# Contents

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

2 Getting Started ........................................ 11
   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

3 Basic Algebra .......................................... 33
   3.1 Basic Number Theory ................................ 33
   3.2 Prime Fields ........................................ 39
   3.3 Extension Fields .................................... 45
   3.4 Linear Algebra Over Finite Fields .................. 52
   3.5 Advanced Topics in Finite Fields .................... 64
   3.6 Basic Ring Theory ................................... 73

4 Geometry ............................................... 75
   4.1 Finite Projective Spaces ......................... 75
   4.2 Indexing Points and Lines ......................... 82
   4.3 Finite Desarguesian Projective Planes ............. 87
   4.4 The Grassmannian .................................. 91
   4.5 Algebraic Sets ..................................... 93
   4.6 The Klein Quadric and the Plücker Map ............ 98
   4.7 Orthogonal Spaces ................................ 100
   4.8 Hermitian Varieties ................................ 107
   4.9 Advanced Topics ................................... 109
   4.10 Geometric Objects ................................ 116
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Group Theory</td>
<td>121-180</td>
</tr>
<tr>
<td>5.1</td>
<td>Permutation Groups</td>
<td>121</td>
</tr>
<tr>
<td>5.2</td>
<td>Linear Groups</td>
<td>127</td>
</tr>
<tr>
<td>5.3</td>
<td>Subgroups</td>
<td>145</td>
</tr>
<tr>
<td>5.4</td>
<td>Linear Groups, Advanced Topics</td>
<td>157</td>
</tr>
<tr>
<td>5.5</td>
<td>Induced Actions</td>
<td>161</td>
</tr>
<tr>
<td>5.6</td>
<td>Group Theoretic Activities</td>
<td>172</td>
</tr>
<tr>
<td>5.7</td>
<td>Group Theoretic Activities Based on Magma</td>
<td>180</td>
</tr>
<tr>
<td>6</td>
<td>Orbit Algorithms</td>
<td>187-213</td>
</tr>
<tr>
<td>6.1</td>
<td>Schreier Trees</td>
<td>187</td>
</tr>
<tr>
<td>6.2</td>
<td>Poset Classification</td>
<td>195</td>
</tr>
<tr>
<td>6.3</td>
<td>Orbits on Subsets</td>
<td>199</td>
</tr>
<tr>
<td>6.4</td>
<td>Orbits on Subspaces</td>
<td>204</td>
</tr>
<tr>
<td>6.5</td>
<td>Orbits on Set-Partitions</td>
<td>206</td>
</tr>
<tr>
<td>6.6</td>
<td>Arcs and Caps in Projective Spaces</td>
<td>208</td>
</tr>
<tr>
<td>6.7</td>
<td>Cubic Curves</td>
<td>213</td>
</tr>
<tr>
<td>7</td>
<td>Cubic Surfaces</td>
<td>215-248</td>
</tr>
<tr>
<td>7.1</td>
<td>Creation</td>
<td>215</td>
</tr>
<tr>
<td>7.2</td>
<td>Quartic Curves</td>
<td>220</td>
</tr>
<tr>
<td>7.3</td>
<td>Classification</td>
<td>228</td>
</tr>
<tr>
<td>7.4</td>
<td>Isomorphism Testing and Recognition</td>
<td>243</td>
</tr>
<tr>
<td>7.5</td>
<td>Dickson Surfaces</td>
<td>246</td>
</tr>
<tr>
<td>7.6</td>
<td>ATLAS and Tables</td>
<td>248</td>
</tr>
<tr>
<td>8</td>
<td>Ring Theory</td>
<td>249-258</td>
</tr>
<tr>
<td>8.1</td>
<td>Polynomials Over Finite Fields</td>
<td>249</td>
</tr>
<tr>
<td>8.2</td>
<td>Multivariate Polynomial Rings</td>
<td>258</td>
</tr>
<tr>
<td>9</td>
<td>Applications</td>
<td>263-268</td>
</tr>
<tr>
<td>9.1</td>
<td>Number Theory</td>
<td>263</td>
</tr>
<tr>
<td>9.2</td>
<td>Representation Theory</td>
<td>266</td>
</tr>
<tr>
<td>9.3</td>
<td>Cryptography</td>
<td>268</td>
</tr>
<tr>
<td>10</td>
<td>Coding Theory</td>
<td>277-309</td>
</tr>
<tr>
<td>10.1</td>
<td>Introduction</td>
<td>277</td>
</tr>
<tr>
<td>10.2</td>
<td>Hamming Codes</td>
<td>281</td>
</tr>
<tr>
<td>10.3</td>
<td>Golay Codes</td>
<td>286</td>
</tr>
<tr>
<td>10.4</td>
<td>CRC Codes</td>
<td>287</td>
</tr>
<tr>
<td>10.5</td>
<td>Reed-Muller Codes</td>
<td>294</td>
</tr>
<tr>
<td>10.6</td>
<td>BCH Codes</td>
<td>296</td>
</tr>
<tr>
<td>10.7</td>
<td>Reed-Solomon Codes</td>
<td>302</td>
</tr>
<tr>
<td>10.8</td>
<td>Bounds</td>
<td>307</td>
</tr>
<tr>
<td>10.9</td>
<td>Classification of Optimal Linear Codes</td>
<td>309</td>
</tr>
</tbody>
</table>
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 [10].
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. [7]). Docker [24] 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. [27]). This is a software tool that allows collecting short command snippets in the form of text files that can easily be handled. However, the conventions in the tool involve some subtleties regarding the use of whitespace, which can cause problems to novice users. We will point out possible pitfalls along the way. Note that it is not necessary to use makefiles. Another possibility would be to use shell scripts. Ultimately, it would be possible to type out all commands into a terminal window. This could be a little tedious though, considering the fact that most Orbiter commands expect lengthy parameters from the command line.

Let us start by discussing how to run Orbiter as a native application. To do so, a unix-like compile environment is required, including a modern C++ compiler and the tools git and make. Windows users may need to install Cygwin [21]. The following steps are required: Using git, clone the repository. Then enter the directory orbiter and type

make

Once compiled, the Orbiter executable is

src/apps/orbiter/ orbiter.out

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

```
orbiter
  work
```

In the work directory, create a small makefile like so:
OP=../orbiter
ORBITER_PATH=$(OP)/src/apps/orbiter/

test:
    $(ORBITER_PATH)orbiter.out

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

make test

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

test:
    $(ORBITER_PATH)orbiter.out

is the makefile target “test.” It is important that the indentation after the makefile target is
done using tab characters (no spaces). There can be multiple targets in one makefile, as long
as they are separated by an empty line. for more information about the syntax of makefiles,
see [27].

A second way to run Orbiter is through Docker [24]. 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 \
    /mnt abetten/orbiter
ORBITER_PATH= docker $(DOCKER_OPTIONS)

test:
    $(ORBITER_PATH)orbiter.out

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

make test

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

Orbiter comes with a version numbering system called a build number. The build number should match the commit number on the github tree, shown in Figure 2.1. When Orbiter starts up, the build number is displayed. In order to update to a more recent version of Orbiter, Docker needs to be instructed to discard the local image. To do so, the command

```bash
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).

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

ORBITER_PATH=$MY_PATH/src/apps/orbiter/
ORBITER=$ORBITER_PATH/ orbiter.out
```

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

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

(a) Install Docker from [www.docker.com](http://www.docker.com), including the Linux kernel.

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

(c) Type
docker run -it --volume ${PWD}:/mnt -w /mnt abetten/orbiter orbiter.out

This will produce an output similar to the following:

sh-3.2$ docker run -it --volume $PWD:/mnt -w /mnt abetten/orbiter orbiter.out
Unable to find image 'abetten/orbiter:latest' locally
latest: Pulling from abetten/orbiter
004f1ee87df: Pull complete
5d6f1e8117db: Pull complete
48c2f6f66abe: 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:

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

where `<github-orbiter-path>` has to be replaced by the actual address provided by github. To find this path, find Orbiter on github, then click on the green box that says “Code” and copy the address into the clipboard by clicking the clipboard symbol (see Figure 2.2). Back in the terminal, paste this text after the `git clone` command. After cloning is complete, enter the orbiter directory (`cd orbiter`).

(c) Issue the following commands to compile Orbiter:

```bash
make
make install
```

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

Figure 2.2: GitHub Orbiter Page
2.2 The Orbiter Session

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

Let us take a closer look at an Orbiter session. Any orbiter session is invoked through the orbiter command `orbiter.out`, which is the name of the executable. Unless the executable resides in a directory contained in the search path of the shell, a path must be given. Several options apply to the orbiter session. They are listed in Table 2.1. 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_args</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**

17
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 a 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

\[ \text{Label:} \]
\[ \text{Do something} \]

Here, label is the name of the rule, and \text{Do something} is the code that is executed whenever make is called with the given label in the command line. For instance

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

\begin{verbatim}
  orbiter.out -v 3 -define F -finite_field -q 16 -end \n  -with F -do -finite_field_activity -cheat_sheet_GF -end
\end{verbatim}

The purpose of this command is to produce a file called

\begin{verbatim}
  GF_16.tex
\end{verbatim}

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 \texttt{F_16}. We can create a makefile like this:

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

With this file present, type the terminal command \texttt{make F_16} to execute the two line Orbiter command. Windows users can use \texttt{SuperShell}. The program \texttt{make} will look for the file \texttt{makefile} in the current directory. Once found, it will search for the label \texttt{F_16} in it and execute the commands beneath it. The given commands will invoke Orbiter and produce the \texttt{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 \texttt{ORBITER_PATH} below.

One difficulty in installing Orbiter is the path of installation. In the sample makefile, there is a makefile variable called \texttt{ORBITER_PATH} which contains the path to the orbiter executable \texttt{orbiter.exe}. Depending on the local installation of orbiter, the makefile variable needs to be changed accordingly. The actual command to run the \texttt{F_16} example is as follows:
The orbiter installation directory `orbiter` and a second directory called `work` should be next to each other. The orbiter example makefile should be copied into the `work` directory. The top of the file should contain the line

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

This will set `ORBITER_PATH` to point to the correct location of the orbiter executable. Inside the `work` directory, any of the commands listed in this guide will function correctly. Another possibility is to install `orbiter.out` in a central location. 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

Orbiter follows the object oriented paradigm. Mathematical objects of various types can be defined. Objects are maintained in a symbol table. New objects can be created from old. Activities can be applied to objects according to their type. By associating activities to objects of a certain type, Orbiter becomes more structured. It is easier to find the place where a certain functionality is defined, simply by searching by the type of object. This resembles the object oriented programming paradigm, where global functions are to be avoided and member functions of classes are preferred.

Objects can be of two types: primary or secondary. Objects of primary type can be created directly from scratch. Secondary objects depend on other objects that have to be created first. For instance, a finite field object is an object of primary type. A projective space is an object of secondary type because it needs a finite field object to be created first (the field over which the projective space is defined). Yet another type of objects are created from activities that are applied to objects. For instance, a cubic surface object can be created from a projective space object using the `-define_surface` command.

In this section, a brief overview of the types of objects is given, as well as the activities that can be applied. More details will be provided in later sections of this guide.

The syntax to create an object is

```
-define LABEL KEYWORD EXTRAS -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 `EXTRAS` depend on the type of object created. The command `-end` is necessary to finish the definition. 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 3 F -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
<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
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
```

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 DH$(k, n)$. Orbiter knows the classification of dual hyperovals DH$(4, 7)$ and 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 \n  ▶ ▶ -define F -finite_field -q 5 -end \n  ▶ ▶ ▶ -define O -orthogonal_space 0 5 F -end \n```
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 3 F -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 
  > -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 surfaces, see Section 7.1.
2.6 Set Builder

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

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

The next command creates the interval [0, 63]. We use the \(-\text{loop}\) command to save us from typing out all elements of the set. The \(-\text{loop}\) command has three arguments: the start value, the end value plus one, and the increment.

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

For C programmers, \(-\text{loop} a b c\) is equivalent to

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

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

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

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

Here is an example. We first create the field \(\mathbb{F}_5\), and then create the vector \(v = (0, 1, 2, 3, 4)\). The -field option refers to the finite field created previously. The -dense option allows to enter the vector coefficients on the command line.

vector_example1:

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

Vectors can also be read from file. The -file option can be used to name a csv file. In this case, the -dense option should not be used. A vector can also serve as a matrix. The -format \(k\) option can be used to specify the number \(k\) of rows. The number of columns is determined as \(n/k\), where \(n\) is the length of the vector given. For instance, the next example creates a \(2 \times 3\) matrix over \(\mathbb{F}_5\):

vector_example2:

```
$ (ORBITER) -v 2 \
  > -define F -finite_field -q 5 -end \n  > -define v -vector -field F -format 2 -dense "0,1,2,3,4,0" -end \n  > -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,\n0,0,0,1,1,1,1"
```

```
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
```

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="\n11011000100001010000\n11101011111010000101\n00000100000100010101\n11110011010001001110\n01010100000001001101\n00000100000001001101\n01000000000001001101\n00100000000001001101\n00010001100000100110\n11101001011001001101\n00000000000001001101\n00000000000001001101\n01010110100110010101\n01110111110100111011\n00000000000001000111\n00000000000001010101\n00000000000001000101\n00000000000001000101\n00000000000001000101\n00000000000001000101"
```

30
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 \texttt{\textbf{sparse}} command can be used to enter non-zero coefficients as a list of pairs. For instance,

\texttt{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:

\begin{verbatim}
1 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 1
\end{verbatim}

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.

\texttt{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 \texttt{-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:

\begin{verbatim}
1 1 1 1 1 1 1 1 1 1 1
\end{verbatim}
vector_example_all_one_11:
  $\texttt{ORDITER} -v 2\
  \texttt{-define v -vector -repeat 1 11 -end}\
  \texttt{-print_symbols}

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

\[(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1).\]
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:
\[
\begin{align*}
\text{\$} & \text{(ORBITER) -v 1 -smallest\_primitive\_root 29} \\
\end{align*}
\]
computes a primitive root modulo 29. The answer in this case is 2. For a large example, consider

PR_915839:
\[
\begin{align*}
\text{\$} & \text{(ORBITER) -v 5 -primitive\_root 915839} \\
\end{align*}
\]
which computes a primitive root modulo 915839. The answer is 43085. The command

PR_915839\_check:
\[
\begin{align*}
\text{\$} & \text{(ORBITER) -v 5 -power\_mod 43085 49842 915839} \\
\end{align*}
\]
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 $a^n \mod p$.</td>
</tr>
<tr>
<td>-discrete_log</td>
<td>b a p</td>
<td>Computes $n$ such that $a^n \equiv b \mod p$.</td>
</tr>
<tr>
<td>-extended_gcd</td>
<td>a b</td>
<td>Computes integers $g, u,$ and $v$ such that $g = \gcd(a, b) = ua + vb$.</td>
</tr>
<tr>
<td>-square_root_mod</td>
<td>a p</td>
<td>Computes a square root of $a$ modulo $p$, i.e. an integer $b$ such that $b^2 \equiv a \mod p$.</td>
</tr>
<tr>
<td>-square_root</td>
<td>a</td>
<td>Computes $\lfloor \sqrt{a} \rfloor$ of an integer $a$.</td>
</tr>
<tr>
<td>-inverse_mod</td>
<td>a p</td>
<td>Computes the modular inverse of $a$ modulo $p$, i.e. an integer $b$ with $ab \equiv 1 \mod p$.</td>
</tr>
<tr>
<td>-draw_mod_n</td>
<td>descr</td>
<td>Draws the integers modulo $n$ on a circle.</td>
</tr>
<tr>
<td>-order_of_q_mod_n</td>
<td>q n_min n_max</td>
<td>Computes the order $\text{ord}(q, n)$ of $q$ modulo $n$ for all $n$ with $n_{\text{min}} \leq n \leq n_{\text{max}}$ 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 \texttt{-inverse\_mod} command. The command

\texttt{IM\_723:}
\texttt{\textgreater{} $(ORBITER) -v 5 -inverse\_mod 723 4060}$

computes the inverse of 1865025205 modulo 2147483647 which is 579785381.

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

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

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

\texttt{IM\_gcd:}
\texttt{\textgreater{} $(ORBITER) -v 5 -extended\_gcd 1865025205 2147483647}$

The output is

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

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

In order to compute the modular power

$$a^e \mod n,$$

the \texttt{-power\_mod} command can be used. For instance,

\texttt{PM3a:}
\texttt{\textgreater{} $(ORBITER) -v 5 -power\_mod 16807 1073741823 2147483647}$

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

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

The command `order_of_2_mod_n` computes ord($q, n$), the order of $q$ modulo $n$, for all $n$ with $n_{\min} \leq n \leq n_{\max}$ and gcd($n, q$) = 1. It also computes Euler’s totient function $\varphi(n)$ and the cofactor $\varphi(n)/\text{ord}(q, n)$. For instance,

```
$ (ORBITER) -v 2 -order_of_2_mod_n 2 3 151
```

produces the output shown in Table 3.2.

The command

```
$ (ORBITER) -v 3 -square_root_mod 33 41
```

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

\[ 19^2 \equiv 33 \mod 41. \]
Table 3.3: The values of the Eulerfunction

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

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

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

The result is shown in Table 3.3.
Table 3.4: The function \( a \mapsto a^2 \mod 11 \)

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

Figure 3.1: Cycle of powers of 2 modulo 13

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

```
draw_mod_13:
▷ $(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,

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

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

Table 3.6 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 3.5: Options for Creating Finite Fields

<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-q</td>
<td>$q$</td>
<td>Specify the order of the field. Here, $q = p^k$ for some prime $p$ and some positive integer $k$.</td>
</tr>
<tr>
<td>-override_polynomial</td>
<td>$n$</td>
<td>Specify the polynomial used to create the finite field. The polynomial is given as integer, using the base $p$ representation. See Section 3.3.</td>
</tr>
<tr>
<td>-without_tables</td>
<td></td>
<td>Create the field without precomputing the tables.</td>
</tr>
<tr>
<td>Command</td>
<td>Arguments</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<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 DNE</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>$4 = \alpha^4$</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>$5 = \alpha^5$</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>$6 = \alpha^3$</td>
<td>1</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

$$+\begin{array}{cccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 \\
0 & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\
1 & 1 & 2 & 3 & 4 & 5 & 6 & 0 \\
2 & 2 & 3 & 4 & 5 & 6 & 0 & 1 \\
3 & 3 & 4 & 5 & 6 & 0 & 1 & 2 \\
4 & 4 & 5 & 6 & 0 & 1 & 2 & 3 \\
5 & 5 & 6 & 0 & 1 & 2 & 3 & 4 \\
6 & 6 & 0 & 1 & 2 & 3 & 4 & 5
\end{array}$$
Suppose we want to check Wilson’s theorem that the product of all nonzero field elements of negative one. The following command so so, assuming that \( p = 11 \). We first create a vector of all nonzero field elements, which we take as the integers from 1 to 10. After that, we use the product_of finite field activity to compute the product of these elements. The answer is 10 which is congruent to \(-1 \mod 11\):

\[
\begin{array}{cccccccc}
3^0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
3^1 & 3 & 6 & 2 & 5 & 1 & 4 & 8 \\
3^2 & 9 & 4 & 1 & 5 & 2 & 6 & 3 \\
3^3 & 3 & 6 & 2 & 5 & 1 & 4 & 10 \\
\end{array}
\]

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.

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 \\
2 & 4 & 6 & 1 & 3 & 5 & 8 \\
3 & 6 & 2 & 5 & 1 & 4 & 9 \\
4 & 1 & 5 & 2 & 6 & 3 & 10 \\
5 & 3 & 1 & 6 & 4 & 2 & 11 \\
6 & 5 & 4 & 3 & 2 & 1 & 12 \\
\end{array}
\]

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:

```bash
F_101_wo:
  $\$(ORBITER) -v 3 \
  $\$ -define F -finite_field -q 101 -without_tables -end \
```
\texttt{\textasciitilde\textasciitilde -with F -do -finite\_field\_activity -cheat\_sheet\_GF \textasciitilde\textasciitilde end}\n\texttt{pdflatex GF\_101.tex}\n\texttt{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 $|F| = q^e$. Every finite field $E$ is of this form, where $F = \mathbb{F}_p$ and $p$ is the characteristic of $E$.

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

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

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

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

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

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

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

$$\mathbb{F}_4 = \{0, 1, X, X + 1\}.$$
Figure 3.4: Addition and multiplication tables of $\mathbb{F}_4$

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

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>X</th>
<th>X + 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>X</td>
<td>X + 1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>X + 1</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X + 1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>X + 1</td>
<td>X + 1</td>
<td>X</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

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

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

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>X</th>
<th>X + 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>X</td>
<td>X + 1</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
<td>X</td>
<td>X + 1</td>
<td>1</td>
</tr>
<tr>
<td>X + 1</td>
<td>0</td>
<td>X + 1</td>
<td>1</td>
<td>X</td>
</tr>
</tbody>
</table>

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

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

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

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

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

The primitive polynomials used by Orbiter to create small finite fields are listed in Table 3.8. The relation is given using the Greek letter that is used in orbiter cheat sheets for that particular field.

Let us look at a few examples. The command
<table>
<thead>
<tr>
<th>$q$</th>
<th>Polynomial</th>
<th>Numerical</th>
<th>Relation</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>$X^2 + X + 1$</td>
<td>7</td>
<td>$\omega^2 = \omega + 1$</td>
</tr>
<tr>
<td>8</td>
<td>$X^3 + X^2 + 1$</td>
<td>13</td>
<td>$\gamma^3 = \gamma^2 + 1$</td>
</tr>
<tr>
<td>9</td>
<td>$X^2 + X + 2$</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>$X^4 + X^3 + 1$</td>
<td>25</td>
<td>$\delta^4 = \delta^3 + 1$</td>
</tr>
<tr>
<td>25</td>
<td>$X^2 + X + 2$</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>$X^3 + 2X + 1$</td>
<td>34</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>$X^5 + X^2 + 1$</td>
<td>37</td>
<td>$\eta^5 = \eta^2 + 1$</td>
</tr>
<tr>
<td>49</td>
<td>$X^2 + X + 3$</td>
<td>59</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>$X^6 + X^5 + 1$</td>
<td>97</td>
<td></td>
</tr>
<tr>
<td>81</td>
<td>$X^4 + X^3 + 2$</td>
<td>110</td>
<td></td>
</tr>
<tr>
<td>121</td>
<td>$X^2 + 4X + 2$</td>
<td>167</td>
<td></td>
</tr>
<tr>
<td>125</td>
<td>$X^3 + X^2 + X + 2$</td>
<td>86</td>
<td></td>
</tr>
<tr>
<td>128</td>
<td>$X^7 + X^6 + 1$</td>
<td>193</td>
<td>$\zeta^7 = \zeta^6 + 1$</td>
</tr>
<tr>
<td>169</td>
<td>$X^2 + X + 2$</td>
<td>184</td>
<td></td>
</tr>
<tr>
<td>243</td>
<td>$X^5 + 2X + 1$</td>
<td>250</td>
<td></td>
</tr>
<tr>
<td>256</td>
<td>$X^8 + X^4 + X^3 + X^2 + 1$</td>
<td>285</td>
<td></td>
</tr>
<tr>
<td>289</td>
<td>$X^2 + X + 3$</td>
<td>309</td>
<td></td>
</tr>
<tr>
<td>343</td>
<td>$X^3 + 3X + 2$</td>
<td>366</td>
<td></td>
</tr>
<tr>
<td>361</td>
<td>$X^2 + X + 2$</td>
<td>382</td>
<td></td>
</tr>
<tr>
<td>512</td>
<td>$X^9 + X^4 + 1$</td>
<td>529</td>
<td></td>
</tr>
<tr>
<td>529</td>
<td>$X^2 + 2X + 5$</td>
<td>580</td>
<td></td>
</tr>
<tr>
<td>625</td>
<td>$X^4 + X^3 + X + 2$</td>
<td>326</td>
<td></td>
</tr>
<tr>
<td>729</td>
<td>$X^6 + X^5 + 2$</td>
<td>974</td>
<td></td>
</tr>
<tr>
<td>841</td>
<td>$X^2 + 5X + 2$</td>
<td>988</td>
<td></td>
</tr>
<tr>
<td>961</td>
<td>$X^2 + 2X + 3$</td>
<td>1026</td>
<td></td>
</tr>
<tr>
<td>1024</td>
<td>$X^{10} + X^3 + 1$</td>
<td>1033</td>
<td></td>
</tr>
</tbody>
</table>

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

$\mathbb{F}_4$:

\begin{verbatim}
$\$(ORBITER) -v 3 \\
$\$(ORBITER) -v 3 \\
$\$(ORBITER) -define F -finite_field -q 4 -end \\
$\$(ORBITER) -with F -do -finite_field_activity -cheat_sheet_GF -end
\end{verbatim}

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

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

\begin{verbatim}
$\$(ORBITER) -v 3 \\
$\$(ORBITER) -define F -finite_field -q 16 -end \\
$\$(ORBITER) -with F -do -finite_field_activity -cheat_sheet_GF -end
\end{verbatim}

\begin{verbatim}
pdflatex GF_16.tex
\end{verbatim}

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

Unlike other computer algebra systems (GAP [28] and Magma [14]), Orbiter does not use Conway polynomials to create field extensions. Instead, it provides the option to override the polynomial used to create the finite field. For subfield relationships, the cheat sheet will indicate the irreducible polynomials of all subfields for a given field. For instance, Table 3.10
<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>

Table 3.10: The subfields of $\mathbb{F}_{64}$

![Table of subfields](image)

Figure 3.5: Addition and multiplication table of $\mathbb{F}_3$ and $\mathbb{F}_9$ using the lexicographic ordering of elements

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 $\mathbb{F}_q$</td>
</tr>
<tr>
<td>-nullspace</td>
<td>$m \ n \ L$</td>
<td>Compute a basis for the right nullspace of the $m \times n$ matrix $L$</td>
</tr>
<tr>
<td>-normalize_from_the_right</td>
<td></td>
<td>Normalizes the result of -RREF or nullspace from the right</td>
</tr>
<tr>
<td>-normalize_from_the_left</td>
<td></td>
<td>Normalizes the result of -RREF or nullspace from the left</td>
</tr>
<tr>
<td>-eigenstuff</td>
<td>$d \ M$</td>
<td>Computes the eigenvalues and eigenvectors of the given $d \times d$ matrix $M$ over $\mathbb{F}_q$</td>
</tr>
<tr>
<td>-all_rational_normal_forms</td>
<td>$d$</td>
<td>Produces a report of all rational normal forms of endomorphisms of $\mathbb{F}^d_q$</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:
▷ $(\text{ORBITER}) -v 2 \ \$
▷ ▷ -define F -finite_field -q 2 -end \$
▷ ▷ -define v -vector -field F -format 2 \$
▷ ▷ ▷ -dense "1,1,1,0,1,1,0,0,1" \$
▷ ▷ ▷ -end \$
▷ ▷ -with F -do -finite_field_activity \$
▷ ▷ -RREF v -normalize_from_the_right \$
▷ ▷ -end
```

computes the RREF form of the matrix

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

over $\mathbb{F}_2$. The output is the matrix
The \texttt{-\textsc{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 \texttt{-\textsc{rref}} command:

```
V7\_VANDERMONDE\_EXTENDED="\n 1,0,0,0,0,0,1,0,0,0,0,0,0, \\
 1,1,1,1,1,0,0,0,0,0,0,0,0, \\
 1,2,4,1,2,4,1,0,0,1,0,0,0,0, \\
 1,3,2,6,4,5,1,0,0,1,0,0,0, \\
 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"
```

```
RREF\_V7:
  $\text{(ORBITER)} -v 2 \$
  \quad -define F -finite_field -q 7 -end \$
  \quad -define V -vector -format 7 \$
  \quad \quad -dense $(V7\_VANDERMONDE\_EXTENDED) \$
  \quad \quad -end \$
  \quad -with F -do -finite_field_activity \$
  \quad \quad -RREF V \$
  \quad -end
```

The following (long) output is produced. Observe how the inverse matrix appears in the second half once the \texttt{-\textsc{rref}} algorithm is finished:

```
\begin{pmatrix}
  1 & 1 & 0 & 0 & 1 \\
  0 & 0 & 1 & 1 & 1
\end{pmatrix}
```

A matrix over the field $\mathbb{F}_7$
Position \((i,j) = (0,0)\), found pivot in column 0

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

After making pivot 1:

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

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

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

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

After 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

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

After making pivot 1:
After elimination below pivot:

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

Position \((i,j) = (3,3)\), found pivot in column 3

After making pivot 1:

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

After elimination below pivot:
\[
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 1 & 6 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 3 & 0 & 1 & 3 & 4 & 6 & 4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 6 & 4 & 6 & 1 & 4 & 3 & 6 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 3 & 2 & 6 & 1 & 3 & 6 & 3 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 4 & 3 & 4 & 6 & 6 & 4 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 3 & 1 & 4 & 3 & 6 & 3 & 1 & 0 & 0 & 1 & 0 \\
\end{bmatrix}
\]

Position \((i, j) = (4, 4)\), found pivot in column 4

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

After making pivot 1:

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

After elimination below pivot:

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

Position \((i, j) = (5, 5)\), found pivot in column 5
\[
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 1 & 6 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 3 & 0 & 1 & 3 & 4 & 6 & 4 & 0 & 0 \\
0 & 0 & 0 & 1 & 6 & 4 & 6 & 1 & 4 & 3 & 6 & 0 \\
0 & 0 & 0 & 0 & 1 & 3 & 2 & 5 & 1 & 2 & 1 & 5 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 6 & 5 & 4 & 3 & 2 \\
0 & 0 & 0 & 0 & 0 & 6 & 5 & 2 & 3 & 4 & 5 & 6 \\
\end{bmatrix}
\]

After making pivot 1:

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

After elimination below pivot:

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

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

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

After making pivot 1:
After elimination below pivot:

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

Did not find pivot. The rank of the matrix is 7.

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

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

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

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

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

\[
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 3 & 3 & 4 & 5 & 6 & 0 \\
0 & 0 & 1 & 3 & 0 & 0 & 0 & 0 & 3 & 2 & 6 & 0 & 1 & 2 \\
0 & 0 & 0 & 1 & 6 & 0 & 0 & 0 & 0 & 3 & 3 & 1 & 5 & 2 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 6 & 3 & 5 & 5 & 3 & 6 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 6 & 5 & 4 & 3 & 2 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 6 & 6 & 6 & 6 & 6 & 6 & 6 \\
\end{bmatrix}
\]
After elimination above pivot 0 in position (0,0):

\[
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 4 & 0 & 6 & 0 & 3 & 1 \\
0 & 0 & 1 & 3 & 0 & 0 & 0 & 0 & 3 & 2 & 6 & 0 & 1 & 2 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 6 & 6 & 1 & 6 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 6 & 3 & 5 & 5 & 3 & 6 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 6 & 5 & 4 & 3 & 2 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 6 & 6 & 6 & 6 & 6 & 6 & 6 \\
\end{bmatrix}
\]
The inverse matrix agrees with the output obtained in Section 3.2.

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

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

compares 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"
```

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"
```

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"
```

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"
```

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"
```

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"
```

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"
```

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"
```

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"
```

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"
```

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"
```

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

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

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:

\[
\text{\texttt{\{(ORBITER) -v 7 \}}}
\]

\[
\text{\texttt{\{define F -finite_field -q 2 -end \}}}
\]

\[
\text{\texttt{\{all_rational_normal_forms F 3 \}}}
\]

\[
\text{\texttt{pdflatex Class_reps_GL_3_2.tex}}
\]

\[
\text{\texttt{open Class_reps_GL_3_2.pdf}}
\]

produces a list of all conjugacy classes of \( \text{GL}(3, 2) \). 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 [40]. This command relies on the Orbiter catalogue of irreducible polynomials. For an introduction to the rational normal form of endomorphisms, see [46].

**Conjugacy Classes of \( \text{GL}(3, 2) \)**

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

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

Class 0 / 6

\(3, 1, 0\)

\[
\begin{pmatrix}
0 & 0 & 1 \\
1 & 0 & 1 \\
0 & 1 & 0
\end{pmatrix}
\]
<table>
<thead>
<tr>
<th>Class</th>
<th>Order</th>
<th>Size</th>
<th>Representation</th>
</tr>
</thead>
</table>
| 1 / 6 | 7     | 24   | \[
\begin{bmatrix}
0 & 0 & 1 \\
1 & 0 & 0 \\
0 & 1 & 1
\end{bmatrix}
\] |
| 2 / 6 | 7     | 24   | \[
\begin{bmatrix}
0 & 1 & 0 \\
1 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\] |
| 3 / 6 | 3     | 56   | \[
\begin{bmatrix}
1 & 0 & 0 \\
1 & 1 & 0 \\
0 & 1 & 1
\end{bmatrix}
\] |
| 4 / 6 | 4     | 42   | \[
\begin{bmatrix}
1 & 0 & 0 \\
1 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\] |
| 5 / 6 | 8     | 21   | \[
\begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\] |
| 6 / 6 | 168   | 1    | \[
\begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\] |
3.5 Advanced Topics in Finite Fields

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

First, in Tables 3.12-3.13, a summary of finite field activities is shown.

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

```
normal_basis.2.3:
▷ $\$(ORBITER) -v 2 \n▷ ▷ -define F -finite_field -q 2 -end \n▷ ▷ -with F -do -finite_field_activity \n▷ ▷ -normal_basis 3 -end
```

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

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

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

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

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

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

Thus,

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

under $\Phi$, 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

\[\mathbb{F}_{q^r} \rightarrow \mathbb{F}_{q^r}, x \mapsto ax.\]

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 $x_i^j$ 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 $2 \times 2$. 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 $3 \times 3$ and $6 \times 6$, respectively.

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

```
F_8_Nth_roots_21:
▷ $(\text{ORBITER})\ -v\ 3\ \backslash
▷ \ (define\ F\ \ -finite\_field\ -q\ 8\ \ -override\_polynomial\ 11\ \ -end)\ \backslash
▷ \ (with\ F\ \ do\ \ -finite\_field\_activity\ \ -nth\_roots\ 21\ \ -end)
▷ \ pdfflatex\ Nth\_roots\_q8\_n21.tex
▷ \ open\ Nth\_roots\_q8\_n21.pdf
```

The output is:

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

- $\beta^0 = 100000 = 1$
- $\beta^1 = 000100 = \alpha^3$
- $\beta^2 = 100001 = \alpha^5 + 1$
- $\beta^3 = 111011 = \alpha^5 + \alpha^3 + \alpha^2 + \alpha + 1$
- $\beta^4 = 011111 = \alpha^5 + \alpha^4 + \alpha^3 + \alpha^2 + \alpha$
- $\beta^5 = 101010 = \alpha^4 + \alpha^2 + 1$
\[ \beta^6 = 110100 = \alpha^3 + \alpha + 1 \]
\[ \beta^7 = 100111 = \alpha^5 + \alpha^4 + \alpha^3 + 1 \]
\[ \beta^8 = 101101 = \alpha^5 + \alpha^3 + \alpha^2 + 1 \]
\[ \beta^9 = 011101 = \alpha^5 + \alpha^3 + \alpha^2 + \alpha \]
\[ \beta^{10} = 011011 = \alpha^5 + \alpha^4 + \alpha^2 + \alpha \]
\[ \beta^{11} = 001011 = \alpha^5 + \alpha^4 + \alpha^2 \]
\[ \beta^{12} = 001001 = \alpha^3 + \alpha^2 \]
\[ \beta^{13} = 111000 = \alpha^2 + \alpha + 1 \]
\[ \beta^{14} = 000111 = \alpha^5 + \alpha^4 + \alpha^3 \]
\[ \beta^{15} = 101001 = \alpha^5 + \alpha^2 + 1 \]
\[ \beta^{16} = 111100 = \alpha^3 + \alpha^2 + \alpha + 1 \]
\[ \beta^{17} = 100110 = \alpha^4 + \alpha^3 + 1 \]
\[ \beta^{18} = 010100 = \alpha^3 + \alpha \]
\[ \beta^{19} = 100011 = \alpha^5 + \alpha^4 + 1 \]
\[ \beta^{20} = 001100 = \alpha^3 + \alpha^2 \]

Let \( \gamma \) 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 \ \\
  \$\text{-define F -finite_field -q 8 -end} \ \\
  \$\text{-with F -do -finite_field_activity} \ \\
  \$\text{-Vandermonde_matrix} \ \\
```
The output is shown below. Again, the first matrix is $V = (x_i^j)$. The second matrix is $V^{-1}$:

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

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

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

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

Orbiter can create code for the number theoretic transform. This is the discrete Fourier transform performed over finite fields. The generated code can be compiled with the Orbiter library. 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:

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

71
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.

```
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
```

The command

```
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 \texttt{-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 \n\end{verbatim}

For more on rings, see Chapter 8.
4.1 Finite Projective Spaces

Orbiter can create the projective space $\text{PG}(n, q)$. In order to do so, an object of type \texttt{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{verbatim}
  $(ORBITER) -v 3 \\
  \triangleright \triangleright -define F -finite_field -q 2 -end \\
  \triangleright \triangleright -define P -projective_space 3 F -end
\end{verbatim}

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 th 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{verbatim}
  $(ORBITER) -v 2 \\
  \triangleright \triangleright -define F -finite_field -q 4 -end \\
  \triangleright \triangleright -define P -projective_space 2 F -end
\end{verbatim}
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 cheat sheet contains a drawing of the plane as shown in Figure 4.1. The affine plane is shown in the cartesian plane, while the line at infinity is wrapped around the top right corner. The cheat sheet continues by listing the points, including the canonical Baer subgeometry PG(2, 2). After that, the points are listed again, but with left-normalized vectors. Finally, the lines are shown.

PG(2, 4) has 21 points:
\[ P_0 = (1, 0, 0) = (1, 0, 0) \quad P_{11} = (2, 1, 1) = (\alpha, 1, 1) \]
\[ P_1 = (0, 1, 0) = (0, 1, 0) \quad P_{12} = (3, 1, 1) = (\alpha^2, 1, 1) \]
\[ P_2 = (0, 0, 1) = (0, 0, 1) \quad P_{13} = (0, 2, 1) = (0, \alpha, 1) \]
\[ P_3 = (1, 1, 1) = (1, 1, 1) \quad P_{14} = (1, 2, 1) = (1, \alpha, 1) \]
\[ P_4 = (1, 1, 0) = (1, 1, 0) \quad P_{15} = (2, 2, 1) = (\alpha, \alpha, 1) \]
\[ P_5 = (2, 1, 0) = (\alpha, 1, 0) \quad P_{16} = (3, 2, 1) = (\alpha^2, \alpha, 1) \]
\[ P_6 = (3, 1, 0) = (\alpha^2, 1, 0) \quad P_{17} = (0, 3, 1) = (0, \alpha^2, 1) \]
\[ P_7 = (1, 0, 1) = (1, 0, 1) \quad P_{18} = (1, 3, 1) = (1, \alpha^2, 1) \]
\[ P_8 = (2, 0, 1) = (\alpha, 0, 1) \quad P_{19} = (2, 3, 1) = (\alpha, \alpha^2, 1) \]
\[ P_9 = (3, 0, 1) = (\alpha^2, 0, 1) \quad P_{20} = (3, 3, 1) = (\alpha^2, \alpha^2, 1) \]
\[ P_{10} = (0, 1, 1) = (0, 1, 1) \]

Baer subgeometry:

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

There are 7 elements in the Baer subgeometry.

Normalized from the left:

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

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

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

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

The projective space PG(3, 2)

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

PG(3, 2) has 15 points:

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

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:

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

Lines sorted by Pluecker coordinates

\[
0 = \text{Pl}(1, 0, 0, 0, 0, 0) = L_0 = \begin{bmatrix} 1000 \\ 0100 \end{bmatrix}
\]
1 = Pl(0, 1, 0, 0, 0) = L_{34} = \begin{bmatrix} 0010 \\ 0001 \end{bmatrix}

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

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

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

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

\vdots

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

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

\begin{align*}
L_0 &= \begin{bmatrix} 1000 \\ 0100 \end{bmatrix} = Pl(1, 0, 0, 0, 0) \\
L_4 &= \begin{bmatrix} 1000 \\ 0010 \end{bmatrix} = Pl(0, 0, 1, 0, 0) \\
L_6 &= \begin{bmatrix} 1000 \\ 0001 \end{bmatrix} = Pl(0, 0, 0, 1, 0) \\
L_{28} &= \begin{bmatrix} 0100 \\ 0010 \end{bmatrix} = Pl(0, 0, 0, 0, 1) \\
L_{30} &= \begin{bmatrix} 0100 \\ 0001 \end{bmatrix} = Pl(0, 0, 0, 1, 0) \\
L_{34} &= \begin{bmatrix} 0010 \\ 0001 \end{bmatrix} = Pl(0, 1, 0, 0, 0) \\
\end{align*}

The planes of PG(3, 2)

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

\begin{align*}
L_0 &= \begin{bmatrix} 1000 \\ 0100 \\ 0010 \end{bmatrix} \\
L_1 &= \begin{bmatrix} 1000 \\ 0100 \\ 0011 \end{bmatrix} \\
\end{align*}
\[
L_{14} = \begin{bmatrix}
0100 \\
0010 \\
0001
\end{bmatrix}
\]

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 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 PG($n, q$) is

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

This is the \textit{standard frame}. We start the labeling of points with the standard frame. After these $n + 2$ points, we list the remaining points in lexicographic ordering (utilizing right-normalized representative). Thus, for 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 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 PGL(3, 2) acting on PG($2, 2$), for instance. The points of 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 PG($2, 4$):

```
PG_2_4_rank_point:
  $(ORBITER) -v 2 \$
  $-define F -finite_field -q 4 -end \$
  $-with F -do -finite_field_activity \$
  $-rank_point_in_PG 2 "3,3,1" -end$
```
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 $x = e_i$ 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 $j = i - 1, \ldots, 1, 0$ 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 $a \geq \sum_{j=0}^{i-1} q^j$ then
15:        $a \leftarrow a - 1$
16:    $a \leftarrow a + n - i + \sum_{j=0}^{i-1} q^j$
17: return $a$

$a = \text{Rank}(x)$ | $x = \text{Unrank}(a)$
---|---
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 | $(0, 1, 1)$

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

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

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

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

The projective spaces $\text{PG}(2,q)$ deserve special attention. They are examples of a more general structure called projective planes. The $\text{PG}(2,F)$, $F$ a field, are distinguished in the class of projective planes by the fact that the theorem of Desargues always holds. They are called the desarguesian projective planes. For other projective planes, see Section 12.2.

The points in the dearguesian projective plane $\text{PG}(2,q)$ have the coordinates $P(x,y,z)$, with $x,y,z \in F_q$. 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

\[ \text{PG}_2.16: \]
\[ \text{\texttt{\$\{ORBITER\} \}} \]
\[ \text{\texttt{\quad \texttt{-draw_options -xin 20000 -yin 20000 \}}} \]
\[ \text{\texttt{\quad \texttt{-radius 200 -line_width 0.3 -nodes_empty -end \}}} \]
\[ \text{\texttt{\quad \texttt{-define F -finite_field -q 16 -end \}}} \]
\[ \text{\texttt{\quad \texttt{-define P -projective_space 2 F -end \}}} \]
\[ \text{\texttt{\quad \texttt{-with P -do -projective_space_activity \}}} \]
\[ \text{\texttt{\quad \texttt{\quad \texttt{-cheat_sheet \}}} \]
\[ \text{\texttt{\quad \texttt{-end \}}} \]
\[ \text{\texttt{\texttt{\texttt{pdflatex PG}_2.16.tex \}}} \]
\[ \text{\texttt{\texttt{\texttt{open PG}_2.16.pdf \}}} \]

produces the drawing of $\text{PG}(2,16)$ shown in Figure 4.3. The $\texttt{-nodes_empty}$ command is used to suppress the drawing of the nodes. The $\texttt{-xin 20000}$ and $\texttt{-yin 20000}$ 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 $\text{PG}(2,4)$

\[ \text{PG}_2.4_{\text{with decomposition}}: \]
\[ \text{\texttt{\$\{ORBITER\} -v 2 \}} \]
\[ \text{\texttt{\quad \texttt{-define F -finite_field -q 4 -end \}}} \]
\[ \text{\texttt{\quad \texttt{-define P -projective_space 2 F -end \}}} \]
\[ \text{\texttt{\quad \texttt{-with P -do -projective_space_activity \}}} \]
\[ \text{\texttt{\quad \texttt{\quad \texttt{-cheat_sheet_for_decomposition_by_element_PG \}}} \]
\[ \text{\texttt{\quad \texttt{\quad \texttt{\texttt{\texttt{\texttt{1 "0,1,0, 0,0,1, 2,1,1, 0" \}}} \]
\[ \text{\texttt{\quad \texttt{\quad \texttt{\quad \texttt{\quad \texttt{\quad \texttt{PG}_2.4_{\text{singer}} \}}} \]
\[ \text{\texttt{\quad \texttt{\quad \texttt{\quad \texttt{\quad \texttt{\quad \texttt{-end \}}} \]
\[ \text{\texttt{\texttt{\texttt{pdflatex PG}_2.4_{\text{singer}}.tex \}}} \]
\[ \text{\texttt{\texttt{\texttt{\texttt{open PG}_2.4_{\text{singer}}.pdf \}}} \]

87
Figure 4.3: The plane $\text{PG}(2, 16)$

The output is shown below:

Considering the cyclic group generated by

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

The group is transitive on points and on lines.

