 push m mult add mult add
push b push d push m mult add mult
store n002
push m
17548 push c push d push m mult add push b mult push m push d push c push 1 push h m mult add mult mult add \n17549 push a mult \n17550 push b push c mult push d push 1 push m mult add mult add mult \n17551 push a push b push c push m mult push b push c mult mult add mult mult \n17552 store n122 \n17553 clebsch_up_create_points:
17554 $(ORBITER) -v 2 \n17555 -smooth_curve "Clebsch_map_of_circle_to_defining_eqn_r2" \n17556 -const a $(CLEBSCH_A) b $(CLEBSCH_B) c $(CLEBSCH_C) d $(CLEBSCH_D) \n17557 t00 $(T00) t01 $(T01) t02 $(T02) t03 $(T03) \n17558 t10 $(T10) t11 $(T11) t12 $(T12) t13 $(T13) \n17559 t20 $(T20) t21 $(T21) t22 $(T22) t23 $(T23) \n17560 t30 $(T30) t31 $(T31) t32 $(T32) t33 $(T33) \n17561 r 2 one 1 m -1 \n17562 -const_end \n17563 -var t \n17564 c001 c002 c011 c012 \n17565 d001 d011 d012 d112 \n17566 m002 m012 m022 m122 \n17567 n002 n012 n112 n022 n122 \n17568 y0 y1 y2 \n17569 y001 y002 y011 y012 y022 y112 y122 \n17570 x0 x1 x2 x3 \n17571 -var_end \n874
\(\text{-code} \)
\begin{verbatim}
17574 \# push t cos push r mult store y0 \\
17575 \# push t sin push r mult store y1 \\
17576 \# push one store y2 \\
17577 \# push y0 push y0 push y1 mult mult store y001 \\
17578 \# push y0 push y0 push y2 mult mult store y002 \\
17579 \# push y0 push y1 push y1 mult mult store y011 \\
17580 \# push y0 push y1 push y2 mult mult store y012 \\
17581 \# push y0 push y2 push y2 mult mult store y022 \\
17582 \# push y1 push y1 push y2 mult mult store y112 \\
17583 \# push y1 push y2 push y2 mult mult store y122 \\
17584 \# $\text{CLEBSCH\_CUBICS}$ \\
17585 \# push c001 push y001 mult \\
17586 \# push c002 push y002 mult add \\
17587 \# push c011 push y011 mult add \\
17588 \# push c012 push y012 mult add \\
17589 \# store x0 \\
17590 \# push d001 push y001 mult \\
17591 \# push d011 push y011 mult add \\
17592 \# push d012 push y012 mult add \\
17593 \# push d112 push y112 mult add \\
17594 \# store x1 \\
17595 \# push m002 push y002 mult \\
17596 \# push m012 push y012 mult add \\
17597 \# push m022 push y022 mult add \\
17598 \# push m122 push y122 mult add \\
17599 \# store x2 \\
17600 \# push n002 push y002 mult \\
17601 \# push n012 push y012 mult add \\
17602 \# push n022 push y022 mult add \\
17603 \# push n112 push y112 mult add \\
17604 \# push n122 push y122 mult add \\
17605 \# store x3 \\
17606 \# push x0 push t00 mult \\
17607 \# push x1 push t10 mult add \\
17608 \# push x2 push t20 mult add \\
17609 \# push x3 push t30 mult add \\
17610 \# return \\
17611 \# push x0 push t01 mult \\
17612 \# push x1 push t11 mult add \\
17613 \# push x2 push t21 mult add \\
17614 \# push x3 push t31 mult add \\
17615 \# return \\
17616 \# push x0 push t02 mult \\
17617 \# push x1 push t12 mult add \\
17618 \# push x2 push t22 mult add \\
17619 \# push x3 push t32 mult add \\
\end{verbatim}

875
17621  ▷ ▷ ▷ ▷ return \  
17622  ▷ ▷ ▷ ▷ push x0 push t03 mult \  
17623  ▷ ▷ ▷ ▷ push x1 push t13 mult add \  
17624  ▷ ▷ ▷ ▷ push x2 push t23 mult add \  
17625  ▷ ▷ ▷ ▷ push x3 push t33 mult add \  
17626  ▷ ▷ ▷ ▷ return \  
17627  ▷ ▷ -code_end  
17628  
17629  17630 Clebsch_surface:  
17631  ▷ $(ORBITER) -v 2 -povray \  
17632  ▷  ▷ -round 0 -nb_frames_default 30 \  
17633  ▷  ▷ -output_mask Clebsch_%d_%03d.pov \  
17634  ▷  ▷ -video_options -W 1024 -H 768 \  
17635  ▷  ▷ -global_picture_scale 0.9 \  
17636  ▷  ▷ -default_angle 75 \  
17637  ▷  ▷ -clipping_radius 2.4 \  
17638  ▷  ▷ -camera 0 "1,1,1" "-3,1,3" "0.12,0.12,0.12" \  
17639  ▷  ▷ -end \  
17640  ▷  ▷ -scene_objects \  
17641  ▷  ▷  ▷ -cubic_orbiter "0,0,0,0,0,-4.236067972,\  
17642  ▷  ▷  ▷  0,0,4.236067972,4.236067972,17.94427188,\  
17643  ▷  ▷  ▷  -17.94427188,0,0,- 9.472135941,0,0,5.236067971,\  
17644  ▷  ▷  ▷  8.472135938,- 27.41640782" \  
17645  ▷  ▷  ▷ -group_of_things "0" \  
17646  ▷  ▷  ▷ -cubics 0 $(SURFACE_COLOR_SEETHROUGH) \  
17647  ▷  ▷  ▷ -point_list_from_csv_file \  
17648  ▷  ▷  ▷ function_Clebsch_map_of_circle_N1000_points.csv \  
17649  ▷  ▷  ▷ -group_of_things_as_interval 0 954 \  
17650  ▷  ▷  ▷ -spheres 1 0.07 $(COLOR_RED) \  
17651  ▷  ▷  ▷ -scene_objects_end \  
17652  ▷  ▷ -povray_end  
17653  ▷  - rm -rf POV  
17654  ▷  mkdir POV  
17655  ▷  mv Clebsch_0*.pov POV  
17656  ▷  mv makefile_animation POV  
17657  
17658  17659 Clebsch_surface_defining_equation:  
17660  ▷ $(ORBITER) -v 2 -povray \  
17661  ▷  ▷ -round 0 -nb_frames_default 30 \  
17662  ▷  ▷ -output_mask Clebsch_%d_%03d.pov \  
17663  ▷  ▷ -video_options -W 1024 -H 768 \  
17664  ▷  ▷ -global_picture_scale 0.6 \  
17665  ▷  ▷ -default_angle 75 \  
17666  ▷  ▷ -clipping_radius 1.6 \  
17667  ▷  ▷ -camera 0 "1,1,1" "-2,0,2" "0,0,0" \  

876
Clebsch_surface_defining_equation_and_curves:

Clebsch_surface_defining_equation_and_curves:

Clebsch_surface_defining_equation_and_curves:

Clebsch_surface_defining_equation_and_curves:

Clebsch_surface_defining_equation_and_curves:

Clebsch_surface_defining_equation_and_curves:

Clebsch_surface_defining_equation_and_curves:

Clebsch_surface_defining_equation_and_curves:

Clebsch_surface_defining_equation_and_curves:

Clebsch_surface_defining_equation_and_curves:

Clebsch_surface_defining_equation_and_curves:

Clebsch_surface_defining_equation_and_curves:

Clebsch_surface_defining_equation_and_curves:

Clebsch_surface_defining_equation_and_curves:

Clebsch_surface_defining_equation_and_curves:
spheres 1 0.07 "texture{ pigment{ color Red } finish { diffuse 0.9 phong 1}}" 