Orbits on points:
There are 1 orbits, the orbit lengths are 21

Orbits on lines:
There are 1 orbits, the orbit lengths are 21

Fixed points:
Fixed lines:
Row scheme:
\[
\begin{array}{c|c}
\rightarrow & 21 \\
21 & 5
\end{array}
\]

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

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

```
PG_2_4_incma_cyclic:
  $ (ORBITER) -v 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
  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:

```plaintext
PG_2_4_incma_singer_sub_3:
▷ $(ORBITER) -v 4 \
▷ ▷ -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 $\mathcal{G}r_k(V)$ be the Grassmannian of $k$-dimensional subspaces of $V$. If $\dim(V) = n$, the notation $\mathcal{G}r_{n,k}$ is used for $\mathcal{G}r_k(V)$. If $V = \mathbb{F}^n_q$, the notation $\mathcal{G}r_{n,k,q}$ is used for $\mathcal{G}r_k(V)$. The order of the set $\mathcal{G}r_{n,k,q}$ can be computed as

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

using the $q$-binomial coefficient.

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

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

$$\text{GR}_322:\text{\texttt{\$(ORBITER) \backslash}}}$$
$$\text{\texttt{\triad{define F -finite_field -q 2 -end \backslash}}}$$
$$\text{\texttt{\triad{with F -do -finite_field_activity \backslash}}}$$
$$\text{\texttt{\triad{cheat_sheet_Gr 3 2 -end}}}$$
$$\text{pdflatex Gr}_322\text{.tex}$$
$$\text{open Gr}_322\text{.pdf}$$

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

$$L_0 = \left[\begin{array}{c} 100 \\ 010 \end{array}\right]$$
$$L_1 = \left[\begin{array}{c} 100 \\ 011 \end{array}\right]$$
$$L_2 = \left[\begin{array}{c} 100 \\ 001 \end{array}\right]$$
$$L_3 = \left[\begin{array}{c} 101 \\ 010 \end{array}\right]$$
$$L_4 = \left[\begin{array}{c} 101 \\ 011 \end{array}\right]$$
$$L_5 = \left[\begin{array}{c} 110 \\ 001 \end{array}\right]$$
$$L_6 = \left[\begin{array}{c} 010 \\ 001 \end{array}\right]$$

The following command illustrates how to rank lines. In th 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.

91
rank_lines:

```bash
$ (ORBITER) -v 2 \
  > define v1 -vector -format 3 \
  > dense "1,0,2,0,1,1,2, 1,0,2,0,0,1,1,2, 1,0,2,2,0,1,2,1" \
  > end \
  > define v2 -vector -format 3 \
  > dense "1,0,0,0,1,0,0, 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 3 F -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:

```bash
$ (ORBITER) -v 2 \
  > define F -finite_field -q 8 -end \
  > define P -projective_space 3 F -end \
  > with P -do \
  > -projective_space_activity \
  >   > -planes_through_line 0 \
  >   > end
```

92
Table 4.2: The partition ordering of monomials of degree 1, 2, 3 and 4 in a plane

<table>
<thead>
<tr>
<th>$h$</th>
<th>mon</th>
<th>vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$X_0$</td>
<td>(1, 0, 0)</td>
</tr>
<tr>
<td>1</td>
<td>$X_1$</td>
<td>(0, 1, 0)</td>
</tr>
<tr>
<td>2</td>
<td>$X_2$</td>
<td>(0, 0, 1)</td>
</tr>
<tr>
<td>3</td>
<td>$X_0^2$</td>
<td>(2, 0, 0)</td>
</tr>
<tr>
<td>4</td>
<td>$X_1^2$</td>
<td>(0, 2, 0)</td>
</tr>
<tr>
<td>5</td>
<td>$X_2^2$</td>
<td>(0, 0, 2)</td>
</tr>
<tr>
<td>6</td>
<td>$X_0X_1$</td>
<td>(1, 1, 0)</td>
</tr>
<tr>
<td>7</td>
<td>$X_0X_2$</td>
<td>(0, 1, 0)</td>
</tr>
<tr>
<td>8</td>
<td>$X_1X_2$</td>
<td>(0, 1, 1)</td>
</tr>
<tr>
<td>9</td>
<td>$X_0X_1X_2$</td>
<td>(1, 1, 1)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$h$</th>
<th>mon</th>
<th>vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$X_0^3$</td>
<td>(3, 0, 0)</td>
</tr>
<tr>
<td>1</td>
<td>$X_1^3$</td>
<td>(0, 3, 0)</td>
</tr>
<tr>
<td>2</td>
<td>$X_2^3$</td>
<td>(0, 0, 3)</td>
</tr>
<tr>
<td>3</td>
<td>$X_0^2X_1$</td>
<td>(2, 1, 0)</td>
</tr>
<tr>
<td>4</td>
<td>$X_0^2X_2$</td>
<td>(2, 0, 1)</td>
</tr>
<tr>
<td>5</td>
<td>$X_0X_1^2$</td>
<td>(1, 2, 0)</td>
</tr>
<tr>
<td>6</td>
<td>$X_1X_2^2$</td>
<td>(0, 2, 1)</td>
</tr>
<tr>
<td>7</td>
<td>$X_0X_1X_2$</td>
<td>(1, 0, 2)</td>
</tr>
<tr>
<td>8</td>
<td>$X_0X_1X_2^2$</td>
<td>(0, 1, 2)</td>
</tr>
<tr>
<td>9</td>
<td>$X_1X_2^2$</td>
<td>(0, 1, 1)</td>
</tr>
<tr>
<td>10</td>
<td>$X_0X_1X_2$</td>
<td>(2, 1, 1)</td>
</tr>
<tr>
<td>11</td>
<td>$X_0^2X_1X_2$</td>
<td>(1, 2, 1)</td>
</tr>
<tr>
<td>12</td>
<td>$X_0^2X_1X_2$</td>
<td>(1, 1, 2)</td>
</tr>
</tbody>
</table>

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 $V = v(p_1, \ldots, p_r)$. 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 $(n+1)(q-1)$ [29]. 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 $X^3 + 3Z^3 - Y^2Z + ZX = 0$. Using $X_0, X_1, X_2$ instead of $X, Y, Z$ yields

$$X_0^3 + 3X_2^3 + 10X_1X_2 + X_0X_2^2 = 0.$$
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

\[
\text{EC}11\_\text{EQUATION} = "1,0,3,0,0,0,10,1,0,0"
\]

\[
\text{EC}11\_\text{txt}: \\
\text{\texttt{\$\{(ORBITER) -v 2 \}} \\
\text{\texttt{\textbackslash \texttt{-define F -finite_field -q 11 -end \}} \\
\text{\texttt{-define R -polynomial_ring -field F \}} \\
\text{\texttt{-number_of_variables 3 \}} \\
\text{\texttt{-homogeneous_of_degree 3 \}} \\
\text{\texttt{-end \}} \\
\text{\texttt{-define P -projective_space 2 F -end \}} \\
\text{\texttt{-define EC -geometric_object P \}} \\
\text{\texttt{-projective_variety R \}} \\
\text{\texttt{-end \}} \\
\text{\texttt{-with EC -do -combinatorial_object_activity -save \}} \\
\text{\texttt{-end \}}
\]

creates the algebraic set associated to the cubic curve \(y^2 = x^3 + x + 3\) in \(\text{PG}(2, 11)\). It turns out that there are exactly 18 points over \(\mathbb{F}_{11}\) (cf. Figure 4.6). Suppose we want to create the Hirschfeld surface with equation

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

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

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

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

\[
\text{HIRSCHFELD\_SURFACE\_EQUATION} = "0,0,0,0,0,0,1,0,1,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0"
\]

\[
\text{Hirschfeld\_surface\_q4\_txt}: \\
\text{\texttt{\$\{(ORBITER) -v 2 \}} \\
\text{\texttt{\textbackslash \texttt{-define F -finite_field -q 4 -end \}} \\
\text{\texttt{-define R -polynomial_ring -field F \}} \\
\text{\texttt{-number_of_variables 4 \}}
\]

94
Table 4.3: The Orbiter ordering of monomials of degree 1, 2 and 3 in PG(3, q)
Figure 4.6: Elliptic curve $y^2 \equiv x^3 + x + 3 \mod 11$

```
> > > -homogeneous_of_degree 3 \
> > > -end \
> > -define P -projective_space 3 F -end \
> > -define H4 -geometric_object P \
> > > -projective_variety R \
> > > > "Hirschfeld_surface_q4" \
> > > > "Hirschfeld\_surface\_q4" \
> > > > $(HIRSCHFELD_SURFACE_EQUATION) \
> > > > -end \
> > > -with H4 -do -combinatorial_object_activity -save \
> > > -end
```

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

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

```
ENDRASS_SPARSE="\n6,0,4,4,2,7,5,9,6,20,6,23,1,25,3,30,1,32,3,34,4,56,6,59,1,61,6,66, \n2,68,6,70,3,77,2,79,6,83,6,120,2,123,5,125,3,130,1,132,3,134,3,141, \n2,143,6,147,3,156"
```

Endrass_F7.txt:
Suppose we want to create the monomials of degree 8 in 4 variables. We use an integer program to do so. The following command also applies the unix sort command to sort the monomials:

```plaintext
$ (ORBITER) -v 2 \n  -define F -finite_field -q 7 -end \n  -define R -polynomial_ring -field F \n  -number_of_variables 4 \n  -homogeneous_of_degree 8 \n  -end \n  -define eqn -vector -field F -sparse 165 \n  -ENDRASS -end \n  -define P -projective_space 3 F -end \n  -define Endrass_F7 -geometric_object P \n  -projective_variety R \n  "Endrass_F7" \n  -end \n  with Endrass_F7 -do \n  -combinatorial_object_activity -save \n  -end
```

There are 165 monomials. They are listed in the file `octic_monomials_sorted.txt`. 

```plaintext
$ (ORBITER) -v 4 \n  -define A -vector -format 1 -dense "1,1,1,1" -end \n  -define D -diophant \n  -label octic_monomials \n  -coefficient_matrix A \n  -RHS "8,8,1" \n  -x_min_global 0 -x_max_global 8 \n  -end \n  -with D -do \n  -diophant_activity -solve_mckay \n  -end
```

sort -r octic_monomials.sol >octic_monomials_sorted.txt

There are 165 monomials. They are listed in the file `octic_monomials_sorted.txt`. 

97
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 $\mathfrak{Gr}_{4,2}(V)$, additional functionality is possible. The Plücker coordinates allow to identify $\mathfrak{Gr}_{4,2}(V)$ with the $Q^+(5, q)$ quadric.

The command

```
GR.4.2.2:
▷ $(ORBITER) -v 2 \n▷ ▷ -define F -finite_field -q 2 -end \n▷ ▷ -with F -do -finite_field_activity \n▷ ▷ ▷ -cheat_sheet_Gr 4 2 -end
▷ pdflatex Gr.4.2.2.tex
▷ open Gr.4.2.2.pdf
```

creates the elements of $\mathfrak{Gr}_{4,2,2}$ and lists them together with their Plücker coordinates. The following output is shortened:

```
There are 35 lines:

$L_0 = \begin{bmatrix} 1000 \\ 0100 \end{bmatrix} = \text{Pl}(1,0,0,0,0,0)$
$L_1 = \begin{bmatrix} 1000 \\ 0110 \end{bmatrix} = \text{Pl}(1,0,1,0,0,0)$
$L_2 = \begin{bmatrix} 1000 \\ 0101 \end{bmatrix} = \text{Pl}(1,0,0,0,1,0)$
$L_3 = \begin{bmatrix} 0100 \\ 0101 \end{bmatrix} = \text{Pl}(0,1,0,0,0,0)$
```

The Plücker coordinates satisfy

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

and hence belong to the quadric $Q^+(5, q)$. This quadric is also known as the Klein quadric. 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 $\text{PG}(3,q)$. In the example of $\text{PG}(3,2)$ this yields the following list (output shortened):
\[
0 = \mathbf{P}(1, 0, 0, 0, 0) = L_0 = \begin{bmatrix} 1000 \\ 0100 \end{bmatrix} \\
1 = \mathbf{P}(0, 1, 0, 0, 0) = L_{34} = \begin{bmatrix} 0010 \\ 0001 \end{bmatrix} \\
2 = \mathbf{P}(0, 0, 1, 0, 0) = L_4 = \begin{bmatrix} 1000 \\ 0010 \end{bmatrix} \\
\vdots \\
34 = \mathbf{P}(0, 1, 1, 1, 1) = L_{26} = \begin{bmatrix} 1101 \\ 0011 \end{bmatrix}
\]
<table>
<thead>
<tr>
<th>Type</th>
<th>Quadratic Form</th>
<th># Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q^+(n, q)$ Hyperbolic ($n$ is odd)</td>
<td>$\frac{n-1}{2} \sum_{i=0}^{n-1} X_{2i}X_{2i+1}$</td>
<td>$\frac{(q^{(n+1)/2} - 1)(q^{(n-1)/2} + 1)}{q - 1}$</td>
</tr>
<tr>
<td>$Q^-(n, q)$ Elliptic ($n$ is odd)</td>
<td>$p(X_{n-1}, X_n) + \sum_{i=0}^{n-1} X_{2i}X_{2i+1}$</td>
<td>$\frac{(q^{(n+1)/2} + 1)(q^{(n-1)/2} - 1)}{q - 1}$</td>
</tr>
<tr>
<td>$Q(n, q)$ Parabolic ($n$ is even)</td>
<td>$X_n^2 + \sum_{i=0}^{n-1} X_{2i}X_{2i+1}$</td>
<td>$\frac{q^n - 1}{q - 1}$</td>
</tr>
</tbody>
</table>

Table 4.4: Nondegenerate Quadrics in $\text{PG}(n, q)$ and the canonical form adopted in Orbiter

### 4.7 Orthogonal Spaces

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

```
-orthogonal_space \( \epsilon \) \( d \) \( q \) -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:
▷ $\$(ORBITER) -v 2 \$
▷ ▷ -define F -finite_field -q 2 -end \$
▷ ▷ -define O -orthogonal_space 1 4 F -without_group -end \$
▷ ▷ ▷ -with O -do -orthogonal_space_activity \$
▷ ▷ ▷ ▷ -cheat_sheet_orthogonal -end
▷ pdflatex 0.1_4.2_report.tex
▷ open 0.1_4.2_report.pdf
```
<table>
<thead>
<tr>
<th>Modifier</th>
<th>Arguments</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-label_txt</td>
<td>L</td>
<td>Set the ascii-label of the space. The label is used for things like file names etc. A default label will be used if this option is not given.</td>
</tr>
<tr>
<td>-label_tex</td>
<td>L</td>
<td>Set the tex-label of the space. The label is used within latex reports. A default label will be used if this option is not given.</td>
</tr>
<tr>
<td>-without_group</td>
<td></td>
<td>Do not create the orthogonal group.</td>
</tr>
</tbody>
</table>

Table 4.5: Command options to create an orthogonal space

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 0 -orthogonal_space 0 5 F -without_group -end \
  >   -with 0 -do -orthogonal_space_activity \
  >   > -export_point_line_incidence_matrix \
  >   > -end \
  > $(ORBITER) -v 2 \
  >   -define all_one_r -vector -repeat 1 15 -end \
  >   -define all_one_c -vector -repeat 1 15 -end \
  >   -draw_matrix \
  >   > -input_csv_file 0.5.2_incidence_matrix.csv \
  >   > -box_width 20 -bit_depth 8 \
  >   > -partition 2 \
  >   > > all_one_r all_one_c \
  >   > -end \
  > open 0.5.2_incidence_matrix_draw.bmp
```

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

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

```
  -define 0 -orthogonal_space 1 6 2 -end
```
Figure 4.7: Incidence matrix of $Q(4, 2)$

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

The command

```
Op_6.2:
▷ $(ORBITER) -v 2 \ 
▷ ▷ -define F -finite_field -q 2 -end \ 
▷ ▷ -define O -orthogonal_space 1 6 F -without_group -end \ 
▷ ▷ -with O -do -orthogonal_space_activity \ 
▷ ▷ ▷ -cheat_sheet_orthogonal -end 
▷ pdflatex 0_1_6_2_report.tex
▷ open 0_1_6_2_report.pdf
```

produces a cheat sheet for the quadric $Q^+(5, 2)$. This is the Klein quadric from Section 4.6. Orbiter produces the following output. At the top is the tactical decomposition of the incidence matrix between points and lines with respect to a hyperbolic pair. After that, the points and lines are listed (output shortened):
<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</td>
<td>descr</td>
<td>Creates a BLT-set of $Q(4,q)$. See Section 12.4.</td>
</tr>
</tbody>
</table>

Table 4.6: Activities related to orthogonal spaces

<table>
<thead>
<tr>
<th>→</th>
<th>9</th>
<th>36</th>
<th>18</th>
<th>18</th>
<th>6</th>
<th>9</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>3</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>↓</th>
<th>9</th>
<th>36</th>
<th>18</th>
<th>18</th>
<th>6</th>
<th>9</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3</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>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

The number of points is 35 points:
- $P_0 = (1,0,0,0,0,0)$
- $P_1 = (0,1,0,0,0,0)$
- $P_2 = (0,0,1,0,0,0)$
- $P_3 = (1,0,1,0,0,0)$
- $P_4 = (0,1,1,0,0,0)$
- $P_5 = (0,0,0,1,0,0)$
- $P_6 = (1,0,0,1,0,0)$
Orbiter has enumerators for points and lines in orthogonal spaces. For small spaces, the cheat sheet lists points and lines in the Orbiter ordering. Creating the groups can be expensive. For large spaces, it may be necessary to disable the group using the \texttt{-without\_group} option. The command

\texttt{Op\_6\_64\_line\_rank\_problem:}

\begin{verbatim}
▷ $(ORBITER) -v 4 \
\end{verbatim}
computes the Orbiter rank of the line through the points with rank 15447347 and 15225451, respectively. The rank of the line is 16767254. These ranks refer to the orthogonal geometry. They are different from the ranks of points and lines in projective spaces.

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

```
Op_6.64_report:
  $(ORBITER) -v 4 \
  -define F -finite_field -q 64 -end \
  -define O -orthogonal_space 1 6 F -without_group -end \
  -with O -do -orthogonal_space_activity \
  -unrank_line_through_two_points 15447347 15225451 \
  -end
```

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

The group is not available.
The quadratic form is:

$$X_0X_1 + X_2X_3 + X_4X_5 = 0$$

<table>
<thead>
<tr>
<th>→</th>
<th>16769025</th>
<th>1090252800</th>
<th>532350</th>
<th>532350</th>
<th>130</th>
<th>4225</th>
<th>4225</th>
</tr>
</thead>
<tbody>
<tr>
<td>16511040</td>
<td>65</td>
<td>4160</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>266175</td>
<td>0</td>
<td>4096</td>
<td>128</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>266175</td>
<td>0</td>
<td>4096</td>
<td>0</td>
<td>128</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4225</td>
<td>3969</td>
<td>0</td>
<td>126</td>
<td>126</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>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>
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:
\$\$(ORBITER) -v 2 \$
\$\$-define F -finite_field -q 4 -end \$
\$\$-with F -do -finite_field_activity \$
\$\$\$ -cheat_sheet/hermitian 2 -end
\$\$pdflatex H_2.4.tex
\$\$open H_2.4.pdf
```

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

The Hermitian variety $H(2, 4)$ 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:
\$\$(ORBITER) -v 2 \$
\$\$-define F -finite_field -q 4 -end \$
\$\$-with F -do -finite_field_activity \$
\$\$\$ -cheat_sheet/hermitian 3 -end
\$\$pdflatex H_3.4.tex
\$\$open H_3.4.pdf
```

produces a cheat sheet for the variety $H(3, 4)$.
The Hermitian variety $H(3, 4)$ contains 45 points:

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

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 \(\text{PG}(3, 2)\). The command

\[
\text{fix\_structure\_2A}:
\]

\[
\begin{align*}
\text{fix\_structure\_2A} & : \\
\text{define F -finite\_field -q 4 -end} & : \\
\text{define P -projective\_space 3 F -end} & : \\
\text{with P -do} & : \\
\text{projective\_space\_activity} & : \\
\text{cheat\_sheet\_for\_decomposition\_by\_element\_PG 1} & : \\
\text{"1,0,0,0, 0,1,0,0, 0,0,1,0, 0,0,0,1, 1"} & : \\
\text{fix\_structure\_2A} & : \\
\text{-end} & :
\end{align*}
\]

\[
\text{pdflatex fix\_structure\_2A.tex} \\
\text{open fix\_structure\_2A.pdf}
\]

can be used.

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

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

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

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 m n matrix</td>
<td>Compute the automorphism group of a linear code using Nauty. See Section 10.</td>
</tr>
<tr>
<td>-map</td>
<td>label parameters</td>
<td>evaluate a formula using the given parameters</td>
</tr>
<tr>
<td>-analyze_del_Pezzo_surface</td>
<td>label parameters</td>
<td></td>
</tr>
<tr>
<td>-cheat_sheet_for_decomposition_by_element_PG</td>
<td>power elt fname</td>
<td>Analyzes the orbit structure of the cyclic group generated by the given element in the action on PG(n,q).</td>
</tr>
<tr>
<td>-cheat_sheet_for_decomposition_by_subgroup</td>
<td>label descr</td>
<td>Analyzes the orbit structure of the subgroup H in the action on PG(n,q). The subgroup must be a linear group, and the description of H must come from the commands from Section 5.2.</td>
</tr>
<tr>
<td>-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><code>-classify_surfaces_through_arcs_and_two_lines</code></td>
<td></td>
<td>See Section 7.3.</td>
</tr>
<tr>
<td><code>-test_nb_Eckardt_points</code></td>
<td>nbE</td>
<td>See Section 7.3.</td>
</tr>
<tr>
<td><code>-classify_surfaces_through_arcs_and_trihedral_pairs</code></td>
<td></td>
<td>See Section 7.3.</td>
</tr>
<tr>
<td><code>-sweep</code></td>
<td>fname</td>
<td></td>
</tr>
<tr>
<td><code>-sweep_4</code></td>
<td>fname</td>
<td>surface-descr</td>
</tr>
<tr>
<td><code>-sweep_4_27</code></td>
<td>fname</td>
<td>surface-descr</td>
</tr>
<tr>
<td><code>-six_arcs_not_on_conic</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>-filter_by_nb_Eckardt_points</code></td>
<td>nbE</td>
<td></td>
</tr>
<tr>
<td><code>-surface_quartic</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>-surface_clebsch</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>-surface_codes</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>-trihedral_control</code></td>
<td>poset-control</td>
<td></td>
</tr>
<tr>
<td><code>-trihedra2_control</code></td>
<td>poset-control</td>
<td></td>
</tr>
<tr>
<td><code>-control_six_arcs</code></td>
<td>poset-control</td>
<td></td>
</tr>
<tr>
<td><code>-make_gilbert_varshamov_code</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>$k$ control</td>
<td>See Section 12.1.</td>
</tr>
<tr>
<td><code>-classify_semifields</code></td>
<td>descr</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>fname-mask $N$</td>
<td>Classify quartic curves using Nauty.</td>
</tr>
<tr>
<td><code>-classify_quartic_curves_with_substructure</code></td>
<td>fname-mask $N k d$ fname</td>
<td>Classify quartic curves using substructure algorithm.</td>
</tr>
<tr>
<td><code>-set_stabilizer</code></td>
<td>$k$ fname-mask $N$ col-label</td>
<td>Compute canonical form of sets using the substructure algorithm.</td>
</tr>
<tr>
<td><code>-lift_skew_hexagon</code></td>
<td>text</td>
<td>Lift a skew-hexagon.</td>
</tr>
<tr>
<td><code>-lift_skew_hexagon_with_polarity</code></td>
<td>polarity</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>$sz d d_{\text{min}} s$</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>$sz d d_{\text{min}} s t T$</td>
<td>Finds arcs with the two given sets as $s$-lines and $t$-lines, respectively.</td>
</tr>
<tr>
<td><code>-arc_with_three_given_sets_of_lines_after_dualizing</code></td>
<td>$sz d d_{\text{min}} s t T u U$</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 $\text{PG}(2,q)$. 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>$\epsilon$, $N$, $b$, $t_{\text{min}}$, $t_{\text{max}}$, function</td>
<td>Creates at least $N$ points on a continuous curve given by “function”. Consecutive points are no more than $\epsilon$ apart. The function must be in terms of a parameter $t$. The values of $t$ are taken from the interval $[t_{\text{min}}, t_{\text{max}}]$.</td>
</tr>
<tr>
<td>-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
trans:
\$\{\text{ORBITER}\} -v 5 \$
\$\{\text{ORBITER}\} \text{-define } F \text{-finite_field -q 16 -end} \$
\$\{\text{ORBITER}\} \text{-define } P \text{-projective_space 3 F -end} \$
\$\{\text{ORBITER}\} \text{-with } P \text{-do} \$
\$\{\text{ORBITER}\} \text{-projective_space_activity} \$
\$\{\text{ORBITER}\} \text{-move_two_lines_in_hyperplane_stabilizer_text} \$
\$\{\text{ORBITER}\} \text{-"1,0,0,0, 0,0,0,1" "1,1,0,2, 0,0,1,0"} \$
\$\{\text{ORBITER}\} \text{-"1,0,0,0, 0,0,0,1" "0,1,0,1, 0,0,1,0"} \$
\$\{\text{ORBITER}\} \text{-end} \$

computes a projectivity (transvection) to do so:

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
\delta^{14} & 0 & 0 & \delta^{14}
\end{bmatrix}
\]

Here, $\delta$ is the primitive element in the built-in field $F_{16}$, 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:

\[
x*x*x*x+y*y*y*y+z*z*z*z+8*x*x*y*y+8*x*x*z*z+8*y*y*z*z
\]
for $f_3$ and

\[
x*x*x*x+y*y*y*y+z*z*z*z-x*x*y*y
\]
for $f_4$. The following command can be used to produce a report on the two surfaces over the field $F_{13}$.

\$\{\text{ORBITER}\} \text{-define } F \text{-finite_field -q 13 -end} \$

114
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 \texttt{-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 \n  -define F -finite_field -q 8 -end \n  -define P -projective_space 3 F -end \n  -define O -geometric_object P \n  -elliptic_quadric_ovoid \n  -end \n  -with O -do -combinatorial_object_activity -save \n  -end
```

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

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

The Edge curve is given by the equation

\[ X^4 - Y^4 - Z^4 + 2f^2Y^2Z^2 + 4fX^2YZ = 0 \]

where \( f \) is a primitive element of \( \mathbb{F}_q \). Let us pick \( q = 17 \). The next example creates the Edge curve in PG(2, 17) and saves it to file. The equation is encoded using the ordering of quartic monomials from Table 4.2.

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

EDGE_CURVE_Q17_AS_POINTS="4, 7, 16, 19, 20, 23, 32, 35, 89, 100, 244, 251"
```
<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 Adelaide hyperoval</td>
</tr>
<tr>
<td>-translation</td>
<td>i</td>
<td>Create the translation hyperoval with exponent i</td>
</tr>
<tr>
<td>-Segre</td>
<td></td>
<td>Create the Segre hyperoval</td>
</tr>
<tr>
<td>-Payne</td>
<td></td>
<td>Create the Payne hyperoval</td>
</tr>
<tr>
<td>-Cherowitzo</td>
<td></td>
<td>Create the Cherowitzo hyperoval</td>
</tr>
<tr>
<td>-OKeefe_Penttila</td>
<td></td>
<td>Create the O’Keefe, Penttila hyperoval</td>
</tr>
<tr>
<td>-BLT_database</td>
<td>k</td>
<td>Create the $k$th BLT-set of order $q$ from the database ($k = 0, 1, \ldots$)</td>
</tr>
<tr>
<td>-elliptic_quadric_ovoind</td>
<td></td>
<td>Create an elliptic quadric ovoid in PG(3, $q$).</td>
</tr>
<tr>
<td>-ovoid_ST</td>
<td></td>
<td>Create the Suzuki Tits ovoid in PG(3, $q$). Here, $q = 2^{2r+1}$.</td>
</tr>
<tr>
<td>-Baer</td>
<td></td>
<td>Create the (standard) Baer subgeometry</td>
</tr>
<tr>
<td>-orthogonal</td>
<td>$\epsilon$</td>
<td>Create the $Q^\epsilon(n, q)$ quadric</td>
</tr>
<tr>
<td>-hermitian</td>
<td></td>
<td>Create the Hermitian variety given by $\sum_{i=0}^{n} X_i^{\sqrt{q}+1} = 0$</td>
</tr>
<tr>
<td>-cuspidal_cubic</td>
<td></td>
<td>Create the cuspidal cubic $(s^3, ts^2, t^3)$ in 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 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 $A$ [8]</td>
</tr>
<tr>
<td>-ttp_construction_A_hyperoval</td>
<td></td>
<td>Create the twisted tensor product code of type $A$ [8]</td>
</tr>
<tr>
<td>-ttp_construction_B</td>
<td></td>
<td>Create the twisted tensor product code of type $B$ [8]</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 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 $l \ d \ n \ C_1 \ldots \ C_n$</td>
<td></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 $l \ r \ d \ C$</td>
<td></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>Create the BLT-set with ranks in PG($n, q$) instead of orthogonal point ranks</td>
</tr>
<tr>
<td>-BLT_in_PG</td>
<td></td>
<td>Create the BLT-set with ranks in 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
FILE_Q17="orbit,curve,pts_on_curve,bitangents,go\n0,"$(EDGE_CURVE_Q17_EQUATION)""$(EDGE_CURVE_Q17_AS_POINTS)""",-1\n\nEdge_curve_17:
   ▶ $(ORBITER) -v 2 
   ▶ ▶ -define F -finite_field -q 17 -end 
   ▶ ▶ -define R -polynomial_ring -field F 
   ▶ ▶ ▶ -number_of_variables 3 
   ▶ ▶ ▶ -homogeneous_of_degree 4 
   ▶ ▶ ▶ -end 
   ▶ ▶ -define P -projective_space 2 F -end 
   ▶ ▶ -define C -geometric_object P 
   ▶ ▶ ▶ -projective_variety R 
   ▶ ▶ ▶ ▶ "Edge_q17" "Edge\_q17" 
   ▶ ▶ ▶ ▶ $(EDGE_CURVE_Q17_EQUATION) 
   ▶ ▶ ▶ -end 
   ▶ ▶ -with C -do -combinatorial_object_activity -save 
   ▶ ▶ -end

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

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 2 F -end 
   ▶ ▶ -define C -geometric_object P 
   ▶ ▶ ▶ -projective_variety R 
   ▶ ▶ ▶ ▶ "Edge_q17" "Edge\_q17" 
   ▶ ▶ ▶ ▶ $(EDGE_CURVE_Q17_EQUATION) 
   ▶ ▶ ▶ -end 
   ▶ ▶ -with C -do 
   ▶ ▶ -combinatorial_object_activity 
   ▶ ▶ ▶ -line_type 
   ▶ ▶ ▶ -end 
   ▶ ▶ -print_symbols

The line type is

\( (4^6, 2^{30}, 1^{132}, 0^{139}) \)
This means that there are 6 4-secants, 30 2-secants, 132 tangent lines, and 139 external lines to the curve.
Chapter 5

Group Theory

5.1 Permutation Groups

Permutation groups can be represented on a computer using the technique of stabilizer chains, or Sims chains (cf. [35, 59]). 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 an 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 \\
 ▶ pdflatex 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 \
```
Next, let us consider the symmetric group Sym(\(n\)). The following command creates Sym(3):

\[
\text{Symmetric}_3: \\
\text{\$\text{ORBITER} -v 3 \ \\
\text{-define G -permutation_group -symmetric_group 3 -end \ \\
\text{-with G -do \ \\
\text{-group_theoretic_activity \ \\
\text{-report \ \\
\text{-end \ \\
\text{pdflatex Perm3_report.tex \ \\
\text{open Perm3_report.pdf}}}
\]

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

\[0\]

1

\[1\]

2

\[2\]
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 \\n▷▷ -define G -permutation_group -symmetric_group 3 -end \\n▷▷ -with G -do \\n▷▷ -group_theoretic_activity \\

124
The next command produces a graphical representation of the elements of the symmetric group Sym(3), shown in Figure 5.3.

```
$ (ORBITER) -v 2 
  -define all_one_r -vector -repeat 1 6 -end 
  -define all_one_c -vector -repeat 1 6 -end 
  -draw_matrix 
  -input_csv_file Perm3_group_table.csv 
  -box_width 50 -bit_depth 24 
  -partition 3 all_one_r all_one_c 
  -end 
open Perm3_group_table_draw.bmp
```

```
$ (ORBITER) -v 3 
  -define G -permutation_group -symmetric_group 3 -end 
  -with G -do 
  -group_theoretic_activity 
  -save_elements_csv "Symmetric3_elts.csv" 
  -end 
$ (ORBITER) -v 2 
  -define Sym3_elts -vector -load_csv_data_column 
  -Symmetric3_elts.csv 1 -end 
  -save_matrix_csv Sym3_elts 
$ (ORBITER) -v 2 
  -define all_one_r -vector -repeat 1 6 -end 
```

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

Symmetric_3_elements:
```
$ (ORBITER) -v 3 
  -define G -permutation_group -symmetric_group 3 -end 
  -with G -do 
  -group_theoretic_activity 
  -save_elements_csv "Symmetric3_elts.csv" 
  -end 
$ (ORBITER) -v 2 
  -define Sym3_elts -vector -load_csv_data_column 
  -Symmetric3_elts.csv 1 -end 
  -save_matrix_csv Sym3_elts 
$ (ORBITER) -v 2 
  -define all_one_r -vector -repeat 1 6 -end 
```
\begin{verbatim}
\define all_one_c -vector -repeat 1 3 -end \n\define -draw_matrix \n\define -input_csv_file Sym3_elts_matrix.csv \n\define -box_width 50 -bit_depth 8 \n\define -partition 3 \n\define all_one_r all_one_c \n\define -end
\define open Sym3_elts_matrix_draw.bmp
\end{verbatim}
5.2 Linear Groups

Orbiter provides support for matrix groups and their various permutation representationes. For background information about the classical groups of matrices over finite fields, see cf. [64]. 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

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

The identity element is \( I_0 \), where \( I \) is the identity matrix and 0 is the residue class modulo \( h \). The rules for multiplication and inversion in the collineation group are given as
\[
M_k \cdot N_l = \left( M \cdot N^{\Phi^{-k}} \right)_{k+l},
\]
\[ (M_k)^{-1} = \left( (M^{-1})^{\Phi^k} \right)_{-k}. \] (5.3)

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

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

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

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>$Gr_1((\mathbb{F}_q^d)^n)$ 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:
- $(ORBITER) -v 2 \$
- $-define F -finite_field -q 2 -end \$
- $-define G -linear_group -PGL 4 F -end \$
- $-with G -do \$
- $-group_theoretic_activity \$
- $-report \$
- $-end$
- `pdflatex PGL_4.2_report.tex`
- `open PGL_4.2_report.pdf`
```

creates the group $PGL(4,2)$ acting on the 15 elements of $Gr_1(\mathbb{F}_2^4)$. 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>GL($n, q$)</td>
</tr>
<tr>
<td>-GGL</td>
<td>$n \ q$</td>
<td>$\Gamma L(n, q)$</td>
</tr>
<tr>
<td>-SL</td>
<td>$n \ q$</td>
<td>SL($n, q$)</td>
</tr>
<tr>
<td>-SSL</td>
<td>$n \ q$</td>
<td>$\Sigma L(n, q)$</td>
</tr>
<tr>
<td>-PGL</td>
<td>$n \ q$</td>
<td>PGL($n, q$)</td>
</tr>
<tr>
<td>-PGGL</td>
<td>$n \ q$</td>
<td>PGL($n, q$)</td>
</tr>
<tr>
<td>-PSL</td>
<td>$n \ q$</td>
<td>PSL($n, q$)</td>
</tr>
<tr>
<td>-PSSL</td>
<td>$n \ q$</td>
<td>PSL($n, q$)</td>
</tr>
<tr>
<td>-AGL</td>
<td>$n \ q$</td>
<td>AGL($n, q$)</td>
</tr>
<tr>
<td>-AGGL</td>
<td>$n \ q$</td>
<td>AGL($n, q$)</td>
</tr>
<tr>
<td>-ASL</td>
<td>$n \ q$</td>
<td>ASL($n, q$)</td>
</tr>
<tr>
<td>-ASSL</td>
<td>$n \ q$</td>
<td>ASL($n, q$)</td>
</tr>
<tr>
<td>-PGO</td>
<td>$n \ q$</td>
<td>PGO($n, q$)</td>
</tr>
<tr>
<td>-PGOp</td>
<td>$n \ q$</td>
<td>$\text{PGO}^+(n, q)$</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>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$_d \ q \ _{\text{wr}} \ _{\text{Sym}} \ n$</td>
<td>$d \ q \ n$</td>
<td>GL($d, q$) $\wr$ Sym($n$)</td>
</tr>
</tbody>
</table>

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

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

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

\[
\begin{align*}
0 &= (1, 0, 0, 0) \\
1 &= (0, 1, 0, 0) \\
2 &= (0, 0, 1, 0) \\
3 &= (0, 0, 0, 1) \\
4 &= (1, 1, 1, 1) \\
5 &= (1, 1, 0, 0) \\
6 &= (1, 0, 1, 0) \\
7 &= (0, 1, 1, 0) \\
8 &= (1, 1, 1, 0) \\
9 &= (1, 0, 0, 1) \\
10 &= (0, 1, 0, 1) \\
11 &= (1, 1, 0, 1) \\
12 &= (0, 0, 1, 1) \\
13 &= (1, 0, 1, 1) \\
14 &= (0, 1, 1, 1)
\end{align*}
\]

The group is a matrix group.
The group acts on projective space PG(3,2)
\( q = 2 \)
\( p = 2 \)
\( e = 1 \)
\( n = 3 \)
Number of points = 15
Number of lines = 35
Number of lines on a point = 7
Number of points on a line = 3

The finite field $\mathbb{F}_2$

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

<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>DNE</td>
</tr>
<tr>
<td>1</td>
<td>1 = 1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>DNE</td>
</tr>
</tbody>
</table>

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

$\begin{array}{ccc}
\cdot & 1 & \\
1 & 1 & \\
\end{array}$

$1^0 \equiv 1$

$1^1 \equiv 1$

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:

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

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

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>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

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.

**PGL.4.2.export:**

```bash
$(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 PGL_4_2_report.tex`

`open PGL_4_2_report.pdf`

The file PGL_4_2.makefile is created:

**PGL.4.2.generated:**

```bash
$(ORBITER) -v 2 \
  -define gens -vector -file PGL_4_2 gens.csv -end \n  -define G -permutation_group \n  -bsgs PGL_4_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 PGL_4_2_gens.csv:

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

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

\texttt{L\_5\_3:}
\begin{itemize}
  \item \texttt{\$(ORBITER) -v 2 \}
  \item \texttt{\quad -define F -finite_field -q 3 -end \}
  \item \texttt{\quad -define G -linear_group -PSL 5 F -end \}
  \item \texttt{\quad -with G -do \}
  \item \texttt{\quad -group\_theoretic\_activity \}
  \item \texttt{\quad -report \}
  \item \texttt{\quad -end \}
  \item \texttt{pdflatex PSL\_5\_3\_report.tex}
  \item \texttt{open PSL\_5\_3\_report.pdf}
\end{itemize}

creates PSL(5,3) of order 237783237120.

The command

\texttt{PSP\_4\_4:}
\begin{itemize}
  \item \texttt{\$(ORBITER) -v 2 \}
  \item \texttt{\quad -define F -finite_field -q 4 -end \}
  \item \texttt{\quad -define G -linear_group -PGL 4 F \}
  \item \texttt{\quad -symplectic_group \}
  \item \texttt{\quad -end \}
  \item \texttt{\quad -with G -do \}
  \item \texttt{\quad -group\_theoretic\_activity \}
  \item \texttt{\quad -report \}
  \item \texttt{\quad -end \}
  \item \texttt{pdflatex PGL\_4\_4\_Sp\_4\_4\_report.tex}
  \item \texttt{open PGL\_4\_4\_Sp\_4\_4\_report.pdf}
\end{itemize}

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

The command

\texttt{PGO\_5\_2:}
\begin{itemize}
  \item \texttt{\$(ORBITER) -v 2 \}
\end{itemize}
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:

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

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

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

\[
\begin{align*}
0 &= (0, 1, 0, 0, 0) \\
1 &= (0, 0, 1, 0, 0) \\
2 &= (0, 0, 0, 1, 0) \\
3 &= (0, 1, 0, 1, 0) \\
4 &= (0, 0, 1, 1, 0) \\
5 &= (0, 0, 0, 0, 1) \\
6 &= (0, 1, 0, 0, 1) \\
7 &= (0, 0, 1, 0, 1) \\
8 &= (0, 1, 1, 1, 1) \\
9 &= (1, 1, 1, 0, 0) \\
10 &= (1, 1, 1, 1, 0) \\
11 &= (1, 1, 1, 0, 1) \\
12 &= (1, 0, 0, 1, 1) \\
13 &= (1, 1, 0, 1, 1) \\
14 &= (1, 0, 1, 1, 1)
\end{align*}
\]

The group is a matrix group.
The base action is on projective space \( \text{PG}(4, 2) \)
\( q = 2 \)
\( p = 2 \)
\( e = 1 \)
\( n = 4 \)
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 \( \mathbb{F}_2 \)

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

\[
\begin{array}{cccccc}
\hline
i & \gamma_i & -\gamma_i & \gamma_i^{-1} & \log_\alpha(\gamma_i) & \alpha^i & Z_i \\
\hline
0 & 0 = 0 & 0 & \text{DNE} & \text{DNE} & 1 & \text{DNE} \\
1 & 1 = 1 & 1 & \text{DNE} & \text{DNE} & 1 & \text{DNE} \\
\hline
\end{array}
\]

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

**Base and Stabilizer Chain**

Group order 720
$tl=15, 8, 3, 1, 1, 2,$
Base: $(0, 1, 2, 3, 4, 5)$

Strong generators for a group of order 720:

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

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 1 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 \n▷ ▷ -define F -finite_field -q 2 -end \n▷ ▷ -define G -linear_group -PGL 6 F \n▷ ▷ ▷ -symplectic_group \n▷ ▷ -end \n▷ ▷ -with G -do \n▷ ▷ -group_theoretic_activity \n
143
The group PGO(7, 2), isomorphic to PSp(6, 2), can be created using the following command:

```
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
```

`pdflatex PGO_7_2_report.tex`
`open PGO_7_2_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 $F_{q^n}$ over $F_q$</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 \texttt{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 \
```

146
The activity `-report` produces a report for the cyclic group, shown below:

```
Stabilizer chain

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

Basic Orbit 0

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

Basic orbit 0 has size 13
```

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"
```

147
It is possible to create a permutation group as a subgroup of the symmetric group, using the known base for the symmetric group. Because the base of the symmetric group is large, this way of creating the group is less efficient than creating the group with a known (small) base. Here is an example. We create $C_{13}$ as a subgroup of Sym(13).

```plaintext
C13_as_subgroup:
▷ $(ORBITER) -v 2 \n ▷ ▷ -define G -permutation_group -symmetric_group 13 \n ▷ ▷ ▷ -subgroup_by_generators C13 13 1 $(GEN_C13) -end \n ▷ ▷ -with G -do \n ▷ ▷ -group_theoretic_activity \n ▷ ▷ ▷ -export_orbiter \n ▷ ▷ -end \n ▷ -with G -do \n ▷ ▷ -group_theoretic_activity \n ▷ ▷ ▷ -report \n ▷ ▷ -end \n ▷ -with G -do \n ▷ ▷ -group_theoretic_activity \n ▷ ▷ ▷ -save_elements_csv "C13_elts.csv" \n ▷ ▷ -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 \n ▷ ▷ -define G -linear_group -PGL 7 11 -Janko1 -end \n ▷ ▷ -with G -do \n ▷ ▷ -group_theoretic_activity \n ▷ ▷ ▷ -report \n```

148
creates the first Janko group as a subgroup of PGL(7, 11).

The command

PGL\_3\_11\_singer:
\[\text{pdflatex PGL\_3\_11\_Singer\_3\_11\_19\_report.tex}\]
\[\text{open PGL\_3\_11\_Singer\_3\_11\_19\_report.pdf}\]

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

The command

PGL\_3\_11\_singer\_and\_frobenius:
\[\text{pdflatex PGL\_3\_11\_Singer\_and\_Frob\_3\_11\_19\_report.tex}\]
\[\text{open PGL\_3\_11\_Singer\_and\_Frob\_3\_11\_19\_report.pdf}\]

creates a subgroup of index 19 of the Singer cycle of PG(2, 11), extended by a group of order 3 that arises from the field extension \(F_{11}^3\) over \(F_{11}\). The group created by this command has order 21.

The quaternion group is a group of order 8 generated by the following matrices over \(\mathbb{R}\):

\[
i = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad j = \begin{bmatrix} -1 & 1 \\ 1 & 1 \end{bmatrix}, \quad k = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}.
\]

It is isomorphic to a subgroup of 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
```
The group of the cube can be created over the field $\mathbb{F}_3$:

cube\_group:
$\texttt{\$\{ORBITER\} -v 2 \$
$\texttt{\> \> \> -define gens -vector -dense \$
$\texttt{\> \> \> \> \> "0,1,0,2,0,0,0,0,1, \$
$\texttt{\> \> \> \> \> 0,0,1,0,1,0,2,0,0, \$
$\texttt{\> \> \> \> \> 2,0,0,0,1,0,0,0,1" \$
$\texttt{\> \> \> -end \$
$\texttt{\> \> \> -define G -linear\_group -GL 3 3 \$
$\texttt{\> \> \> -subgroup\_by\_generators "cube" "48" 3 \$
$\texttt{\> \> \> \> \> gens \$
$\texttt{\> \> \> -end \$
$\texttt{\> \> \> -with G -do \$
$\texttt{\> \> \> \> -group\_theoretic\_activity \$
$\texttt{\> \> \> \> \> -print\_elements\_tex \$
$\texttt{\> \> \> \> \> -report \$
$\texttt{\> \> \> -end \$
$\texttt{\> pdflatex GL\_3\_3\_Subgroup\_cube\_48\_report.tex \$
$\texttt{\> open GL\_3\_3\_Subgroup\_cube\_48\_report.pdf \$
$\texttt{\> pdflatex GL\_3\_3\_Subgroup\_cube\_48\_elements.tex \$
$\texttt{\> open GL\_3\_3\_Subgroup\_cube\_48\_elements.pdf \$

The tetrahedral subgroup can be created as well:

tetra\_group:
$\texttt{\$\{ORBITER\} -v 3 \$
$\texttt{\> \> \> -define G -linear\_group -GL 3 3 \$
$\texttt{\> \> \> -subgroup\_by\_generators "tetra" "12" 2 \$
$\texttt{\> \> \> \> \> "0,1,0,0,0,1,1,0,0, 0,0,1,2,0,0,0,2,0" \$
$\texttt{\> \> \> -end \$
$\texttt{\> \> \> -with G -do \$
$\texttt{\> \> \> \> -group\_theoretic\_activity \$
$\texttt{\> \> \> \> \> -print\_elements\_tex \$
$\texttt{\> \> \> \> \> -report \$
$\texttt{\> \> \> -end \$
$\texttt{\> pdflatex GL\_3\_3\_Subgroup\_tetra\_12\_report.tex \$
$\texttt{\> open GL\_3\_3\_Subgroup\_tetra\_12\_report.pdf \$
$\texttt{\> pdflatex GL\_3\_3\_Subgroup\_tetra\_12\_elements.tex \$
$\texttt{\> open GL\_3\_3\_Subgroup\_tetra\_12\_elements.pdf \$

152
The Hesse group of order 216 extended by the automorphism group of the field can be created in PG(3, 4)

GENERATORS_HESSE_GROUP="\
3000300030 \\
2000201230 \\
1000100111 \\
1000220200 \\
1002312010 \\
0331003211 \\
2200011331"

Hesse group:
▷ $(ORBITER) -v 3 \\
▷ ▷ -define gens -vector -compact \\
▷ ▷ ▷ $(GENERATORS_HESSE_GROUP) \\
▷ ▷ -end \\
▷ ▷ -define G -linear_group -PGGL 3 4 \\
▷ ▷ -subgroup_by_generators "Hesse" "432" 7 gens \\
▷ ▷ -end \\
▷ ▷ -with G -do \\
▷ ▷ -group_theoretic_activity \\
▷ ▷ ▷ -print_elements.tex \\
▷ ▷ ▷ -report \\
▷ ▷ -end \\
▷ pdflatex PGGL_3_4_Subgroup_Hesse_432_report.tex \\
▷ open PGGL_3_4_Subgroup_Hesse_432_report.pdf

The group has order 432.

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

GENERATORS_WEYL_GROUP_E8="\
-1,-1,-1,-1,0,0,0,0, \\
0,0,0,1,0,0,0,0, \\
1,0,0,0,0,0,0,0, \\
0,0,1,0,0,0,0,0, \\
0,1,0,1,1,0,0,0, \\
0,0,0,0,0,1,0,0, \\
0,0,0,0,0,0,1,0, \\
0,0,0,0,0,0,0,1, \\
-1,0,-1,-1,-1,-1,-1, \\
0,1,0,1,1,1,1,1, \\
0,1,0,1,1,1,1,1, \\
0,1,0,1,1,1,1,1, \\
0,1,0,1,1,1,1,1, \\
0,1,0,1,1,1,1,1, \\
0,1,0,1,1,1,1,1, \\
0,1,0,1,1,1,1,1, \\
0,1,0,1,1,1,1,1, \\
0,1,0,1,1,1,1,1, \\
0,1,0,1,1,1,1,1, \\
0,1,0,1,1,1,1,1, \\
0,1,0,1,1,1,1,1, \\
0,1,0,1,1,1,1,1, \\
0,1,0,1,1,1,1,1, \\
0,1,0,1,1,1,1,1, \\
0,1,0,1,1,1,1,1, \\
0,1,0,1,1,1,1,1, \\
0,1,0,1,1,1,1,1, \\
0,1,0,1,1,1,1,1, \\
0,1,0,1,1,1,1,1, \\
0,1,0,1,1,1,1,1, \\
0,1,0,1,1,1,1,1, \\
0,1,0,1,1,1,1,1, \\
0,1,0,1,1,1,1,1, \\
0,1,0,1,1,1,1,1,
Weyl_E8:

\texttt{\$ (ORBITER) -v 3 \ 
  \$-define gens -vector -dense \ 
  \$ (GENERATORS_WEYL_GROUP_E8) \ 
  \$-end \ 
  \$-define G -linear_group -GL 8 3 \ 
  \$-subgroup_byGenerators \ 
  \$ "Weyl_E8" "696729600" 2 \ 
  \$ (GENERATORS_WEYL_GROUP_E8) \ 
  \$-end \ 
  \$-with G -do \ 
  \$-group_theoretic_activity \ 
  \$-report \ 
  \$-end \ 
  pdflatex GL_8_3_Subgroup_Weyl_E8_696729600_report.tex \ 
  open GL_8_3_Subgroup_Weyl_E8_696729600_report.pdf}

A latex report is generated in the file \texttt{GL_8_3_Subgroup_Weyl_E8_696729600_report.tex}. This command uses generators found by Gabi Nebe:

\url{http://www.math.rwth-aachen.de/~Gabriele.Nebe/LATTICES/E8.b.html}

We can test if a group is a subgroup of another. In the following example, we test whether 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:

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

coset_reps:
▷ $(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▷ -coset_reps \n▷ -end
▷ pdflatex PGOp_6_2.coset_reps.tex
▷ open PGOp_6_2.coset_reps.pdf

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

\[
\begin{array}{cccccc}
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1
\end{array} = \begin{array}{c}
100000 \\
010000 \\
001000 \\
000100 \\
000010 \\
000001
\end{array}
\]

\[\vdots\]
\[
\begin{array}{cccccc}
1 & 0 & 1 & 1 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 1 \\
0 & 1 & 1 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 & 1 & 0 \\
1 & 1 & 0 & 1 & 0 & 0
\end{array} = \begin{array}{c}
101110 \\
101000 \\
011101 \\
011111 \\
100010 \\
110100
\end{array}
\]

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

coset_reps_read:
$\text{(ORBITER)} -v 2 \\
\text{define F -finite_field -q 2 -end } \\
\text{define G1 -linear_group -PGOp 6 F -end } \\
\text{define G2 -linear_group -PGL 6 F } \\
\text{define CR -vector_ge -action G2 } \\
\text{read csv } \\
\text{PGOp.6.2.coset.reps.csv Element } \\
\text{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 [67].

The electronic Atlas of finite simple groups [67] 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 [62]):

```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 \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
```

Group theoretic activities will be discussed in Section 5.6.

As an example of a large group, consider the Conway group Co3. Following [61], the group can be generated using two matrices of dimension 22 over \( \mathbb{F}_2 \). We use the makefile variables to give each generator in compact form. Then we define vectors for each of the generators. We concatenate the two generators to form one long vector of generators, which is passed to the \(-\text{subgroup}\_\text{by}\_\text{generators}\) command. Finally, we create a report for the group.

```
CONWAY_GEN1="\n  110111000000001010000\n  11110101111110100001011\n  00000010000001000101\n  111110011010001000111\n  01010100000001001101\n  00000000001000101001\n  00100000000100010101\n  00000000100000000001\n  1101000110100001011\n  00000000001100000010\n  00000000001100000010\n  0110111110100011111\n  00000000000110000010\n  00000000001100000010\n  00000000011000000010\n  00000000100000000010\n  00000000100000100010\n  00000000100000000010\n  00000000000000000010\n  00000000000000000010\n  00000000000000000010\n  00000000000000000010\n  00000000000000000010\n  00000000000000000010\n  00000000000000000010\n  00000000000000000010\n`
```
```
CONWAY_GEN2="\n010100001011101011111\n0110010001111011000\n001101000111111101111\n000110111001011010011\n10100100010001011110\n110100000001010100011\n11001011001111010101\n10001101010101010101\n010011000101000001111\n110000010100101010010\n0101110110011000000101\n010111110101001111101\n100001010101010101001\n000101000111100010111\n0011010010111011001111\n010011001011001111101\n110101100111110110001\n010010100101000010001\n010111011001010000001\n0000001101111000101110\n1101101010101110000101"

Co3:
▷ $(ORBITER) -v 6 \\
▷▷ -define F -finite_field -q 2 -end \\
▷▷ -define g1 -vector -field F -format 22 -compact $(CONWAY_GEN1) -end \\
▷▷ -define g2 -vector -field F -format 22 -compact $(CONWAY_GEN2) -end \\
▷▷ -define gens -vector -concatenate g1 -concatenate g2 -end \\
▷▷ -define G -linear_group -PGL 22 2 \\
▷▷ -subgroup_by_generators "Co3" "495766656000" 2 gens \\
▷▷▷ -end \\
▷▷ -with G -do \\
▷▷ -group_theoretic_activity \\
▷▷▷ -report \\
▷▷▷ -end \\
▷ pdflatex PGL_22_2_Subgroup_Co3_495766656000_report.tex \\
▷ open PGL_22_2_Subgroup_Co3_495766656000_report.pdf
```

The next example creates the Ree group in 7 dimensions over the field $\mathbb{F}_{27}$. 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, 21,3,21,22,6,14, \ 
14,18,1,5,13,6,7, 3,3,2,1,24,16,3, 17,3,22,10,16,24,26, \ 
21,21,6,18,20,2,5"
```

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

```
Ree_27:
  $\text{-(ORBITER)} \ -v \ 6 \ 
  -\text{-define} \ F \ -\text{finite_field} \ -q \ 27 \ -\text{override_polynomial} \ "34" \ -\text{-end} \ 
  -\text{-define} \ g1 \ -\text{-vector} \ -\text{-field} \ F \ -\text{-format} \ 7 \ -\text{-dense} \ $(Ree_gen1) \ -\text{-end} \ 
  -\text{-define} \ g2 \ -\text{-vector} \ -\text{-field} \ F \ -\text{-format} \ 7 \ -\text{-dense} \ $(Ree_gen2) \ -\text{-end} \ 
  -\text{-define} \ gens \ -\text{-vector} \ -\text{-concatenate} \ g1 \ -\text{-concatenate} \ g2 \ -\text{-end} \ 
  -\text{-define} \ G \ -\text{-linear_group} \ -\text{PGL} \ 7 \ F \ 
  -\text{-subgroup_by_generators} \ "Ree_27" \ "10073444472" \ 2 \ gens \ 
  -\text{-with} \ G \ -\text{-do} \ 
  -\text{-group_theoretic_activity} \ 
  -\text{-report} \ 
  -\text{-end}
```

### Table 5.4: Commands for creating new actions from old

<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>

### 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

```bash
T3_on_tensors:
  ▶ $(ORBITER) -v 4 \
  ▶ ▶ -define G \
  ▶ ▶ -linear_group -GL_d_q_wr_Sym_n 2 2 3 \
  ▶ ▶ ▶ -on_tensors -end \
  ▶ ▶ -with G -do \
  ▶ ▶ -group_theoretic_activity \
  ▶ ▶ ▶ -report \
  ▶ ▶ -end \
  ▶ pdflatex GL_2_2_wreath_Sym3_report.tex \
  ▶ open GL_2_2_wreath_Sym3_report.pdf
```

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 $\mathbb{F}_2$. Elements of this group are denoted in the notation of the semidirect product. A vector of elements in the linear group is followed by a permutation of the components.
The Group $\text{GL}(2, 2) \wr \text{Sym}(3)$

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.
$t_l=3,2,1,3,2,3,2,3,2,3,2,$

Strong generators for a group of order 1296.

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

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

0,1,2,1,0,0,1,1,0,0,1,1,0,1,1,1,0,2,1,1,0,1,1,0,0,0,1,1,0,1,1,0,1,1,0,1,1,0,1,1,0,1,1,0,1,1,0,1,1,0,1,1,0,1,1,0,1,1,0,1,1,0,1,1,0,1,1,0,1,1,0,1,1,0,1,1,0,1,1,0,1,1,0,1,1,0,1,1,0,1,1,0,1,1,0,1,1,0,1,1,0,1,1,0,1,1,0,1,1,0,1,1,
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>3</td>
<td>1296</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>432</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>216</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>3</td>
<td>216</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>2</td>
<td>72</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>3</td>
<td>36</td>
</tr>
<tr>
<td>6</td>
<td>9</td>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td>7</td>
<td>12</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>13</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

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

\[ T3r1: \]
\[ \$ \text{ORBITER} -v 4 \]
\[ \text{define} \ G \]
\[ \text{linear_group} \ -\text{GL}\_d\_q\_wr\_Sym\_n \ 2 \ 2 \ 3 \]
\[ \text{on_rank_one_tensors} \ -\text{end} \]
\[ \text{with} \ G \ -\text{do} \]
\[ \text{group_theoretic_activity} \]
\[ \text{report} \]
\[ \text{end} \]
\[ \text{pdflatex} \ \text{GL}\_2\_2\_wreath\_Sym3\_report.tex \]
\[ \text{open} \ \text{GL}\_2\_2\_wreath\_Sym3\_report.pdf \]

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 $GL(2, 2) \wr Sym(3)res27$ of degree 27
We act on the following set:

$0 = ( 1, 0, 0, 0, 0, 0, 0, 0 )$
$1 = ( 0, 1, 0, 0, 0, 0, 0, 0 )$
$2 = ( 1, 1, 0, 0, 0, 0, 0, 0 )$
$3 = ( 0, 0, 1, 0, 0, 0, 0, 0 )$
$4 = ( 0, 0, 0, 1, 0, 0, 0, 0 )$
$5 = ( 0, 0, 1, 1, 0, 0, 0, 0 )$
$6 = ( 1, 0, 1, 0, 0, 0, 0, 0 )$
$7 = ( 0, 1, 0, 1, 0, 0, 0, 0 )$
$8 = ( 1, 0, 1, 1, 0, 0, 0, 0 )$
$9 = ( 0, 0, 0, 0, 1, 0, 0, 0 )$
$10 = ( 0, 0, 0, 0, 0, 1, 0, 0 )$
$11 = ( 0, 0, 0, 0, 1, 1, 0, 0 )$
$12 = ( 0, 0, 0, 0, 0, 0, 1, 0 )$
$13 = ( 0, 0, 0, 0, 0, 0, 0, 1 )$
$14 = ( 0, 0, 0, 0, 0, 0, 1, 1 )$
$15 = ( 0, 0, 0, 0, 1, 0, 1, 0 )$
$16 = ( 0, 0, 0, 0, 0, 1, 0, 1 )$
$17 = ( 0, 0, 0, 0, 1, 1, 1, 1 )$
$18 = ( 1, 0, 0, 0, 1, 0, 0, 0 )$
$19 = ( 0, 1, 0, 0, 0, 1, 0, 0 )$
$20 = ( 1, 1, 0, 0, 1, 1, 0, 0 )$
$21 = ( 0, 0, 1, 0, 0, 0, 1, 0 )$
$22 = ( 0, 0, 0, 1, 0, 0, 0, 1 )$
$23 = ( 0, 0, 1, 1, 0, 0, 1, 1 )$
$24 = ( 1, 0, 1, 0, 1, 0, 1, 0 )$
$25 = ( 0, 1, 0, 1, 0, 1, 0, 1 )$
$26 = ( 1, 1, 1, 1, 1, 1, 1, 1 )$

Base and Stabilizer Chain

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

\[
\begin{pmatrix}
1 & 0 \\
0 & 1 \\
\end{pmatrix}
\begin{pmatrix}
1 & 0 \\
0 & 1 \\
\end{pmatrix}
\begin{pmatrix}
1 & 0 \\
0 & 1 \\
\end{pmatrix}; id),
\begin{pmatrix}
1 & 0 \\
0 & 1 \\
\end{pmatrix}
\begin{pmatrix}
1 & 0 \\
0 & 1 \\
\end{pmatrix}
\begin{pmatrix}
1 & 1 \\
0 & 0 \\
\end{pmatrix}; id),
\begin{pmatrix}
1 & 0 \\
0 & 1 \\
\end{pmatrix}
\begin{pmatrix}
1 & 0 \\
0 & 1 \\
\end{pmatrix}
\begin{pmatrix}
1 & 0 \\
0 & 1 \\
\end{pmatrix}; id),
\begin{pmatrix}
1 & 0 \\
0 & 1 \\
\end{pmatrix}
\begin{pmatrix}
1 & 0 \\
0 & 1 \\
\end{pmatrix}
\begin{pmatrix}
1 & 0 \\
0 & 1 \\
\end{pmatrix}; id),
\begin{pmatrix}
1 & 0 \\
0 & 1 \\
\end{pmatrix}
\begin{pmatrix}
1 & 0 \\
0 & 1 \\
\end{pmatrix}
\begin{pmatrix}
1 & 0 \\
0 & 1 \\
\end{pmatrix}; (1, 2),
\begin{pmatrix}
1 & 0 \\
0 & 1 \\
\end{pmatrix}
\begin{pmatrix}
1 & 0 \\
0 & 1 \\
\end{pmatrix}
\begin{pmatrix}
1 & 0 \\
0 & 1 \\
\end{pmatrix}; (0, 1)
\]
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 \texttt{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 $\text{PGL}(2,8)$ of $\text{PG}(1,8)$ and act on $\text{PG}(2,8)$:

\begin{verbatim}
PGGL_2.8_on_conic:
  $(\text{ORBITER}) -v 4 \$
  $\text{define G }$
  $\text{linear_group -PGGL 2 8 -PGL2OnConic -end }$
  $\text{with G -do }$
  $\text{group_theoretic_activity }$
  $\text{report }$
  $\text{end}$
  $\text{pdflatex PGGL_2.8_OnConic_2.8_report.tex}$
  $\text{open PGGL_2.8_OnConic_2.8_report.pdf}$
\end{verbatim}

This produces the following report. The generators are elements of $\text{PGL}(2,8)$ 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 $\text{PGL}(2,8)\text{OnConic}(2,8)$**

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

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

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

Group action $\text{PGL}(2, 8) \text{OnConic}$ of degree 73
We act on the following set:

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

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

The finite field $\mathbb{F}_8$

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

<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>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 = 0</td>
<td>0</td>
<td>DNE</td>
<td>DNE</td>
<td>1</td>
<td>DNE</td>
<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>DNE</td>
<td>7</td>
<td>2</td>
<td>5</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>$\alpha + 1 = \gamma^5$</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>0</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>
\[ \begin{array}{c|cccccccc} + & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \hline 0 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 1 & 0 & 3 & 2 & 5 & 4 & 7 & 6 \\ 2 & 2 & 3 & 0 & 1 & 6 & 7 & 4 & 5 \\ 3 & 3 & 2 & 1 & 0 & 7 & 6 & 5 & 4 \\ 4 & 4 & 5 & 6 & 7 & 0 & 1 & 2 & 3 \\ 5 & 5 & 4 & 7 & 6 & 1 & 0 & 3 & 2 \\ 6 & 6 & 7 & 4 & 5 & 2 & 3 & 0 & 1 \\ 7 & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\ \end{array} \]

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

\[ 2^0 = 1 \quad \quad 2^5 = 3 \]
\[ 2^1 = 2 \quad \quad 2^6 = 6 \]
\[ 2^2 = 4 \quad \quad 2^7 = 1 \]
\[ 2^3 = 5 \]
\[ 2^4 = 7 \]

**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>

167
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

168
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="1,0,0,0,12,0,0,0,12,0,0,0,1, \ 
1,0,0,0,12,0,0,0,0,1,0,0,0,12, \ 
1,0,0,0,0,12,0,0,12,0,0,0,0,1, \ 
0,1,0,0,1,0,0,0,0,0,1,0,0,0,1"
```

```
surface_q13_stab_on_tritangents_orbits:
  $(ORBITER) -v 30 \ 
  -define F -finite_field -q 13 -end \ 
  -define P -projective_space 3 F -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) -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 \ 

170
-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:
$\text{ORBITER} \ -v \ 5 \ \\
\quad \ -define \ G \ -linear \ group \ -PGL \ 3 \ 2 \ -end \ \\
\quad \ -with \ G \ -do \ \\
\quad \quad \ -group_theoretic_activity \ \\
\quad \quad \ \ -save_elements.csv \ "PGL_3_2_elements.csv" \ \\
\quad \ -end
\end{verbatim}

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

The command

\begin{verbatim}
Sym_3_elements:
$\text{ORBITER} \ -v \ 3 \ \\
\quad \ -define \ G \ -permutation \ group \ -symmetric \ group \ 3 \ -end \ \\
\quad \ -with \ G \ -do \ \\
\quad \quad \ -group_theoretic_activity \ \\
\quad \quad \ \ -print_elements.tex \ \\
\quad \ -end
$\text{ORBITER} \ -v \ 2 \ \\
\quad \ -draw_options \ \\
\quad \quad \ -nodes \ \\
\quad \quad \ \ -embedded \ -radius \ 250 \ \\
\quad \quad \ \ -xin \ 10000 \ -yin \ 10000 \ \\
\quad \quad \ \ -xout \ 100000 \ -yout \ 600000 \ \\
\quad \quad \ \ -scale \ 0.3 \ -line_width \ 1.0 \ \\
\quad \ -end \ \\
\quad \ -tree_draw \ -file \ Perm3_elements_tree.txt \ -end
pdflatex \ Perm3_elements_tree_draw.tex
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 [28].</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>see Table 6.2</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 $\text{PGL}(r,q)$ or $\text{PΓL}(r,q)$. For each $n \leq n_{\text{max}}$, the $[n,k,\geq d]$ codes are classified with $n-k=r$. See Section 10.</td>
</tr>
<tr>
<td>-tensor_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_morphism_exterior_square</td>
<td></td>
<td>Given a set of generators of a subgroup of $\text{PGO}^+(6,q)$ as $6 \times 6$ matrixes, compute the inverse image of the generators in $\text{PGL}(4,q)$ (if possible).</td>
</tr>
<tr>
<td>-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](image)

174
We create the 12 powers of the cycle

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

The output is

<table>
<thead>
<tr>
<th>$i$</th>
<th>$(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11)^i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11)$</td>
</tr>
<tr>
<td>2</td>
<td>$(0, 2, 4, 6, 8, 10)(1, 3, 5, 7, 9, 11)$</td>
</tr>
<tr>
<td>3</td>
<td>$(0, 3, 6, 9)(1, 4, 7, 10)(2, 5, 8, 11)$</td>
</tr>
<tr>
<td>4</td>
<td>$(0, 4, 8)(1, 5, 9)(2, 6, 10)(3, 7, 11)$</td>
</tr>
<tr>
<td>5</td>
<td>$(0, 5, 10, 3, 8, 1, 6, 11, 4, 9, 2, 7)$</td>
</tr>
<tr>
<td>6</td>
<td>$(0, 6)(1, 7)(2, 8)(3, 9)(4, 10)(5, 11)$</td>
</tr>
<tr>
<td>7</td>
<td>$(0, 7, 2, 9, 4, 11, 6, 1, 8, 3, 10, 5)$</td>
</tr>
<tr>
<td>8</td>
<td>$(0, 8, 4)(1, 9, 5)(2, 10, 6)(3, 11, 7)$</td>
</tr>
<tr>
<td>9</td>
<td>$(0, 9, 6, 3)(1, 10, 7, 4)(2, 11, 8, 5)$</td>
</tr>
<tr>
<td>10</td>
<td>$(0, 10, 8, 6, 4, 2)(1, 11, 9, 7, 5, 3)$</td>
</tr>
<tr>
<td>11</td>
<td>$(0, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1)$</td>
</tr>
<tr>
<td>12</td>
<td>id</td>
</tr>
</tbody>
</table>

The command

\texttt{PGL\_3\_4\_singer:}

\texttt{$(ORBITER) -v 5$}

\texttt{-define G -linear\_group -PGL 3 4 -end}$
finds all Singer cycles in PGL(3, 4) whose matrix is the companion matrix of a polynomial. The first one found is

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

whose projective order is 21. Here, we are using the numeric form of field elements, so 2 is \(\omega\) and 3 is \(\omega + 1\).

Suppose we want to multiply two elements in a group. The following command shows an example in GL(2, 8). We multiply the elements coded by 0,1,2,3 and 4,5,6,7:

GL_2.8_multiply:

```
$ (ORBITER) -v 5 \
-define G -linear_group -GL 2 8 -end \n-with G -do \
-group_theoretic_activity \n-multiply "0,1,2,3" "4,5,6,7" \n-end \
pdflatex GL_2.8_mult.tex \
open GL_2.8_mult.pdf
```

The output is

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

0,1,2,3,
4,5,6,7,
6,7,2,3,

Note that the output shows the codings of the three group elements. This way, the result of this computation can be processed further easily. The same example over \(\mathbb{F}_7\), noting that 7 \(\equiv 0 \mod 7\) is:
GL_2.7_multiply:
\begin{verbatim}
$\text{(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
\end{verbatim}
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:
\begin{verbatim}
$\text{(ORBITER)} -v 5 \ \\
  -define G -linear_group -GL 2 7 -end \ \\
  -with G -do \ \\
  -group_theoretic_activity \ \\
  -inverse "0,1,2,3" \ \\
  -end
\end{verbatim}
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:

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

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 3 F -end \\
$\$ -define G -linear_group -PGL 4 F -end \\
$\$ -define G_on_planes -modified_group -from G \\
$\$ -on_k_subspaces 3 \\
$\$ -end \\
$\$ -with G_on_planes -do \\
$\$ -group_theoretic_activity \\
$\$ -apply "0,8,1,6,4,3,7,2,5" "1,0,0,0, 0,1,0,0, 0,0,0,2, 0,0,1,1" \\
$\$ -end
```

The output is

```
178
```
\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & \gamma \\
0 & 0 & 1 & 1
\end{bmatrix}
= 
\begin{bmatrix}
1000 \\
0100 \\
0002 \\
0011
\end{bmatrix}
\]

1,0,0,0,1,0,0,0,0,2,0,0,1,1,
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 5.7 shows the 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 lists 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

**PGGL2_4_classes:**

\[
\texttt{$(ORBITER) -v 3$
}\]

\[
\texttt{$(ORBITER) -v 3$
}\]

\[
\texttt{define G$
}\]

\[
\texttt{-linear_group -PGGL 2 4$
}\]

\[
\texttt{-end$
}\]

\[
\texttt{-with G -do$
}\]

\[
\texttt{-group_theoretic_activity$
}\]

\[
\texttt{-classes$
}\]

\[
\texttt{-end$
}\]

\[
\texttt{$(MAGMA\_PATH)magma\_PGGL2_4\_classes.magma$
}\]

\[
\texttt{$(ORBITER) -v 3$
}\]

\[
\texttt{-define G$
}\]

\[
\texttt{-linear_group -PGGL 2 4$
}\]

\[
\texttt{-end$
]\]

\[
\texttt{-with G -do$
]\]

\[
\texttt{-group_theoretic_activity$
]\]

\[
\texttt{-classes$
]\]

\[
\texttt{-end$
]\]
computes the classes of elements in \(\text{PGL}(2,4)\) using Orbiter-Magma-Orbiter. The first Orbiter command produces the file \(\text{PGGL}_2_4\_\text{classes.magma}\). The magma command reads this file and produces the file \(\text{PGGL}_2_4\_\text{classes_out.txt}\). The second Orbiter command reads the file \(\text{PGGL}_2_4\_\text{classes_out.txt}\) and produces the latex report \(\text{PGGL}_2_4\_\text{classes_out.tex}\).

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

Order of element = 2  
Class size = 10  
Centralizer order = 12  
Normalizer order = 12  
Representing element is

\[
c_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}
\]

of order 2 and with 3 fixed points. 0,1,1,0,1,

The normalizer is generated by:

Strong generators for a group of order 12:

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

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

The command sequence

\texttt{PGGL}_2_4\_\text{cent}_2\text{A}:

\texttt{pdflatex PGGL}_2_4\_\text{classes_out}\_\text{tex}

\texttt{open PGGL}_2_4\_\text{classes_out}\_\text{pdf}

\texttt{open PGGL}_2_4\_\text{classes_out}\_\text{csv}

\texttt{\$(ORBITER) -v 3 \}

\texttt{\% define G \}

\texttt{\% linear_group -PGGL 2 4 -end \}

\texttt{\% with G -do \}

\texttt{\% group_theoretic_activity \}

181
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).Z₂. Orbiter produces a list of strong generators, shown below:

<table>
<thead>
<tr>
<th>Strong generators for a group of order 40320:</th>
</tr>
</thead>
</table>
| \[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}, \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}, \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
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 & 1 & 1 \\
0 & 1 & 0 & 0
\end{bmatrix}, \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 \\
0 & 0 & 0 & 1
\end{bmatrix}, \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 \\
0 & 1 & 0 & 0
\end{bmatrix}, \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 \\
0 & 1 & 1 & 1
\end{bmatrix} |

| 182 |
The end of the report has a list of generators in coded form. This list can be used to create the centralizer in Orbiter.

Orbiter can compute the normalizer of a cyclic subgroup. For instance, the element

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

generates a cyclic subgroup of PGL(4,5) of order 31. The command

```
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, 0,0,1,0, 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
```

computes the normalizer, which is a group of order 372. The following report is produced:

The subgroup generated by

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

has order 31
The normalizer has order 372
Strong generators for a group of order 372:

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

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

For general normalizers, the group must be constructed as a subgroup \( H \) of a larger group \( G \) containing \( H \). Then, the normalizer of \( H \) in \( G \) is computed. Consider this example. The group

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

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

\begin{verbatim}
Normalizer_of_Z22_in_PGL_2_9:
  $ (ORBITER) -v 2 \n  $ define G -linear_group -PGL 2 9 \n  $ subgroup G -linear_group -PGL 2 9
  $ subgroup_by_generators Z22 4 2 \n  $ with G -do \n  $ group_theoretic_activity \n  $ normalizer \n  $ end
pdflatex PGL_2_9_Subgroup_Z22_4_normalizer.tex
open PGL_2_9_Subgroup_Z22_4_normalizer.pdf
\end{verbatim}

It produces a report showing that the normalizer is a group of order 24 (it is isomorphic to \( \text{Sym}(4) \), 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, 35, 59]. 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 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 6.1.

```latex
T3r1_orbits:
\>$\text{ORBITER}$ -v 4 \\
\>$\text{define } G \$
\>$\text{-linear group } -GL_d_q_wr_Sym_n 2 2 3 \$
\>$\text{-on rank_one_tensors -end } \$
\>$\text{-with } G \text{-do } \$
\>$\text{-group_theoretic_activity } \$
\>$\text{-report } \$
\>$\text{-orbits_on_points } \$
\>$\text{-export_trees } \$
\>$\text{-end } \$
\>$\text{pdflatex GL_2_2_wreath_Sym3_orbits_report.tex} \$
\>$\text{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:
<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>fname, $f$, $l$</td>
<td>Reads the csv file “fname” and extract sets from columns $[f, ..., f + l - 1]$.</td>
</tr>
<tr>
<td>-orbit_of_set_from_file</td>
<td>fname</td>
<td>Reads a set from the text file “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>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>name-C, 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 name-C and name-T are the files containing the elements of the class and the transporter, respectively.</td>
</tr>
</tbody>
</table>

Table 6.1: Basic Orbit algorithms
This command computes the orbits of on all cubic forms in 4 variables, confirming the work of Dickson [23] 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 \n  > define G \n  > -linear_group -PGGL 2 8 -PGL2OnConic -end \n  > -with G -do \n```
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 \( \text{PGL}(2, 8) \) OnConic:

Strong generators for a group of order 1512:

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

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

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

- \((1, 9, 63)\)

\[ i : \text{orbit length} : \text{number of orbits} \]
- 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 : 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 1:
Orbit 1: ( 1 )

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

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

1,0,0,1,1,
1,0,0,6,0,
1,0,2,1,0,
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, 38, 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: \((4, 5, 18, 7, 57, 25, 11, 37, 56, 10, 8, 33, 66, 45, 32, 41, 34, 14, 64, 9, 30, 47, 68, 52, 59, 71, 62, 6, 49, 65, 26, 21, 72, 54, 39, 13, 20, 43, 70, 50, 61, 17, 22, 44, 35, 23, 46, 40, 51, 28, \ldots 12, 31, 16)\)
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:
\[
\begin{bmatrix}
1 & 0 \\
0 & 1 \\
\end{bmatrix}_1
\]

Generator 1 / 4 is:
\[
\begin{bmatrix}
1 & 0 \\
\gamma^5 & 1 \\
\end{bmatrix}_2
\]

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 6.2 shows the power set lattice of a four-element subset.

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

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

For background on poset actions, see Plesken [54]. The orbits of $G$ on $\mathcal{P}$ form another poset, the poset of orbits. The problem of classification of combinatorial objects can often be attacked by using group invariant relations. A layered poset can be decomposed into a series of relations. The layers allow to reduce the classification problem into small steps, namely from 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 [58] to perform poset classification. Two versions are available: one for subset-type posets and one for subspace-type posets. Figure 6.3 shows the subspace lattice of $V(3, 2) = F_2^3$. The basis elements are listed, using the enumerator for elements on the projective geometry $\text{PG}(2, 2)$ explained in Section 4.1.
The commands shown in 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 6.3: 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>fname</td>
<td>Recover from the given file.</td>
</tr>
<tr>
<td>-lex</td>
<td></td>
<td>Use the lexicographic ordering to speed up the search.</td>
</tr>
<tr>
<td>-w</td>
<td></td>
<td>Save orbits at level d only.</td>
</tr>
<tr>
<td>-W</td>
<td></td>
<td>Save orbits at all levels.</td>
</tr>
<tr>
<td>-write_data_files</td>
<td></td>
<td>Save data to files.</td>
</tr>
<tr>
<td>-t</td>
<td></td>
<td>Write a file containing the search tree at level d.</td>
</tr>
<tr>
<td>-T</td>
<td></td>
<td>Write a file containing the search tree at all levels.</td>
</tr>
<tr>
<td>-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>-list_all</td>
<td></td>
<td>List orbits at all levels.</td>
</tr>
<tr>
<td>-table_of_nodes</td>
<td></td>
<td>Produce a spreadsheet of all orbits.</td>
</tr>
<tr>
<td>-make_relations_</td>
<td></td>
<td>Produce a bitmap drawing of the neighboring relations in the poset with</td>
</tr>
<tr>
<td>with_flag_orbits</td>
<td></td>
<td>flag orbits.</td>
</tr>
<tr>
<td>-Kramer_Mesner_matrix</td>
<td>( t k )</td>
<td>Compute the Kramer-Mesner matrix ( M_{t,k} ).</td>
</tr>
<tr>
<td>-level_summary_csv</td>
<td></td>
<td>Write a summary of number of orbits at each level to a csv file.</td>
</tr>
<tr>
<td>-orbit_reps_csv</td>
<td></td>
<td>Write orbit representatives to a csv file.</td>
</tr>
<tr>
<td>-report.. -end</td>
<td></td>
<td>Produce a latex report. Requires -orbiter_path option from Section 2.2.</td>
</tr>
<tr>
<td>-node_label_</td>
<td></td>
<td>When drawing the poset of orbits, display the group order in the orbit</td>
</tr>
<tr>
<td>is_group_order</td>
<td></td>
<td>nodes.</td>
</tr>
<tr>
<td>-node_label_</td>
<td></td>
<td>When drawing the poset of orbits, display the element rank in the orbit</td>
</tr>
<tr>
<td>is_element</td>
<td></td>
<td>nodes.</td>
</tr>
<tr>
<td>-show_orbit_decomposition</td>
<td></td>
<td>Show the orbits of the stabilizers.</td>
</tr>
<tr>
<td>-show_stab</td>
<td></td>
<td>Show the stabilizers.</td>
</tr>
<tr>
<td>-save_stab</td>
<td></td>
<td>Save the stabilizer generators.</td>
</tr>
<tr>
<td>-show_whole_orbits</td>
<td></td>
<td>Show the whole orbits.</td>
</tr>
<tr>
<td>-recognize</td>
<td>( L )</td>
<td>Recognize the given object in the classified list and compute a transpor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ter that maps the given object to the canonical form. Here, ( L )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>must be a list of integers (comma separated and enclosed in double quo</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tesse) encoding an object. This option can be repeated.</td>
</tr>
<tr>
<td>-export_schreier_trees</td>
<td></td>
<td>Export all Schreier trees.</td>
</tr>
<tr>
<td>-draw_schreier_trees</td>
<td>args</td>
<td>Draw all Schreier trees.</td>
</tr>
<tr>
<td>-preferred_choice</td>
<td>( n a b )</td>
<td>At node ( n ), choose ( b ) instead of ( a ) as orbit representat</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:

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

The join of two subsets in the poset may or may not belong to the poset. Let us consider the action of the Singer cycle on $\text{PG}(3,2)$. The following command computes the orbits of the group $G$ generated by a Singer cycle in $\text{PG}(3,2)$:

```
PGL_3_2_singer:
  $(ORBITER) -v 3 \$
  $-orbi$t_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 \$
  $-report -end \$
  $-end \$
  $-orbits_on_subsets 7 \$
  $-report \$
  $-end$
```

The next command computes the orbits of the projective group $\text{PGL}(4,2)$ acting on all subsets of $\text{PG}(3,2)$:

```
PG_3_2_subsets:
  $(ORBITER) -v 3 \$
  $-orbi$t_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 \$
```

199
Figure 6.4: The orbits of $\text{PGL}(4, 2)$ on subsets

A drawing of the poset of orbits as in Figure 6.4 is produced.

Orbiter can compute orbits of groups acting in various different actions. The following

```latex
\$\text{pdflatex PGL_4_2_poset.tex}$
\$\text{open PGL_4_2_poset.pdf}$
```
example computes the orbits of \( \text{PGL}(3, 2) \) on the subsets of lines of \( \text{PG}(2, 2) \).

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

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

\begin{verbatim}
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
\end{verbatim}
The poset of orbits is shown in Figure 6.5.
Figure 6.5: 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 6.6. PG(3,2) has 15 points:

$$P_0 = (1, 0, 0, 0) \quad P_1 = (0, 1, 0, 0) \quad P_2 = (0, 0, 1, 0) \quad P_3 = (0, 0, 0, 1)$$

$$P_4 = (1, 1, 1, 1) \quad P_5 = (1, 1, 0, 0) \quad P_6 = (1, 0, 1, 0) \quad P_7 = (0, 1, 1, 0)$$

$$P_8 = (1, 1, 1, 0) \quad P_9 = (1, 0, 0, 1) \quad P_{10} = (0, 1, 0, 1) \quad P_{11} = (1, 1, 0, 1)$$

$$P_{12} = (0, 0, 1, 1) \quad P_{13} = (1, 0, 1, 1) \quad P_{14} = (0, 1, 1, 1)$$

The $Q^+(3,2)$ quadric given by the equation above consists of the nine points $P_0, P_1, P_2, P_3, P_4, P_6, P_7, P_9, P_{10}$.

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

```
subspaces.Op.4.2:
> $(ORBITER) -v 5 \$
> -orbiter_path $(ORBITER_PATH) \$
> -define G -linear_group -PGL 4 2 -orthogonal 1 -end \$
> -with G -do \$
```

204
Figure 6.7: Hasse-diagram for the orbits of the orthogonal group $\text{PGO}^+(4, 2)$ on subspaces of $\text{PG}(3, 2)$

```
>> -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 \\
pdflatex PGL_4_2_Orthogonal_plus_4_2_poset.tex \\
open PGL_4_2_Orthogonal_plus_4_2_poset.pdf
```

produces a classification of all subspaces of $\text{PG}(3, 2)$ under $\text{PGO}^+(4, 2)$. The option `-draw_poset` creates a Hasse diagram of the classification as shown in Figure 6.7. 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

\[
\text{C6_on_partition:} \\
\quad \text{\textbackslash or}$(\text{ORBITER}) -v 5 \text{\textbackslash or}
\quad \quad \text{-orbiter_path $(\text{ORBITER\_PATH}) \text{\textbackslash or}
\quad \quad \quad \text{-define G -permutation\_group -cyclic\_group 6 -end \text{\textbackslash or}
\quad \quad \quad \quad \text{-with G -do \text{\textbackslash or}
\quad \quad \quad \quad \quad \text{-group\_theoretic\_activity \text{\textbackslash or}
\quad \quad \quad \quad \quad \quad \text{-poset\_classification\_control \text{\textbackslash or}
\quad \quad \quad \quad \quad \quad \quad \text{-problem\_label C6 \text{\textbackslash or}
\quad \quad \quad \quad \quad \quad \quad \quad \text{-depth 2 \text{\textbackslash or}
\quad \quad \quad \quad \quad \quad \quad \quad \quad \text{-W \text{\textbackslash or}
\quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \text{-draw\_options \text{\textbackslash or}
\quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \text{-radius 200 -embedded \text{\textbackslash or}
\quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \text{-end \text{\textbackslash or}
\quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \text{-end \text{\textbackslash or}
\quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \text{-orbits\_on\_partition 2 \text{\textbackslash or}
\quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \text{-end}
\]

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

\[
\text{PGL\_2\_17_on_partition:} \\
\quad \text{\textbackslash or}$(\text{ORBITER}) -v 5 \text{\textbackslash or}
\quad \quad \text{-define G -linear\_group -PGL 2 17 -end \text{\textbackslash or}
\quad \quad \quad \text{-with G -do \text{\textbackslash or}
\quad \quad \quad \quad \text{-group\_theoretic\_activity \text{\textbackslash or}
\quad \quad \quad \quad \quad \text{-poset\_classification\_control \text{\textbackslash or}
\quad \quad \quad \quad \quad \quad \text{-problem\_label PGL\_2\_17 \text{\textbackslash or}
\quad \quad \quad \quad \quad \quad \quad \text{-depth 6 \text{\textbackslash or}
\quad \quad \quad \quad \quad \quad \quad \quad \text{-W \text{\textbackslash or}
\quad \quad \quad \quad \quad \quad \quad \quad \quad \text{-end \text{\textbackslash or}
\quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \text{-orbits\_on\_partition 6 \text{\textbackslash or}
\quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \text{-end}
\]

206
computes the orbits of the group PGL(2, 17) on set-partitions of type 6 + 6 + 6. The number of set-partitions is
\[
\frac{\binom{18}{6} \cdot \binom{12}{6}}{3!} = 2858856
\]
There are 720 orbits. The orbit stabilizer statistic is
\[
(1^{480}, 2^{184}, 3^{11}, 4^{20}, 6^{15}, 8, 12^{6}, 18, 24, 36).
\]
The orbit-stabilizer count confirms that
\[
4896 \left(\frac{480}{1} + \frac{184}{2} + \frac{11}{3} + \frac{20}{4} + \frac{15}{6} + \frac{1}{8} + \frac{6}{12} + \frac{1}{18} + \frac{1}{24} + \frac{1}{36} + \right) = 2858856
\].
### 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. [47]) and the Pellegrino cap in $\text{AG}(4,3)$. The uniqueness of this cap was proven by Hill [30].

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 Op4_2:
  ▶ $(\text{ORBITER}) -v 5 \$
  ▶ ▶ -orbiter_path $(\text{ORBITER\_PATH}) \$
  ▶ ▶ -define G -linear_group -PGL 4 2 -orthogonal 1 -end \$
  ▶ ▶ -with G -do \$
```

<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 $x_0 = 0$ 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
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

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

<table>
<thead>
<tr>
<th>Orbit</th>
<th>Node</th>
<th>Coordinates</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>( 1, 0, 0 )</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>( 0, 1, 0 )</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>( 0, 0, 1 )</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>( 1, 1, 1 )</td>
</tr>
<tr>
<td>4</td>
<td>52</td>
<td>( 3, 2, 1 )</td>
</tr>
<tr>
<td>5</td>
<td>67</td>
<td>( 2, 3, 1 )</td>
</tr>
<tr>
<td>6</td>
<td>89</td>
<td>( 8, 4, 1 )</td>
</tr>
<tr>
<td>7</td>
<td>106</td>
<td>( 9, 5, 1 )</td>
</tr>
<tr>
<td>8</td>
<td>126</td>
<td>(13, 6, 1 )</td>
</tr>
<tr>
<td>9</td>
<td>141</td>
<td>(12, 7, 1 )</td>
</tr>
<tr>
<td>10</td>
<td>159</td>
<td>(14, 8, 1 )</td>
</tr>
<tr>
<td>11</td>
<td>176</td>
<td>(15, 9, 1 )</td>
</tr>
<tr>
<td>12</td>
<td>184</td>
<td>( 7, 10, 1 )</td>
</tr>
<tr>
<td>13</td>
<td>199</td>
<td>( 6, 11, 1 )</td>
</tr>
<tr>
<td>14</td>
<td>220</td>
<td>(11, 12, 1 )</td>
</tr>
<tr>
<td>15</td>
<td>235</td>
<td>(10, 13, 1 )</td>
</tr>
<tr>
<td>16</td>
<td>245</td>
<td>( 4, 14, 1 )</td>
</tr>
<tr>
<td>17</td>
<td>262</td>
<td>( 5, 15, 1 )</td>
</tr>
</tbody>
</table>
Strong generators for a group of order 144:

\[
\begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
\delta^4 & \delta^9 & 1
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & \delta^7 & \delta^{13} \\
\delta^8 & \delta^9 & \delta^{10} \\
\delta^5 & \delta^5 & \delta^6
\end{bmatrix}_1
\]

\[
\begin{bmatrix}
\delta^5 & \delta^5 & \delta^6 \\
\delta^5 & \delta^2 & \delta^{11} \\
\delta^5 & \delta^{14} & 1
\end{bmatrix}_3
\]

1,0,0,1,0,9,5,1,1,
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}


\[
\begin{align*}
0 : 0 &= (1, 0, 0) \\
1 : 1 &= (0, 1, 0) \\
2 : 2 &= (0, 0, 1) \\
3 : 3 &= (1, 1, 1) \\
4 : 52 &= (3, 2, 1) \\
5 : 70 &= (5, 3, 1) \\
6 : 83 &= (2, 4, 1) \\
7 : 109 &= (12, 5, 1) \\
8 : 127 &= (14, 6, 1) \\
9 : 139 &= (10, 7, 1) \\
10 : 156 &= (11, 8, 1) \\
11 : 174 &= (13, 9, 1) \\
12 : 186 &= (9, 10, 1) \\
13 : 199 &= (6, 11, 1) \\
14 : 217 &= (8, 12, 1) \\
15 : 229 &= (4, 13, 1) \\
16 : 256 &= (15, 14, 1) \\
17 : 264 &= (7, 15, 1)
\end{align*}
\]

Strong generators for a group of order 16320:

\[
\begin{bmatrix}
\delta^6 & 0 & 0 \\
0 & \delta^3 & 0 \\
0 & 0 & 1
\end{bmatrix}_2
\]

\[
\begin{bmatrix}
\delta^9 & 0 & 0 \\
0 & \delta^7 & 0 \\
0 & 0 & 1
\end{bmatrix}_3
\]

\[
\begin{bmatrix}
\delta^2 & 0 & 0 \\
0 & \delta^{11} & 0 \\
0 & \delta^4 & \delta^7 & 1
\end{bmatrix}_3
\]

\[
\begin{bmatrix}
\delta^{10} & 0 & 0 \\
0 & \delta^3 & 0 \\
\delta & \delta^{11} & 1
\end{bmatrix}_3
\]

\[
\begin{bmatrix}
\delta & 0 & 0 \\
\delta^{12} & \delta^2 & \delta^5 \\
\delta^{14} & \delta^{10} & 1
\end{bmatrix}_1
\]

\[
\begin{bmatrix}
\delta^5 & \delta^3 & \delta^6 \\
\delta^5 & \delta^3 & \delta^6 \\
\delta^5 & \delta^3 & \delta^6
\end{bmatrix}_0
\]

\[
\begin{bmatrix}
\delta^{12} & 1 & \delta^2 \\
\delta^4 & \delta^3 & \delta^7 \\
\delta^6 & \delta^3 & 1
\end{bmatrix}_2
\]

\[
\begin{bmatrix}
\delta^{11} & \delta^6 & \delta^{10} \\
\delta^{10} & \delta^6 & 1
\end{bmatrix}_3
\]

210
There are 0 extensions
Number of generators 8

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:

```
nc_arcs_16:
▷ $(ORBITER) -v 4 \
▷ ▷ -define F -finite_field -q 16 -end \
▷ ▷ -define P -projective_space 2 F -end \
▷ ▷ -with P -do \
▷ ▷ -projective_space_activity \
▷ ▷ ▷ -classify_arcs \
▷ ▷ ▷ ▷ -poset_classification_control \
▷ ▷ ▷ ▷ ▷ -problem_label nc_arcs_q16_d2 \
▷ ▷ ▷ ▷ ▷ -W -depth 6 \
▷ ▷ ▷ ▷ ▷ -report -end \
▷ ▷ ▷ ▷ -end \
▷ ▷ ▷ ▷ -target_size 6 \
▷ ▷ ▷ ▷ -d 2 \
▷ ▷ ▷ ▷ -conic_test \
▷ ▷ ▷ ▷ -end \
▷ ▷ -end
pdflatex nc_arcs_q16_d2_poset.tex
open nc_arcs_q16_d2_poset.pdf
```

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 $\mathbb{F}_{32}$:

```
nc_arcs_32_E13:
```

211
\texttt{$(ORBITER) -v 4 \ \
-\texttt{orbiter\_path $(ORBITER\_PATH) \ 
-\texttt{define F -finite\_field -q 32 -end \ 
-\texttt{define P -projective\_space 2 F -end \ 
-\texttt{with P -do \ 
-\texttt{projective\_space\_activity \ 
-\texttt{classify\_arcs \ 
-\texttt{poset\_classification\_control \ 
-\texttt{problem\_label nc\_arcs\_q32\_d2 \ 
-\texttt{W -depth 6 \ 
-\texttt{draw\_poset -draw\_options -end \ 
-\texttt{report -end \ 
-\texttt{target\_size 6 \ 
-\texttt{test\_nb\_Eckardt\_points 13 \ 
-\texttt{d 2 \ 
-\texttt{conic\_test \ 
-\texttt{-end \ 
-\texttt{-end \ 
-pdflatex nc\_arcs\_q32\_d2\_poset.tex
-\texttt{open nc\_arcs\_q32\_d2\_poset.pdf}
6.7 Cubic Curves

Orbiter can classify cubic curves in PG(2, q). To this end, the (9, 3)-arcs in 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:
  $(ORBITER) -v 3 -define G \
  -define F -finite_field -q 8 -end \
  -define P -projective_space 2 F -end \
  -with P -do \
  -projective_space_activity \
  -classify_cubic_curves -q 8 -target_size 9 -n 3 -d 3 \
  -poset_classification_control \
  -problem_classification_label cc_8 -W -depth 9 \
  -draw_options -radius 200 -embedded -end \
  -recognize "0,1,2,3,35,28" \
  -recognize "1,2,3,51,28,61,46,71,40" \
  -draw_poset \
  -Kramer_Mesner_matrix 6 9 \
  -end \
  -end \
  $(ORBITER) -v 2 -draw_matrix \
  -input_csv_file cc_8_KM_6_9.csv \
  -box_width 50 -bit_depth 8 -end 

pdflatex Cubic_curves_q8.tex
open Cubic_curves_q8.pdf
#pdflatex cc_8_tree_lvl_9.tex
#open cc_8_tree_lvl_9.pdf

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

classifies the cubic curves in PG(2, 8).
Chapter 7

Cubic Surfaces

7.1 Creation

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

Orbiter contains a built-in catalogue of cubic surfaces with 27 lines for small finite fields $\mathbb{F}_q$. The surfaces in the catalogue all come with their automorphism group. It is also possible to create surfaces from known families, or to create surfaces from associated objects like 6-arcs. Some of these constructions only create the surface, not the automorphism group.