F7_povray: $\langle\text{ORBITER}\rangle -v 2 -povray \ 
\langle\text{round 0 -nb_frames_default 30} \ 
\langle\text{-output_mask F7.15_lines_%03d.pov} \ 
\langle\text{-video_options -W 1024 -H 768} \ 
\langle\text{-global_picture_scale 1.5} \ 
\langle\text{-default_angle 80} \ 
\langle\text{-clipping_radius 4.4} \ 
\langle\text{-omit_bottom_plane} \ 
\langle\text{-camera 0 "1,1,1" ",4.5,3.5,6" ",0,0,0"} \ 
\langle\text{-end} \ 
\langle\text{-scene_objects} \ 
\langle\text{-cubiclex "0, 0, 6, 0, 0, -13.39014946, -3.341901346, -6.972931640, 5.827182718, 0, 0, 7.390149464, 7.390149464, 6.972931640, -1.512349728, -8.485281372, 0, 0, 0"} \ 
\langle\text{-group_of_things "0"} \ 
\langle\text{-cubics 0 $(\text{SURFACE\_COLOR\_SEETHROUGH})} \ 
\langle\text{-line_through_point_with_direction "0, 0, 0, 1, 0, 0"} \ 
\langle\text{-line_through_point_with_direction "0, 0, -1, 0, 1, 0"} \ 
\langle\text{-line_through_point_with_direction "0, 0, 0, 0, -1"} \ 
\langle\text{-line_through_point_with_direction "1, 0, 0, 1, 1"} \ 
\langle\text{-line_through_point_with_direction ",1.414213562, 0, 0, 4.146264370, 1.7320050808, 1.732050808"} \ 
\langle\text{-line_through_point_with_direction "0, 1.732050808, -1, 2.414213562, -0.317837246, 2.414213562"} \ 
\langle\text{-line_through_point_with_direction ",2.133352390, 0, -1, 1.674708020, 1, 0"} \ 
\langle\text{-line_through_point_with_direction ",2.539058015, 0, 2.211360755, 1, 0"} \ 
\langle\text{-line_through_point_with_direction "0, 1.148188060, 0, 0, -0.9435440612, 1"} \ 
\langle\text{-line_through_point_with_direction ",0.9711971171, 0, 0, 1.162155272, 0, 1"} \ 
\langle\text{-line_through_point_with_direction "2.096037870, 2.096037870, 0, -1.065851905, 1.722456585, -1.722456585, 1"} \ 
\langle\text{-line_through_point_with_direction "3.921555783, 2.921555781, 0, -1.722456585, -1.722456585, 1"} \ 
\langle\text{-group_of_things_as_interval 0 12} \ 
\langle\text{-lines 1 0.04 $(\text{COLOR\_YELLOW})} \ 
\langle\text{-scene_objects_end} \ 
\langle\text{-povray_end} \ 

878
17750 \> - rm -rf POV
17751 \> mkdir POV
17752 \> mv F7_15_lines_0_*.pov POV
17753 \> mv makefile_animation POV
17754
17755
17756
17757
17758
17759
17760 F7_video:
17761 \> - rm -r FRAMES
17762 \> - mkdir FRAMES
17763 \> - rm fifteen_with_lines.mp4
17764 \> $(ORBITER) \\n17765 \> \> -prepare_frames \\
17766 \> \> \> -i 0 30 F7b/F7_15_lines_0_003d.png \\
17767 \> \> \> -output_start_at 0 \\
17768 \> \> \> -o FRAMES/frame%04d.png \\
17769 \> \> -end
17770 \> ffmpeg -r 5 -f image2 -i FRAMES/frame%04d.png \\
17771 \> \> -f mp4 -q:v 0 -vcodec mpeg4 fifteen_with_lines.mp4
17772
17773
17774 McKean_povray:
17775 \> $(ORBITER) -v 2 -povray \\
17776 \> -round 0 -nb_frames_default 30 \\
17777 \> -output_mask McKean_0%03d.pov \\
17778 \> -video_options -W 1024 -H 768 \\
17779 \> -global_picture_scale 1.5 \\
17780 \> -default_angle 80 \\
17781 \> -clipping_radius 4.4 \\
17782 \> -omit_bottom_plane \\
17783 \> -camera 0 "1,1,1" "-4.5,3.5,6" "0,0,0" \\
17784 \> -end \\
17785 \> -scene_objects \\
17786 \> -cubic_lex "0, 0, 1, 0, 0, -1, -2, 1, \\
17787 \> 2, 0, 0, 1, 1, -1, -1, -1, 0, 0, 0" \\
17788 \> -group_of_things "0" \\
17789 \> -cubics 0 $(SURFACE_COLOR_SEETHROUGH) \\
17790 \> -scene_objects.end \\
17791 \> -povray_end
17792 \> - rm -rf POV
17793 \> mkdir POV
17794 \> mv McKean_0_*.pov POV
17795 \> mv makefile_animation POV
17796
879
# Section 16.4: Continuous Function Plotter

`lissajous:`

```bash
$(ORBITER) -v 2
```

```bash
-smooth_curve "lissajous" 0.07 2000 15 0 18.85
```

```bash
-const a 3 b 2 c 1.57 r 7 -const_end
```

```bash
-var t -var_end
```

```bash
-code
```

```bash
push t push a mult push c add sin push r mult return
```

```bash
push t push b mult sin push r mult return
```

```bash
-code_end
```