Tables 7.1-7.2 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:
  $(ORBITER) -v 3 \
  ▶ -define F -finite_field -q 4 -end \
  ▶ -define P -projective_space 3 F -end \
  ▶ -with P -do \n  ▶ -projective_space_activity \n  ▶  ▶ -define_surface S -q 4 -catalogue 0 -end \n  ▶  ▶ -end \n  ▶ -with S -do \n  ▶ -cubic_surface_activity \n  ▶  ▶ -report \n  ▶  ▶  ▶ -report_with_group \n  ▶  ▶ -end \n  pdflatex surface_catalogue_q4_iso0_report.tex
  open surface_catalogue_q4_iso0_report.pdf
```
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-catalogue</td>
<td>$i$</td>
<td>Create the $i$-th surface in the Orbiter catalogue. Here, $i$ is an index variable used to index all surfaces in 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 $X_3^3 - b^2(X_0^2 + X_1^2 + X_2^2)X_3 + \frac{b^3}{a}(a^2 + 1)X_0X_1X_2 = 0$. 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_abcld</td>
<td>$a \ b \ c \ d$</td>
<td>Create a member of the general family with parameters $a,b,c,d$.</td>
</tr>
<tr>
<td>-arc_lifting</td>
<td>$A$</td>
<td>Create the surface associated with the arc $A = a_1,\ldots,a_6$ in PG($2,q$) by means of the Clebsch map. Each of the $a_i$ is the rank of a point in PG($2,q$). Use the trihedral pair algorithm. Here, $A$ is a comma-separated string containing the numerical ranks of the $P_i$ in 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><strong>-arc_lifting_with_two_lines</strong></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><strong>-select_double_six</strong></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><strong>-transform</strong></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><strong>-transform_inverse</strong></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,

\begin{verbatim}
eckardt_13_4_12:
  $\$(ORBITER) -v 6 \
  -define F -finite_field -q 13 -end \
  -define P -projective_space 3 F -end \
  -with P -do \
  -projective_space_activity \ 
  -define_surface S.2.1 -q 13 \ 
  -family_Eckardt 4 12 -end \ 
  -end \ 
  -with S.2.1 -do \ 
  -cubic_surface_activity \ 
  -report \ 
  -report_with_group \ 
  -end
\end{verbatim}

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:

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

The following command parses the formula and creates the surface with \((a, b, c, d) = (4, 2, 2, 4)\):
surface_F_abcd:
▷ $(ORBITER) -v 3 \$
▷▷ -define F -finite_field -q 7 -end \$
▷▷ -with F -do \$
▷▷ -finite_field_activity \$
▷▷ -parse_and_evaluate "Fabcd" "X0,X1,X2,X3" \$
▷▷▷ $(F_{abcd}\text{eqn}) \"a=4,b=2,c=2,d=4\" \$
▷▷ -end

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

F_alpha_beta_gamma_delta_q7_override_group:
▷ $(ORBITER) -v 3 \$
▷▷ -define F -finite_field -q 7 -end \$
▷▷ -define P -projective_space 3 F -end \$
▷▷ -with P -do \$
▷▷ -projective_space_activity \$
▷▷ -define_surface F_{2345} -q 7 \$
▷▷▷ -by_equation "F\_alpha\_beta\_gamma\_delta" \$
▷▷▷ "\DF{\{\alpha,\beta,\gamma,\delta\}}\x0,\x1,\x2,\x3" \$
▷▷▷ $(F_{ALPHA_BETA_GAMMA_DELTA}) \$
▷▷▷ "\alpha=2,\beta=3,\gamma=4,\delta=5" \$
▷▷▷ "\D\alpha=2,\beta=3,\gamma=4,\delta=5\D" \$
▷▷▷ -override_group 6 2 \$
▷▷▷ "1,5,0,0,3,6,0,0,1,1,3,0,5,5,0,3, \$
▷▷▷ 1,0,2,5,0,1,6,1,0,0,3,5,0,0,4,4" \$
▷▷ -end \$
▷▷ -end \$
▷▷ -with F_{2345} -do \$
▷▷ -cubic_surface_activity \$
▷▷▷ -report \$
▷▷▷ -report_with_group \$
▷▷ -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 [31]), 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. Here is an example. 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:

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

The report contains the following information:

**The equation**

The equation of the quartic curve is:

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

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

**The gradient**

The gradient of the quartic curve is:

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

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

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

\[
egin{align*}
0 : P_7 &= (4, 1, 0) = c_{13} \cap c_{14} \cap c_{36} \cap c_{46} \\
1 : P_8 &= (5, 1, 0) = a_2 \cap a_4 \cap c_{25} \cap c_{45} \\
2 : P_9 &= (6, 1, 0) = b_1 \cap b_6 \cap c_{12} \cap c_{26} \\
3 : P_{10} &= (7, 1, 0) = a_3 \cap b_5 \cap c_{35} \cap d \\
4 : P_{11} &= (8, 1, 0) = b_2 \cap b_3 \cap c_{24} \cap c_{34} \\
5 : P_{12} &= (1, 0, 1) = a_3 \cap a_4 \cap c_{23} \cap c_{24}
\end{align*}
\]
<table>
<thead>
<tr>
<th>Index</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>( P_{13} = (2, 0, 1) = c_{34} \cap c_{36} \cap c_{45} \cap c_{56} )</td>
</tr>
<tr>
<td>7</td>
<td>( P_{15} = (4, 0, 1) = b_3 \cap b_6 \cap c_{13} \cap c_{16} )</td>
</tr>
<tr>
<td>8</td>
<td>( P_{16} = (5, 0, 1) = a_5 \cap a_6 \cap c_{25} \cap c_{26} )</td>
</tr>
<tr>
<td>9</td>
<td>( P_{18} = (7, 0, 1) = a_2 \cap b_1 \cap c_{35} \cap c_{46} )</td>
</tr>
<tr>
<td>10</td>
<td>( P_{19} = (8, 0, 1) = b_4 \cap b_5 \cap c_{14} \cap c_{15} )</td>
</tr>
<tr>
<td>11</td>
<td>( P_{20} = (0, 1, 1) = a_2 \cap b_3 \cap c_{14} \cap c_{56} )</td>
</tr>
<tr>
<td>12</td>
<td>( P_{21} = (2, 1, 1) = b_2 \cap b_4 \cap c_{26} \cap c_{46} )</td>
</tr>
<tr>
<td>13</td>
<td>( P_{22} = (3, 1, 1) = a_4 \cap b_5 \cap c_{12} \cap c_{36} )</td>
</tr>
<tr>
<td>14</td>
<td>( P_{23} = (4, 1, 1) = a_6 \cap b_1 \cap c_{23} \cap c_{45} )</td>
</tr>
<tr>
<td>15</td>
<td>( P_{26} = (7, 1, 1) = c_{16} \cap c_{25} \cap c_{34} \cap d )</td>
</tr>
<tr>
<td>16</td>
<td>( P_{27} = (8, 1, 1) = a_3 \cap a_5 \cap c_{13} \cap c_{15} )</td>
</tr>
<tr>
<td>17</td>
<td>( P_{28} = (0, 2, 1) = c_{12} \cap c_{13} \cap c_{25} \cap c_{35} )</td>
</tr>
<tr>
<td>18</td>
<td>( P_{29} = (1, 2, 1) = b_1 \cap b_5 \cap c_{16} \cap c_{56} )</td>
</tr>
<tr>
<td>19</td>
<td>( P_{31} = (3, 2, 1) = a_3 \cap a_6 \cap c_{34} \cap c_{46} )</td>
</tr>
<tr>
<td>20</td>
<td>( P_{33} = (5, 2, 1) = a_2 \cap b_4 \cap c_{24} \cap d )</td>
</tr>
<tr>
<td>21</td>
<td>( P_{35} = (7, 2, 1) = b_2 \cap b_6 \cap c_{23} \cap c_{36} )</td>
</tr>
<tr>
<td>22</td>
<td>( P_{36} = (8, 2, 1) = a_4 \cap b_3 \cap c_{15} \cap c_{26} )</td>
</tr>
<tr>
<td>23</td>
<td>( P_{37} = (0, 3, 1) = a_5 \cap b_1 \cap c_{24} \cap c_{36} )</td>
</tr>
<tr>
<td>24</td>
<td>( P_{39} = (2, 3, 1) = a_2 \cap a_6 \cap c_{12} \cap c_{16} )</td>
</tr>
<tr>
<td>25</td>
<td>( P_{40} = (3, 3, 1) = b_3 \cap b_4 \cap c_{35} \cap c_{45} )</td>
</tr>
<tr>
<td>26</td>
<td>( P_{42} = (5, 3, 1) = a_4 \cap b_2 \cap c_{13} \cap c_{56} )</td>
</tr>
<tr>
<td>27</td>
<td>( P_{43} = (6, 3, 1) = a_3 \cap b_6 \cap c_{14} \cap c_{25} )</td>
</tr>
<tr>
<td>28</td>
<td>( P_{45} = (8, 3, 1) = c_{15} \cap c_{23} \cap c_{46} \cap d )</td>
</tr>
<tr>
<td>29</td>
<td>( P_{47} = (1, 4, 1) = a_6 \cap b_2 \cap c_{14} \cap c_{35} )</td>
</tr>
<tr>
<td>30</td>
<td>( P_{48} = (2, 4, 1) = b_3 \cap b_5 \cap c_{23} \cap c_{25} )</td>
</tr>
<tr>
<td>31</td>
<td>( P_{49} = (3, 4, 1) = a_5 \cap b_6 \cap c_{56} \cap d )</td>
</tr>
<tr>
<td>32</td>
<td>( P_{50} = (4, 4, 1) = a_2 \cap a_3 \cap c_{26} \cap c_{36} )</td>
</tr>
<tr>
<td>33</td>
<td>( P_{52} = (6, 4, 1) = b_1 \cap b_4 \cap c_{13} \cap c_{34} )</td>
</tr>
<tr>
<td>34</td>
<td>( P_{54} = (8, 4, 1) = c_{12} \cap c_{15} \cap c_{24} \cap c_{45} )</td>
</tr>
<tr>
<td>35</td>
<td>( P_{55} = (0, 5, 1) = a_4 \cap a_6 \cap b_4 \cap b_6 )</td>
</tr>
<tr>
<td>36</td>
<td>( P_{56} = (1, 5, 1) = c_{13} \cap c_{26} \cap c_{45} \cap d )</td>
</tr>
<tr>
<td>37</td>
<td>( P_{59} = (4, 5, 1) = c_{24} \cap c_{25} \cap c_{46} \cap c_{56} )</td>
</tr>
<tr>
<td>38</td>
<td>( P_{60} = (5, 5, 1) = c_{12} \cap c_{14} \cap c_{23} \cap c_{34} )</td>
</tr>
<tr>
<td>39</td>
<td>( P_{61} = (6, 5, 1) = a_2 \cap a_5 \cap b_2 \cap b_5 )</td>
</tr>
<tr>
<td>40</td>
<td>( P_{63} = (8, 5, 1) = c_{15} \cap c_{16} \cap c_{35} \cap c_{36} )</td>
</tr>
<tr>
<td>41</td>
<td>( P_{64} = (0, 6, 1) = a_1 \cap b_5 \cap c_{26} \cap c_{34} )</td>
</tr>
<tr>
<td>42</td>
<td>( P_{65} = (1, 6, 1) = a_1 \cap b_4 \cap c_{25} \cap c_{36} )</td>
</tr>
<tr>
<td>43</td>
<td>( P_{66} = (2, 6, 1) = a_1 \cap b_6 \cap c_{24} \cap c_{35} )</td>
</tr>
<tr>
<td>44</td>
<td>( P_{67} = (3, 6, 1) = a_1 \cap a_2 \cap c_{13} \cap c_{23} )</td>
</tr>
<tr>
<td>45</td>
<td>( P_{68} = (4, 6, 1) = a_1 \cap b_2 \cap c_{12} \cap d )</td>
</tr>
<tr>
<td>46</td>
<td>( P_{69} = (5, 6, 1) = a_1 \cap a_3 \cap b_1 \cap b_3 )</td>
</tr>
<tr>
<td>47</td>
<td>( P_{70} = (6, 6, 1) = a_1 \cap a_4 \cap c_{16} \cap c_{46} )</td>
</tr>
<tr>
<td>48</td>
<td>( P_{71} = (7, 6, 1) = a_1 \cap a_5 \cap c_{14} \cap c_{45} )</td>
</tr>
<tr>
<td>49</td>
<td>( P_{72} = (8, 6, 1) = a_1 \cap a_6 \cap c_{15} \cap c_{56} )</td>
</tr>
</tbody>
</table>
50 : \( P_{73} = (0, 7, 1) = a_3 \cap b_2 \cap c_{16} \cap c_{45} \)
51 : \( P_{74} = (1, 7, 1) = a_5 \cap b_3 \cap c_{12} \cap c_{16} \)
52 : \( P_{75} = (2, 7, 1) = a_4 \cap b_1 \cap c_{14} \cap d \)
53 : \( P_{79} = (6, 7, 1) = c_{23} \cap c_{26} \cap c_{35} \cap c_{56} \)
54 : \( P_{80} = (7, 7, 1) = a_6 \cap b_5 \cap c_{13} \cap c_{24} \)
55 : \( P_{81} = (8, 7, 1) = a_2 \cap b_6 \cap c_{15} \cap c_{34} \)
56 : \( P_{85} = (3, 8, 1) = c_{14} \cap c_{16} \cap c_{24} \cap c_{26} \)
57 : \( P_{86} = (4, 8, 1) = a_4 \cap a_5 \cap c_{34} \cap c_{35} \)
58 : \( P_{87} = (5, 8, 1) = b_5 \cap b_6 \cap c_{45} \cap c_{46} \)
59 : \( P_{88} = (6, 8, 1) = a_6 \cap b_3 \cap c_{36} \cap d \)
60 : \( P_{89} = (7, 8, 1) = a_3 \cap b_4 \cap c_{12} \cap c_{56} \)
61 : \( P_{90} = (8, 8, 1) = b_1 \cap b_2 \cap c_{15} \cap c_{25} \)
62 : \( P_6 = (3, 1, 0) = a_5 \cap b_4 \cap c_{16} \cap c_{23} \)

The Kovalevskii 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:

<table>
<thead>
<tr>
<th>Rank</th>
<th>Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( P_6 = (3, 1, 0) )</td>
</tr>
<tr>
<td>1</td>
<td>( P_7 = (4, 1, 0) )</td>
</tr>
<tr>
<td>2</td>
<td>( P_8 = (5, 1, 0) )</td>
</tr>
<tr>
<td>3</td>
<td>( P_9 = (6, 1, 0) )</td>
</tr>
<tr>
<td>4</td>
<td>( P_{10} = (7, 1, 0) )</td>
</tr>
<tr>
<td>5</td>
<td>( P_{11} = (8, 1, 0) )</td>
</tr>
<tr>
<td>6</td>
<td>( P_{12} = (1, 0, 1) )</td>
</tr>
<tr>
<td>7</td>
<td>( P_{13} = (2, 0, 1) )</td>
</tr>
<tr>
<td>8</td>
<td>( P_{15} = (4, 0, 1) )</td>
</tr>
<tr>
<td>9</td>
<td>( P_{16} = (5, 0, 1) )</td>
</tr>
<tr>
<td>10</td>
<td>( P_{18} = (7, 0, 1) )</td>
</tr>
<tr>
<td>11</td>
<td>( P_{19} = (8, 0, 1) )</td>
</tr>
<tr>
<td>12</td>
<td>( P_{20} = (0, 1, 1) )</td>
</tr>
<tr>
<td>13</td>
<td>( P_{21} = (2, 1, 1) )</td>
</tr>
<tr>
<td>14</td>
<td>( P_{22} = (3, 1, 1) )</td>
</tr>
<tr>
<td>15</td>
<td>( P_{23} = (4, 1, 1) )</td>
</tr>
<tr>
<td>16</td>
<td>( P_{26} = (7, 1, 1) )</td>
</tr>
<tr>
<td>17</td>
<td>( P_{27} = (8, 1, 1) )</td>
</tr>
<tr>
<td>18</td>
<td>( P_{28} = (0, 2, 1) )</td>
</tr>
<tr>
<td>19</td>
<td>( P_{29} = (1, 2, 1) )</td>
</tr>
<tr>
<td>20</td>
<td>( P_{31} = (3, 2, 1) )</td>
</tr>
<tr>
<td>21</td>
<td>( P_{33} = (5, 2, 1) )</td>
</tr>
</tbody>
</table>

223
The lines and their points of contact are:

\[
\begin{align*}
a_1 &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & \alpha^3 \end{bmatrix}^8 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 8 \end{bmatrix}^8, \quad P_0 = P(1,0,0) \times 4 \\
a_2 &= \begin{bmatrix} 1 & 0 & \alpha^6 \\ 0 & 1 & 1 \end{bmatrix}^{51} = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 1 \end{bmatrix}^{51}, \quad P_{33} = P(1,8,1) \times 4 \\
a_3 &= \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & \alpha^6 \end{bmatrix}^{15} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 5 \end{bmatrix}^{15}, \quad P_{57} = P(2,5,1) \times 4 \\
a_4 &= \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & \alpha^2 \end{bmatrix}^{17} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 7 \end{bmatrix}^{17}, \quad P_{53} = P(7,4,1) \times 4 \\
a_5 &= \begin{bmatrix} 1 & 0 & \alpha^2 \\ 0 & 1 & \alpha^7 \end{bmatrix}^{74} = \begin{bmatrix} 1 & 0 & 7 \\ 0 & 1 & 4 \end{bmatrix}^{74}, \quad P_{30} = P(2,2,1) \times 4 \\
a_6 &= \begin{bmatrix} 1 & 0 & \alpha^2 \\ 0 & 1 & \alpha^7 \end{bmatrix}^{77} = \begin{bmatrix} 1 & 0 & 7 \\ 0 & 1 & 7 \end{bmatrix}^{77}, \quad P_5 = P(2,1,0) \times 4 \\
b_1 &= \begin{bmatrix} 1 & 0 & \alpha^6 \\ 0 & 1 & \alpha^7 \end{bmatrix}^{54} = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 4 \end{bmatrix}^{54}, \quad P_{58} = P(3,5,1) \times 4 \\
b_2 &= \begin{bmatrix} 1 & 0 & \alpha^7 \\ 0 & 1 & \alpha^6 \end{bmatrix}^{45} = \begin{bmatrix} 1 & 0 & 6 \\ 0 & 1 & 5 \end{bmatrix}^{45}, \quad P_{14} = P(3,0,1) \times 4 \\
b_3 &= \begin{bmatrix} 1 & 0 & \alpha \\ 0 & 1 & 1 \end{bmatrix}^{31} = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 1 \end{bmatrix}^{31}, \quad P_{62} = P(7,5,1) \times 4 \\
b_4 &= \begin{bmatrix} 1 & 0 & \alpha^5 \\ 0 & 1 & \alpha^2 \end{bmatrix}^{67} = \begin{bmatrix} 1 & 0 & 6 \\ 0 & 1 & 7 \end{bmatrix}^{67}, \quad P_{77} = P(4,7,1) \times 4 \\
b_5 &= \begin{bmatrix} 1 & 0 & \alpha^5 \\ 0 & 1 & \alpha^3 \end{bmatrix}^{68} = \begin{bmatrix} 1 & 0 & 6 \\ 0 & 1 & 8 \end{bmatrix}^{68}, \quad P_{41} = P(4,3,1) \times 4 \\
b_6 &= \begin{bmatrix} 1 & 0 & \alpha \\ 0 & 1 & \alpha^2 \end{bmatrix}^{37} = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 7 \end{bmatrix}^{37}, \quad P_3 = P(1,1,1) \times 4 \\
c_{12} &= \begin{bmatrix} 1 & 0 & \alpha^3 \\ 0 & 1 & \alpha^4 \end{bmatrix}^{82} = \begin{bmatrix} 1 & 0 & 8 \\ 0 & 1 & 2 \end{bmatrix}^{82}, \quad P_{17} = P(6,0,1) \times 4 \\
c_{13} &= \begin{bmatrix} 1 & 0 & \alpha \\ 0 & 1 & \alpha^4 \end{bmatrix}^{32} = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 2 \end{bmatrix}^{32}, \quad P_{84} = P(2,8,1) \times 4 \\
c_{14} &= \begin{bmatrix} 1 & 0 & \alpha^5 \\ 0 & 1 & 1 \end{bmatrix}^{61} = \begin{bmatrix} 1 & 0 & 6 \\ 0 & 1 & 1 \end{bmatrix}^{61}, \quad P_{32} = P(4,2,1) \times 4 \\
\end{align*}
\]
\[ \begin{align*}
\mathbf{c}_{15} &= \begin{bmatrix} 1 & 0 & \alpha^5 \\ 0 & 1 & 0 \end{bmatrix}, & P_1 = \mathbf{P}(0, 1, 0) 4 \times 60 \\
\mathbf{c}_{16} &= \begin{bmatrix} 1 & 0 & \alpha \\ 0 & 1 & \alpha^6 \end{bmatrix}, & P_{51} = \mathbf{P}(5, 4, 1) 4 \times 35 \\
\mathbf{c}_{23} &= \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & \alpha^5 \end{bmatrix}, & P_{82} = \mathbf{P}(0, 8, 1) 4 \times 16 \\
\mathbf{c}_{24} &= \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & \alpha^7 \end{bmatrix}, & P_{25} = \mathbf{P}(6, 1, 1) 4 \times 14 \\
\mathbf{c}_{25} &= \begin{bmatrix} 1 & 0 & \alpha^2 \\ 0 & 1 & \alpha^4 \end{bmatrix}, & P_{76} = \mathbf{P}(3, 7, 1) 4 \times 72 \\
\mathbf{c}_{26} &= \begin{bmatrix} 1 & 0 & \alpha^2 \\ 0 & 1 & \alpha^3 \end{bmatrix}, & P_{44} = \mathbf{P}(7, 3, 1) 4 \times 78 \\
\mathbf{c}_{34} &= \begin{bmatrix} 1 & 0 & \alpha^4 \\ 0 & 1 & \alpha^3 \end{bmatrix}, & P_{38} = \mathbf{P}(1, 3, 1) 4 \times 28 \\
\mathbf{c}_{35} &= \begin{bmatrix} 1 & 0 & \alpha^6 \\ 0 & 1 & \alpha^4 \end{bmatrix}, & P_{24} = \mathbf{P}(5, 1, 1) 4 \times 52 \\
\mathbf{c}_{36} &= \begin{bmatrix} 1 & 0 & \alpha^4 \\ 0 & 1 & \alpha^7 \end{bmatrix}, & P_{78} = \mathbf{P}(5, 7, 1) 4 \times 24 \\
\mathbf{c}_{45} &= \begin{bmatrix} 1 & 0 & \alpha^4 \\ 0 & 1 & \alpha^6 \end{bmatrix}, & P_{34} = \mathbf{P}(6, 2, 1) 4 \times 25 \\
\mathbf{c}_{46} &= \begin{bmatrix} 1 & 0 & \alpha^6 \\ 0 & 1 & \alpha \end{bmatrix}, & P_{46} = \mathbf{P}(0, 4, 1) 4 \times 53 \\
\mathbf{c}_{56} &= \begin{bmatrix} 1 & 0 & \alpha^4 \\ 0 & 1 & 1 \end{bmatrix}, & P_{4} = \mathbf{P}(1, 1, 0) 4 \times 21 \\
\mathbf{d} &= \begin{bmatrix} 1 & \alpha^6 & 0 \\ 0 & 0 & 1 \end{bmatrix}, & P_{2} = \mathbf{P}(0, 0, 1) 4 \times 59 
\end{align*} \]

Rank of lines: (8, 51, 15, 17, 74, 77, 54, 45, 31, 67, 68, 37, 82, 32, 61, 60, 35, 16, 14, 72, 78, 28, 52, 24, 25, 53, 21, 59 )
Line type: $1^{28}$

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

point types: $1^{28}$
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>1</td>
</tr>
<tr>
<td>63</td>
<td>4</td>
</tr>
</tbody>
</table>

### Point Types for Points Off the Curve: $4^{63}$

#### 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: $\{8, 9, 22, 24\}$

226
<table>
<thead>
<tr>
<th>Off point</th>
<th>( P )</th>
<th>Lies on 4 Bisecants</th>
</tr>
</thead>
<tbody>
<tr>
<td>27</td>
<td>( P_{42} = (5, 3, 1) )</td>
<td>{ 3, 7, 13, 26 }</td>
</tr>
<tr>
<td>28</td>
<td>( P_{43} = (6, 3, 1) )</td>
<td>{ 2, 11, 14, 19 }</td>
</tr>
<tr>
<td>29</td>
<td>( P_{45} = (8, 3, 1) )</td>
<td>{ 15, 17, 25, 27 }</td>
</tr>
<tr>
<td>30</td>
<td>( P_{47} = (1, 4, 1) )</td>
<td>{ 5, 7, 14, 22 }</td>
</tr>
<tr>
<td>31</td>
<td>( P_{48} = (2, 4, 1) )</td>
<td>{ 8, 10, 17, 19 }</td>
</tr>
<tr>
<td>32</td>
<td>( P_{50} = (3, 4, 1) )</td>
<td>{ 4, 11, 26, 27 }</td>
</tr>
<tr>
<td>33</td>
<td>( P_{52} = (4, 4, 1) )</td>
<td>{ 1, 2, 20, 23 }</td>
</tr>
<tr>
<td>34</td>
<td>( P_{53} = (6, 4, 1) )</td>
<td>{ 6, 9, 13, 21 }</td>
</tr>
<tr>
<td>35</td>
<td>( P_{54} = (8, 4, 1) )</td>
<td>{ 12, 15, 18, 24 }</td>
</tr>
<tr>
<td>36</td>
<td>( P_{55} = (0, 5, 1) )</td>
<td>{ 3, 5, 9, 11 }</td>
</tr>
<tr>
<td>37</td>
<td>( P_{56} = (1, 5, 1) )</td>
<td>{ 13, 20, 24, 27 }</td>
</tr>
<tr>
<td>38</td>
<td>( P_{60} = (5, 5, 1) )</td>
<td>{ 18, 19, 25, 26 }</td>
</tr>
<tr>
<td>39</td>
<td>( P_{60} = (5, 5, 1) )</td>
<td>{ 12, 14, 17, 21 }</td>
</tr>
<tr>
<td>40</td>
<td>( P_{61} = (6, 5, 1) )</td>
<td>{ 1, 4, 7, 10 }</td>
</tr>
<tr>
<td>41</td>
<td>( P_{63} = (8, 5, 1) )</td>
<td>{ 15, 16, 22, 23 }</td>
</tr>
<tr>
<td>42</td>
<td>( P_{64} = (0, 6, 1) )</td>
<td>{ 0, 10, 20, 21 }</td>
</tr>
<tr>
<td>43</td>
<td>( P_{65} = (1, 6, 1) )</td>
<td>{ 0, 9, 19, 23 }</td>
</tr>
<tr>
<td>44</td>
<td>( P_{66} = (2, 6, 1) )</td>
<td>{ 0, 11, 18, 22 }</td>
</tr>
<tr>
<td>45</td>
<td>( P_{67} = (3, 6, 1) )</td>
<td>{ 0, 1, 13, 17 }</td>
</tr>
<tr>
<td>46</td>
<td>( P_{68} = (4, 6, 1) )</td>
<td>{ 0, 7, 12, 27 }</td>
</tr>
<tr>
<td>47</td>
<td>( P_{69} = (5, 6, 1) )</td>
<td>{ 0, 2, 6, 8 }</td>
</tr>
<tr>
<td>48</td>
<td>( P_{70} = (6, 6, 1) )</td>
<td>{ 0, 3, 16, 25 }</td>
</tr>
<tr>
<td>49</td>
<td>( P_{71} = (7, 6, 1) )</td>
<td>{ 0, 4, 14, 24 }</td>
</tr>
<tr>
<td>50</td>
<td>( P_{72} = (8, 6, 1) )</td>
<td>{ 0, 5, 15, 26 }</td>
</tr>
<tr>
<td>51</td>
<td>( P_{73} = (0, 7, 1) )</td>
<td>{ 2, 7, 16, 24 }</td>
</tr>
<tr>
<td>52</td>
<td>( P_{74} = (1, 7, 1) )</td>
<td>{ 4, 8, 12, 25 }</td>
</tr>
<tr>
<td>53</td>
<td>( P_{75} = (2, 7, 1) )</td>
<td>{ 3, 6, 14, 27 }</td>
</tr>
<tr>
<td>54</td>
<td>( P_{76} = (3, 7, 1) )</td>
<td>{ 17, 20, 22, 26 }</td>
</tr>
<tr>
<td>55</td>
<td>( P_{80} = (7, 7, 1) )</td>
<td>{ 5, 10, 13, 18 }</td>
</tr>
<tr>
<td>56</td>
<td>( P_{81} = (8, 7, 1) )</td>
<td>{ 1, 11, 15, 21 }</td>
</tr>
<tr>
<td>57</td>
<td>( P_{85} = (3, 8, 1) )</td>
<td>{ 14, 16, 18, 20 }</td>
</tr>
<tr>
<td>58</td>
<td>( P_{86} = (4, 8, 1) )</td>
<td>{ 3, 4, 21, 22 }</td>
</tr>
<tr>
<td>59</td>
<td>( P_{87} = (5, 8, 1) )</td>
<td>{ 10, 11, 24, 25 }</td>
</tr>
<tr>
<td>60</td>
<td>( P_{88} = (6, 8, 1) )</td>
<td>{ 5, 8, 23, 27 }</td>
</tr>
<tr>
<td>61</td>
<td>( P_{90} = (7, 8, 1) )</td>
<td>{ 2, 9, 12, 26 }</td>
</tr>
<tr>
<td>62</td>
<td>( P_{90} = (8, 8, 1) )</td>
<td>{ 6, 7, 15, 19 }</td>
</tr>
</tbody>
</table>
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 [57]. The approach described in [36] utilizes the relation to non-conical six-arcs in a plane. A third approach is described in [37]. 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 [32]. 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:
▷ $(\text{ORBITER}) -v 5 \$
▷ ▷ -define F -finite_field -q 4 -end \$
▷ ▷ -define P -projective_space 3 F -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 \$
▷ ▷ ▷ -report -end \$
▷ ▷ -end \$
▷ ▷ -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

\[ \vdots \]

The orthogonal group

The Action

Group action $\Gamma L(4, 4) \text{OnWedge}$ of degree 1365
The group is a matrix group.
The base action is on projective space $PG(3, 4)$
\[ q = 4 \]
\[ p = 2 \]
\[ e = 2 \]
\[ n = 3 \]

Number of points = 85
Number of lines = 357
Number of lines on a point = 21
Number of points on a line = 5

\[ \vdots \]

The group stabilizing the fixed line

The Action

Group action $\Gamma L(4, 4) \text{OnWedge}_{res100}$ of degree 100

\[ \vdots \]

Strong generators for a group of order 5529600: \[ \vdots \]

The classification of five-plus-ones

Poset classification up to depth 5
The Orbits

Number of Orbits By Level

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

Summary of Orbit Representatives

N = node
D = depth or level
O = orbit with a level
Rep = orbit representative
(S,O) = (order of stabilizer, orbit length)
L = number of live points
F = number of flags
Gen = number of generators for the stabilizer of the orbit rep.

Table 7.3: 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>
<tr>
<td>10</td>
<td>5</td>
<td>3</td>
<td>{ 0, 3, 56, 80, 93 }</td>
<td>(120, 46080)</td>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Poset of Orbits in Detail

Classification of 5 + 1 Configurations in 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:

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & \omega^2 & 0 & 0 \\
0 & 0 & \omega & 0 \\
0 & 0 & 0 & 1
\end{bmatrix},
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & \omega & 0 & 0 \\
\omega^2 & 0 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix},
\begin{bmatrix}
1 & 0 & 0 & 0 \\
\omega & 0 & 0 & 0 \\
\omega & \omega & 0 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix},
\begin{bmatrix}
1 & 0 & 0 & 0 \\
\omega^2 & \omega^2 & 0 & 0 \\
\omega & 0 & 0 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix},
\begin{bmatrix}
1 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix},
\begin{bmatrix}
\omega^2 & \omega^2 & 0 & 0 \\
\omega & 0 & 0 & 0 \\
\omega & \omega & 0 & 0 \\
0 & 0 & 0 & 0
\end{bmatrix},
\begin{bmatrix}
\omega & 1 & \omega & 1 \\
\omega^2 & 1 & \omega^2 & 1 \\
\omega & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{bmatrix},
\begin{bmatrix}
\omega^2 & 0 & 0 & 0 \\
\omega^2 & 0 & 0 & 0 \\
\omega & 0 & 0 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}.
\]
The order of the group is 1974067200
The group has 1 orbits:

The orbits are:

(1) 0/1 \{16, 340, 38, 61, 156, 165, 155, 72, 54, 25, 356, 0\}_{1440} orbit length 1370880
Strong generators for a group of order 1440:

\[
\begin{bmatrix}
1 & \omega & 0 & 0 \\
0 & 1 & 0 & 0 \\
\omega & \omega^2 & 1 & \omega^2 \\
0 & \omega^2 & 0 & 1
\end{bmatrix}_1
\]

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

nb received = 0

Double Sixes

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

The orbits are:

(1) 0/1 \{16, 340, 38, 61, 156, 165, 155, 72, 54, 25, 356, 0\}_{1440} orbit length 1370880
Strong generators for a group of order 1440:

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

1,0,0,0,0,3,0,0,0,0,3,0,0,0,0,1,1,
1,0,0,0,0,3,0,0,0,3,0,3,0,0,1,0,1,
1,0,0,0,3,0,2,0,2,2,0,0,3,3,1,1,0,
1,0,0,0,2,0,2,0,1,2,0,0,2,1,2,1,1,
0,0,1,0,0,0,2,1,1,0,3,0,3,1,3,2,0,
1,1,0,0,3,0,0,0,0,0,3,3,0,0,1,0,1,
The overall number of objects is: 1370880

**Flag orbits for surfaces**

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

(1) Flag orbit 0 / 1 down=(0,0) up=(0,-1) is ( 16, 340, 38, 61, 156, 165, 155, 72, 54, 25, 356, 0, 71, 55, 26, 100, 1, 138, 109, 345, 84, 85, 122, 110, 145, 139, 81 )
with a stabilizer of order 1440

Strong generators for a group of order 1440:

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

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

nb received = 0

**Surfaces**

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

The orbits are:

(1) 0/1 \{16, 340, 38, 61, 156, 165, 155, 72, 54, 25, 356, 0, 71, 55, 26, 100, 1, 138, 109, 345, 84, 85, 122, 110, 145, 139, 81\}_{51840} orbit length 38080
Strong generators for a group of order 51840:

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

The overall number of objects is: 38080

**The Group** 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:

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

The overall number of objects is: 38080

**Surface 4\#0**

**The equation**

The equation of the surface is:

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

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

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

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

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

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

237
\begin{align*}
\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} = \text{Pl}(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} = \text{Pl}(0, 1, 0, 0, 0, 0)_{1} \\
\ell_4 &= a_5 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}_{0} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}_{0} = \text{Pl}(1, 0, 0, 0, 0, 0)_{0} \\
\ell_5 &= a_6 = \begin{bmatrix} 1 & 0 & \omega^2 & 1 \\ 0 & 1 & 0 & \omega \end{bmatrix}_{155} = \begin{bmatrix} 1 & 0 & 3 & 1 \\ 0 & 1 & 0 & 2 \end{bmatrix}_{155} = \text{Pl}(3, 2, 0, 2, 3, 1)_{314} \\
\ell_6 &= b_1 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_{340} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_{340} = \text{Pl}(0, 0, 0, 1, 0, 0)_{9} \\
\ell_7 &= b_2 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}_{38} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}_{38} = \text{Pl}(0, 0, 1, 1, 1, 1)_{198} \\
\ell_8 &= b_3 = \begin{bmatrix} 1 & \omega & 0 & 0 \\ 0 & 0 & 1 & \omega^2 \end{bmatrix}_{61} = \begin{bmatrix} 1 & 2 & 0 & 0 \\ 0 & 0 & 1 & 3 \end{bmatrix}_{61} = \text{Pl}(0, 0, 2, 3, 2, 1)_{265} \\
\ell_9 &= b_4 = \begin{bmatrix} 1 & 0 & \omega^2 & 1 \\ 0 & 1 & 1 & \omega \end{bmatrix}_{156} = \begin{bmatrix} 1 & 0 & 3 & 1 \\ 0 & 1 & 1 & 2 \end{bmatrix}_{156} = \text{Pl}(3, 0, 3, 2, 3, 1)_{335} \\
\ell_{10} &= b_5 = \begin{bmatrix} 1 & \omega & 0 & 1 \\ 0 & 0 & 1 & \omega \end{bmatrix}_{165} = \begin{bmatrix} 1 & 3 & 0 & 1 \\ 0 & 0 & 1 & 2 \end{bmatrix}_{165} = \text{Pl}(0, 2, 3, 2, 3, 1)_{337} \\
\ell_{11} &= b_6 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}_{16} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}_{16} = \text{Pl}(0, 0, 1, 0, 0, 0)_{2} \\
\ell_{12} &= c_{12} = \begin{bmatrix} 1 & 0 & \omega & 1 \\ 0 & 1 & 0 & \omega^2 \end{bmatrix}_{138} = \begin{bmatrix} 1 & 0 & 2 & 1 \\ 0 & 1 & 0 & 3 \end{bmatrix}_{138} = \text{Pl}(2, 3, 0, 3, 2, 1)_{256} \\
\ell_{13} &= c_{13} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}_{109} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}_{109} = \text{Pl}(1, 1, 0, 1, 1, 1)_{189} \\
\ell_{14} &= c_{14} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_{345} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_{345} = \text{Pl}(0, 1, 0, 1, 0, 0)_{13} \\
\ell_{15} &= c_{15} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}_{84} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}_{84} = \text{Pl}(1, 0, 0, 1, 0, 0)_{10} \\
\end{align*}
\[ \ell_{16} = c_{16} = \begin{bmatrix} 1 & 0 & \omega^2 & 0 \\ 0 & 1 & 0 & \omega \end{bmatrix}_{71} = \begin{bmatrix} 1 & 0 & 3 & 0 \\ 0 & 1 & 0 & 2 \end{bmatrix}_{71} = \text{Pl}(3, 2, 0, 0, 3, 1)_{299} \]

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

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

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

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

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

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

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

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

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

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

Rank of lines: \( (72, 54, 25, 356, 0, 155, 340, 38, 61, 156, 165, 16, 138, 109, 345, 84, 71, 85, 122, 110, 55, 145, 139, 26, 81, 100, 1) \)

Rank of points on Klein quadric: \( (308, 238, 177, 1, 0, 314, 9, 198, 265, 335, 337, 2, 256, 189, 13, 10, 299, 16, 202, 199, 244, 271, 267, 180, 332, 6, 3) \)

**All Points on surface**

The surface has 45 points
Eckardt Points

The surface has 45 Eckardt points:

0 : $E_{56} = a_5 \cap b_6 \cap c_{56} = P_0 = P_0 = P(1,0,0,0) = P(1,0,0,0), T = 0$
1 : $E_{51} = a_5 \cap b_1 \cap c_{15} = P_1 = P_1 = P(0,1,0,0) = P(0,1,0,0), T = 4$
2 : $E_{46} = a_4 \cap b_6 \cap c_{46} = P_2 = P_2 = P(0,0,1,0) = P(0,0,1,0), T = 20$
3 : $E_{41} = a_4 \cap b_1 \cap c_{41} = P_3 = P_3 = P(0,0,0,1) = P(0,0,0,1), T = 84$
4 : $E_{32} = a_3 \cap b_2 \cap c_{32} = P_4 = P_4 = P(1,1,1,1) = P(1,1,1,1), T = 27$
5 : $E_{52} = a_5 \cap b_2 \cap c_{52} = P_5 = P_5 = P(1,1,0,0) = P(1,1,0,0), T = 1$
6 : $E_{54} = a_5 \cap b_4 \cap c_{54} = P_6 = P_6 = P(\omega,1,0,0) = P(2,1,0,0), T = 2$
7 : $E_{33} = a_5 \cap b_3 \cap c_{33} = P_7 = P_7 = P(\omega^2,1,0,0) = P(3,1,0,0), T = 3$
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 : $E_{14,23,56} = c_{14} \cap c_{23} \cap c_{56} = P_{11} = P_{11} = P(0,1,0,0) = P(0,1,0,0), T = 9$
12 : $E_{13,24,56} = c_{13} \cap c_{24} \cap c_{56} = P_{12} = P_{12} = P(1,1,0,0) = P(1,1,0,0), T = 6$
13 : $E_{69} = a_6 \cap b_5 \cap c_{56} = P_{13} = P_{13} = P(\omega,1,1,0) = P(2,1,1,0), T = 12$
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 : $E_{21} = a_2 \cap b_1 \cap c_{12} = P_{18} = P_{30} = P(0,0,0,1) = P(0,2,0,1), T = 46$
19 : $E_{15,26,34} = c_{15} \cap c_{26} \cap c_{34} = P_{19} = P_{31} = P(1,0,1,1) = P(1,2,0,1), T = 24$
20 : $E_{61} = a_6 \cap b_1 \cap c_{16} = P_{20} = P_{34} = P(0,0,2,0,1) = P(0,3,0,1), T = 67$
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 : $E_{62} = a_6 \cap b_2 \cap c_{26} = P_{25} = P_{47} = P(\omega,1,1,1) = P(2,2,1,1), T = 53$
26 : $E_{25} = a_1 \cap b_5 \cap c_{25} = P_{26} = P_{48} = P(\omega^2,\omega,1,1) = P(3,2,1,1), T = 80$
27 : $E_{16,25,34} = c_{16} \cap c_{25} \cap c_{34} = P_{27} = P_{51} = P(\omega^2,1,1,1) = P(2,3,1,1), T = 55$
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 : $E_{35} = a_3 \cap b_5 \cap c_{35} = P_{31} = P_{59} = P(\omega,1,\omega,1) = P(2,1,2,1), T = 49$
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 : $E_{14,26,35} = c_{14} \cap c_{26} \cap c_{35} = P_{33} = P_{61} = P(0,0,\omega,1) = P(0,2,2,1), T = 51$
34 : $E_{23} = a_2 \cap b_3 \cap c_{23} = P_{34} = P_{62} = P(1,\omega,\omega,1) = P(1,2,2,1), T = 39$
35 : $E_{13} = a_1 \cap b_3 \cap c_{13} = P_{35} = P_{67} = P(\omega,\omega,\omega,1) = P(2,3,2,1), T = 50$
36 : $E_{16,24,35} = c_{16} \cap c_{24} \cap c_{35} = P_{36} = P_{68} = P(\omega^2,\omega^2,\omega,1) = P(3,3,2,1), T = 74$
37 : $E_{45} = a_4 \cap b_5 \cap c_{45} = P_{37} = P_{69} = P(0,0,\omega^2,1) = P(0,0,3,1), T = 83$
38 : $E_{64} = a_6 \cap b_4 \cap c_{46} = P_{38} = P_{70} = P(1,0,\omega^2,1) = P(1,0,3,1), T = 31$
39 : $E_{12,36,45} = c_{12} \cap c_{36} \cap c_{45} = P_{39} = P_{75} = P(\omega,1,\omega^2,1) = P(2,1,3,1), T = 59$
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 = $E_{12,36,45}, E_{14,26,35}, E_{13,25,46}$

241
**Tritangent planes**

The 45 tritangent planes are:

\[
\pi_{12} = \pi_0 = 79 = \begin{bmatrix} 1 & 0 & 0 & \omega^2 \\ 0 & 1 & 0 & \omega^2 \\ 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 1 \end{bmatrix} = V(\omega^2X_0 + \omega^2X_1 + X_2 + X_3) = V(3X_0 + 3X_1 + X_2 + X_3)
\]

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

\[
\pi_{16,25,34} = \pi_{44} = 55 = \begin{bmatrix} 1 & 0 & 0 & \omega \\ 0 & 1 & 0 & \omega \\ 0 & 0 & 1 & \omega^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 3 \end{bmatrix} = V(\omega X_0 + \omega X_1 + \omega^2 X_2 + X_3) = V(2X_0 + 2X_1 + 3X_2 + X_3)
\]

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

Karaoglu [36] 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 \\
▷ ▷ -define F -finite_field -q 4 -end \\
▷ ▷ -define P -projective_space 3 F -end \\
▷ ▷ -with P -do \\
▷ ▷ -projective_space_activity \\
▷ ▷ ▷ -control_six_arcs -problem_label sixarcs_q4 -end \\
▷ ▷ ▷ -classify_surfaces_through_arcs_and_two_lines \\
▷ ▷ -end \\
▷ pdflatex surfaces_arc_lifting_4.tex
▷ open surfaces_arc_lifting_4.pdf
```

classifies all cubic surfaces with 27 lines over the field $\mathbb{F}_4$ using the algorithm of Karaoglu. The result agrees with the previous algorithm. The only surface with 27 lines in PG(3, 4) is the Hirschfeld surface.
Table 7.4: Projective space activities related to the recognition of cubic surfaces

<table>
<thead>
<tr>
<th>Modifier</th>
<th>Arguments</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-surface_identify_Eckardt</td>
<td></td>
<td>Identifies the isomorphism type of the Eckardt surface with parameter ( a ). All values of ( a ) are considered.</td>
</tr>
<tr>
<td>-surface_identify_F13</td>
<td></td>
<td>Identifies the isomorphism type of the ( F_{13} ) surface with parameter ( a ). All values of ( a ) are considered.</td>
</tr>
<tr>
<td>-surface_identify_Bes</td>
<td></td>
<td>Identifies the isomorphism type of the Bes surface with parameters ( a ) and ( c ). All values of ( a, c ) are considered.</td>
</tr>
<tr>
<td>-surface_identify_general_abcd</td>
<td></td>
<td>Identifies the isomorphism type of the general surface with parameters ( a,b,c,d ). All values of ( a,b,c,d ) are considered.</td>
</tr>
<tr>
<td>-surface_isomorphism_testing</td>
<td>surface-descr-1 surface-descr-2</td>
<td>Computes an isomorphism between two given surfaces or concludes that none exists.</td>
</tr>
<tr>
<td>-surface_recognize</td>
<td>surface-descr</td>
<td>Identifies the isomorphism type of the given surface.</td>
</tr>
<tr>
<td>-create_surface</td>
<td>surface-descr</td>
<td>Creates a surface from a description. See Section 7.1.</td>
</tr>
</tbody>
</table>

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 3 F -end \n  > ▶ -with P -do \n  > ▶ -projective_space_activity \n  > ▶ ▶ -classify_surfaces_with_double_sixes Surf -W -end \n  > ▶ -end \n  > ▶ -with Surf -do \n  > ▶ ▶ -classification_of_cubic_surfaces_with_double_sixes_activity \n  > ▶ ▶ -recognize \n```

243
identifies the surface (cf. Table 4.3)

\[ X_0^2X_3 + X_1^2X_2 + X_1X_2^2 + X_2X_3^2 + X_1X_2X_3 = 0 \]  

(7.1)

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

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

(7.2)

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

```
surface_isomorph_16:
```

```bash
$ (ORBITER) -v 3 \
  -define F -finite_field -q 16 -end \
  -define P -projective_space 3 F -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 \
  -q 16 -by_coefficients \
  "13,6,3,8,3,11,13,13,1,19" -end \
  -end \
  -end \
  -print_symbols
```
computes an isomorphism between two cubic surfaces with 27 lines

\[
\begin{align*}
X_0^2X_2 + X_1^2X_2 + X_2^2X_3 + X_0X_2^2 + X_1X_2^2 + X_2X_3^2 + \delta^{13}X_1X_3^2 + \delta^{12}X_2X_3^2 + \\
\delta^7X_0X_2X_3 + \delta^7X_1X_2X_3
\end{align*}
\]

and

\[
\begin{align*}
\delta^{11}X_0^2X_3 + \delta^{12}X_1^2X_2 + \delta^{12}X_1X_2^2 + \delta^{11}X_0X_3^2 + X_1X_2X_3 = 0
\end{align*}
\]

over the field \( \mathbb{F}_{16} \).

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

Orbiter can recognize the isomorphism type of a cubic surface with 27 lines. 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

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

```
surface_recognize_8:
  ▶ $ (ORBITER) -v 3 \ 
  ▶ ▶ -define F -finite_field -q 8 -end \ 
  ▶ ▶ -define P -projective_space 3 F -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 \ 
  ▶ ▶ ▶ -recognize \ 
  ▶ ▶ ▶ ▶ -q 8 \ 
  ▶ ▶ ▶ ▶ -by_coefficients "1,6,1,8,1,11,1,13,1,19" \ 
  ▶ ▶ ▶ ▶ -end \ 
  ▶ ▶ ▶ -end \ 
  ▶ ▶ -end \ 
  ▶ -print_symbols
```

The command find that the surface is isomorphic to the surface with OCN=0. An isomorphism will be computed as well.
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 [23]. 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:

```bash
orbits_cubic_curves.q2:
  $\text{ORBITER} -v 4 \$
  $\text{-define G -linear group -PGL 3 2 -end} \$
  $\text{-with G -do} \$
  $\text{-group_theoretic_activity} \$
  $\text{-orbits_on_polynomials 3} \$
  $\text{-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:

```bash
poly_orbits_d3_n3_q2_F2.csv: poly_orbits_d3_n3_q2.csv
  $\text{ORBITER} -v 4 \$
  $\text{-define F -finite field -q 2 -end} \$
  $\text{-define P -projective space 3 F -end} \$
  $\text{-with P -do} \$
  $\text{-projective_space_activity} \$
  $\text{-table_of_cubic_surfaces_compute_properties} \$
  $\text{-poly_orbits_d3_n3_q2.csv 2 0} \$
  $\text{-end} \$
```

and

```bash
Dickson_q2_analyze: poly_orbits_d3_n3_q2_F2.csv
  $\text{ORBITER} -v 4 \$
  $\text{-define F -finite field -q 2 -end} \$
  $\text{-define P -projective space 3 F -end} \$
  $\text{-with P -do} \$
  $\text{-projective_space_activity} \$
  $\text{-cubic_surface_properties_analyze} \$
  $\text{-poly_orbits_d3_n3_q2_F2.csv 2} \$
  $\text{-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
  $\text{ORBITER} -v 4 \backslash$
  $\text{-define F -finite_field -q 4 -end} \backslash$
  $\text{-define P -projective_space 3 F -end} \backslash$
  $\text{-with P -do} \backslash$
  $\text{-projective_space_activity} \backslash$
  $\text{-table_of_cubic_surfaces_compute_properties} \backslash$
  $\text{poly_orbits_d3_n3_q2.csv 2 0} \backslash$
  $\text{-end} \backslash$

and

Dickson_q4_analyze: poly_orbits_d3_n3_q2_F4.csv
  $\text{ORBITER} -v 4 \backslash$
  $\text{-define F -finite_field -q 4 -end} \backslash$
  $\text{-define P -projective_space 3 F -end} \backslash$
  $\text{-with P -do} \backslash$
  $\text{-projective_space_activity} \backslash$
  $\text{-cubic_surface_properties_analyze} \backslash$
  $\text{poly_orbits_d3_n3_q2_F4.csv 2} \backslash$
  $\text{-end} \backslash$
  $\text{pdflatex poly_orbits_d3_n3_q2_F4_report.tex} \backslash$
  $\text{open poly_orbits_d3_n3_q2_F4_report.pdf} \backslash$
```
7.6 ATLAS and Tables

The data in Orbiter can be exported to be used for automated processing. It is possible to create a csv file with the cubic surfaces with 27 lines for a given $q$. The following example shows how to export the data about cubic surfaces with $q = 17$:

```
MAKE_TABLE_OF_CUBIC_SURFACES=-define P -projective_space 3 F -end \  
▷ ▷ -with P -do \  
▷ ▷ ▷ -projective_space.activity \  
▷ ▷ ▷ ▷ -table_of_cubic_surfaces \  
▷ ▷ ▷ -end
```

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

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

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

produces a latex table from the csv file.
Chapter 8
Ring Theory

8.1 Polynomials Over Finite Fields

For $p$ prime, the finite field $\mathbb{F}_p$ of order $p$ can be constructed as factor ring 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 \texttt{-finite_field_activity} command can be used to define a finite field. The \texttt{-q $q$} option can be used to specify the order of the finite field. The \texttt{-override_polynomial $a$} option can be used to specify the polynomial $m(X)$ as integer $a$ in the base $p$ representation. This option can be omitted, 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:
  > $(ORBITER) -v 2 \n  >  -define F -finite_field -q 2 -end \n  >  -with F -do \n  >  -finite_field_activity \n  >  -polynomial_division "1,0,0,0,0,0,0,0,0,1" "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.

249
<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.

```plaintext
poly_division2:
> $(ORBITER) -v 2 \n> -define F -finite_field -q 2 -end \n> -define A -vector -field F -sparse 11 "1,0,1,10" -end \n> -define B -vector -field F -dense "1,0,1,1" -end \n> -with F -do \n> -finite_field_activity \n> -polynomial_division A B -end
```

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,

```plaintext
poly_gcd:
> $(ORBITER) -v 2 \n> -define F -finite_field -q 2 -end \n> -with F -do \n> -finite_field_activity \n> -extended_gcd_for_polynomials "1,0,0,0,0,0,0,0,0,0,1" "1,0,1,1" -end
```

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

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

```plaintext
poly_mult_mod1:
> $(ORBITER) -v 2 \n> -define F -finite_field -q 7 -end \n> -with F -do \n> -finite_field_activity \n> -polynomial_mult_mod "1,2,3" "3,4,5" "6,0,0,1" -end
```
which has a result of

\[ X^2 + 4X + 4. \]

Observe how the coefficients are given from the lowest to the highest term. For the opposite order, the following command computes

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

\texttt{poly\_mult\_mod2:}
\begin{verbatim}
> $(ORBITER) -v 2 \
>   > -define F -finite_field -q 7 -end \n>   > -with F -do \n>   >   > -finite_field_activity \n>   >   > -polynomial_mult_mod "3,1,2" "5,3,4" "6,0,0,1" -end
\end{verbatim}

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:

\texttt{poly\_mult\_mod\_F4:}
\begin{verbatim}
> $(ORBITER) -v 2 \
>   > -define F -finite_field -q 2 -end \n>   > -with F -do \n>   >   > -finite_field_activity \n>   >   > -polynomial_mult_mod "1,1" "1,1" "1,1,1" -end
> $(ORBITER) -v 2 \
>   > -define F -finite_field -q 2 -end \n>   > -with F -do \n>   >   > -finite_field_activity \n>   >   > -polynomial_mult_mod "0,1" "1,1" "1,1,1" -end
> $(ORBITER) -v 2 \
>   > -define F -finite_field -q 2 -end \n>   > -with F -do \n>   >   > -finite_field_activity \n>   >   > -polynomial_mult_mod "0,1" "0,1" "1,1,1" -end
\end{verbatim}

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 \( \mathbb{F}_2 \):

\texttt{mult\_polynomials\_2.5.7:}
\begin{verbatim}
> $(ORBITER) -v 2 \
>   > -define F -finite_field -q 2 -end \n\end{verbatim}
The next command performs polynomial long division based on numerical polynomials:

```
polynomial.division.ranked.2.27.13:
  $(ORBITER) -v 2 \
  -define F -finite_field -q 2 -end \
  -with F -do \
  -finite_field_activity \
  -polynomial.division.ranked 27 13 \
  -end
```
```
\pdflatex polynomial.division.27.13.tex
```
```
\open polynomial.division.27.13.pdf
```

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

```
mult.polynomials.1024.999.997:
  $(ORBITER) -v 2 \
  -define F -finite_field -q 2 -end \
  -with F -do \
  -finite_field_activity \
  -mult.polynomials 999 997 \
  -end
```
```
\pdflatex polynomial.mult.999.997.tex
```
```
\open polynomial.mult.999.997.pdf
```

The next command performs division with remainder:

```
polynomial.division.ranked.2.349147.1033:
  $(ORBITER) -v 2 \
  -define F -finite_field -q 2 -end \
  -with F -do \
  -finite_field_activity \
  -polynomial.division.ranked 349147 1033 \
  -end
```
```
\pdflatex polynomial.division.349147.1033.tex
```
```
\open polynomial.division.349147.1033.pdf
```

```
253
```
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 \\
  ▶ ▶ -define F -finite_field -q 1024 -end \\
  ▶ ▶ -with F -do \\
  ▶ ▶ -finite_field_activity -parse_and_evaluate \\
  ▶ ▶ "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 \\
  ▶ ▶ -define F -finite_field -q 2 -end \\
  ▶ ▶ -define M -vector -field F -sparse 33 $(CRC32_SPARSE) -end \\
  ▶ ▶ -define A -vector -field F -sparse 2 "1,1" -end \\
  ▶ ▶ -with F -do \\
  ▶ ▶ -finite_field_activity \\
  ▶ ▶ -polynomial_power_mod A $(TWO_TO_THE_32_MINUS_2) M \\
  ▶ ▶ -end
```

This command produces the polynomial

$$B(X) = X^{31} + X^{25} + X^{22} + X^{21} + X^{15} + X^{11} + X^{10} + X^9 + X^7 + X^6 + X^4 + X^3 + X + 1$$

In order to test that this polynomial really is the multiplicative inverse of $X$ modulo CRC32, we perform the following command:
INVERSE\_SPARSE="1,31,1,25,1,22,1,21,1,15,\
1,11,1,10,1,9,1,7,1,6,1,4,1,3,1,1,1,0"

mult\_mod\_to\_get\_one:

```
▷ $(ORBITER) -v 2 \
▷ ▷ -define F -finite_field -q 2 -end \
▷ ▷ -define M -vector -field F -sparse 33 $(CRC32\_SPARSE) -end \
▷ ▷ -define A -vector -field F -sparse 2 "1,1" -end \
▷ ▷ -define B -vector -field F -sparse 33 $(INVERSE\_SPARSE) -end \
▷ ▷ -with F -do \
▷ ▷ -finite_field_activity \
▷ ▷ ▷ -polynomial_mult_mod A B M \
▷ ▷ -end
```

The product is indeed 1.

The Berlekamp matrix can be used to test if a polynomial is irreducible over a given finite field. The polynomial is irreducible if and only if the rank of the Berlekamp matrix is \(d - 1\), where \(d\) is the degree of the polynomial. For instance, the command

Berlekamp\_matrix\_2,3:

```
▷ $(ORBITER) -v 2 \
▷ ▷ -define F -finite_field -q 2 -end \
▷ ▷ -define v -vector -field F -dense "1,1,0,1" -end \
▷ ▷ -with F -do \
▷ ▷ -finite_field_activity \
▷ ▷ -Berlekamp\_matrix v -end
```

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 \n▷ ▷ -search_for_primitive_polynomial_in_range 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 \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
```

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.

```
CREMONA_MAP_Y0 = "3*y0*y0*y0*y0*y0*y2+4*y0*y0*y0*y1*y1*y2+
+2*y0*y0*y0*y2*y2*y2+y0*y1*y1*y1*y1*y2+
+6*y0*y1*y1*y2*y2*y2+9*y0*y2*y2*y2*y2*y2"

CREMONA_MAP_Y1 = "y0*y0*y0*y0*y0*y1+5*y0*y0*y1*y1*y1*y1+
+12*y0*y0*y1*y1*y2*y2+3*y0*y1*y1*y1*y1*y1+
+5*y0*y1*y1*y1*y2*y2+y0*y1*y2*y2*y2*y2"

CREMONA_MAP_Y2 = "10*y0*y0*y0*y0*y0+y0+11*y0*y0*y0*y0*y1*y1*y1+
+11*y0*y0*y0*y0*y2*y2+4*y0*y0*y1*y1*y1*y1+
+9*y0*y0*y1*y1*y2*y2+4*y0*y0*y2*y2*y2*y2"

CREMONA_MAP_Y3 = "0"
```

Cremona_map:

```
> $(ORBITER) -v 3 \
>   -define F -finite_field -q 13 -end \n>   -define P -projective_space 2 F -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 \
```

258
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 $\text{PG}(2,11)$:

```
$>$ $>$ $>$ -projective_space_activity \\
$>$ $>$ $>$ $>$ -map R Cremona "" \\
$>$ $>$ $>$ -end
```

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

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

They are stored in the file `arcs_5_2_q11_lvl_5`. Let us now create the ideal in the quadratic component of the polynomial ring in three variables over $\mathbb{F}_{11}$:

```
$>$ $>$ $>$ $>$ arcs_5_2_q11:  \\
$>$ $>$ $(ORBITER)$ -v 2  \\
$>$ $>$ $>$ -define F -finite_field -q 11 -end  \\
$>$ $>$ -define R -projective_space 2 F -end  \\
$>$ $>$ -with F -do  \\
$>$ $>$ $>$ -projective_space_activity  \\
$>$ $>$ $>$ $>$ -classify_arcs  \\
$>$ $>$ $>$ $>$ $>$ -poset_classification_control  \\
$>$ $>$ $>$ $>$ $>$ $>$ -problem_label arcs_5_2_q11  \\
$>$ $>$ $>$ $>$ $>$ $>$ -W -depth 5  \\
$>$ $>$ $>$ $>$ $>$ $>$ -report -end  \\
$>$ $>$ $>$ $>$ -end  \\
$>$ $>$ $>$ -target_size 5  \\
$>$ $>$ $>$ $>$ -d 2  \\
$>$ $>$ $>$ -end  \\
$>$ $>$ $>$ -end
```

```
pdflatex arcs_5_2_q11_poset.tex  \\
open arcs_5_2_q11_poset.pdf
```

```
arcs_5_2_q11_ideal:  \\
$>$ $(ORBITER)$ -v 2  \\
$>$ $>$ -define F -finite_field -q 11 -end  \\
$>$ $>$ -define R -polynomial_ring  \\
$>$ $>$ $>$ -field F  \\
$>$ $>$ $>$ -number_of_variables 3  \\
$>$ $>$ $>$ -homogeneous_of_degree 2  \\
$>$ $>$ $>$ -monomial_ordering_lex  \\
$>$ $>$ $>$ -variables "x0, x1, x2" "x_0, x_1, x_2"  \\
$>$ $>$ $>$ -end  \\
$>$ $>$ -define C -combinatorial_objects
```
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.

```
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:

```
surface_9lines_4E_ideal:
> $(ORBITER) -v 2 \n> -define Pts -vector -dense \n> $(PTS_OF_SURFACE_ORBIT211_Q3_L9_E4) \n> -end \n> -define F -finite_field -q 3 -end \n> -define R -polynomial_ring \n> -field F \n> -number_of_variables 4 \n> -homogeneous_of_degree 3 \n> -monomial_ordering.lex \n> -variables "x0,x1,x2,x3" "x_0,x_1,x_2,x_3" \n> -end \n> -with R -do \n> -ring_theoretic_activity \n> -ideal R "surf_eqn" "surf\_eqn" Pts \n> -end
```
We find a two-dimensional ideal. Generators are:

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

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

\[
\text{SURFACE_F_9} = x_0 x_0 x_1 - x_0 x_1 x_1 - x_0 x_1 x_3 - x_2 x_2 x_3 - x_2 x_3 x_3
\]

\[
\begin{align*}
\text{F}_9.q7: & \quad $(\text{ORBITER}) \ -v \ 3 \ \\
& \quad \ -define \ F \ -\text{finite\_field} \ -q \ 7 \ -end \ \\
& \quad \ -define \ P \ -\text{projective\_space} \ 3 \ F \ -end \ \\
& \quad \ -with \ P \ -do \ \\
& \quad \ -\text{projective\_space\_activity} \ \\
& \quad \ -define \ surface \ F_9 \ -q \ 7 \ \\
& \quad \ -by\text{-equation} \ "F_9" \ \\
& \quad \ -define \ \"DF_9\" \ "x_0,x_1,x_2,x_3" \ \\
& \quad \ -$(\text{SURFACE_F_9}) \ \\
& \quad \ -"" \ \\
& \quad \ -end \ \\
& \quad \ -end \ \\
& \quad \ -with \ F_9 \ -do \ \\
& \quad \ -\text{cubic\_surface\_activity} \ \\
& \quad \ -report \ \\
& \quad \ -end \ \\
& \quad \ \text{pdflatex surface\_equation_F_9\_q7\_report.tex} \\
& \quad \ \text{open surface\_equation_F_9\_q7\_report.pdf}
\end{align*}
\]

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:

\[
\text{random_k\_subsets:} \\
\quad $(\text{ORBITER}) \ -v \ 4 \ \\
\quad \ -create\_random\_k\_subsets \ 133 \ 5 \ 20
\]

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

\[
\text{line\_type\_in\_PG.2.11:} \\
\quad $(\text{ORBITER}) \ -v \ 3 \ \\
\]

261
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:

```plaintext
random_arc_5_2.q11_ideal:
  $(ORBITER) -v 2 \
  -define F -finite_field -q 11 -end \
  -define R -polynomial_ring \
  -field F \
  -number_of_variables 3 \
  -homogeneous_of_degree 2 \
  -monomial_ordering lex \
  -variables "x0,x1,x2" "x_0,x_1,x_2" \
  -end \
  -define C -combinatorial_objects \
  -set_of_points "3,33,40,83,102" \
  -end \
  -with C -do \
  -combinatorial_object_activity \
  -ideal R \
  -end
```

The ideal is generated by

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

The conic contains the following 12 points:

\[\{3, 15, 19, 33, 40, 42, 46, 50, 83, 88, 102, 108\}\]
Chapter 9

Applications

9.1 Number Theory

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

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

computes the inverse of 18059241 modulo 58014043.

The Legendre symbol tells us if a number \(a\) is a square modulo an odd prime \(p\). By definition,

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

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

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

where

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

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

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

263
<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 \texttt{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 \texttt{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:

\[
\begin{align*}
\left( \frac{2221}{7817} \right) \\
&= \left( \frac{7817}{2221} \right) \cdot \left( -1 \right)^{\frac{2221-1}{2}} \cdot \left( \frac{7817}{2} \right) \\
&= \left( \frac{7817}{2221} \right) \\
&= \left( \frac{1154}{2221} \right) \\
&= \left( \frac{2}{2221} \right) \cdot \left( \frac{577}{2221} \right) \\
&= \left( -1 \right)^{\frac{2221^2-1}{2}} \cdot \left( \frac{577}{2221} \right) \\
&= \left( -1 \right) \cdot \left( \frac{577}{2221} \right) \\
&= \left( -1 \right) \cdot \left( \frac{2221}{577} \right) \cdot \left( -1 \right)^{\frac{577-1}{2}} \cdot \left( \frac{2221}{2} \right) \\
&= \left( -1 \right) \cdot \left( \frac{2221}{577} \right) \\
&= \left( -1 \right) \cdot \left( \frac{490}{577} \right)
\end{align*}
\]
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:
```

```bash
$\text{ORBITER} -v 2 \text{-square_root_mod} 2221 7817
```

yields that 7634 is a square root. Indeed,

\[
7634^2 \equiv 2221 \mod 7817.
\]
Table 9.2: Representation Theory Commands

### 9.2 Representation Theory

Orbiter has some commands for representations of finite groups. Table 9.2 list the commands available to classify arcs. The command

```bash
representation_on_polynomials_of_degree_3:
  ▶ $(ORBITER) -v 4 \n      ▶ ▶ -define G -linear_group -PGL 4 3 -end \n      ▶ ▶ -with G -do \n      ▶ ▶ -group_theoretic_activity \n      ▶ ▶ ▶ -representation_on_polynomials 3 \n      ▶ ▶ -end
      ▶ $(ORBITER) -v 2 \n      ▶ ▶ -loop L 0 9 1 -draw_matrix \n      ▶ ▶ ▶ -input_csv_file PGL_4_3_rep_3_%L.csv \n      ▶ ▶ ▶ -box_width 40 -bit_depth 24 -partition 3 20 20 -end \n      ▶ ▶ -end_loop
```

creates $G = \text{PGL}(4,3)$ and computes the representation on polynomials of degree 3 in 4 variables. The representation has degree 20. The second command produces bitmap drawings for the representing matrices associated with a generating set of the group. Figure 9.1 shows the representing matrices for a generating set of size 9.
Figure 9.1: Representation of PGL(4,3) on cubic polynomials
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>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 n₁ n₂ n₃</td>
<td>Computes a pseudoprime which survives n₁ Fermat tests, n₂ Miller Rabin tests, n₃ Solovay Strassen tests</td>
</tr>
<tr>
<td>-find_strong_pseudoprime</td>
<td>a n₁ n₂</td>
<td>Computes a pseudoprime which survives n₁ Fermat tests and n₂ Miller Rabin tests</td>
</tr>
<tr>
<td>-RSA_encrypt_text</td>
<td>d n b text</td>
<td>Using blocks of b letters at a time, encrypt “text” using RSA with exponent d modulo n</td>
</tr>
<tr>
<td>-RSA</td>
<td>d n list-of-integers</td>
<td>encrypt the given sequence of integers using RSA with exponent d modulo n</td>
</tr>
</tbody>
</table>

Table 9.3: Cryptographic Commands

9.3 Cryptography

In Table 9.3, some cryptographic commands are shown. In Table 9.3, some cryptographic commands depending on a finite field are shown. We assume that the field \( \mathbb{F}_q \) has been defined. For instance,

EC_add:

```bash
$ (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:

```bash
$ (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 i₁ i₂</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</td>
<td>Decodes the ciphertext “cipher” on the elliptic curve $y^2 \equiv x^3 + ax + b$ over $\mathbb{F}_q$ using the base point pt of order n and the round keys “keys”.</td>
</tr>
<tr>
<td>-EC_discrete_log</td>
<td>a b pt base-pt</td>
<td>Computes the elliptic curve discrete log analogue of pt with respect to base-pt on the elliptic curve $y^2 \equiv x^3 + ax + b$ over $\mathbb{F}_q$.</td>
</tr>
<tr>
<td>-NTRU_encrypt</td>
<td>N p H R M</td>
<td>NTRU encryption for the message $M(X)$ using the public key $H(X)$ and one-time-key $R(X)$.</td>
</tr>
<tr>
<td>-polynomial_center_lift</td>
<td>A(X)</td>
<td>Compute the center lift mod $q$ for the coefficients of $A$.</td>
</tr>
<tr>
<td>-polynomial_reduce_mod_p</td>
<td>p A(X)</td>
<td>Reduce the coefficients of the polynomial $A$ modulo $p$.</td>
</tr>
</tbody>
</table>

Table 9.4: Finite Field Activities related to Cryptography
computes the cyclic subgroup generated by the point \((1, 4)\) on the curve \(y^2 = x^3 + x + 3 \mod 11\). The command

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

\[
\text{EC.Koblitz_encoding:}
\]
\[
> \ $(ORBITER) -v 6 -seed 17 \ \\
> \ > \ -define F -finite_field -q 199 -end \ \\
> \ > \ -with F -do \ \\
> \ > \ -finite_field_activity \ \\
> \ > \ -EC.Koblitz_encoding 5 7 67 "147,164" "DEADBEEF" \ \\
> \ > \ -end
\]

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

\[
\text{EC.bgs_decode:}
\]
\[
> \ $(ORBITER) -v 2 \ \\
> \ > \ -define F -finite_field -q 199 -end \ \\
> \ > \ -with F -do \ \\
> \ > \ -finite_field_activity \ \\
> \ > \ -EC.bgs_decode 5 7 "129,176" 212 \ \\
> \ > \ "127,188,51,141,85,29,106,90,41,105,179,71,171,2,16,197,183,72,27,129,37,10" \ \\
> \ > \ "50,179,169,13,153,169,115,116,188,110,176" \ \\
> \ > \ -end
\]

271
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 [34]). In the example, we choose the parameters of the cryptosystem to be \((N, p, q, d) = (7, 41, 3, 2)\). Orbiter uses the following convention for polynomials over a finite field \(\mathbb{F}_q\): The coefficients of \(A(X) = a_0 + a_1X + \cdots + a_dX^d\) are listed as a sequence, starting with the constant term and ending with the leading coefficient. The cryptosystem requires coefficients \(a_i\) in the range \(-\frac{p}{2} \leq a_i \leq \frac{p}{2}\). So, in an extension to the conventions for field elements in \(\mathbb{F}_q\), Orbiter allows negative coefficients as well. The assumption is that \(q\) is prime and negative coefficients are considered modulo \(q\). In the example, Alice picks the private polynomials \(f(x) = x^6 - x^4 + x^3 + x^2 - 1\) (with \(d + 1\) coefficients equal to plus one and \(d\) coefficients equal to minus one) and \(g(x) = x^6 + x^4 - x^2 - x\) with \(d\) coefficients plus one and \(d\) coefficients minus one. We also need the polynomial \(x^N - 1\). The makefile commands

\[
\text{NTRU N}=7 \\
\text{NTRU P}=3 \\
\text{NTRU Q}=41 \\
\text{NTRU D}=2 \\
\text{ALICE PRIVATE F}="-1,0,1,1,-1,0,1" \\
\text{ALICE PRIVATE G}="0,-1,-1,0,1,0,1"
\]

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:

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

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:

```makefile
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) $(NTRUE_XN1) \
  -end
```

273
The public key is assigned to the makefile variable \texttt{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).

\texttt{BOB\_MESSAGE="1,-1,1,1,0,-1"}

\texttt{BOB\_ONE\_TIME\_KEY="-1,1,0,0,0,-1,1"}

The encryption proceeds using the \texttt{NTRU\_encrypt} command, and the result is stored in the makefile variable \texttt{BOB\_ENCRYPT}:

\begin{verbatim}
NTRU\_encrypt:
  $(ORBITER) -v 2 \
  -define F -finite_field -q $(NTRU\_Q) -end \
  -with F -do \n  -finite_field_activity \n  -NTRU\_encrypt $(NTRU\_N) $(NTRU\_P) $(ALICE\_PUBLIC\_KEY) \n  $(BOB\_ONE\_TIME\_KEY) $(BOB\_MESSAGE) -end
\end{verbatim}

\texttt{BOB\_ENCRYPT \textasciitilde "25,3,40,2,4,19,31"}

Decryption is done in five steps.

\texttt{NTRU\_decrypt1:}
\begin{verbatim}
  $(ORBITER) -v 2 \
  -define F -finite_field -q $(NTRU\_Q) -end \
  -with F -do \n  -finite_field_activity \n  -polynomial_mult_mod $(ALICE\_PRIVATE\_F) \n  $(BOB\_ENCRYPT) $(NTRUE\_XN1) \n  -end
\end{verbatim}

\texttt{ALICE\_C1="40,1,40,40,33,10,1"}

\texttt{NTRU\_decrypt2:}
\begin{verbatim}
  $(ORBITER) -v 2 \
  -define F -finite_field -q $(NTRU\_Q) -end \
\end{verbatim}

274
ALICE_C2="-1,1,-1,-1,-8,10,1"

NTRU_decrypt3:
> $(ORBITER) -v 2 \
>   -define F -finite_field -q $(NTRU_P) -end \
>   -with F -do \
>   -finite_field_activity \
>   -polynomial_reduce_mod_p $(ALICE_C2) -end

ALICE_C3="2,1,2,2,1,1,1"

NTRU_decrypt4:
> $(ORBITER) -v 2 \
>   -define F -finite_field -q $(NTRU_Q) -end \
>   -with F -do \
>   -finite_field_activity \
>   -polynomial_mult_mod $(ALICE_PRIVATE_FP) \
>   $(ALICE_C3) $(NTRUE_XN1) \n>   -end

ALICE_C4="1,2,1,1,0,2"

NTRU_decrypt5:
> $(ORBITER) -v 2 \
>   -define F -finite_field -q $(NTRU_P) -end \
>   -with F -do \
>   -finite_field_activity \
>   -polynomial_center_lift $(ALICE_C4) -end

Decryption produces Bob’s message to Alice.

ToDo:

- RSA
- \text{sqrt \ mod}
- quadratic sieve
- pseudoprimes
Chapter 10

Coding Theory

10.1 Introduction

In Table 10.1, global coding theoretic commands of Orbiter are shown. The commands

```
Hamming_space_4_2_distance_matrix:
> $(ORBITER) -Hamming_space_distance_matrix 4 2
> $(ORBITER) -v 2 -draw_matrix \
>   -input_csv_file Hamming_n4_q2.csv \
>   -box_width 20 -bit_depth 24 -partition 4 16 16 -end
> open Hamming_n4_q2_draw.bmp
```

create the csv-file `Hamming_n4_q2.csv` and produce the bitmap file `Hamming_n4_q2_draw.bmp` shown in Figure 10.1. Table 10.2 lists coding theoretic activities in Orbiter.

The following command creates the $[5,2]_2$ code whose codewords are $\{0,7,25,30\}$:

```
CODE_5_2_3_CODEWORDS="0,7,25,30"
```

code_5_2_3_diagram:
```
> $(ORBITER) -v 2 -code_diagram "code_5_2_3" \
>   $(CODE_5_2_3_CODEWORDS) 5 -metric_balls 1
> $(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
```

The Hamming graph $H(5,2)$ can be created with the following command:
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-Hamming_graph</td>
<td>$n \ q$</td>
<td>Creates the distance matrix of the Hamming graph $H(n, q)$. The vertices are the elements of $\mathbb{F}_q^n$, and the $i, j$-entry is the distance between the vectors whose affine ranks are $i$ and $j$, respectively. The matrix is written as csv-file.</td>
</tr>
<tr>
<td>-general_code_binary</td>
<td>$n \ R$</td>
<td>Creates the binary code of length $n$ containing the elements corresponding to the integers in the list $R$ under the binary representation.</td>
</tr>
<tr>
<td>-linear_code_through_basis</td>
<td>$n \ R$</td>
<td>Creates the binary linear code of length $n$ generated by the elements corresponding to the integers in the list $R$ under the binary representation.</td>
</tr>
<tr>
<td>-long_code</td>
<td>$n \ k \ r_1 \ldots \ r_k$</td>
<td>Creates the binary code of length $n$ and dimension $k$ whose generators are given as $r_1, \ldots, r_k$.</td>
</tr>
<tr>
<td>-make_macwilliams_system</td>
<td>$q \ n \ k$</td>
<td>Creates the MacWilliams system for a linear $[n, k]_q$-code.</td>
</tr>
<tr>
<td>-table_of_bounds</td>
<td>$N \ q$</td>
<td>Compute Singleton, Hamming, Plotkin, Griesmer upper bounds on $d$ for a $[n, k]_q$ code for all $n \leq N$ and all $k \leq n$. The results are written to a csv file.</td>
</tr>
</tbody>
</table>

Table 10.1: Global Coding Theoretic Commands
Figure 10.1: The color-coded distance matrix of the Hamming graph $H(4, 2)$

<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-weight_enumerator</td>
<td>$m \ n \ L$</td>
<td>Compute the complete weight enumerator of the linear code generated by the $m \times n$ matrix $L$</td>
</tr>
<tr>
<td>-field_reduction</td>
<td>$q_0 \ m \ n \ L$</td>
<td>Perform field reduction. The input is a $m \times n$ generator matrix $L$ over the field $\mathbb{F}<em>{q}$. The output is the $sm \times sn$ generator matrix of the code obtained by field reduction. The code is defined over the field of order $q_0$, which must be a subfield of $\mathbb{F}</em>{q}$, with $q_0^s = q$. A latex report is written.</td>
</tr>
</tbody>
</table>

Table 10.2: Coding Theoretic Activities
Hamming\_5\_2\_graph:
\[\text{\verb|\$(ORBITER) -v 2 \}\text{\verb| -define G -graph -Hamming 5 2 -end \}\text{\verb| -with G -do \}\text{\verb| -graph_theoretic_activity -export_csv -end \}\text{\verb| -with G -do \}\text{\verb| -graph_theoretic_activity -export_graphviz -end \}\text{\verb| -with G -do \}\text{\verb| -graph_theoretic_activity -save -end \}\text{\verb|\}$ORBITER) -v 2 -draw_matrix \}\text{\verb| -input_csv_file Hamming\_5\_2.csv \}\text{\verb| -box_width 8 -bit_depth 24 -partition 4 32 32 -end \}\text{\verb| dot -Tpng Hamming\_5\_2.gv >Hamming\_5\_2.png \}\]

Using the unix dot program, this command sequence creates the drawing of $H(5, 2)$ shown in Figure 10.2.
10.2 Hamming Codes

The Hamming code is the dual of the simplex code. The simplex code has a generator matrix whose columns are the coordinate vectors of the points of PG(2, 2). To compute the dual, we need to compute the nullspace of this matrix. The following command does that:

\[
\text{SIMPLEX\_CODE\_GENERATOR} = \\
1,0,1,0,1,0,1, \\
0,1,1,0,0,1,1, \\
0,0,0,1,1,1,1
\]

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:

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

This produces the following output:
Suppose we want to look at the codewords of the Hamming code in the Hamming space. The following command will produce Figure 10.3.

```
Hamming_code_words:
▷ $(ORBITER) -v 2 \n▷ ▷ -define v -vector -dense $(HAMMING_CODE_ROWS_IN_BINARY_RANKS) -end \n▷ ▷ -linear_code_through_basis 7 v
▷ pdflatex code_n7_k4_q2.tex
▷ open code_n7_k4_q2.pdf
```

Suppose we want to compute the weight enumerator of the Hamming code. We use the following command:

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

```
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```

282
We find that the weight enumerator is
\[(1, 0, 0, 7, 0, 0, 1)\].

Suppose we want to establish the MacWilliams relations for the Hamming code. The following command creates the matrix of Kravtchuck numbers:

```
Hamming_code_macwilliams:
▷ $(ORBITER) -v 2 \n▷ -make_macwilliams_system 7 4 2
▷ pdflatex MacWilliams_n7_k4_q2.tex
▷ open MacWilliams_n7_k4_q2.pdf
```

This produces the following output:

\[
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
7 & 5 & 3 & 1 & -1 & -3 & -5 & -7 \\
21 & 9 & 1 & -3 & -3 & 1 & 9 & 21 \\
35 & 5 & -5 & -3 & 3 & 5 & -5 & -35 \\
35 & -5 & -5 & 3 & 3 & -5 & -5 & 35 \\
21 & -9 & 1 & 3 & -3 & -1 & 9 & -21 \\
7 & -5 & 3 & -1 & -1 & 3 & -5 & 7 \\
1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\
\end{bmatrix}
\]

The Hamming code is cyclic. To see this, we need to consider the action of the Singer cycle on the set of points of $\text{PG}(2, 2)$. The following command creates the Singer cycle:

```
Hamming_singer:
▷ $(ORBITER) -v 3 \n▷ -define G -linear_group -PGL 3 2 -singer 1 -end \n▷ -with G -do \n▷ -group_theoretic_activity \n▷ -report \n▷ -orbits_on_points \n▷ -end
▷ pdflatex PGL_3_2_Singer_3_2_1_report.tex
▷ open PGL_3_2_Singer_3_2_1_report.pdf
```

283
Strong generators for a group of order 7:

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

Basic Orbit 0

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

From this, we know how to rearrange the points of PG(2, 2) to exhibit the cyclic structure. We issue the following command to recreate the Hamming code:

```
SIMPLEX_CODE_GENMA_CYCLIC="
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 \\
> -define F -finite_field -q 2 -end \\
> -define v -vector -format 3 -field F \\
> -dense $(SIMPLEX_CODE_GENMA_CYCLIC) \\
> -end \\
> -with F -do -finite_field_activity \
```

284
This produces the following output:

Input matrix:
\[
\begin{bmatrix}
1 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 1 & 0 & 1 \\
\end{bmatrix}
\]

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

Basis for Perp:
\[
\begin{bmatrix}
1 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 1 \\
\end{bmatrix}
\]
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 \n  -define v -vector -dense $(GOLAY_23_COLUMN_RANKS_PROJECTIVELY) -end \n  -linear_code_through_columns_of_parity_check_projectively 12 v \n  pdflatex code_n23_k12_q2.tex \n  open code_n23_k12_q2.pdf
```
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.

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  -finite_field_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 \n  -define F -finite_field -q 2 -end \n  -with F -do \n  -finite_field_activity \n  -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 =
1010110101101010101011100010111100 / 1101 =
1101101110001101111111010000101
==================================
1101 | 10101101011010101010111000010111100
```

287
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 \
  ▷   "Hithere" "text.csv"
  ▷ $(ORBITER) -v 2 \n  ▷   -define F -finite_field -q 2 -end \n  ▷   -with F -do \n  ▷   -finite_field_activity \n  ▷   -polynomial_division_from_file_all_k_bit_error_patterns \n  ▷   text.csv 13 1 -end
  ▷ pdflatex polynomial_division_file_all_1_error_patterns_13.tex
  ▷ open polynomial_division_file_all_1_error_patterns_13.pdf
```

The following output is created:

```
: 01010110100110101010101011000011111100
0 : 01010110100110101010101011000011111101 : 100 : 4 : X^{-2}
1 : 01010110100110101010101011000011111110 : 111 : 7 : X^{-2} + X + 1
2 : 01010110100110101010101011000011111000 : 001 : 1 : 1
3 : 01010110100110101010101011000011110100 : 000 : 0 : 0
4 : 0101011010011010101010101100001101100 : 010 : 2 : X
5 : 01010110100110101010101011000011001100 : 111 : 6 : X^{-2} + X
6 : 01010110100110101010101011000011001100 : 010 : 3 : X + 1
7 : 01010110100110101010101011000010111100 : 100 : 4 : X^{-2}
8 : 01010110100110101010101011000010111100 : 111 : 7 : X^{-2} + X + 1
9 : 01010110100110101010101011000010111100 : 001 : 1 : 1
10 : 01010110100110101010101011000010111100 : 000 : 0 : 0
11 : 01010110100110101010101011000010111100 : 010 : 2 : X
```

\[ /13 = 1841528453 \text{ Remainder } 5 \]

289
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 $n = 128$.