`lissajous_plot:`

```bash
$(ORBITER) -v 2 -povray
```

```bash
-round 0 -nb_frames_default 1
```

```bash
-output_mask lissajous_%d_%03d.pov
```

```bash
-video_options -W 1024 -H 768
```

```bash
-global_picture_scale 0.40
```

```bash
-default_angle 45
```

```bash
-clipping_radius 5
```

```bash
-omit_bottom_plane
```

```bash
-camera 0 "0,-1,0" "0,0,12" "0,0,0"
```

```bash
-rotate_about_z_axis
```
-end \\
-scene_objects \\
-line_through_two_points_recentered_from_csv_file \\
-coordinate_grid.csv \\
-group_of_things "0" \\
-group_of_things "1" \\
-group_of_things "2" \\
-lines 0 0.09 "texture{ pigment{ color Yellow } }" \\
-lines 1 0.09 "texture{ pigment{ color Yellow } }" \\
-lines 2 0.09 "texture{ pigment{ color Yellow } }" \\
-group_of_things_as_interval 3 39 \\
-lines 3 0.02 "texture{ pigment{ color Black } }" \\
-point_list_from_csv_file \\
-function_lissajous_N2000.points.csv \\
-group_of_things_as_interval 0 6524 \\
-spheres 4 0.1 "texture{ pigment{ color Red } }" \\
-finish { diffuse 0.9 phong 1 }" \\
-plane_by_dual_coordinates "0,0,1,0" \\
-group_of_things "0" \\
-group_of_things_as_interval 0 6524 \\
-planes 5 "texture{ pigment{ color Blue*0.5 \ 
-transmit 0.5 } }" \\
-scene_objects_end \\
-povray_end \\
- rm -rf POV \\
-mkdir POV \\
-mv lissajous_0.*.pov POV \\
-mv makefile_animation POV \\

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 \\
lissajous_3d_plot: \\
- $(ORBITER) -v 2 -povray \\
-round 0 -nb_frames_default 30 \\
-output_mask lissajous_3d_%d.%03d.pov \\
-video_options -W 1024 -H 768 \\
-global_picture_scale 0.40 \\
-default_angle 45 \\
-clipping_radius 5
-omit_bottom_plane \n-camera 0 "0,0,1" "7,7,5" "0,0,1" \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_3d_N2000_points.csv \n-group_of_things_as_interval 0 6538 \n-spheres 4 0.1 "texture{ pigment{ color Red } finish { diffuse 0.9 phong 1}}" \n-group_of_things "0" \n-scene_objects_end \n-povray_end \n-rm -rf POV \n-mkdir POV \n-mv lissajous_3d_0.*.pov POV \n-mv makefile_animation POV

# Chapter 17 - Miscellaneous
# Section 17.1: Miscellaneous
SECTION MISCELLANEOUS:
misc_select:
  $(ORBITER) -v 3 \n  -define F -finite_field -q 7 -end \n  -with F -do -finite_field_activity -cheat_sheet_GF -end \n$(ORBITER) -v 4 -csv_file_select_rows_and_cols \n  GF_q7_multiplication_table_reordered.csv \n  "0,2,4" "0,2,4"

misc_join:
  $(ORBITER) -v 4 \n  -csv_file_join poly_orbits_d3_n3_q2_select_F2.csv Orbit_idx \n  -csv_file_join poly_orbits_d3_n3_q2_select_F4.csv Orbit_idx \n  -csv_file_join poly_orbits_d3_n3_q2_select_F8.csv Orbit_idx \n  -csv_file_join poly_orbits_d3_n3_q2_select_F16.csv Orbit_idx \n  -csv_file_join poly_orbits_d3_n3_q2_select_F32.csv Orbit_idx

table_mod_12:
  $(ORBITER) -v 2 \n  -define M -vector -load_csv_no_border clock_mult_excel.csv -end \n  -define all_one_r -vector -repeat 1 12 -end \n  -define all_one_c -vector -repeat 1 12 -end \n  -draw_matrix \n  -input_object M \n  -box_width 50 -bit_depth 24 \n  -partition 3 \n  all_one_r all_one_c \n  -end

# Section 17.2: Limitations
SECTION_LIMITATIONS:

###

# unclassified:
extract:
extract_from_file makefile Cremona_map make_Cremona_map.txt
~/bin/a2tex.out -numbers -text_width 80 <make_Cremona_map.txt >make_Cremona_map.tex

draw.eigenvalue.diag23:

draw.options 

-draw_mod_n -n 20 

-file ev.diag23 

-eigenvalues 2 0 0 3 

-pdflatex ev.diag23.draw.tex

-open ev.diag23.draw.pdf
Bibliography


[38] Fatma Karaoglu and Anton Betten. The number of cubic surfaces with 27 lines over a finite field. To appear in Journal of Algebraic Combinatorics.


[59] L. Schläfi, 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.