```
$ (ORBITER) -v 1 \
  -define F -finite_field -q 2 -end \
  -with F -do -finite_field_activity \
  -find_CRC_polynomials 3 128 10 \
  -end
```

The program finds 244 polynomials in about 1 minute.

Here is a collection of CRC polynomials from various sources:

CRC4="1, 4, 1, 2, 1, 1, 1, 0"

CRC7="1, 7, 1, 3, 1, 0"

CRC8_ATM="1, 8, 1, 2, 1, 1, 1, 0"
CRC16_CCITT="1,16,1,12,1,5,1,0"

CRC32_ETHERNET="1,32,1,26,1,23,1,22,1,16,1,12,1,11,1,10,1,8,1,7,1,5,1,4,1,2,1,1,1,0"

CRC32_CASTAGNOLI="1,32,1,28,1,27,1,26,1,25,1,23,1,22,1,20,1,19,1,18,1,14,1,13,1,11,1,10,1,9,1,8,1,6,1,0"

CRC64_ECMA182="1,64,1,62,1,57,1,55,1,54,1,53,1,52,1,47,1,46,1,45,1,40,1,39,1,38,1,37,1,35,1,33,1,32,1,31,1,29,1,27,1,24,1,23,1,22,1,21,1,19,1,17,1,13,1,12,1,10,1,9,1,7,1,4,1,1,1,0"

CRC64_ROCKSOFT="1,64,1,63,1,61,1,59,1,58,1,56,1,55,1,54,1,53,1,52,1,49,1,48,1,47,1,46,1,44,1,41,1,37,1,36,1,34,1,32,1,31,1,28,1,26,1,23,1,22,1,19,1,16,1,13,1,12,1,10,1,9,1,6,1,4,1,3,1,0"

We test whether the polynomial crc32 is irreducible:

crc32_Berlekamp_matrix:
  ▷ $(ORBITER) -v 2 \
  ▷ ▷ -define F -finite_field -q 2 -end \
  ▷ ▷ -define v -vector -field F -sparse 33 $(CRC32_ETHERNET) -end \
  ▷ ▷ -with F -do \
  ▷ ▷ -finite_field_activity \
  ▷ ▷ -Berlekamp_matrix v -end

Now, we create some new CRC polynomials over the field $\mathbb{F}_{256}$. To begin with, we create the 771st roots over $\mathbb{F}_{256}$:

CRC_F256_roots_771:
  ▷ $(ORBITER) -v 3 \
  ▷ ▷ -define F -finite_field -q 256 -end \
  ▷ ▷ -with F -do -finite_field_activity \
  ▷ ▷ ▷ -nth_roots 771 \
  ▷ ▷ -end

We create a BCH code of length 771 over $\mathbb{F}_{256}$ with designed distance 2:
The polynomial in dense coding

```
CRC_POLY_Q256_DEG2_DENSE="214,167,1"
```

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

```
CRC_F256_BCH_write_code_for_division_d2:
  $(ORBITER) -v 2 \\
  -define F -finite_field -q 256 -end \\
  -define A -vector -field F -sparse 772 "1,771,1,0" -end \\
  -define B -vector -field F -dense $(CRC_POLY_Q256_DEG2_DENSE) -end \\
  -with F -do \\
  -finite_field_activity \\
  -write_code_for_division \\
  check_q256_n771_r2.cpp A B \\
  -end \\
  g++ check_q256_n771_r2.cpp -o check_q256_n771_r2.out \\
  ./check_q256_n771_r2.out
```

We create a BCH code of length 771 over $\mathbb{F}_{256}$ with designed distance 16:

```
F256_BCH_code_d16:
  $(ORBITER) -v 3 \\
  -define F -finite_field -q 256 -end \\
  -with F -do -finite_field_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:

```
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:

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

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

```
F256_BCH_write_code_for_division_d16:
▷ $(ORBITER) -v 2 
▷ ▷ -define F -finite_field -q 256 -end 
▷ ▷ -define A -vector -field F -sparse 772 "1,771,1,0" -end 
▷ ▷ -define B -vector -field F -dense $(POLY_Q256_DEG30_DENSE) -end 
▷ ▷ -with F -do 
▷ ▷ -finite_field_activity 
▷ ▷ -write_code_for_division check_q256_n771_r30.cpp A B -end 
▷ g++ check_q256_n771_r30.cpp -o check_q256_n771_r30.out 
▷ ./check_q256_n771_r30.out
```

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

```
F256_BCH_code_d16_division:
▷ $(ORBITER) -v 2 
▷ ▷ -define F -finite_field -q 256 -end 
▷ ▷ -define A -vector -field F -sparse 772 "1,771,1,0" -end 
▷ ▷ -define B -vector -field F -dense $(POLY_Q256_DEG30_DENSE) -end 
▷ ▷ -with F -do 
▷ ▷ -finite_field_activity 
▷ ▷ -polynomial_division A B -end
```

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

```
F256_BCH_code_d16_error:
▷ $(ORBITER) -v 2 
▷ ▷ -define F -finite_field -q 256 -end 
▷ ▷ -define A -vector -field F -sparse 771 "2,30,3,31,55,770" -end 
▷ ▷ -define B -vector -field F -dense $(POLY_Q256_DEG30_DENSE) -end 
▷ ▷ -with F -do 
▷ ▷ -finite_field_activity 
▷ ▷ -polynomial_division A B -end
```

293
10.5 Reed-Muller Codes

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

\begin{verbatim}
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
\end{verbatim}

The output is shown in Figure 10.4.

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

\begin{verbatim}
RM_3_1_Hamming_space_diagram:
  > $(ORBITER) -v 2 -code_diagram "RM_3_1" \
  >   $(REED_MULLER_3_1_CODEWORDS) 8 \
  >   -metric_balls 1
  >   $(ORBITER) -v 2 -draw_matrix \
  >   -input_csv_file RM_3_1_diagram_01_8_16.csv \
\end{verbatim}
Figure 10.5: Boolean function representation of $RM_{3,1}$ in $H(8,2)$

```bash
drover -o ORBITER -v 2 -draw_matrix
  -input_csv_file RM_3_1_diagram_8_16.csv
  -box_width 25 -bit_depth 8 -partition 4 16 16 -end
open RM_3_1_diagram_8_16.draw.bmp
```

produces a representation of the code as boolean function in the Hamming space $H(8,2)$, shown in Figure 10.5. 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

```bash
draw_cyclotomic_mod_21_q8:
```

```bash
draw_cyclotomic_mod_21_q8:
  > $(ORBITER) -v 2 \\
  >   -draw_options \\
  >     -radius 100 \\
  >     -line_width 1.0 -embedded \\
  > -end \\
  >   -draw_mod_n -n 21 -file mod_21_cyclotomic \\
  >   -cyclotomic_sets 8 "1,2,4,5,7,10,13" -end \\
  > pdflatex mod_21_cyclotomic_draw.tex \\
  > open mod_21_cyclotomic_draw.pdf
```

The output is shown in Figure 10.6. We will try BCH-codes with minimum distances 3, 5 and 7. Here is distance 3:

```bash
F_8_BCH_code_d3:
```

```bash
F_8_BCH_code_d3:
  > $(ORBITER) -v 3 \\
  >   -define F -finite_field -q 8 -override_polynomial 11 -end \\
  >   -with F -do -finite_field_activity -make_BCH_code 21 3 -end \\
  > pdflatex BCH_codes_q8_n21_d3.tex \\
  > open BCH_codes_q8_n21_d3.pdf
```

The code is described in a latex output file:

```latex
BCH-code:
\begin{itemize}
  \item $n = 21$, $k = 17$, $d_0 = 3$, $q = 8$,  
  \item $g(x) = m_1 m_2 = X^4 + 4X^3 + 4X^2 + 3X + 4$
\end{itemize}
Chosen cyclotomic sets:
\begin{itemize}
  \item \{ 1, 8 \}
  \item \{ 2, 16 \}
\end{itemize}
```
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 $d = 5$:

F_8_BCH_code_d5:

\[
\text{\$ (ORBITER) \ -v 3 \ \\\n\text{\% define F \ -finite_field \ -q 8 \ -override_polynomial 11 \ -end \ \\\n\text{\% with F \ -do \ -finite_field_activity \ -make_BCH_code 21 5 \ -end \ \\\n\text{\% pdflatex BCHP_codes_q8_n21_d5.tex \ \\\n\text{\% open BCHP_codes_q8_n21_d5.pdf}
\]

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 \}
\{ 4, 11 \}

The generator polynomial has degree 7
The generator matrix is:

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

Finally, \( d = 7 \):

\[
F_8.BCH_code_d7: \quad \begin{array}{c}
\text{\textasciitilde ORBITER) -v 3} \\
\text{\textasciitilde -define F -finite_field -q 8 -override_polynomial 11 -end} \\
\text{\textasciitilde -with F -do -finite_field_activity -make_BCH_code 21 7 -end} \\
pdflatex BCH_codes_q8_n21_d7.tex \\
\text{\textasciitilde open BCH_codes_q8_n21_d7.pdf}
\end{array}
\]

The output file is:

BCH-code:

\( n = 21, \ k = 11, \ d_0 = 7, \ q = 8 \),

\( g(x) = m_1 m_2 m_3 m_4 m_5 m_6 = X^{10} + X^9 + 2X^8 + 5X^7 + 2X^6 + 4X^4 + 6X^3 + 5X^2 + 6X + 6 \)

Chosen cyclotomic sets:

\{ 1, 8 \}

\{ 2, 16 \}

\{ 3 \}
The generator polynomial has degree 10

- dense "6,6,5,6,4,0,2,5,2,1,1"
- sparse "6,0,6,1,5,2,6,3,4,4,2,6,5,7,2,8,1,9,1,10"

The generator matrix is:

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

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:

```
$\text{(ORBITER)} -v 2 \$
```

The drawing is shown in Figure 10.7.

Suppose we want to make a BCH-code over $F_{256}$. In order to keep the degree of the generator polynomial low, we try a quadratic field extension. This way, each cyclotomic set has size 300.
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:

\[ \text{BCH}\_\text{F256}\_\text{roots}\_771: \]
\[ \text{\textasciitilde ORBITER} -v 3 \]
\[ \text{\textasciitilde -define F -finite field -q 256 -end} \]
\[ \text{\textasciitilde -with F -do -finite field activity -nth roots 771 -end} \]

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

\[ \text{BCH}\_\text{F256}\_\text{BCH code d16:} \]
\[ \text{\textasciitilde ORBITER} -v 3 \]
\[ \text{\textasciitilde -define F -finite field -q 256 -end} \]
\[ \text{\textasciitilde -with F -do -finite field activity -make BCH code 771 16 -end} \]
\[ \text{pdflatex BCH_codes_q256_n771_d16.tex} \]
\[ \text{open BCH_codes_q256_n771_d16.pdf} \]
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

$$\text{CODE} \_	ext{RS} \_6 \_4 \_7=$"\$
621000 \\
062100 \$
006210 \\
000621"

$$\text{RREF} \_	ext{RS} \_6 \_4 \_7 \_\text{weight} \_\text{enumerator}:$$ 

```
$\text{ORBITER} -v 2 \\
$-define F -finite_field -q 7 -end \\
$-define v -vector -format 4 -field F \\
$-compact $(\text{CODE} \_	ext{RS} \_6 \_4 \_7) \\
$-end \\
$-with F -do \\
$-finite_field_activity \\
$-weight\_enumerator v \\
$-end
```

computes the weight enumerator, which turns out to be

$$(1, 0, 0, 120, 360, 972, 948).$$

In polynomial form, this is

$$1y^6 + 120x^3y^3 + 360x^4y^2 + 972x^5y + 948x^6.$$ 

This confirms that the minimum distance is three.
Let us consider an example of a Reed-Solomon code in characteristic two: 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

$$\begin{bmatrix}
5 & 6 & 1 & 0 & 0 & 0 \\
0 & 5 & 6 & 1 & 0 & 0 \\
0 & 0 & 5 & 6 & 1 & 0 \\
0 & 0 & 0 & 5 & 6 & 1 \\
0 & 0 & 0 & 0 & 5 & 6 \\
\end{bmatrix}.$$ 

We use the makefile variable `CODE_RS_8` to hold this generator matrix. The following command computes the weight enumerator

```bash
CODE_RS_8="\n5610000 \n0561000 \n0056100 \n0005610 \n0000561"
```

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 \
  -finite_field_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 \
```

303
The reduced matrix is shown in Figure 10.8. Let us compute the weight enumerator of the reduced code. The command

```plaintext
RS_8_reduced="\n01000110000000000000000\n00111001000000000000000\n11001100100000000000000\n00001000110000000000000\n00000111001000000000000\n00011001100100000000000\n00000001000110000000000\n00000000111001000000000\n00000000011100100000000\n00000000000000000000000\nend
```

Figure 10.8: Field reduction of a Reed-Solomon code
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 \
>   -finite_field_activity \
>   -weight Enumerator v \
>   -end

computes the weight enumerator of the binary code. It is

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

In particular, the field reduced Reed-Solomon code is not optimal. It has minimum distance three, but there are codes of minimum distance 4. Here is one. We store the code to a file and then draw the generator matrix as bitmap.

CODE_21_15_4="
111000100000000000000000 
110100010000000000000000 
101100001000000000000000 
011100000100000000000000 
110010000010000000000000 
101010000001000000000000 
011010000000100000000000 
100110000000010000000000 
010110000000001000000000 
001110000000000100000000 
111110000000000100000000 
110001000000000010000000 
1010010000000000010000000"
0110010000000000000101
1001010000000000000101

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

We compute the weight enumerator

CODE_21_15_4_weight Enumerator:
  ▶ $(ORBITER) -v 2 \n  ▶ ▶ -define F -finite_field -q 2 -end \n  ▶ ▶ -define v -vector -format 15 -field F \n  ▶ ▶ ▶ -compact $(CODE_21_15_4) \n  ▶ ▶ -end \n  ▶ ▶ -with F -do \n  ▶ ▶ -finite_field_activity \n  ▶ ▶ -weight Enumerator v \n  ▶ ▶ -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

\[
\text{bounds\ for\ d\ given\ n15\ k6\ q2:}
\]

\[
\text{\textgreater\textgreater\ $(\text{ORBITER})\ -v\ 2\ \backslash}
\]

\[
\text{\textgreater\textgreater\ \textgreater\textgreater\ -make\_bounds\ for\ d\ given\ n\ and\ k\ and\ q\ 15\ 6\ 2}
\]

gives upper and lower bounds on the optimal minimum distance \( d_{\text{max}} \) of a \([15, 6]_2\) code. The values of the Gilbert-Varshamov lower bound and the Singleton, Hamming, Plotkin and Griesmer upper bounds are computed. The output is:

\[
d_{\text{GV}} = 5
\]
\[
d_{\text{singleton}} = 10
\]
\[
d_{\text{hamming}} = 6
\]
\[
d_{\text{plotkin}} = 7
\]
\[
d_{\text{griesmer}} = 6
\]

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

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

\[
\text{\textgreater\textgreater\ $(\text{ORBITER})\ -v\ 2\ -table\ of\ bounds\ 20\ 2}
\]

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

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

\[
\text{\textgreater\textgreater\ $(\text{ORBITER})\ -v\ 2\ \backslash}
\]

\[
\text{\textgreater\textgreater\ \textgreater\textgreater\ -define\ F\ -finite\_field\ -q\ 2\ -end\ \backslash}
\]

\[
\text{\textgreater\textgreater\ \textgreater\textgreater\ -define\ P\ -projective\_space\ 8\ F\ -end\ \backslash}
\]

\[
\text{\textgreater\textgreater\ \textgreater\textgreater\ -with\ P\ -do\ \backslash}
\]

\[
\text{\textgreater\textgreater\ \textgreater\textgreater\ -projective\_space\ activity\ \backslash}
\]

\[
\text{\textgreater\textgreater\ \textgreater\textgreater\ -make\_gilbert\_varshamov\_code\ 15\ 5\ \backslash}
\]

\[
\text{\textgreater\textgreater\ \textgreater\textgreater\ -end}
\]

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:

```
CODEGVN15_K6="
111111111100000
111111000010000
111001100001000
1101010000100
10101011000010
1011010000001"
```

The weight enumerator is

\[ 1y^{15} + 27x^6y^9 + 24x^8y^7 + 9x^{10}y^5 + 3x^{12}y^3. \]

From this, we see that the code has minimum distance 6, which is better than predicted.
10.9 Classification of Optimal Linear Codes

The classification problem of optimal codes in coding theory is the problem of determining the equivalence classes of codes for a given set of values of $n$ and $k$ and $q$ with a lower bound on $d$. Orbiter can be used to classify linear codes with given redundancy and bounded minimum distance. The redundancy of a linear $[n,k]$ code is the parameter $r = n - k$. Codes with redundancy $r$ can be identified with subsets of $\text{PG}(r-1,q)$. Under this correspondence, a code with minimum distance at least $d$ corresponds to a subset such that any $d-1$ elements are independent. We use the notation $\Lambda_{r-1,s}(q)$ to denote the poset of subsets of $\text{PG}(r-1,q)$ for which any $d-1$-subset (if any) is independent. Under the correspondence, the action of $\text{PGL}(r,q)$ on $\Lambda_{r-1,s}(q)$ corresponds to the orbits of equivalent linear codes. For this reason, we are interested in determining the orbits of $\text{PGL}(r,q)$ on $\Lambda_{r-1,s}(q)$. An orbit of size $n$ represents an isometry class of $[n,n-r,d;q]$ codes with $d \geq s + 1$. The projective stabilizer of the subset is the automorphism group of the code. The Orbiter command

```
codes_8_4_4:
  $(\text{ORBITER})$ -v 6 \n  -orbit_path $(\text{ORBITER\_PATH})$ \n  -define G \n  -linear_group -PGL 4 2 -end \n  -with G -do \n  -group_theoretic_activity \n  -poset_classification_control \n    -problem_label codes_8_4_4 \n    -draw_poset \n    -draw_options -embedded -radius 250 \n    -line_width 1.0 -spanning_tree -end \n    -report -end \n  -end \n  -linear_codes 3 8 \n  -end
```

classifies linear codes with redundancy 4 and minimum distance at least 4. Orbiter confirms that there is exactly one such code, and it computes the code together with the projective stabilizer. Orbiter creates the action of the group $\text{PGL}(4,2)$ on the poset $\Lambda_{3,3}(2)$. Using poset classification, Orbiter then produces the poset of orbits shown in Figure 10.9. In this diagram, the numbers stand for Orbiter ranks of points in $\text{PG}(3,2)$. All nodes except for the root node have a number attached to it. The nodes represent subsets. In order to determine the set associated to a node, follow the path from the root node to the node and collect the points according to their labels. The root node represents the empty set. The $[8,4,4;2]$-code is represented by the set $\{0,1,2,3,8,11,13,14\}$. The fact that there is only one node at level
Figure 10.9: Orbits of PGL(4, 2) on the poset $\Lambda_{3,3}(2)$
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

\texttt{Sym\_10.conj\_classes:}
\begin{itemize}
  \item \texttt{\$(ORBITER) -v 2 -conjugacy\_classes\_Sym\_n 10}
  \item \texttt{open classes\_Sym\_10.csv}
\end{itemize}

produces a list of the conjugacy classes of \(\text{Sym}(10)\). The list is written to a csv file. On
Macintosh, the open command invokes Numbers, which can be used to create a pie chart of
the class size distribution (see Fig. 11.1).

The next command computes the character table of the symmetric group \(\text{Sym}(4)\):

\texttt{Char\_Sym\_4:}
\begin{itemize}
  \item \texttt{\$(ORBITER) -v 2 \_character\_table\_symmetric\_group 4}
\end{itemize}

The command produces the following output:

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

Table 11.1: Commands related to Combinatorics (Part 1)
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-draw_layered_graph</code></td>
<td>options</td>
<td>Draws a graph.</td>
</tr>
<tr>
<td><code>-make_elementary_symmetric_functions</code></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><code>-Dedekind_numbers</code></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><code>-rank_k_subset</code></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><code>-geometry_builder</code></td>
<td></td>
<td>See Section 11.4.</td>
</tr>
<tr>
<td><code>-character_table_symmetric_group</code></td>
<td>$n$</td>
<td>Computes the character table of $\text{Sym}(n)$ using the algorithm of Burnside.</td>
</tr>
<tr>
<td><code>-domino_portrait</code></td>
<td>$D \ s \ \text{fname}$</td>
<td>Computes a domino portrait for a graphics file in r/g/b format using double $D$ domino sets.</td>
</tr>
</tbody>
</table>

Table 11.2: Commands related to Combinatorics (Part 2)
The following command creates the elementary symmetric functions in 4 variables.

```
$ (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.
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.

domino_portrait:

```bash
$ (ORBITER) -v 3 -domino_portrait 7 4 anton_28x32 -end
```

domino_portrait_input:

```bash
$ (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 \
```

Figure 11.2: Domino Portrait
Figure 11.3: Domino portrait input file in grayscale

```
▷▷ -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 TEST_SYSTEM for the coefficient matrix and TEST_RHS for the right hand side.

```
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"
```

```
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 \n  ▶ ▶ -define A -vector -format 7 -dense $(TEST_SYSTEM) -end \n  ▶ ▶ -define D -diophant \n  ▶ ▶ ▶ -label test_system \n  ▶ ▶ ▶ -coefficient_matrix A \n  ▶ ▶ ▶ -RHS $(TEST_RHS) \n  ▶ ▶ ▶ -x_min_global 0 -x_max_global 1 \n  ▶ ▶ -end
```

319
<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 PG(2,q). Use the given set of two pencil lines. The subset picks the lines from the given pencils which are external.</td>
</tr>
<tr>
<td>-q</td>
<td>q</td>
<td>Use PG(2,q) for maximal arcs.</td>
</tr>
<tr>
<td>-override_polynomial</td>
<td>a</td>
<td>Use polynomial numerically coded as a for creating $F_q$.</td>
</tr>
</tbody>
</table>

Table 11.3: Orbiter Commands to create Diophantine systems
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>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 jth solution to the restricted system consisting of the ith 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 ith and the j-th row whose number is congruent to r mod m.</td>
</tr>
</tbody>
</table>

Table 11.4: Orbiter activities for Diophantine systems
There are two commands to solve a diophantine system: -solve_mckay and -solve_DLX. The latter is more restrictive, as it allows only 0,1 variables. Here is the McKay solver:

```
McKay_test:
  ▶ $(ORBITER) -v 4 \\
  ▶ ▶ -define A -vector -format 7 -dense $(TEST_SYSTEM) -end \\
  ▶ ▶ -define D -diophant \\
  ▶ ▶ ▶ -label test_system \\
  ▶ ▶ ▶ -coefficient_matrix A \\
  ▶ ▶ ▶ -RHS $(TEST_RHS) \\
  ▶ ▶ ▶ -x_min_global 0 -x_max_global 1 \\
  ▶ ▶ ▶ -end \\
  ▶ ▶ -with D -do \\
  ▶ ▶ ▶ -diophant_activity -solve_mckay \\
  ▶ ▶ -end
```

The solutions are written to the file DLX_test.sol. And now the dancing links solver:

```
DLX_test:
  ▶ $(ORBITER) -v 4 \\
  ▶ ▶ -define A -vector -format 7 -dense $(TEST_SYSTEM) -end \\
  ▶ ▶ -define D \\
  ▶ ▶ -diophant -label test_system \\
  ▶ ▶ ▶ -coefficient_matrix A \\
  ▶ ▶ ▶ -RHS $(TEST_RHS) \\
  ▶ ▶ ▶ -x_min_global 0 -x_max_global 1 \\
  ▶ ▶ ▶ -end \\
  ▶ ▶ -with D -do \\
  ▶ ▶ ▶ -diophant_activity -solve_DLX \\
  ▶ ▶ -end
```
11.3 Combinatorial Linear Spaces

A linear space is a pair \((S, \mathcal{L})\) where \(S\) is a set and \(\mathcal{L}\) is a set of subsets of \(S\) such that each set \(L \in \mathcal{L}\) satisfies \(|L| \geq 2\) and moreover for any two \(a, b \in S\) there is exactly one element \(L \in \mathcal{L}\) such that both \(a\) and \(b\) belong to \(L\). The usual notions of isomorphism and automorphism apply. For finite linear spaces, a first combinatorial property is the number \(a_i\) which counts the number of sets \(L \in \mathcal{L}\) of size \(i\). The vector \((a_2, \ldots, a_n)\) is the line type of \((S, \mathcal{L})\). The equation

\[
\binom{n}{2} = \sum_{j=2}^{n} a_j \binom{j}{2}
\]

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

\[
\begin{bmatrix}
15 & 10 & 6 & 3 & 1 \\
15 & 1 & 0 & 0 & 0
\end{bmatrix}
\]

The Orbiter command

```
linsp6:
  > $(ORBITER) -v 4 \\
  >   -define A -vector -format 1 -dense "15,10,6,3,1" -end \\
  >   -define D -diophant -label linsp6 \\
  >   -coefficient_matrix A \\
  >   -RHS "15,15,1" \\
  >   -x_min_global 0 \\
  >   -x_max_global 15 \\
  >   -end \\
  >   -with D -do \\
  >     -diophant_activity -solve_mckay \\
  >     -end \\
```

solves the system using McKay’s program possolve [49]. The program finds 15 solutions, written to the file `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

\[
p_j = \#j\text{-lines though } P.
\]

323
We are trying to precompute the matrix of point types
\[
(p_{ij})
\]
where \( j = 7, 5, 4 \) and \( i \) belongs to an index set of all possible point types. Fixing a point \( P \), counting points \( Q \neq P \) collinear with \( P \) yields
\[
6p_7 + 4p_5 + 3p_4 = 29, \quad p_7 \leq 1, \; p_5 \leq 27, \; p_4 \leq 24.
\]

Using the Orbiter commands

```plaintext
linsp30_pt_types:
▷ $\$(ORBITER) -v 4 \`
▷ ▷ -define A -vector -format 1 -dense "6,4,3" -end \`
▷ ▷ -define D -diophant \`
▷ ▷ ▷ -label linsp30_pt.types \`
▷ ▷ ▷ -coefficient_matrix A \`
▷ ▷ ▷ -RHS "29,29,1" -x_bounds "0,1,0,27,0,24" \`
▷ ▷ -end \`
▷ ▷ -with D -do \`
▷ ▷ ▷ -diophant_activity -solve_mckay \`
▷ ▷ -end
```

we determine the possibilities

\[
(p_7, p_5, p_4) = \begin{cases} 
1 & 5 & 1 \\
1 & 2 & 5 \\
0 & 5 & 3 \\
0 & 2 & 7 
\end{cases}
\]

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

324
linsp30.pt_distribution:
\begin{verbatim}
$\text{(ORBITER)} -v 4 \\
  -define A -vector -format 6 -dense \\
  "1,1,1,1,1,0,0,5,2,5,2,1,5,3,7,10,1,10,1,0,10,3,21" \\
  -end \\
  -define D -diophant \\
  -label linsp30.pt_distribution \\
  -coefficient_matrix A \\
  -RHS "30,30,1,7,7,1,135,135,1,96,96,1,0,351,2,0,276,2" \\
  -x_min_global 0 -x_max_global 30 \\
  -end \\
  -with D -do \\
  -diophant_activity -solve_mckay \\
  -end \\
  -with D -do \\
  -diophant_activity -draw_as_bitmap 20 8 \\
  -end
\end{verbatim}

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 $x I y$, if either $x \in y$ or $y \in x$. We require that any line has at least two points and any two points are contained in at most one line. A decomposition of a linear space is a partition $\Pi = (C_1, \ldots, C_n)$ of $V \cup \mathcal{B}$ 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

$$\iota(C_i, C_j) = \# \{ y \in C_j, x I y \}$$

does not depend on the choice of $x \in C_i$. Any linear space has a tactical decomposition, as the discrete partition (every element is in its own class) is tactical. Let $\text{Aut}(\mathcal{L})$ 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 \texttt{-geometry\_builder} option can answer these kinds of questions.

The command

\begin{verbatim}
geo_10_3:
  \$\{$\text{ORBITER}\}$ -v 2 \\
  \$\{$\text{-define Test\_lines -set -loop 4 11 1 -end \}$} \\
  \$\{$\text{-define Geo -geometry\_builder \$} \\
  \$\{$\{$\text{-V 10 -B 10 -TDO 3 -fuse 1 \}$} \\
  \$\{$\{$\text{-fname\_GEO 10_3 \$} \\
  \$\{$\text{-test Test\_lines \$}
\end{verbatim}
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 $\text{Test\_lines}$. 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 $10\_3\_\text{inc}$ 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 $10\_3\_\text{inc}$:

```
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 67 69 73 76 79 85 88 89 96 97 98
0 1 2 10 13 14 20 25 26 31 33 35 41 44 47 52 56 58 62 67 69 73 78 79 84 86 89 95 97 98
0 1 2 10 13 14 20 25 26 31 33 35 41 44 47 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 77 79 84 86 89 95 97 99
0 1 2 10 13 14 20 25 26 31 33 35 41 47 48 52 54 57 62 66 68 73 77 79 84 86 89 95 97 99
0 1 2 10 13 14 20 25 26 31 33 35 41 47 48 52 54 57 62 66 68 73 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 68 73 78 79 84 86 89 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 86 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
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 $10\_3\_\text{blocks}$:

```
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 $10\_3\_\text{blocks\_long}$ 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 $\text{fname\_GEO}$ option. Here, the $\text{fname\_GEO}$ 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:
\begin{verbatim}
geo_10_3_tree: 
  $(ORBITER) -v 20 \ 
  -define Test_lines -set -loop 0 11 1 -end \ 
  -define GEO -geometry_builder \ 
  -V 10 -B 10 -TDO 3 -fuse 1 \ 
  -fname GEO 10_3 \ 
  -search_tree \ 
  -test Test_lines \ 
  -end \ 
  $(ORBITER) -v 20 \ 
  -draw_options -embedded -radius 20 \ 
  -paperheight 220 \ 
  -paperwidth 330 \ 
  -xin 10000 -yin 10000 \ 
  -xout 1000000 -yout 500000 \ 
  -scale 2 -line_width 0.3 \ 
  -nodes_empty \ 
  -end \ 
  -tree_draw \ 
  -file 10_3_tree.txt \ 
  -end 
\end{verbatim}

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 $\text{Sym}(5)$ 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:
$\text{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 \
  -search_tree \
  -test Test_lines \
  -end
```

329
This command takes about 8 minutes of time to complete. The next command classifies the \(15_3\) with girth 4. Only one configuration arises, the Cremona Richmond, with automorphism group \(\text{Sym}(6)\) of order 720.

```plaintext
geo_15.3.g4:
  $(ORBITER) -v 2 
  -define Test_lines -set -loop 4 16 1 -end 
  -define Geo -geometry_builder 
  -V 15 -B 15 -TDO 3 
  -fuse 1 -fname_GEO 15_3.g4 
  -girth 4 
  -search_tree 
  -test Test_lines 
  -end 
  $(ORBITER) -v 2 
  -draw_options -embedded -radius 50 
  -nodes_empty 
  -scale 0.5 -line_width 0.3 -end 
  -tree_draw -file 15_3.g4.tree.txt -end 
pdflatex 15_3.g4.tree_draw.tex
  open 15_3.g4.tree_draw.pdf
```
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 \( \mathbb{P}_q^2 \). 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

\[
\text{design PG}_2_3: \\
\text{\{ORBITER\}} -v 8 \ \\
\text{\{ORBITER\}} -define D -design -q 3 -family PG_2_q -end \ \\
\text{\{ORBITER\}} -with D -do \ \\
\text{\{ORBITER\}} -design_activity \ \\
\text{\{ORBITER\}} -export_inc \ \\
\text{\{ORBITER\}} -end
\]

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:

331
Strong generators for a group of order 5616:

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

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

The blocks of the design are encoded in the lexicographic ordering of \(k\)-subsets (here \(k = 4\)).

The program also displays the tactical decomposition schemes of the design, which are

\[
\begin{array}{c|c}
\rightarrow & 13\downarrow \\
13 & 13
\end{array}
\]

In Section 15.4, we will show how to compute further properties of the design.

The command

\[
\text{wreath\_product\_designs\_n4\_k2\_inc.txt:}
\]

\[
\begin{verbatim}
$(ORBITER) -v 8 \\
-define D -design -wreath\_product\_designs 4 2 -end \\
-with D -do \\
-design\_activity \\
-export\_inc \\
-end
\end{verbatim}
\]

creates a design on 8 points invariant under the wreath product \(\text{Sym}(4) \bowtie \text{Sym}(2)\). The design has 12 blocks of size 4. The command

\[
\text{wreath\_product\_designs\_n8\_k6\_inc.txt:}
\]

\[
\begin{verbatim}
$(ORBITER) -v 8 \\
-define D -design -wreath\_product\_designs 8 6 -end \\
\end{verbatim}
\]

332
create a design on 16 points invariant under the wreath product Sym(8)≀Sym(2). The design has 3920 blocks of size 6. We will compute the automorphism groups of these two designs in Section 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. [39]). 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-Mesner matrix has 32 rows and 97 columns. The matrix is stored in the csv-file

```
KM_PGGL_2_32_KM_7_8.csv.
```

The second command produces the graphical representation of the matrix shown 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.
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  ▶ ▶ ▶ -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 17 18 24 31 32 33 37 40 43 46 49 53 56 59 62 64 69 71 74 78 80 82 87 89 93 94 99 102 103 107
-1 1
432
```

This shows that the design is unique, and has an automorphism group of order 432. For the following commands, we will treat blocks of the design as sets of ranks of \(k\)-subsets. We can now create a table of all designs AG(2, 3), as orbit under the group Sym(9). The following command does that:

```
AG_2_3_BLOCKS="0,13,22,27,35,41,47,53,55,59,71,76"
```

```
LS_AG_2_3_design_table_create:
  $(ORBITER) -v 20 \
  -define D -design -list_of_blocks \
  9 3 $(AG_2_3_BLOCKS) -end \
  -define Sym9 -permutation_group -symmetric_group 9 -end \
  -define T -design_table D "AG_2_3" Sym9
```

The number of designs is \(|\text{Sym}(9)|/432 = 840.\) To find all large sets, we establish the block-disjointness graph on this set of designs. After that, we find all cliques of size 7:
The files `AG_2_3_design_table_disjoint_sets_sol.txt` and `AG_2_3_design_table_disjoint_sets_sol.csv` are created, each containing the cliques of size 7. There are exactly 15360 cliques of size 7. It remains to classify the resulting 15360 large sets up to isomorphism. To do that, we first need to create the actual large sets from the cliques. The following command does that:

```bash
LS_AG_2_3_disjoint_sets_graph_and_cliques:
▸ $(ORBITER) -v 2 \n ▸ ▸ -define Gamma -graph \n ▸ ▸ ▸ -disjoint_sets_graph \n ▸ ▸ ▸ AG_2_3_design_table.csv \n ▸ ▸ -end \n ▸ ▸ -with Gamma -do \n ▸ ▸ ▸ -graph_theoretic_activity \n ▸ ▸ ▸ ▸ -save \n ▸ ▸ -end \n ▸ ▸ -with Gamma -do \n ▸ ▸ ▸ -graph_theoretic_activity \n ▸ ▸ ▸ ▸ -find_cliques -target_size 7 -end \n ▸ ▸ -end \n ▸ ▸ -print_symbols
```

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 [22] study line-transitive and point-imprimitive designs and show that they are rare in a certain sense. Orbiter can be used to construct such designs assuming that there is a grid structure on the set of points and assuming that the design is invariant under a chosen group $G$. The group $G$ is assumed to be a subgroup of the group $\text{AGL}(d_1, q_1) \times \text{AGL}(d_2, q_2)$ acting on a grid of size $q_1^{d_1} \times q_2^{d_2}$ in product action.

Finite projective planes often arise in this context. However, not all examples are projective planes. Orbit 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=\n  ▶ -nb_orbits_on_blocks 1 \n  ▶ -depth 5 \n  ▶ -mask_label "no_mask"
```

The command `DD_PP4` sets up the orbits of the group on pairs and writes the file `PP4_pair_covering.csv`.

```
DD_PP4:
  ▶ $(\text{ORBITER}) -v 6 \n  ▶ ▶ $\text{Delandtsheer_Doyen} \$(PP4) \$(PP4\_GROUP1) \$(PP4\_MASK1) \n  ▶ ▶ ▶ -end \n```

The command `DD_PP4_system` creates a diophantine system of Steiner type and solves it.
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} |
```

340
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-lambda3</td>
<td>( \lambda_3 ) s</td>
<td>Refine as 3-design with ( \lambda_3 ) and with block-size s</td>
</tr>
<tr>
<td>-solution</td>
<td>i fname</td>
<td>Use solutions to system ( i ) from file name.</td>
</tr>
<tr>
<td>-range</td>
<td>f l</td>
<td>Refine cases ( i ) with ( f \leq i &lt; f + l ) only.</td>
</tr>
<tr>
<td>-select</td>
<td>label</td>
<td>Select the case for refinement by label.</td>
</tr>
<tr>
<td>-o1</td>
<td>s</td>
<td>Omit ( s ) variables from the first refinement system.</td>
</tr>
<tr>
<td>-o2</td>
<td>s</td>
<td>Omit ( s ) variables from the second refinement system.</td>
</tr>
<tr>
<td>-D1_upper_bound_x0</td>
<td>b</td>
<td>Add the bound ( x_0 \leq b ) in the first refinement.</td>
</tr>
<tr>
<td>-reverse</td>
<td></td>
<td>Sort the distributions in reverse order.</td>
</tr>
<tr>
<td>-reverse_inverse</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-nopacking</td>
<td></td>
<td>Do not use packing inequalities.</td>
</tr>
<tr>
<td>-dual_is_linear_space</td>
<td></td>
<td>Assume that the dual incidence structure is a linear space also. This is valid for projective planes, for instance.</td>
</tr>
<tr>
<td>-geometric_test</td>
<td></td>
<td>Subject the distributions to the geometric test.</td>
</tr>
<tr>
<td>-once</td>
<td></td>
<td>Find at most one refinement in each case. This can be used to test which cases can be refined.</td>
</tr>
<tr>
<td>-mckay</td>
<td></td>
<td>Use McKay’s solver instead (by default, a lexicographic solver is used).</td>
</tr>
<tr>
<td>-input_file</td>
<td>fname</td>
<td>Specify the input TDO-file for refinement.</td>
</tr>
</tbody>
</table>

Table 11.5: 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
```

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} |
```

```
row_scheme at level 4 :
is 2 x 2
    | 221_{ 1} 52_{ 2}|
---------------------
      52_{ 0} | 17 0
      221_{ 3} | 13 4
```
<table>
<thead>
<tr>
<th></th>
<th>52_{ 0}</th>
<th>17</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>221_{ 3}</td>
<td>13</td>
<td>4</td>
</tr>
</tbody>
</table>

col_scheme at level 4 :
is 2 x 2

<table>
<thead>
<tr>
<th></th>
<th>221_{ 1}</th>
<th>52_{ 2}</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>52_{ 0}</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>221_{ 3}</td>
<td>13</td>
</tr>
</tbody>
</table>

extra_col_scheme at level 3 :
is 1 x 2

<table>
<thead>
<tr>
<th></th>
<th>221_{ 1}</th>
<th>52_{ 2}</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>273_{ 0}</td>
<td>17</td>
</tr>
</tbody>
</table>
Chapter 12

Finite Geometry

12.1 Spreads

A $t$-spread of $\text{PG}(n,q)$ is a set of disjoint $\text{PG}(t,q)$ that cover all of $\text{PG}(n,q)$ pointwise. $t$-spreads in $\text{PG}(n,q)$ exist if $t+1$ divides $n+1$. The reason is the existence of the Desarguesian spread (also called the regular spread). The Desarguesian spread is created from $\text{PG}(m-1,Q)$ where $Q = q^s$ for some integer $s$. The spread elements are the subspaces which arise by considering the elements of $\text{PG}(m-1,Q)$ as vector spaces over $\mathbb{F}_q$. As such, they are rank $s$ subspaces in $\text{PG}(n-1,q)$. So, with $t = s-1$, we have a $t$-spread in $\text{PG}(n-1,q)$. The following command creates the Desarguesian line-spread in $\text{PG}(3,2)$ (so $s = 2, t = s-1 = 1, m = 2, q = 2,$ and $Q = 4$):

```
$\text{desarguesian\_spread\_in\_PG\_3\_2}$:

\begin{verbatim}
▷ $(ORBITER)$ -v 3 \n▷ ▷ -define FQ -finite\_field -q 4 -end \n▷ ▷ -define Fq -finite\_field -q 2 -end \n▷ ▷ -with FQ -and Fq -do -finite\_field\_activity \n▷ ▷ ▷ -cheat\_sheet\_desarguesian\_spread 2 -end
▷ pdflatex Desarguesian_Spread_3_2.tex
▷ open Desarguesian_Spread_3_2.pdf
\end{verbatim}
```

The cheat sheet contains the following spread:

Spread element 0 is $(1,0) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Spread element 1 is $(0,1) = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Spread element 2 is $(1,1) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$
Spread element 3 is (2, 1) = \[
\begin{bmatrix}
1 & 1 & 1 & 0 \\
1 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 \\
1 & 1 & 0 & 1
\end{bmatrix}
\]

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 PG(5, 2):

desarguesian\_spread\_in\_PG\_5\_2:

\>` $(ORBITER) -v 3 \`

\>` $ -define FQ -finite\_field -q 8 \-end \`

\>` $ -define Fq -finite\_field -q 2 \-end \`

\>` $ -with FQ -and Fq -do -finite\_field\_activity \`

\>` $ -cheat\_sheet\_desarguesian\_spread 2 \-end \`

\>` pdflatex Desarguesian\_Spread\_5\_2.tex

\>` open Desarguesian\_Spread\_5\_2.pdf

Spread element 0 is (1, 0) = \[
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 & 0
\end{bmatrix}
\]

Spread element 1 is (0, 1) = \[
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 \\
1 & 1 & 1 & 1 & 1 & 0
\end{bmatrix}
\]

Spread element 2 is (1, 1) = \[
\begin{bmatrix}
1 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 & 1 \\
0 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 0 & 0
\end{bmatrix}
\]

Spread element 3 is (2, 1) = \[
\begin{bmatrix}
1 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 1 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 & 1 & 0
\end{bmatrix}
\]

Spread element 4 is (3, 1) = \[
\begin{bmatrix}
1 & 1 & 1 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 \\
1 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 \\
1 & 1 & 1 & 0 & 0 & 1
\end{bmatrix}
\]

Spread element 5 is (4, 1) = \[
\begin{bmatrix}
1 & 1 & 1 & 0 & 1 & 0 \\
1 & 1 & 1 & 0 & 0 & 1
\end{bmatrix}
\]
Apart from the second spread element, the left halves of the generator matrices of the subspaces in the Desarguesian spread are the elements of $\mathbb{F}_8$ in a matrix representation over $\mathbb{F}_2$.

Two $t$-spreads are isomorphic if there is a collineation which maps one to the other. The classification problem for $t$-spreads is the problem of determining a complete set of pairwise non-isomorphic $t$-spreads. Orbiter can be used to classify spreads for small parameters. For instance, the command

```
spreads16_4:
  ▶ $(\text{ORBITER})$ -v 6 \n  ▶ ▶ -orbrter_path $(\text{ORBITER\_PATH})$ \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
  ▶ #pdf latex spreads_4_2_poset_detailed_lvl_17.tex
  ▶ #open spreads_4_2_poset_detailed_lvl_17.pdf
```

classifies the line-spreads of PG(3, 4) under the action of PTL(4, 4). Under the André, Bruck-Bose construction [3, 16], these spreads correspond to translation planes of order 16 with kernel $\mathbb{F}_4$. Up to isomorphism, there are exactly three line-spreads in PG(3, 4). They are the
dearguesian spread, the Hall spread, and the semifield spread. Here is the relevant output taken from the latex report:

There are 3 orbits at level 17.

**Orbit 0 / 3 at Level 17**

Node number: 1126

\{0, 25, 50, 75, 90, 107, 122, 140, 144, 157, 179, 204, 213, 238, 268, 334, 345\}_{1200}

Strong generators for a group of order 1200:

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

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

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

348
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},
\begin{bmatrix}
\omega^2 & 0 & 0 & 0 \\
0 & \omega^2 & 0 & 0 \\
\omega & 0 & \omega^2 & 0 \\
0 & 1 & 0 & 1
\end{bmatrix},
\begin{bmatrix}
\omega^2 & 0 & 0 & 0 \\
0 & \omega^2 & 0 & 0 \\
\omega & 0 & 1 & 1 \\
\omega & \omega & \omega & \omega
\end{bmatrix},
\begin{bmatrix}
1 & 0 & \omega^2 & 1 \\
1 & 0 & \omega^2 & 1 \\
0 & 1 & \omega & \omega \\
0 & 0 & 0 & 1
\end{bmatrix},
\begin{bmatrix}
\omega^2 & \omega^2 & 0 \\
\omega^2 & \omega^2 & 0 \\
0 & 0 & 0 & 1 \\
\omega & 1 & \omega & 1
\end{bmatrix},
\begin{bmatrix}
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 \\
1 & 0 & 1 & \omega^2 \\
1 & 0 & 1 & \omega^2
\end{bmatrix},
\begin{bmatrix}
1,0,0,0,0,2,0,0,0,0,0,3,1, \\
1,0,0,0,0,1,0,0,0,3,2,1,0,0,2,0, \\
1,0,0,3,1,0,0,0,0,0,1,0,0,3,2,1, \\
1,0,0,3,3,1,2,1,0,0,2,0,0,0,1,2,1, \\
o,1,1,0,2,0,1,1,0,0,2,1,0,0,0,2,0, \\
o,0,0,1,0,0,2,1,0,1,2,3,2,1,1,1,0,
\end{bmatrix},
\begin{bmatrix}
1,0,0,0,0,1,0,0,0,0,3,0,0,0,0,3,0, \\
1,0,0,0,0,1,0,0,0,0,2,3,0,0,1,1,0, \\
1,0,0,0,0,1,0,0,0,2,1,3,1,2,3,2,2,0, \\
1,0,0,0,3,1,0,0,0,0,0,1,0,0,3,2,1, \\
1,0,0,3,3,1,2,1,0,0,2,0,0,0,1,2,1, \\
0,1,1,0,2,0,1,1,0,0,2,1,0,0,0,2,0, \\
0,0,0,1,0,0,2,1,0,1,2,3,2,1,1,1,0,
\end{bmatrix},
\begin{bmatrix}
0 & 0 & 0 & 1 \\
0 & 0 & \omega & 1 \\
0 & 1 & \omega & \omega^2 \\
\omega & 1 & 1 & 1
\end{bmatrix}.

There are 0 extensions
Number of generators 7
The three spreads in PG(3, 4) can be distinguished by their stabilizer orders. Table 12.1 lists the line spreads in PG(3, 4) according to their orbiter catalogue number (OCN). Table 12.2 lists the solid spreads in PG(7, 2) according to their orbiter catalogue number (OCN).

### Table 12.1: Spreads in PG(3, 4) in the Orbiter Catalogue

| OCN | |Aut| | Name         |
|-----|-----|-----|----------------|
| 0   | 1200|     | Hall spread    |
| 1   | 81600|    | Desarguesian spread |
| 2   | 576 |     | Semifield spread |

### Table 12.2: 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|   |               |
12.2 Translation Planes

Via the André, Bruck, Bose construction (cf. [3, 16]), spreads give rise to translation planes. The orbiter command

```
-Andre_Bruck_Bose_construction
```

constructs a projective plane from a spread. We rely on the catalogue of spreads contained in the knowledge base of Orbiter.

For instance, the command

```
TP_16.4:
▷ $(ORBITER) -v 3 \n▷ ▷ -define F -finite_field -q 4 -end \n▷ ▷ -define PGGL4 -linear_group -PGGL 4 F -end \n▷ ▷ -define PGGL5 -linear_group -PGGL 5 F -end \n▷ ▷ -with PGGL4 -and PGGL5 -do \n▷ ▷ -group_theoretic_activity \n▷ ▷ ▷ -Andre_Bruck_Bose_construction 0 "TP16-4-HALL" \n▷ ▷ -end
▷ $(ORBITER) -v 2 -draw_matrix \n▷ ▷ -input_csv_file TP16-4-HALL_incma.csv \n▷ ▷ -box_width 6 -bit_depth 8 \n▷ ▷ -partition 6 273 273 \n▷ ▷ -end
▷ open TP16-4-HALL_incma_draw.bmp
▷ pdflatex TP16-4-HALL_report.tex
▷ open TP16-4-HALL_report.pdf
```

creates the Hall plane of order 16. Remember from Table 12.1 that the Hall spread has Orbiter Catalogue Number 0. The report lists the spread first, then the automorphism group of the plane and then the tactical decomposition of the incidence matrix:

The spread:
subspace 0 / 17 is 0:

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

subspace 1 / 17 is 356:

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

subspace 2 / 17 is 25:
<table>
<thead>
<tr>
<th>Subspace</th>
<th>Value</th>
<th>Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 / 17</td>
<td>50</td>
<td>$\begin{bmatrix} 1 &amp; 0 &amp; 1 &amp; 0 \ 0 &amp; 1 &amp; 0 &amp; 1 \end{bmatrix}$</td>
</tr>
<tr>
<td>4 / 17</td>
<td>75</td>
<td>$\begin{bmatrix} 1 &amp; 0 &amp; \omega &amp; 0 \ 0 &amp; 1 &amp; 0 &amp; \omega \end{bmatrix}$</td>
</tr>
<tr>
<td>5 / 17</td>
<td>97</td>
<td>$\begin{bmatrix} 1 &amp; 0 &amp; \omega^2 &amp; 0 \ 0 &amp; 1 &amp; 0 &amp; \omega^2 \end{bmatrix}$</td>
</tr>
<tr>
<td>6 / 17</td>
<td>114</td>
<td>$\begin{bmatrix} 1 &amp; 0 &amp; 0 &amp; 1 \ 0 &amp; 1 &amp; 1 &amp; \omega^2 \end{bmatrix}$</td>
</tr>
<tr>
<td>7 / 17</td>
<td>127</td>
<td>$\begin{bmatrix} 1 &amp; 0 &amp; \omega &amp; 1 \ 0 &amp; 1 &amp; 1 &amp; 0 \end{bmatrix}$</td>
</tr>
<tr>
<td>8 / 17</td>
<td>153</td>
<td>$\begin{bmatrix} 1 &amp; 0 &amp; \omega^2 &amp; 1 \ 0 &amp; 1 &amp; \omega &amp; 1 \end{bmatrix}$</td>
</tr>
<tr>
<td>9 / 17</td>
<td>179</td>
<td>$\begin{bmatrix} 1 &amp; 0 &amp; 0 &amp; \omega \ 0 &amp; 1 &amp; \omega^2 &amp; \omega \end{bmatrix}$</td>
</tr>
<tr>
<td>10 / 17</td>
<td>191</td>
<td>$\begin{bmatrix} 1 &amp; 0 &amp; 1 &amp; \omega \ 0 &amp; 1 &amp; \omega &amp; 0 \end{bmatrix}$</td>
</tr>
</tbody>
</table>
subspace 11 / 17 is 224:
\[
\begin{bmatrix}
1 & 0 & \omega & \omega \\
0 & 1 & \omega & \omega^2
\end{bmatrix}
\]

subspace 12 / 17 is 236:
\[
\begin{bmatrix}
1 & 0 & \omega^2 & \omega \\
0 & 1 & 1 & 1
\end{bmatrix}
\]

subspace 13 / 17 is 262:
\[
\begin{bmatrix}
1 & 0 & 0 & \omega^2 \\
0 & 1 & \omega & \omega
\end{bmatrix}
\]

subspace 14 / 17 is 288:
\[
\begin{bmatrix}
1 & 0 & 1 & \omega^2 \\
0 & 1 & \omega^2 & \omega
\end{bmatrix}
\]

subspace 15 / 17 is 297:
\[
\begin{bmatrix}
1 & 0 & \omega & \omega^2 \\
0 & 1 & \omega^2 & 0
\end{bmatrix}
\]

subspace 16 / 17 is 322:
\[
\begin{bmatrix}
1 & 0 & \omega^2 & \omega^2 \\
0 & 1 & \omega^2 & 1
\end{bmatrix}
\]

Automorphism group:
Strong generators for a group of order 921600:
\[
\begin{bmatrix}
\omega & 0 & 0 & 0 & 0 \\
0 & \omega & 0 & 0 & 0 \\
0 & 0 & \omega & 0 & 0 \\
0 & 0 & 0 & \omega & 0 \\
0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]
\[
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
\omega & \omega & \omega & 1 & 0
\end{bmatrix}
\]
\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 \\
\omega & \omega & \omega & \omega
\end{bmatrix}
\]
\[
\begin{bmatrix}
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

353
\[
\begin{bmatrix}
\omega^2 & 0 & 0 & 0 \\
\omega^2 & 0 & 0 & 0 \\
0 & \omega^2 & \omega & 0 \\
\omega^2 & 0 & 0 & \omega^2
\end{bmatrix}
\begin{bmatrix}
1 & 0 & 0 & 0 \\
\omega & \omega & 0 & 0 \\
1 & 0 & \omega^2 & 0 \\
\omega & 1 & 1 & \omega^2
\end{bmatrix}
\begin{bmatrix}
1 & \omega^2 & \omega & 1 \\
0 & \omega^2 & 0 & 1 \\
\omega & 1 & \omega^2 & 0 \\
0 & 1 & 0 & \omega^2
\end{bmatrix}
\begin{bmatrix}
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

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

Tactical decomposition schemes:

\[
\begin{array}{c|ccc}
\rightarrow & 80 & 192 & 14 \\
256_0 & 5 & 12 & 0 \\
5_3 & 16 & 0 & 1 \\
12_2 & 0 & 16 & 1 \\
\end{array}
\quad\quad\quad
\downarrow & 80 & 192 & 14 \\
256_0 & 16 & 16 & 0 \\
5_3 & 1 & 0 & 5 \\
12_2 & 0 & 1 & 12 \\
\end{array}
\]
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 12.3 list Orbiter commands related to the construction of packings with assumed symmetry.

Table 12.4 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 [56] and later placed into an infinite family by Penttila and Williams [53]. 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:

```
spread_table_PG_3_5_regular:
▷ - mkdir SPREAD_TABLES_5_REG
▷ $(ORBITER) -v 6 \\
▷ ▷ -define F -finite_field -q 5 -end \\
▷ ▷ -define P -projective_space 3 F -end \\
▷ ▷ -define T -spread_table P 2 "12" "SPREAD_TABLES_5_REG/" \\
▷ ▷ -print_symbols
```

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

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

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 12.4.</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 12.3: 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 12.4: Orbiter commands related to the construction of packings with assumed symmetry related to picking long orbits
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 \
  "1,0,0,0,0,4,0,0,0,4,0,0,0,4, \ 
  1,0,0,0,3,0,0,0,0,3,0,0,0,3, \ 
  1,0,0,0,0,4,0,0,0,2,1,0,3,2,4, \ 
  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/" \
  -define PW -packing_with_symmetry_assumption T \
  -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
```
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.

```
PG_3.5_assume_31_fpc0_lo_cliques:
  $(ORBITER) -v 2 \n  ▶ -define G -graph -load H31_fpc0_lo.graph -end \n  ▶ -with G -do \n  ▶ ▶ -graph_theoretic_activity \n  ▶ ▶ -find_cliques -target_size 1 -end -end \n  ▶ ▶ -print_symbols
```

There are exactly 8 cliques of size 1. The next command builds the packings arising from these 8 cliques:

```
PG_3.5_assume_31_read_again:
  $(ORBITER) -v 5 \n  ▶ ▶ -define F -finite_field -q 5 -end \n  ▶ ▶ -define P -projective_space 3 F -end \n  ▶ ▶ -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" $(PGL_4.5_SUBGROUP_31_ME) -end \n  ▶ ▶ ▶ -end \n  ▶ ▶ -define PWF -packing_choose_fixed_points PW 0 -end \n  ▶ ▶ -define L -packing_long_orbits PWF \n  ▶ ▶ ▶ -orbit_length 31 -clique_size 1 \n  ▶ ▶ ▶ -read_solutions \n  ▶ ▶ -end \n```
<table>
<thead>
<tr>
<th>Modifier</th>
<th>Arguments</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-catalogue</td>
<td>$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 12.6 for possibilities for $F$.</td>
</tr>
</tbody>
</table>

Table 12.5: Commands for creating BLT-sets

### 12.4 BLT-Sets

A BLT-set of $Q(4, q)$ is a set of $q + 1$ point on the quadric such that no point on the quadric is collinear to more than two points of the set. BLT sets are related to spreads of $PG(3, q)$, to flocks of the quadratic cone in $PG(3, q)$, and to many other objects in combinatorics and finite geometry. They exist whenever $q$ is odd. BLT-sets have been defined in [4]. It is an interesting problem to classify BLT-sets of $Q(4, q)$ under the orthogonal group. Some references are Law [41], Penttila-Royle [52], Penttila-Law [42, 43], Betten [9], 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 12.5 shows options to create known BLT-sets. Table 12.6 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 0 -orthogonal_space 0 5 F -end \n  ▶ ▶ -with 0 -do -orthogonal_space_activity \n  ▶ ▶ ▶ -create_BLT_set -catalogue 0 -end \n  ▶ ▶ -end
  ▶ #pdf1latex 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:
  ▶ $(ORBITER) -v 2 \n  ▶ ▶ -define F -finite_field -q 11 -end \n  ▶ ▶ -define 0 -orthogonal_space 0 5 F -end \n  ▶ ▶ -with 0 -do -orthogonal_space_activity \n```
<table>
<thead>
<tr>
<th>$F$</th>
<th>Condition</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td></td>
<td>Linear BLT-set.</td>
</tr>
<tr>
<td>Fisher</td>
<td></td>
<td>Fisher BLT-set [26].</td>
</tr>
<tr>
<td>Mondello</td>
<td>$q \equiv \pm 1 \mod 10$</td>
<td>Mondello BLT-set due to Penttila [51].</td>
</tr>
<tr>
<td>FTWKB</td>
<td>$q \equiv \pm 2 \mod 3$</td>
<td>Fisher, Thas, Walker [65], Kantor, Betten [13] BLT-set.</td>
</tr>
<tr>
<td>Kantor1</td>
<td>$q = p^e, e &gt; 1$</td>
<td>Kantor’s first family.</td>
</tr>
<tr>
<td>Kantor2</td>
<td>$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 [43].</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 [43].</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 [43].</td>
</tr>
<tr>
<td>LP_71</td>
<td>$q = 71$</td>
<td>BLT-set for $q = 71$ due to Law and Penttila [43].</td>
</tr>
</tbody>
</table>

Table 12.6: Families of BLT-sets

```
▷▷▷ -create_BLT_set -family "Mondello" -end \n▷▷▷ -end
▷ pdflatex BLT_Mondello_q11.tex
▷ open BLT_Mondello_q11.pdf
```

creates the Mondello BLT-set in $Q(4, 11)$. Orbiter creates the following report:

```
The quadratic form is: 
\[ X_0^2 + X_1X_2 + X_3X_4 = 0 \]

The BLT-set is:
```
The classification of BLT-sets proceeds via the poset of partial BLT-sets. For more details, see [1, 9, 41]. The following command classifies the BLT-sets in $Q(4, 13)$:

\begin{verbatim}
BLT_13_deep_14:
  $(ORBITER) -v 2 \
  -define F -finite_field -q 13 -end \
  -define O -orthogonal_space 0 5 F -end \
  -with O -do -orthogonal_space_activity \
  -BLT_set_starter 14 \
\end{verbatim}
▷ ▷ ▷ -problem_label BLT_q13 -W -depth 14 -end \n▷ ▷ -end
Chapter 13
Graph Theory

13.1 Creating Graphs

Table 13.1 shows some Orbiter commands to create graphs.

For instance, the command

```bash
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:

```bash
make_triangle_graph:
  echo $(TRIANGLE_GRAPH) >triangle_graph.csv 
  $(ORBITER) -v 6 \
  -define G -graph \
  -load_csv_no_border \
  triangle_graph.csv \
```

365
<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 [45]</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 [44]</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>-triheiral_pair_</td>
<td></td>
<td>Trihedral pair disjointness graph</td>
</tr>
<tr>
<td>disjointness_graph</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-non_attacking_</td>
<td>n</td>
<td>Create the graph for non-attacking queens on a $n \times n$ chess board.</td>
</tr>
<tr>
<td>queens_graph</td>
<td></td>
<td></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

366
This will create the three-cycle graph.

The command

```
Chain_232:
▷ $(ORBITER) -v 2 \n▷ ▷ -define P1 -vector -dense 2,3,2 -end \n▷ ▷ -define P2 -vector -dense 2,3,2 -end \n▷ ▷ -define Gamma -graph \n▷ ▷ ▷ -chain_graph P1 P2 \n▷ ▷ -end
```

creates the chain graph with respect to the partitions (2, 3, 2) and (2, 3, 2).

The command

```
Paley_13_graph:
▷ $(ORBITER) -v 2 \n▷ ▷ -define Gamma -graph -Paley 13 -end \n```

creates the Paley graph on 13 vertices.

The command

```
trihedral_pair_graph:
▷ $(ORBITER) -v 2 \n▷ ▷ -define Gamma \n▷ ▷ ▷ -graph -trihedral_pair_disjointness_graph \n▷ ▷ -end
```

creates the graph of trihedral pairs. Two vertices are adjacent if the associated trihedral pairs are line-disjoint.

The command

```
small_graph:
▷ $(ORBITER) -v 2 \n▷ ▷ -define G -graph -edges_as_pairs 5 "0,1,0,2,0,3,0,4,1,3,1,4,2,4" -end
```
creates a graph by listing the edges in pairs. In this case, the graph

![Graph Image]

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:
<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 \ 
▷ ▷ -define G -graph \ 
▷ ▷ ▷ -load_csv_no_border \ 
▷ ▷ ▷ halljanko315.csv \ 
▷ ▷ -end \ 

In Section 15.7, we will compute the automorphism group of the graph (of order 1209600). This will create a file halljanko315_gens.csv which we use here to create an orbital graph. An orbital graph is a graph whose adjacency matrix corresponds to an orbit of a permutation group in the action on pairs. The group is the automorphism group of the graph. The following command creates the third orbital graph:

HJ315_orbital_graph_3:
▷ $(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 \ 

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 \ 

369
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_\mathbb{Z}_{11}_{1\text{mod}3}:

```plaintext
$\text{ORBITER}) -v 2 \$
$\text{-define F -finite_field -q 11 -end \$
$\text{-define S -vector -dense \$
$\text{"1,1, 1,4, 1,7, 1,10" -end \$
$\text{-define G -linear_group -AGL 1 F \$
$\text{-define subgroup_by_generators "Z11" 11 1 "1,1" \$
$\text{-end \$
$\text{-define Gamma -graph \$
$\text{-Cayley_graph G S \$
$\text{-end}
```

The vertices of the Cayley graph are ordered. The ordering is determined by the stabilizer chain. This is a total ordering.

In the following example, we create a Cayley graph based on the symmetric group on 4 things. We take the Coxeter generators as connection set:

Cayley_\text{Sym4}_\text{coxeter}:

```plaintext
$\text{ORBITER}) -v 2 \$
$\text{-define S -vector -dense "1,0,2,3, 0,2,1,3, 0,1,3,2" -end \$
$\text{-define G -permutation_group -symmetric_group 4 \$
$\text{-end \$
$\text{-define Gamma -graph \$
$\text{-Cayley_graph G S \$
$\text{-end}
```

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
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 \\
  ▶  ▶  ▶ -load_csv_no_border \\
  ▶  ▶  ▶ triangle_graph.csv \\
  ▶  ▶ -end \\
  ▶  ▶ -with G -do \\
  ▶  ▶  ▶ -graph_theoretic_activity -properties \\
  ▶  ▶ -end
```

Table 13.3: Graph Theoretic Activities

<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 [48].</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>
computes the degree type of the graph. This is the distribution of degrees in the degree sequence of the graph. It prints the distribution of degree values in exponential notation. The multiplicities are indicated as exponent. For instance, the graph in this example has three vertices of degree 2, so the degree sequence is printed as $2^3$.

We can export the adjacency matrix and create a bitmap drawing like so:

```bash
Cycle_13.draw:
  ▶ $(ORBITER) -v 2 \n  ▶ ▶ -define Gamma -graph -cycle 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 Cycle_13.csv \n  ▶ -box_width 20 -bit_depth 8 -partition 4 13 13 -end
  ▶ dot -Tpng Cycle_13.gv >Cycle_13.png
  ▶ #twopi -Tpng Cycle_13.gv >Cycle_13.png
  ▶ #open Cycle_13_draw.bmp
  ▶ #pdflatex Cycle_13_report.tex
  ▶ #open Cycle_13_report.pdf
```

The command first creates the cycle graph of order 13, and then exports the adjacency matrix as csv file. It then draws the adjacency matrix as a bitmap graphics.

Suppose we want to compute the eigenvalues of the adjacency matrix of a graph. In the following example, the command `-eigenvalues` is used to compute the eigenvalues (both regular and Laplace) of the 9-cycle:

```bash
Cycle_9_eigenvalues:
  ▶ $(ORBITER) -v 2 \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
```

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: $\theta_i$

The command

```bash
Paley_13_draw:
$$(\text{ORBITER}) \ -v \ 2 \ \$
-define Gamma -graph -Paley 13 -end 
-with Gamma -do 
-graph_theoretic_activity -export_csv -end 
-with Gamma -do 
-graph_theoretic_activity -export_graphviz -end 
$(\text{ORBITER}) \ -v \ 2 \ -draw_matrix \$
-input_csv_file Paley_13.csv 
-box_width 20 -bit_depth 8 -partition 4 13 13 -end 
dot -Tpng Paley_13.gv >Paley_13.png
open Paley_13_draw.bmp
```
draws the Paley graph of order 13 created in Section 13.1 using the external tool graphviz.

Let us consider the Cayley graphs from Section 13.1. Here is a command that draws the first graph:

```bash
Cayley_Z11_1mod3_draw:
$$(\text{ORBITER}) \ -v \ 2 \ \$
-draw_options -xin 2000000 
-yin 2000000 -embedded -radius 20000 -end 
-define F -finite_field -q 11 -end 
```

The drawing is shown in Figure 13.1. Let us draw the Cayley graph of $\text{Sym}(5)$ with respect to the Coxter generators:

\begin{verbatim}
Cayley_Sym5_coxeter_draw:
  \$\text{ORBITER} -v 2 \n  \$ -draw_options -xin 1000000 -yin 1000000 \n  \$ -embedded -radius 10000 -nodes_empty -end \n  \$ -define S -vector -dense \n  \$ -define, G -permutation_group -symmetric_group 5 \n  \$ -end \n  \$ -define Gamma -graph \n  \$ -Cayley_graph G S \n\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:

```bash
PGO_5_2_collinearity_graph: 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▷ ▷ ▷ -properties \n▷ ▷ ▷ -end
```

376
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 -tournament)</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:

```bash
graph_classify_5:
  > $(ORBITER) -v 2 \n  >   -oribter_path $(ORBITER_PATH) \n  >   -define GC -graph_classification \n  >   > -n 5 \n  >   > -poset_classification_control \n  >   >   > -problem_label graphs_v5 \n  >   >   > -depth 10 -draw_poset \n  >   >   > -draw_options -radius 250 \n  >   >   > -embedded -end \n  >   >   > -report -end \n  >   >   > -end \n  >   >   > -end \n  >   >   > -with GC -do \n  >   >   > -graph_classification_activity \n  >   >   >   > -list_graphs_at_level 5 5 \n  >   >   >   > -end \n  >   >   > -with GC -do \n  >   >   > -graph_classification_activity \n  >   >   >   > -draw_options \n  >   >   >   >   > -radius 300 -nodes_empty \n  >   >   >   > -line_width 1.5 \n  >   >   >   > -scale 0.1 \n  >   >   >   > -end \n  >   >   > -draw_graphs_at_level 5 \n  >   >   > -end \n  >   >   > -print_symbols
  > pdflatex graphs_v5_level_5_reps.tex
  > open graphs_v5_level_5_reps.pdf
  > pdflatex graphs_v5_poset.tex
  > open graphs_v5_poset.pdf
```

378
After the classification, the graphs with 5 edges are shown. The file contains the following graph drawings:

![Graph Drawings](image)

The next command classifies all tournaments on 4 vertices:

```
tournament_classify_4:
  $(ORBITER) -v 2 \n  -define GC -graph_classification \n  -n 4 -tournament \n  -poset_classification_control \n  -problem_label tournament_4 -depth 6 -draw_poset \n  -draw_options -radius 250 -embedded -end \n  -end \n  -end \n  -with GC -do \n  -graph_classification_activity \n  -draw_options -radius 400 \n  -line_width 2 -scale 0.10 -end \n  -draw_graphs_at_level 6 \n  -end \n  -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>
<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>

Table 13.5: Clique Finding Options

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 \n  > > -define G -graph -load graph_v5_e7.colored_graph -end \n  > > -with G -do \n  > > -graph_theoretic_activity \n  > > > -find_cliques -target_size 3 \n  > > -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 \n  > > -define Gamma -graph -Paley 13 -end \n```

381
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 
```

382
finds all cliques of size 3. There are exactly 26 of them. Because of the previous command, we know that they are all equivalent under the automorphism group of the graph.

Let us consider the orbital graph created in Section 13.1. We wish to study the 5-cliques. We first determine the number of 5-cliques, and then the number of orbits of 5-cliques under the automorphism group. The following command computes all 5-cliques:

```
HJ64_cliques5:
  $(ORBITER) -v 6 \
  -define Gamma -graph \
  -load \n  -with Gamma -do \
  -graph_theoretic_activity \
  -find_cliques -target_size 5 -end \n  -end
```

It turns out that there are exactly 1008 5-cliques. Concerning the classification with respect to the automorphism group of the graph, we apply the following command:

```
HJ64_cliques5_classify:
  $(ORBITER) -v 6 \
  -define Gamma -graph \
  -load \n  -with Gamma -do \
  -graph_theoretic_activity \
  -find_cliques -target_size 5 -end \n  -end
```
This command shows that all of the 1008 5-cliques lie in one orbit under the group. The orbit representative picked by Orbiter is \{0, 8, 31, 110, 283\}. These numbers refer to the vertices of the graph. They are zero-based. The stabilizer of the clique has order 1200.

Let us look at the collinearity graph of $Q(4, 2)$ one more time. The following command computes the cliques of size 3:

```
PGO 5 2 cliques: 0 5 2 incidence_matrix.csv

$ (ORBITER) -v 3 \
-define Inc -vector -file 0 5 2 incidence_matrix.csv -end \
-define Gamma -graph -collinearity_graph Inc -end \
-with Gamma -do \
-graph_theoretic_activity \
-find cliques -target_size 3 -end \
-end
```

There are 15 cliques of size 3. They correspond to the lines in the geometry.
Chapter 14

Combinatorial Objects

14.1 Combinatorial Objects

Combinatorial objects are objects that are defined by means of a finite group action. The isomorphism problem for combinatorial objects is the question to decide whether two objects of the same type belong to the same orbit under the relevant group action. Orbiter offers a unified treatment of such questions for various types of objects. The main tool is the computation of a canonical form, as well as the automorphism group.

Combinatorial objects are coded as sequences of integers. Each type of object has its own coding. Coding of objects as integer sequences allows easy handling of objects. For instance, objects can be specified in a command line argument, or they can be stored in a file. Large numbers of objects can be stored in files.

In order to apply Orbiter commands, an input stream is defined. An input stream is a sequence of objects, all of the same kind. The objects can be defined using any of the commands listed in Table 14.1. The file types will be discussed in more detail in the next section. Here are some examples. First, we create the Hirschfeld surface. Since the Hirschfeld surface is a cubic surface, the object is defined using point ranks in the relevant projective space as described in Section 4.2. For instance, the Hirschfeld surface in PG(3,4) is defined as 45 points, coded as 45 integers which are point ranks. A makefile variable is employed to define the set. The makefile variable is then used to define a set-object:

```
HIRSCHFELD_SURFACE_Q4_SET_OF_POINTS="0,1,2,3,4,5,6,7,8,9,\n10,11,12,13,14,23,26,27,30,31,34,35,38,39,42,47,48,51,52,\n53,54,59,60,61,62,67,68,69,70,75,76,79,80,81,82"
```

Hirschfeld q4 from set:

```
> $(ORBITER) -v 4 \
>   -define H -set -here \
>   $(HIRSCHFELD_SURFACE_Q4_SET_OF_POINTS) \
>   -end \
>   -define C -combinatorial_objects \
```
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-set_of_points</td>
<td></td>
<td>A set consisting of points.</td>
</tr>
<tr>
<td>-set_of_lines</td>
<td></td>
<td>A set consisting of lines.</td>
</tr>
<tr>
<td>-set_of_points_and_lines</td>
<td></td>
<td>A set consisting of points and a second set consisting of lines.</td>
</tr>
<tr>
<td>-set_of_packing</td>
<td></td>
<td>A set of packings.</td>
</tr>
<tr>
<td>-file_of_points</td>
<td></td>
<td>A set consisting of points read from file.</td>
</tr>
<tr>
<td>-file_of_lines</td>
<td></td>
<td>A set consisting of lines read from file.</td>
</tr>
<tr>
<td>-file_of_packings</td>
<td></td>
<td>A set consisting of packings read from file.</td>
</tr>
<tr>
<td>-file_of_packings_through_spread_table</td>
<td></td>
<td>A file of packings.</td>
</tr>
<tr>
<td>-file_of_point_set</td>
<td></td>
<td>A file containing point sets.</td>
</tr>
<tr>
<td>-file_of_designs</td>
<td></td>
<td>A file containing designs or large sets.</td>
</tr>
<tr>
<td>-file_of_incidence_geometries</td>
<td>$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 $\text{PG}(2, 16)$. The hyperovals are stored in makefile variables:

\[
\text{HYPEROVAL}_{16.144} = \{0, 1, 2, 3, 52, 67, 89, 106, 126, 141, 159, 176, 184, 199, 220, 235, 245, 262\} \\
\text{HYPEROVAL}_{16.16320} = \{0, 1, 2, 3, 52, 70, 83, 109, 127, 139, 156, 174, 186, 199, 217, 229, 256, 264\}
\]

Hyperoval 16 create:
\[
\$(\text{ORBITER}) -v 2 \lli
\lli -define C -combinatorial_objects \\
\lli -set_of_points $\text{HYPEROVAL}_{16.16320} \\
\lli -set_of_points $\text{HYPEROVAL}_{16.144} \\
\lli -end \\
\]

In the next example, we read the points of an elliptic curve from file (see Section 4.2):

EC read: elliptic_curve_b1_c3_q11.txt
\[
\$(\text{ORBITER}) -v 4 \lli
\lli -define C -combinatorial_objects \\
\lli -file_of_points elliptic_curve_b1_c3_q11.txt \\
\lli -end \\
\]

In the next example, we read a packing, using a previously defined table of spreads, stored in a csv file.

PG 3 5 assume 31 read again:
\[
\$(\text{ORBITER}) -v 5 \lli
\lli -define F -finite_field -q 5 -end \\
\lli -define P -projective_space 3 F -end \\
\lli -define T -spread_table P 2 "12" "SPREAD_TABLESANGO" \\
\lli -define PW -packing_with_symmetry_assumption T \\
\lli -define PWF -packing_choose_fixed_points PW 0 -end \\
\lli -define L -packing_long_orbits PWF \\
\lli -orbit_length 31 -clique_size 1 \\
\]
The following command reads a file of large sets of designs:

```
ls_ag_2_3_read:
  $(ORBITER) -v 2 \
  -define C -combinatorial_objects \
  -file_of_designs \
  solutions.csv 9 84 3 12 \
  -end
```

The next command reads incidence geometries from a file containing the flags:

```
geo_7_3_read:
  $(ORBITER) -v 10 \
  -draw_incidence_structure_description \
  -width 60 -with_10 6 -end \
  -define C -combinatorial_objects \
  -file_of_incidence_geometries \
  7_3.inc 7 7 21 \
  -end
```

The next command creates incidence geometries from a file containing row-ranks:

```
desargues_path_lex_least_read:
  echo $(DESARGUES_PATH_LEX_LEAST) >Desargues_path_lex_least.inc
  $(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
```

388
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 \texttt{pasch.inc} contains:

\[ 6 \ 4 \ 12 \]
Figure 14.2: The incidence matrix of the Pasch configuration

<table>
<thead>
<tr>
<th>ℓ₀</th>
<th>ℓ₁</th>
<th>ℓ₂</th>
<th>ℓ₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>P₀</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>P₁</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>P₂</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>P₃</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>P₄</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>P₅</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 14.3: Row-major ordering of the flag space

<table>
<thead>
<tr>
<th>0 1 4 6 8 11 13 14 17 19 22 23</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1 1</td>
</tr>
<tr>
<td>24</td>
</tr>
</tbody>
</table>

The first line lists the number of rows and columns of the incidence matrix, and the number of incidences. The geometry is encoded on the next line. After that, a marker of -1 shows that this is the only geometry in this file (the file format allows for any number of incidence geometries, all with the same parameters). The final row is the order of the automorphism group of the geometry. This row is optional. In case that there are several geometries in the file, the orders will all be listed. In this case, the possible values will be collected with multiplicities, and listed in decreasing order. The command

```
geo_pasch_read:
▷ $(ORBITER) -v 10 \
▷▷ -define C -combinatorial_objects \
▷▷▷ -file_of_incidence_geometries \
▷▷▷▷ pasch.inc 6 4 12 \
▷▷▷ -end
```

reads the incidence geometry from the file pasch.inc. It is also possible to enter the incidence geometry directly from the command line. The following example uses the `-incidence_geometry` command to do so:
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 [50] 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

\[
\text{elliptic_curve_b1_c3_q11.txt}.
\]

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 refersto the file containing the Orbiter point ranks of points on the curve.

```
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 2 F -end \n▷ ▷ -with C -do \n▷ ▷ -combinatorial_object_activity \n▷ ▷ ▷ -canonical_form_PG P \n▷ ▷ ▷ ▷ -classification_prefix EC \n▷ ▷ ▷ ▷ -label EC \n▷ ▷ ▷ ▷ -save_ago \n▷ ▷ ▷ ▷ -max_TDO_depth 4 \n▷ ▷ ▷ -end \n▷ ▷ ▷ -report \n▷ ▷ ▷ ▷ -prefix EC \n▷ ▷ ▷ ▷ -export_flag_orbits \n▷ ▷ ▷ ▷ -show_TDO \n▷ ▷ ▷ ▷ -show_TDA \n▷ ▷ ▷ ▷ -dont_show_incidence_matrices \n▷ ▷ ▷ ▷ -export_group \n▷ ▷ ▷ ▷ -end \n▷ ▷ -end
▷ pdflatex EC_classification.tex
▷ open EC_classification.pdf
▷ $(ORBITER) -v 2 -draw_matrix \n▷ ▷ -input_csv_file EC_object0_TDA_flag_orbits.csv \n▷ ▷ -secondary_input_csv_file EC_object0_TDA.csv \n▷ ▷ -box_width 20 -bit_depth 24 \n▷ ▷ -end
▷ open EC_object0_TDA_flag_orbits_draw.bmp
```

Orbiter shows that the curve has a collineation stabilizer of order 6, generated by
The following example computes the canonical form and the automorphism group of the Hirschfeld surface in 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.

HIRSCHFELD_SURFACE_Q4_SET_OF_POINTS="0,1,2,3,4,5,6,7,8,9,\
10,11,12,13,14,23,26,27,30,31,34,35,38,39,42,47,48,51,52,\
53,54,59,60,61,62,67,68,69,70,75,76,79,80,81,82"

Hirschfeld_q4.c: Hirschfeld_surface_q4.txt
▶ $(ORBITER) -v 6 \n▶ ▶ -define C -combinatorial_objects \n▶ ▶ ▶ -file_of_points Hirschfeld_surface_q4.txt \n▶ ▶ ▶ -end \n▶ ▶ ▶ -define F -finite_field -q 4 -end \n▶ ▶ ▶ -define P -projective_space 3 F -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
▶ ▶ ▶ pdflatex Hirschfeld_surface_q4_classification.tex
▶ ▶ open Hirschfeld_surface_q4_classification.pdf

Hirschfeld_q4_set_c:
In the next example, we compute the canonical form of the two hyperovals in PG(2, 16).

```
> $(ORBITER) -v 2 \\
> -define C -combinatorial_objects \ 
> -set_of_points $(HYPEROVAL_16_16320) \ 
> -set_of_points $(HYPEROVAL_16_144) \ 
> -end \ 
> -define F -finite_field -q 16 -end \ 
> -define P -projective_space 2 F -end \ 
> -with C -do \ 
> -combinatorial_object_activity \ 
> -canonical_form PG P \ 
> -classification_prefix hyperoval_q16 \ 
> -label hyperoval_q16 \ 
> -save_ago \ 
> -save_transversal \ 
> -max_TDO_depth 10 \ 
> -end \ 
> -report \ 
> -prefix hyperoval_q16 \ 
> -export_flag_orbits \ 
> -show_TDO \ 
> -show_TDA \ 
> -dont_show_incidence_matrices \ 
> -export_group \ 
```

397
In the next example, we compute the set stabilizers of orbits of PGL(4,2) on subsets of PG(3,2), as computed earlier in Section 6.3, using the command \texttt{PG\_3\_2\_subsets}. These orbits are relevant for Section 7.5. Concerning the work in Dickson [23] only subsets whose size is odd are relevant, so we restrict to those subsets:

\textbf{Dickson\_sets\_stabilizer}:

\begin{verbatim}
$\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 \ "0,1,2,5,6,7,8" \ -set\_of\_points \ "0,1,2,3,5,6,7" \ -set\_of\_points \ "0,1,2,3,5,6,9" \ -set\_of\_points \ "0,1,2,3,5,6,10" \ -set\_of\_points \ "0,1,2,3,5,6,4" \ -set\_of\_points \ "0,1,2,3,8,11,13" \ -set\_of\_points \ "3,6,9,7,10,12,8,11,13,14,4" \ -set\_of\_points \ "3,5,6,9,7,10,12,11,13,14,4" \ -set\_of\_points \ "0,1,2,3,5,6,9,7,10,12,4" \ -end \\
$\text{define F} \ -finite\_field \ -q \ 2 \ -end \\
$\text{define P} \ -projective\_space \ 3 \ F \ -end \\
$\text{with C} \ -do \\
$\text{combinatorial\_object\_activity} \ -end \\
$\text{canonical\_form}_{\text{PG P}}$
\end{verbatim}
There are two ovoids in PG(3, 2). The classical ovoid is the elliptic quadric. It was created using the command `elliptic_quadric_ovaloid_q8` in Section 4.10. The following command computes the stabilizer of the ovoid:

```plaintext
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 3 F -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
$$
```

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 $q = 8$.

```plaintext
ovoid_ST_q8_canon: ovoid_ST_q8.txt
```

399
\[ $(\text{ORBITER}) -v 6 \]
\[ -\text{define} \ C -\text{combinatorial_objects} \]
\[ -\text{file_of_points} \ ovoid\_ST\_q8.txt \]
\[ -\text{end} \]
\[ -\text{define} \ F -\text{finite_field} -q 8 -\text{end} \]
\[ -\text{define} \ P -\text{projective_space} 3 \ F -\text{end} \]
\[ -\text{with} \ C -\text{do} \]
\[ -\text{combinatorial_object_activity} \]
\[ -\text{canonical_form} \ PG \ P \]
\[ -\text{classification} \ \text{prefix} \ ovoid\_ST \]
\[ -\text{label} \ ovoid\_ST \]
\[ -\text{save} \ ago \]
\[ -\text{max}\_\text{TDO}_\text{depth} 4 \]
\[ -\text{end} \]
\[ -\text{report} \]
\[ -\text{prefix} \ ovoid\_ST \]
\[ -\text{show}\_\text{TDO} \]
\[ -\text{show}\_\text{TDA} \]
\[ -\text{dont}\_\text{show}\_\text{incidence}\_\text{matrices} \]
\[ -\text{export}\_\text{group} \]
\[ -\text{end} \]
\[ -\text{end} \]
\[ \text{pdflatex ovoid\_ST\_classification.tex} \]
\[ \text{open ovoid\_ST\_classification.pdf} \]

We can store the generators in a makefile variable as follows:

```
SUZUKI_8_GENERATORS="
1,0,0,0,1,0,0,0,1,0,0,0,1,1, \
1,0,0,0,6,0,0,0,2,0,0,0,0,3,0, \\
1,0,0,0,1,1,0,0,0,1,0,0,1,0,1,0, \ 
1,0,0,0,3,6,2,2,5,0,2,0,3,0,6,3,2, \ 
0,1,0,0,1,0,0,0,0,0,1,0,1,0,1,0,2"
```

We can now recover the Suzuki group using the command:

```
Suzuki_8:
\[ $(\text{ORBITER}) -v 6 \]
\[ -\text{define} \ F -\text{finite_field} -q 8 -\text{end} \]
\[ -\text{define} \ gens -\text{vector} -\text{field} \ F \]
\[ -\text{compact} \ $(\text{SUZUKI} \_8 \_\text{GENERATORS}) -\text{end} \]
\[ -\text{define} \ G -\text{linear_group} -\text{PGGL} 4 \ 8 \]
\[ -\text{subgroup_by_generators} \ "Sz8" \ "87360" \ 5 \ gens \]
\[ -\text{end} \]
```

400
-with G -do \\
-group_theoretic_activity \\
-report \\
-end

dflatex PGGL_4_8_Subgroup_Sz8_87360_report.tex
open PGGL_4_8_Subgroup_Sz8_87360_report.pdf
15.3 Canonical Forms of Incidence Geometries

Let us consider a system of subsets. This subset is 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 if 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 73.

```
geo_7_3.c:
▷ $(ORBITER) -v 10 \n▷ ▷ -draw_incidence_structure_description \n▷ ▷ ▷ -width 60 -with_10 6 -end \n```
A bitmap drawing is produced, as shown in Figure 15.1. The command also produces the following report of the geometry:

### Summary of Orbits

<table>
<thead>
<tr>
<th>Rep</th>
<th>#</th>
<th>Ago</th>
<th>Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>168</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:
\[ g_1 = (3, 5)(4, 6)(8, 9)(12, 13) \text{ of order 2 and with 6 fixed points.} \]
\[ g_2 = (3, 4)(5, 6)(10, 11)(12, 13) \text{ of order 2 and with 6 fixed points.} \]
\[ g_3 = (1, 2)(5, 6)(10, 12)(11, 13) \text{ of order 2 and with 6 fixed points.} \]
\[ g_4 = (1, 3)(2, 4)(7, 8)(11, 12) \text{ of order 2 and with 6 fixed points.} \]
\[ g_5 = (0, 1)(4, 5)(8, 10)(9, 11) \text{ of order 2 and with 6 fixed points.} \]

Canonical labeling:
canonical row = 6
canonical orbit number = 0
Flags : (0, 1, 2, 7, 10, 11, 14, 19, 20, 22, 24, 26, 30, 31, 34, 36, 39, 41, 44, 46, 47)

The following command computes the canonical form and a report of the affine plane AG(2, 3), which is a configuration 9_412_3.

AG_2_3.c: AG_2_3.inc
```
> $(ORBITER) -v 2 \
>   -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>   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:

\[
\begin{array}{c|c}
\to & 12_1 \\
\hline
9_0 & 4 \\
\hline
\downarrow & 12_1 \\
\hline
9_0 & 3 \\
\end{array}
\]
Decomposition by automorphism group:

Canonical labeling:
canonical row = 6
canonical orbit number = 0
Flags : ( 0, 1, 2, 3, 12, 16, 17, 18, 24, 31, 32, 33, 37, 40, 43, 46, 49, 53, 56, 59, 62, 64, 69, 71, 75, 78, 79, 83, 87, 89, 93, 94, 98, 102, 104, 106 )

Flag orbits:
orbit length : number of orbits of that length:

36 1

Anti-Flag orbits:
orbit length : number of orbits of that length:

72 1

It is possible to perform isomorph classification for configurations based on incidence files. Suppose we want to check that the configurations in 10_3 are in fact all nonisomorphic. We apply the canonical form algorithm given by Nauty. This produces a transversal of the isomorphism types of incidence geometries from the given list of input objects. The objects are specified by means of the `combinatorial_objects` command. The classification algorithm can print a report which lists the transversal and all elements in it in LaTeX form.

`geo_10_3_c`: converging
$\$(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 \$

408
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>3</td>
<td>9</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>24</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>120</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>12</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>6</td>
<td>0</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, 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:

\[ 1013112141516181719 \]

Canonical labeling:
- canonical row = 5
- canonical orbit number = 1
- Flags: 0, 1, 2, 16, 17, 18, 25, 27, 29, 34, 38, 40, 43, 45, 51, 53, 56, 62, 63, 64, 70, 74, 77, 82, 86, 89, 91, 95, 98,

\[ 16181719151214101311 \]

**Isomorphism type 1 / 10**

Isomorphism type 1 / 10 is original object 1 and appears 1 times:
- This isomorphism type appears 1 times, namely for the following 1 input objects:
  - \{1\}
- incidence structure:
  - \( (0, 1, 2, 10, 13, 14, 20, 25, 26, 31, 33, 35, 41, 44, 47, 52, 54, 58, 62, 66, 69, 73, 78, 79, 85, 87, 89, 96, 97, 98) \)

Generators for the automorphism group:
The stabilizer of order 2 is generated by:
\[ g_1 = (0, 6)(1, 9)(3, 8)(4, 7)(10, 16)(11, 19)(13, 17)(14, 18) \] of order 2 and with 4 fixed points.

Decomposition by automorphism group:

\[
1016119121317141815
\]

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,

\[
1412181013171161915
\]

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.

```plaintext
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 \n132 146 157 158 171 175 183 195 203 208 220 225 233 244 \n258 259 269 272 282 293 300 308 318 321 342 344 352 358 \n367 379 381 392 398 400 417 428 429 442 443 450 466 471 \n479 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 \n132 146 157 158 171 175 183 195 203 208 220 225 233 244 \n258 259 269 272 281 293 301 308 318 324 327 342 354 357 \n```
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:
  $(ORBITER) -v 3 \n  \> -define D -design -q 3 -family PG_2_q -end \n  \> -with D -do \n  \> \> -design_activity \n  \> \> \> -export_inc \n  \> \> \> -end \n  \> \> -end \n  $(ORBITER) -v 3 \n  \> -draw_incidence_structure_description \n  \> \> -width 60 -with_10 6 -end \n  \> -define C -combinatorial_objects \n  \> \> -file_of_incidence_geometries PG_2_3_inc.txt 13 13 52 \n  \> -end \n  \> -with C -do \n  \> \> -combinatorial_object_activity \n  \> \> \> -canonical_form \n  \> \> \> \> -classification_prefix PG_2_3 \n  \> \> \> \> -label PG_2_3 \n  \> \> \> \> -save_ago \n  \> \> \> \> -save_transversal \n  \> \> \> \> -end \n  \> \> \> -report \n  \> \> \> \> -prefix PG_2_3 \n  \> \> \> \> -export_flag_orbits \n  \> \> \> \> -show_incidence_matrices \n  \> \> \> \> -export_group \n  \> \> \> \> -end \n  \> \> -end \n  pdflatex PG_2_3_classification.tex
  open PG_2_3_classification.pdf
  $(ORBITER) -v 2 -draw_matrix \n  \> -input_csv_file PG_2_3_object0_TDA_flag_orbits.csv \n  \> -secondary_input_csv_file PG_2_3_object0_TDA.csv \n  \> -box_width 32 -bit_depth 24 \n  \> -end \n  open PG_2_3_object0_TDA_flag_orbits_draw.bmp
```

415
The command

```
wreath_product_designs_n4_k2_c: wreath_product_designs_n4_k2_inc.txt
  ▶ $(ORBITER) -v 10 \
  ▶ ▶ -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
  ▶ pdflatex wreath_4_2_classification.tex
  ▶ open wreath_4_2_classification.pdf
```

computes the automorphism group of the design on 8 points created in Section 11.5. The group is \( \text{Sym}(4) \wr \text{Sym}(2) \). The command

```
wreath_product_designs_n8_k6_c: wreath_product_designs_n8_k6_inc.txt
  ▶ $(ORBITER) -v 10 \
  ▶ ▶ -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```
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 $\text{AG}(2, 3)$ 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 $\text{AG}(2, 3)$.
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:

```bash
code 3 2 aut:
▷ $(ORBITER) -v 20 \\
▷ ▷ -define F -finite_field -q 2 -end \\
▷ ▷ -define genma -vector -field F -format 2 \\
▷ ▷ ▷ -dense $(CODE_N3_K2_Q2_GENMA) \\
▷ ▷ -end \\
▷ ▷ -define P -projective_space 1 F -end \\
▷ ▷ -with P -do \\
▷ ▷ -projective_space_activity \\
▷ ▷ ▷ -canonical_form_of_code \\
▷ ▷ ▷ ▷ "3_2" genma -save_ago -label "3_2" \\
▷ ▷ ▷ ▷ -classification_prefix "3_2" \\
▷ ▷ ▷ ▷ -end \\
▷ ▷ ▷ -end
▷ pdflatex 3.2_classification.tex
▷ open 3.2_classification.pdf
▷ $(ORBITER) -v 2 -draw_matrix \\
▷ ▷ -input_csv_file 3.2.object0.TDA_flag_orbits.csv \\
▷ ▷ -secondary_input_csv_file 3.2.object0.TDA.csv \\
▷ ▷ -box_width 16 -bit_depth 24 \\
▷ ▷ -end
▷ open 3.2.object0.TDA_flag_orbits_draw.bmp
```

The code has a semilinear automorphism group of order 6. The following report is written:

<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>
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:
g₁ = (1, 2) of order 2 and with 4 fixed points.
g₂ = (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:
g₁ = \[
\begin{bmatrix}
 1 & 0 \\
 0 & 1
\end{bmatrix}
\] of order 2 and with 1 fixed points.
g₂ = \[
\begin{bmatrix}
 0 & 1 \\
 1 & 0
\end{bmatrix}
\] of order 2 and with 1 fixed points.

Decomposition by combinatorial refinement:
Decomposition by automorphism group:

```
\rightarrow 2_1
\downarrow 2_1
\rightarrow 2
\downarrow 3
```

Canonical labeling:
- canonical row = 3
- canonical orbit number = 1
- Flags : ( 0, 1, 2, 3, 4, 5, 7 )

Flag orbits:
- orbit length : number of orbits of that length:
  
<table>
<thead>
<tr>
<th>orbit length</th>
<th>number of orbits of that length</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

Anti-Flag orbits:
- orbit length : number of orbits of that length:
  
<table>
<thead>
<tr>
<th>orbit length</th>
<th>number of orbits of that length</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

The command

```
CODE_RM_3_1_GENMA="\n11111111\n01010101\n00110011\n00001111"

RM_3_1_group:
  ▶ $(ORBITER) -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```
computes the automorphism group of the Reed-Muller code, of order 1344. It is the affine group $\text{AGL}(3,2)$. A report is created, showing the automorphism group and the action on $\text{PG}(3,2)$, with the Reed-Muller code distinguished.

The following command creates a drawing of the incidence matrix between points and lines in $\text{PG}(3,2)$, with the Reed-Muller code distinguished:

```
RM_3_1_group_and_diagram:
  $(\text{ORBITER}) -v 2 \n  -define F -finite_field -q 2 -end \n  -define genma -vector -field F -format 4 \n  -compact $(\text{CODE_RM_3_1_GENMA}) \n  -end \n  -define P -projective_space 3 F -end \n  -with P -do \n  -projective_space_activity \n  -canonical_form_of_code \n  "RM_3_1" genma -save Ago -label "RM_3_1" \n  -classification_prefix "RM_3_1" \n  -end \n  -end
  pdflatex RM_3_1_classification.tex
  open RM_3_1_classification.pdf
```

```
$(\text{ORBITER}) -v 2 -draw_matrix \n  -input_csv_file RM_3_1_object0_INP_flag_orbits.csv \n  -secondary_input_csv_file RM_3_1_object0_INP.csv \n  -box_width 16 -bit_depth 24 \n  -end
$(\text{ORBITER}) -v 2 -draw_matrix \n  -input_csv_file RM_3_1_object0_TDA_flag_orbits.csv \n  -secondary_input_csv_file RM_3_1_object0_TDA.csv \n  -box_width 16 -bit_depth 24 \n  -end
open RM_3_1_object0_INP_flag_orbits_draw.bmp
```
Figure 15.4: PG(3, 2) with the Reed-Muller code distinguished

\[ \text{open RM.3.1_object0_TDA_flag_orbits_draw.bmp} \]

The drawing in Figure 15.4 is created.

The command

```
RS_6.4_7_group:
  $(ORBITER) -v 20 \\
  -define F -finite_field -q 7 -end \\
  -define genma -vector -field F -format 4 \\
  -compact $(CODE_RS_6_4_7) \\
  -end \\
  -define P -projective_space 3 F -end \\
  -with P -do \\
  -projective_space_activity \\
  -canonical_form_of_code \\
  "RS_6" genma -save_ago -label "RS_6" \\
  -classification_prefix "RS_6" \\
  -end \\
  -end
```

shows that the automorphism group has order 12. After some shortening, the output is:

```
Isomorphism type 0 / 1 is original object 0 and appears 1 times:
set of points of size 6: \{(0, 9, 51, 344, 253, 3)\}
```
<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,

\[
\rightarrow 2850_1 \quad 1_2
\]

\[
401_0 \quad 57 \quad 1
\]

The command

```
GV_n15_k6_d5_group:
  $(ORBITER) -v 20 \n  \> -define F -finite_field -q 2 -end \n  \> -define genma -vector -field F -format 6 \n  \> -compact $(CODE_GV_N15_K6) \n  \> -end \n  \> -define P -projective_space 5 F -end \n  \> -with P -do \n```
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, \n51, 112, 15, 76, 42, 105, 25, 90, 60, 127"
```

Hamming_graph_7_with_Hamming_code:

```
▷ $(ORBITER) -v 2 \n▷ ▷ -define G -graph -Hamming 7 2 \n▷ ▷ ▷ -subset "Hamming_code" "\_\_with\n\_Hamming\_\_code" \n▷ ▷ ▷ $(HAMMING_CODE_CODEWORDS) -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 \n▷ ▷ -with G -do \n▷ ▷ -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 \\
  -end \\
  -with Gamma -do \\
  -graph_theoretic_activity -automorphism_group \\
  -end \\
pdflatex chain_graph_report.tex \\
open chain_graph_report.pdf
```

The following report is written:

```
The automorphism group of `chain_graph` has order 1152 and is generated by:
Strong generators for a group of order 1152:

(12, 13),
(3, 4),
(2, 3),
```

427
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 \
  ../JUNTTILA_KASKI/benchmarks/pp/pp16-1 \n  -end \n  -with Gamma -do \
  -graph_theoretic_activity -save -end \
  -with Gamma -do \
  -graph_theoretic_activity -automorphism_group -end 
```

The command shows a group of order 34217164800. As a measurement of the complexity, the number of backtrack nodes in Nauty is recorded:

```
nb_backtrack1 = 6
nb_backtrack2 = 134
nb_backtrack3 = 134
nb_backtrack4 = 1
```

Here, `nb_backtrack1` is the number of calls to `firstpathnode`, `nb_backtrack2` is the number of calls to `othernode`, `nb_backtrack3` is the number of calls to `processnode`,
nb_backtrack4 is the number of calls to firstterminal. These are the four recursive functions in Nauty.

Unfortunately, the complexity of graph isomorphism is not well-understood. We can see this here. While the first projective plane of order 16 can be handled relatively easily, the second one causes problems. The following command hardly finishes:

**JK_graph_pp16_2:**

```bash
$(ORBITER) -v 2
  -define Gamma -graph -load_dimacs
  ../JUNTTILA_KASKI/benchmarks/pp/pp16-2 -end
  with Gamma -do
  -graph_theoretic_activity -save -end
  with Gamma -do
  -graph_theoretic_activity -automorphism_group -end
```

The difference between the two planes is that the first plane has a very large automorphism group, while the second one has not. For any \( q \), the Desarguesian plane \( \text{PG}(2,q) \) 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:**

```bash
$(ORBITER) -v 2
  -define Gamma -graph -load_dimacs
  ../JUNTTILA_KASKI/benchmarks/srg/sts-13 -end
  with Gamma -do
  -graph_theoretic_activity -save -end
  with Gamma -do
  -graph_theoretic_activity -automorphism_group -end
```

The automorphism group has order 39 and is generated by:

\[
(1, 25, 16)(2, 18, 20)(3, 7, 15)(4, 13, 11)(5, 6, 17)(9, 12, 19)(10, 14, 24)(21, 23, 22),
(0, 1, 2)(3, 4, 5)(7, 8, 9)(11, 12, 13)(14, 16, 18)(15, 17, 19)(20, 22, 24)(21, 23, 25)
\]
Graphs can be created from groups by means of orbitals. An orbital is an orbit of a permutation group $G$ on the set of pairs. Here is an example. We start from the Coxeter-Tits graph on 315 vertices, whose automorphism group is the Hall-Janko group $J_2 : 2$. We first read the graph from file `halljanko315.csv` and compute the automorphism group using Nauty:

```
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 \\
```

430
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

```bash
$ (ORBITER) -v 3
```

```bash
> -define Inc -vector -file 0.5.2_incidence_matrix.csv -end
> -define Gamma -graph -collinearity_graph Inc -end
> -with Gamma -do
> -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_edge:
▷ $(ORBITER) -v 3 \\
▷ ▷ -define F -finite_field -q 17 -end \\
▷ ▷ -with F -do -finite_field_activity \\
▷ ▷ ▷ -cheat_sheet_GF -end \\
▷ pdflatex GF_17.tex \\
▷ open GF_17.pdf
```

Next, we compute the canonical form of the Edge quartic. This command also computes generators for the automorphism group of the curve.

```
Edge_curve_17_nauty:
▷ $(ORBITER) -v 3 \\
▷ ▷ -define C -combinatorial_objects \\
▷ ▷ ▷ -file_of_points Edge_q17.txt \\
▷ ▷ -end \\
▷ ▷ -define F -finite_field -q 17 -end \\
▷ ▷ -define P -projective_space 2 F -end \\
▷ ▷ -with C -do \\
▷ ▷ -combinatorial_object_activity \\
▷ ▷ ▷ -canonical_form_PG P \\
▷ ▷ ▷ ▷ -classification_prefix Edge_curve_q17 \\
▷ ▷ ▷ ▷ -label Edge_curve_q17 \\
▷ ▷ ▷ ▷ -save_ago \\
▷ ▷ ▷ ▷ -save_transversal \\
▷ ▷ ▷ ▷ -max_TDO_depth 10 \\
▷ ▷ ▷ ▷ -end \\
▷ ▷ ▷ -report \\
▷ ▷ ▷ ▷ -prefix Edge_curve_q17 \\
▷ ▷ ▷ ▷ -export_flag_orbits \\
▷ ▷ ▷ ▷ -show_TDO \\
▷ ▷ ▷ ▷ -show_TDA \\
▷ ▷ ▷ ▷ -dont_show_incidence_matrices \\
▷ ▷ ▷ ▷ -export_group \\
▷ ▷ ▷ ▷ -end \\
▷ ▷ -end \\
▷ pdflatex Edge_curve_q17_classification.tex \\
▷ open Edge_curve_q17_classification.pdf \\
▷ $(ORBITER) -v 2 -draw_matrix \\
▷ ▷ -input_csv_file Edge_curve_q17_object0_TDA_flag_orbits.csv \\
```

432
Using the generators that have just been computed, we can recreate the group of the quartic curve:

```
Edge_curve_17_group:
▷ $(ORBITER) -v 3 \
▷ ▷ -define G -linear_group -PGL 3 17 \
▷ ▷ -subgroup_by_generators "Stab_Edge" "24" 3 \
▷ ▷ ▷ "1,0,0,0,13,0,0,0,4" \
▷ ▷ ▷ "1,0,0,0,0,16,0,16,0" \
▷ ▷ ▷ "0,1,16,2,4,4,15,4,4" \
▷ ▷ ▷ -end \
▷ ▷ -with G -do \
▷ ▷ -group_theoretic_activities \
▷ ▷ -print_elements_tex \
▷ ▷ -group_table \
▷ ▷ -report \
▷ ▷ -end
```

```
pdflatex PGL_3_17_Subgroup_Stab_Edge_24_report.tex
open PGL_3_17_Subgroup_Stab_Edge_24_report.pdf
```
Chapter 16

Interfaces

16.1 Graphical Output

Orbiter can produce graphical output in a variety of formats:

1. TikZ / Latex [63],
2. Metapost [33],
3. Bitmap files (.bmp) [66],
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:

<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{-input_csv_file}</td>
<td>csv-file</td>
<td>Specify the input csv-file</td>
</tr>
<tr>
<td>\texttt{-partition}</td>
<td>$w \ R \ C$</td>
<td>Specify a partitioning $R$ of rows and $C$ of columns. Use separating lines of with $w$.</td>
</tr>
<tr>
<td>\texttt{-box_width}</td>
<td>$w$</td>
<td>Use $w$ pixels per matrix entry.</td>
</tr>
<tr>
<td>\texttt{-bit_depth}</td>
<td>$d$</td>
<td>Use color bit depth of $d$ bits ($d = 8$ or $d = 24$).</td>
</tr>
<tr>
<td>\texttt{-invert_colors}</td>
<td></td>
<td>Use an inverted color scheme.</td>
</tr>
</tbody>
</table>

Table 16.1: Commands to Create Bitmap Graphics
The finite field activity -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:

```
Row,C0,C1,C2,C3,C4,C5,C6
0,0,1,2,3,4,5,6
1,1,2,3,4,5,6,0
2,2,3,4,5,6,0,1
3,3,4,5,6,0,1,2
4,4,5,6,0,1,2,3
5,5,6,0,1,2,3,4
6,6,0,1,2,3,4,5
END
```

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 2 F -end \
    -with P -do -projective_space_activity \
    -cheat_sheet_for_decomposition_by_element_PG \
```
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 $F_4$. The associated Singer cycle is the projectivity defined by the companion matrix

$$
\begin{bmatrix}
0 & 1 & 0 \\
0 & 0 & 1 \\
2 & 1 & 1
\end{bmatrix}.
$$

The poset classification algorithm from Sections 6.3 and 6.4 computes partially ordered sets. The posets are created using the -draw_poset option in the poset classification control command package, see Table 6.2. The posets are stored in a file with extension .layered_graph. These files can be drawn using the -draw_layered_graph command. The commands in Table 16.2 and Table 16.3 show ways in which to customize the drawings. Let us consider an example. Suppose we are interested in the Schreier trees of a permutation group represented in a Stabilizer chain. We take PGL(4, 2) in its action on the wedge product. The command

```
0 1 2 3 4 5 6
1 2 3 4 5 6 0
2 3 4 5 6 0 1
3 4 5 6 0 1 2
4 5 6 0 1 2 3
5 6 0 1 2 3 4
6 0 1 2 3 4 5
```

Figure 16.1: Addition table of $F_7$
<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 PG(2, 4)

<table>
<thead>
<tr>
<th>Modifier</th>
<th>Args</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-scale</code></td>
<td>$s$</td>
<td>Use tikz global scale-factor of $s$. Default value: $s = 0.45$.</td>
</tr>
<tr>
<td><code>-line_width</code></td>
<td>$s$</td>
<td>Set tikz line width to $s$. Default value: $s = 1.5$.</td>
</tr>
<tr>
<td><code>-nodes_empty</code></td>
<td></td>
<td>Draw nodes empty. Do not label. Default value: off.</td>
</tr>
<tr>
<td><code>-select_layers</code></td>
<td>$S$</td>
<td>Draw layers whose index is given in the list $S$ only.</td>
</tr>
<tr>
<td><code>-paths_in_between</code></td>
<td>$l_1 \ i_1 \ l_2 \ i_2$</td>
<td>Draw all paths from node $(l_1, i_1)$ to node $(l_2, i_2)$. Here, $(l, i)$ is the $i$-th node at layer $l$ (counting from zero). Delete all other edges between layers $l_1$ and $l_2$.</td>
</tr>
</tbody>
</table>

Table 16.3: Drawing Options for Layered Graph Files (Part 2)
Figure 16.3: The first basic orbit of $\text{PGL}(4,2)$ as a subgroup of $\text{PGO}^+(6,2)$

The command

```
schreier_tree_graphical_output:
 $\$(ORBITER) -v 4 \n  $\$ -draw_options \n  $\$  -yout 500000 \n  $\$   -radius 15 -nodes_empty \n  $\$   -line_width 0.5 -y_stretch 0.25 \n  $\$   -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  $\$ -draw_options \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```

440
Figure 16.4: A Schreier tree in the action on polynomials

\begin{itemize}
\item \item \item -orbits_on_polynomials 3 \\
\item \item \item -orbits_on_polynomials_draw_tree 6 \\
\item \item -end
\end{itemize}
\begin{itemize}
\item pdflatex poly_orbits_d3_n3_q2.tex
\item open poly_orbits_d3_n3_q2.pdf
\end{itemize}

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 [55]. 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_group:
```
   $(ORBITER) -v 3 \
   -define gens -vector -dense \
   "0,1,0,2,0,0,0,0,1, \n   0,0,1,0,1,0,2,0,0, \n   2,0,0,0,1,0,0,0,1" \
   -end \n
   -define G -linear_group -GL 3 3 \n   -subgroup_by_generators "cube" "48" 3 \n   gens \n   -end \n
   -with G -do \n   -group_theoretic_activity \n   -print_elements.tex \n   -report \n   -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
```
<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 aₙ aₜ cₛ cₜ</td>
<td>Set zoom angle and clipping with in round r to change from aₙ to aₜ and from cₛ to cₜ, 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><code>-clipping</code></td>
<td><code>r c</code></td>
<td>In round <code>r</code>, set clipping radius to <code>c</code>.</td>
</tr>
<tr>
<td><code>-text</code></td>
<td><code>r a text</code></td>
<td>In round <code>r</code>, produce running text <code>text</code> with sustain value <code>a</code>.</td>
</tr>
<tr>
<td><code>-label</code></td>
<td><code>r s a g text</code></td>
<td>In round <code>r</code>, produce running text <code>text</code> with start value <code>s</code>, sustain <code>s</code> and gravity <code>g</code>.</td>
</tr>
<tr>
<td><code>-latex</code></td>
<td><code>r s a praemable g text l fname</code></td>
<td>In round <code>r</code>, produce running latex text <code>text</code> with start value <code>s</code>, sustain <code>s</code> and gravity <code>g</code>. Put <code>praemable</code> in the latex source code. Use <code>fname</code> for the latex file names (no extension).</td>
</tr>
<tr>
<td><code>-global_picture_scale</code></td>
<td><code>d</code></td>
<td>Set scaling factor to <code>d</code>.</td>
</tr>
<tr>
<td><code>-picture</code></td>
<td><code>r d fname options</code></td>
<td>In round <code>r</code>, place picture <code>fname</code> scaled by <code>d</code> using options.</td>
</tr>
<tr>
<td><code>-picture</code></td>
<td><code>r d fname options</code></td>
<td>In round <code>r</code>, place picture <code>fname</code> scaled by <code>d</code> using options.</td>
</tr>
<tr>
<td><code>-look_at</code></td>
<td><code>L</code></td>
<td>Override camera look-at value to <code>L</code>. <code>L</code> is a three-dimensional vector.</td>
</tr>
<tr>
<td><code>-default_angle</code></td>
<td><code>a</code></td>
<td>Set default camera angle to <code>a</code>.</td>
</tr>
<tr>
<td><code>-clipping_radius</code></td>
<td><code>f</code></td>
<td>Set default clipping radius to <code>f</code>.</td>
</tr>
<tr>
<td><code>-scale_factor</code></td>
<td><code>s</code></td>
<td>Set default scale factor to <code>s</code>.</td>
</tr>
<tr>
<td><code>-line_radius</code></td>
<td><code>s</code></td>
<td>Set default line radius to <code>s</code>.</td>
</tr>
</tbody>
</table>

Table 16.5: Options for Orbiter 3D-graphics (Part 2)
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-cubic_lex</td>
<td>coeffs</td>
<td>Cubic surface given by 20 coefficients in lexicographic ordering</td>
</tr>
<tr>
<td>-cubic_orbiter</td>
<td>coeffs</td>
<td>Cubic surface given by 20 coefficients in Orbiter ordering</td>
</tr>
<tr>
<td>-cubic_Goursat</td>
<td>A B C</td>
<td>Cubic surface with tetrahedral symmetry given by 3 Goursat coefficients as $Axyz + B(x^2 + y^2 + z^2) + C = 0$</td>
</tr>
<tr>
<td>-quadric_lex_10</td>
<td>coeffs</td>
<td>Quadric surface given by 10 coefficients in lexicographic ordering</td>
</tr>
<tr>
<td>-quartic_lex_35</td>
<td>coeffs</td>
<td>Quartic surface given by 35 coefficients in lexicographic ordering</td>
</tr>
<tr>
<td>-octic_lex_165</td>
<td>coeffs</td>
<td>Octic surface given by 165 coefficients in lexicographic ordering</td>
</tr>
<tr>
<td>-point</td>
<td>coeffs</td>
<td>Point given by three coordinates</td>
</tr>
<tr>
<td>-point_list_from_csv_file</td>
<td>fname</td>
<td>List of points with coordinates given in a csv file</td>
</tr>
<tr>
<td>-line_through_two_points_recentered_from_csv_file</td>
<td>fname</td>
<td>List of lines through two points with point coordinates given in a csv file</td>
</tr>
<tr>
<td>-line_through_two_points_from_csv_file</td>
<td>fname</td>
<td>List of lines through two points with point coordinates given in a csv file</td>
</tr>
<tr>
<td>-point_as_intersection_of_two_lines</td>
<td>$i_1 \ i_2$</td>
<td>Create a point from the intersection of two lines $i_1$ and $i_2$</td>
</tr>
<tr>
<td>-edge</td>
<td>$i_1 \ i_2$</td>
<td>Create an edge (line segment) between points $i_1$ and $i_2$</td>
</tr>
<tr>
<td>-text</td>
<td>$i_1 \ s$</td>
<td>Create a label $s$ located at the point $i$</td>
</tr>
<tr>
<td>-triangular_face_given_by_three_lines</td>
<td>$i_1 \ i_2 \ i_3$</td>
<td>Create a triangular face give by three lines $i_1, i_2, i_3$</td>
</tr>
<tr>
<td>-face</td>
<td>pts</td>
<td>Create a face through the vertices pts, ordered cyclically</td>
</tr>
<tr>
<td>-quadric_through_three_skew_lines</td>
<td>$i_1 \ i_2 \ i_3$</td>
<td>Create a quadric through three skew lines</td>
</tr>
<tr>
<td>-plane_defined_by_three_points</td>
<td>$i_1 \ i_2 \ i_3$</td>
<td>Create a plane through three noncollinear points</td>
</tr>
<tr>
<td>-line_through_two_points_recentered</td>
<td>pt-coords</td>
<td>Create a line through two points given by 6 coordinates, recentered</td>
</tr>
</tbody>
</table>

Table 16.6: Scene definition commands (part 1)
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-line_through_two_points</td>
<td>pt-coords</td>
<td>Create a line through two points given by 6 coordinates</td>
</tr>
<tr>
<td>-line_through_two_existing_points</td>
<td>$i_1 i_2$</td>
<td>Create a line through two points</td>
</tr>
<tr>
<td>-line_through_point_with_direction</td>
<td>$x y z u_x u_y u_z$</td>
<td>Create a line through a point $(x, y, z)$ with a given direction $(u_x, u_y, u_z)$</td>
</tr>
<tr>
<td>-plane_by_dual_coordinates</td>
<td>$a b c d$</td>
<td>Create the plane $ax + by + cz + d = 0$ given in dual coordinates</td>
</tr>
<tr>
<td>-dodecahedron</td>
<td></td>
<td>Create a Dodecahedron centered at the origin (20 points, 30 edges, 12 faces)</td>
</tr>
<tr>
<td>-Hilbert_Cohn_Vossen_surface</td>
<td></td>
<td>Create the Hilbert, Cohn-Vossen surface (1 cubic surface, 45 tritangent planes, 27 lines)</td>
</tr>
<tr>
<td>-obj_file</td>
<td>fname</td>
<td>Read points and faces from the given .obj file</td>
</tr>
<tr>
<td>-group_of_things</td>
<td>list</td>
<td>Create a group of things from the given list</td>
</tr>
<tr>
<td>-group_of_things_with_offset</td>
<td>list offset</td>
<td>Create a group of things from the given list, each value is increase by offset</td>
</tr>
<tr>
<td>-group_of_things_as_interval</td>
<td>$a b$</td>
<td>Create a group of things indexed by the interval $a, \ldots, a + b - 1$</td>
</tr>
<tr>
<td>-group_of_things_as_interval_with_exceptions</td>
<td>$a b$ $ex$</td>
<td>Create a group of things indexed by the interval $a, \ldots, a + b - 1$ with the exceptional elements in the list $ex$ removed</td>
</tr>
<tr>
<td>-group_of_all_points</td>
<td></td>
<td>Create a group of things from all points currently defined</td>
</tr>
<tr>
<td>-group_of_all_faces</td>
<td></td>
<td>Create a group of things from all faces currently defined</td>
</tr>
<tr>
<td>-group_subset_at_random</td>
<td>$i f$</td>
<td>Create a group of things from the existing group $i$ by picking a random subset with probability $f$</td>
</tr>
<tr>
<td>-create_regulus</td>
<td>$i N$</td>
<td>Create a regulus for quadric $i$ with $N$ lines</td>
</tr>
</tbody>
</table>

Table 16.7: Scene definition commands (part 2)
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-spheres</td>
<td>i r 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:

![Image](image.jpg)

The coordinates of the cube are stored in an object file `cube_centered.obj`. The content of this file is:

```
v -1 -1 -1
v 1 -1 -1
v -1 1 -1
v 1 1 -1
v -1 -1 1
v 1 -1 1
v -1 1 1
v 1 1 1
f 1 2 4 3
f 1 2 6 5
f 1 3 7 5
f 2 4 8 6
f 3 4 8 7
f 5 6 8 7
```

The monkey saddle is a cubic surface, given by the equation

\[ z = x^3 - 3xy^2 \]

The next example plots the surface known as the monkey saddle. The tangent plane at \((0, 0, 0)\) is drawn as well. An animation is created by rotating the scene around the z-axis.

`MONKEY_SADDLE_CUBIC="1,0,0,0,-3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-1,0"`
monkey:
▷ $(ORBITER) -v 2 -povray \\
▷ ▷ -round 0 -nb_frames_default 30 \\
▷ ▷ -orbit_masks monkey_%d_%03d.pov \\
▷ ▷ -video_options -W 1024 -H 768 \\
▷ ▷ -global_picture_scale 0.8 \\
▷ ▷ -default_angle 75 \\
▷ ▷ -clipping_radius 0.8 \\
▷ ▷ -camera 0 "0,0,1" "1,1,0.5" "0,0,0" \\
▷ ▷ -rotate_about_z_axis \\
▷ ▷ -end \\
▷ ▷ -scene_objects \\
▷ ▷ ▷ -cubic_lex $(MONKEY_SADDLE_CUBIC) \\
▷ ▷ ▷ -plane_by_dual_coordinates "0,0,1,0" \\
▷ ▷ ▷ -group_of_things "0" \\
▷ ▷ ▷ -group_of_things "0" \\
▷ ▷ ▷ -cubics 0 $(COLOR_GOLD) \\
▷ ▷ ▷ -planes 1 $(COLOR_BLUE) \\
▷ ▷ -scene_objects_end \\
▷ ▷ -povray_end \\
▷ - rm -rf POV \\
▷ mkdir POV \\
▷ mv monkey_0_*.pov POV \\
▷ mv makefile_animation POV

Here is one of the frames that are created:

![Eckardt surface image](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 Schl"afli labeling of the lines is indicated.

**Eckardt.13:**

```latex
\begin{verbatim}
$\text{ORBITER} -v 3 \backslash
\text{define } F \text{-finite_field -q 13 -end} \backslash
\text{define } P \text{-projective_space 3 F -end} \backslash
\text{with } P \text{-do} \backslash
\text{projective_space} \backslash
\text{define} \text{surface S_q13 -q 13} \backslash
\text{with } S \text{q13 -do} \backslash
\text{cubic_surface_activity} \backslash
\text{report} \backslash
\text{report with group} \backslash
\text{end} \backslash
\text{pdflatex surface_family_Eckardt_q13_a3_b1_with_group.tex} \backslash
\text{open surface_family_Eckardt_q13_a3_b1_with_group.pdf}
\end{verbatim}
```

Figure 16.5 shows the final product.

The Endrass octic [25] is the algebraic surface given by the equation

\[ x^8 = 64 (-w^2 + x^2) (-w^2 + y^2) ((x+y)^2 - 2w^2) ((x-y)^2 - 2w^2) - (4 (1 + \sqrt{2}) (x^2 + y^2)^2 + (8 (2 - \sqrt{2})) z^2 + 2 (2 + 7 \sqrt{2}) w^2) (x^2 + y^2) - 16 z^4 + 8 (1 - 2 \sqrt{2}) z^2 w^2 - (1 + 12 \sqrt{2}) w^4)^2 \]

The following Orbiter command creates a povray graphics of the octic, shown in Figure 16.6:

```latex
\begin{verbatim}
ENDRASS_OCTIC_LEX_165="-93.2548,0,0,0,-309.019,0,0,527.529,0,395.647,\\ 0,0,0,0,0,0,0,0,0,0,-687.529,0,0,1582.59,0,1186.94,0,0,0,0,-1055.06,0,\\ -1582.59,0,-593.47,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-309.019,\\ 0,0,1582.59,0,1186.94,0,0,0,0,-2110.12,0,-3165.17,0,-1186.94,0,0,0,0,0,\\ 0,874.039,0,1560.63,0,1677.92,0,343.362,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,\\ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-93.2548,0,0,527.529,0,395.647,\\ -93.2548,0,0,0,-309.019,0,0,527.529,0,395.647,\\ -1582.59,0,-593.47,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-309.019,\\ 0,0,1582.59,0,1186.94,0,0,0,0,-2110.12,0,-3165.17,0,-1186.94,0,0,0,0,0,\\ 0,874.039,0,1560.63,0,1677.92,0,343.362,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,\\ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-93.2548,0,0,527.529,0,395.647,\\ -93.2548,0,0,0,-309.019,0,0,527.529,0,395.647\"
\end{verbatim}
```

endrass8:

```latex
\begin{verbatim}
$\text{ORBITER} -v 2 -povray \backslash
\text{-round 0 -nb_frames_default 30} \backslash
\text{-output_mask endrass_octic_%d_%03d.pov} \backslash
\text{-video_options -W 1024 -H 768} \backslash
\end{verbatim}
```

450
Figure 16.5: The Eckardt surface
This illustration includes coordinate axes and the $x,y$-plane.
Figure 16.6: The Endrass Octic
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>$l$ 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

```bash
monkey_video:
  - rm -r FRAMES
  - mkdir FRAMES
  - rm monkey.mp4
  $ (ORBITER) \
  -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 [38].
16.4 Continuous Function Plotter

Orbiter can plot functions using a built-in function tracker. The functions must be continuous apart from a finite number of poles. The function can have multiple components, each described using an expression. Each expression is specified in Reverse Polish Notation (RPN). Consider an example. A Lissajous curve is defined using coordinate functions of the form

\[ x = r \sin(at + c), \quad y = r \sin(bt), \quad a, b, c, r \in \mathbb{R}. \]

The terms

\[ r \sin(at + c), \quad r \sin(bt) \]

are the expressions of the two coordinate functions. RPN means that the operator is listed after the operands. A stack data structure is used to hold temporary values. Operators are pushed to the top of the stack using the push commands. A binary operator pops the two elements from the stack, performs the operation, and pushes the resulting value back onto the stack. For a unary operator, only one element is popped and replaced by the result.

Here are some examples of expressions rewritten in RPN:

- \( \sin(x) \mapsto \text{push } x \text{ sin}, \)
- \( a + b \mapsto \text{push } a \text{ push } b \text{ add}, \)
- \( a \cdot b \mapsto \text{push } a \text{ push } b \text{ mult}. \)

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

lissajous:

\[
\begin{align*}
\text{push } t & \text{ push } a \text{ mult push } c \text{ add sin push } r \text{ mult return } \\
\text{push } t & \text{ push } b \text{ mult sin push } r \text{ mult return } \\
\text{push } t & \text{ push } a \text{ mult push } c \text{ add sin push } r \text{ mult return } \\
\text{push } t & \text{ push } b \text{ mult sin push } r \text{ mult return }
\end{align*}
\]

The sequence

\[
\text{push } t \text{ push } a \text{ mult push } c \text{ add sin push } r \text{ mult}
\]

is \( r \sin(at + c) \) expressed in RPN. The constants are defined in the line

\[-\text{const a 3 b 2 c 1.57 r 7 -const_end}\]
The input variable is defined using the line

```plaintext
-var t -var_end
```

The sequence

```plaintext
-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

```plaintext
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 
```

456
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```
The 3D curve is shown in Figure 16.8.
Figure 16.8: Lissajous Spacecurve
Chapter 17

Miscellaneous

17.1 Miscellaneous

Table 17.1 lists miscellaneous Orbiter commands. The command `csv_file_select_rows` can be used to select rows from a csv file. The command `csv_file_select_cols` can be used to select columns from a csv file. The command `csv_file_select_rows_and_cols` selects rows and columns. Here is an example. We create the multiplication table of the finite field $\mathbb{F}_7$, ordered according to the powers of a primitive element:

$$\alpha^0, \alpha^1, \alpha^2, \alpha^3, \alpha^4, \alpha^5.$$ 

After that, we pull the rows and columns corresponding to even powers $\alpha^0, \alpha^2, \alpha^4$.

```
$($ORBITER$) -v 3 \\
  $define F -finite_field -q 7 -end \\
  $with F -do -finite_field_activity -cheat_sheet_GF -end \\
  $(ORBITER$) -v 4 -csv_file_select_rows_and_cols \\
  GF_q7_multiplication_table_reordered.csv \\
  "$0,2,4" "$0,2,4"
```

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`

```
Row,C0,C1,C2,C3,C4,C5 
0,1,3,2,6,4,5 
1,3,2,6,4,5,1 
2,2,6,4,5,1,3 
3,6,4,5,1,3,2 
4,4,5,1,3,2,6
```
<table>
<thead>
<tr>
<th>Option</th>
<th>Arguments</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-csv_file_select_rows</td>
<td>fname $R$</td>
<td>Selects rows listed in $R$ from the csv-file fname.</td>
</tr>
<tr>
<td>-csv_file_select_cols</td>
<td>fname $R$</td>
<td>Selects columns listed in $R$ from the csv-file fname.</td>
</tr>
<tr>
<td>-csv_file_select_rows_and_cols</td>
<td>fname $R$ $C$</td>
<td>Selects rows listed in $R$ and columns listed in $C$ from the csv-file fname.</td>
</tr>
<tr>
<td>-csv_file_join</td>
<td>fname col-label</td>
<td>Joins csv file fname according to column with label col-label. This option is given once for each file that should be joined.</td>
</tr>
<tr>
<td>-csv_file_latex</td>
<td>fname</td>
<td>Produces a latex table from the given csv-file.</td>
</tr>
<tr>
<td>-store_as_csv_file</td>
<td>fname $m$ $n$ $L$</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_INCIDENCEMATRIX which is defined in src/lib/foundations/geometry/projective_space.cpp. If the number of lines is too big, the table is not created. In this case, the projective geometry class may behave slower.

5. The projective geometry class tries to build a table for the lines if the number of points is less that MAX_NUMBER_OF_POINTS_FOR_POINT_TABLE and the number of lines is less than MAX_NUMBER_OF_LINES_FOR_LINE_TABLE, both of which are defined in src/lib/foundations/geometry/projective_space.cpp. If the number of points is too big, the table is not created. In this case, the projective geometry class may behave slow.

6. The projective geometry class tries to build a table for the lines through any two points if the number of points is less than MAX_NB_POINTS_FOR_LINE_THROUGH_TWO_POINTS_TABLE which is defined in src/lib/foundations/geometry/projective_space.cpp. If the number of points is too big, the table is not created. In this case, the projective geometry class may behave slow.

7. The projective geometry class tries to build a table for the intersection points of pairs of lines if the number of points is less than MAX_NB_POINTS_FOR_LINE_INTERSECTION_TABLE which is defined in src/lib/foundations/geometry/projective_space.cpp. If the number of points or lines is too big, the table is not created. In this case, the projective geometry class may behave slow.

8. For Windows users: Cygwin by default uses 32 bit integers for both int and long int. Using Cygwin 64 to compile Orbiter recommended.

9. A limited list of primitive polynomials are hard-coded in Orbiter. For large fields, the user must provide their own primitive polynomial. The polynomials encoded in orbiter are not guaranteed to be compatible with the subfield relationship.
Chapter 18
Orbiter on Windows

18.1 Using Windows Subsystem Linux

The following quote from https://docs.microsoft.com/en-us/windows/wsl/ summarizes the function of the Windows Subsystem for Linux:

Windows Subsystem for Linux (WSL) lets developers run a GNU/Linux environment – including most command-line tools, utilities, and applications – directly on Windows, unmodified, without the overhead of a traditional virtual machine or dual-boot setup. You can:

1. Choose your favorite GNU/Linux distributions from the Microsoft Store.
2. Run common command-line tools such as grep, sed, awk, or other ELF-64 binaries.
3. Run Bash shell scripts and GNU/Linux command-line applications including:
4. Tools: vim, emacs, tmux
5. Languages: NodeJS, Javascript, Python, Ruby, C/C++, C# & F#, Rust, Go, etc.
6. Services: SSHD, MySQL, Apache, lighttpd, MongoDB, PostgreSQL.
7. Install additional software using your own GNU/Linux distribution package manager.
8. Invoke Windows applications using a Unix-like command-line shell.
9. Invoke GNU/Linux applications on Windows.

The following set of slides will illustrate the installation of Orbiter under WSL.
Resources

• Many of the steps will be taken from the following sources:
  • https://okunhardt.github.io/documents/Installing_WSL.pdf
  • https://docs.microsoft.com/en-us/windows/wsl/basic-commands
• Consult the two links for further help and suggestions.

Installing WSL

• Search “Turn Windows features on or off” in the Windows search bar
• Search for “Windows Subsystem for Linux”, the box must be checked
• Restart the computer
Update

- The Windows Subsystem for Linux kernel does not automatically update due to system settings.
- Updates must be done manually.
- To update, first you need to command prompt as admin.
  - Press Windows + R to open the “Run” box.
  - Type “cmd” into the box.
  - Press Ctrl + Shift + Enter.
  - When the window prompt opens, click “Yes”.
  - Command prompt will now open as admin.
- In command prompt.
  - Type `wsl --update`
  - Type `wsl --shutdown`.

WSL1, WSL2

- When using WSL, you can adjust the configurations according to the Linux distribution that you are using.
- To run Ubuntu distribution, we need the WSL1 configuration.
- To check the status, in the command prompt enter `wsl --status`.
- To change WSL configuration type
  - `wsl --set-default-version 1`
  - `wsl --shutdown`
Ubuntu - installation

• Generally, the Ubuntu distribution is installed by default when WSL is installed
  • `wsl --status`
    • Displays the default distribution

• If you find that Ubuntu was not installed, you can find it in the Microsoft store

• Launch Ubuntu after installation

Ubuntu - launching

• After launching Ubuntu, allow the installation to be initiated

• If you receive an error, this could be a result of the configuration
  • Set configuration to WSL1
    • `wsl --set-default-version 1`
  • Make sure to terminate Ubuntu and reboot
    • `wsl --terminate Ubuntu`
  • Start Ubuntu again

• Once Ubuntu starts correctly
  • Create Username & Password to complete installation
  • Note: the password will not appear when you type it
Ubuntu - update

- Ubuntu does not update automatically, to update run the command
  - `sudo apt update && sudo apt upgrade`
- You will be prompted to enter your password
- When update are ready to be installed the message will appear
  - Do you want to continue? [Y/n]
    - Y + enter

Ubuntu – g++ and make

- At this point, you have successfully installed and setup WSL, and now you can use the terminal as you would on Ubuntu
- Terminate and reboot Ubuntu
- Run the command in Ubuntu
  - `sudo apt install g++`
  - You can now compile C++ in WSL
- Run the command in Ubuntu
  - `sudo apt install make`
  - You can now use makefiles in WSL
Orbiter - installation

• The easiest way to run make is through the command prompt, not Ubuntu

• To run WSL commands in command prompt, use either
  • `wsl <command>`
  • `wsl.exe <command>`

• Open command prompt

• Change directory to Users\username
  • `cd C:\Users\"your username"`

Orbiter - installation

• In web, go to https://github.com/abetten/orbiter

• Click on the green icon “Code” that opens a drop-down menu

• You want to copy HTTPS URL
Orbiter - installation

• In command prompt, once you are in C:\Users\Joel type the command
  • wsl.exe git clone https://github.com/abetten/orbiter.git
  • Hit enter
• Now, orbiter will begin the cloning process

![Cloning Process Screenshot]

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 Content Screenshot]
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

```
C:\Users\Joel\orbiter> wsl.exe make
cd src; make all
make[1]: Entering directory '/mnt/c/Users/Joel/orbiter/src'
cd lib; make all
make[2]: Entering directory '/mnt/c/Users/Joel/orbiter/src/lib'
cd foundations; make
```

Makefile

- Now that orbiter has been successfully compiled, in the directory C:\Users\"your username\"\ orbiter
  - Change directory to C:\Users\"your username\" and create a new directory
    - Ex: mkdir CPP_Workspace
  - Change directory into CPP_Workspace
    - cd CPP_Workspace
  - In C:\Users\"your username\"\"new directory\", run the command
    - wsl.exe vim makefile
  - Vim (an IDE) will create the file “makefile”
  - For Vim commands, go to https://vim.rtorr.com/
  - Remember: all Ubuntu commands must begin with either
    - wsl or wsl.exe
Makefile

- To edit file in vim, click “!”
- 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
###############################################################################

475


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="\n 1,0,0,0,0,0,1,0,0,0,0,0,0,0, \n 1,1,1,1,1,1,0,1,0,0,0,0,0,0, \n 1,2,4,1,2,4,1,0,0,1,0,0,0,0, \n 1,3,2,6,4,5,1,0,0,0,1,0,0,0, \n 1,4,2,1,4,2,1,0,0,0,1,0,0,0, \n 1,5,4,6,2,3,1,0,0,0,0,1,0,0, \n 1,6,1,6,1,6,1,0,0,0,0,0,1"

#Co3 from Conway et al., 1985 (ATLAS)
#order = 495766656000
#Co3 from the paper by Suleiman and Wilson 1997

CONWAY_GEN1="\n 110111000100001010000\n 1111011111110100001011\n 000001000000100010101\n 1111100110110001001110\n 0101010000000100111011"
CONWAY_GEN2="
0101000101110101111111
0110010011111111010000
0011010001111111010111
0001101110001011010011
1010010001000101010011
1101000000101010000111
0101110100111000010111
0101111101011101111101
10001101011010101010101
0100110010100000000111
1100000101001010010010
0101110100111000010111
0101111101011101111101
10011101011010101010101
10001010101010101010001
0001010001111111001011
0011010011111111010111
1001101001111111010111
0100110010111000101111
10001010101010101010001
01001010100100010000001
0101110100111000010111
0101111101011101111101
11001101011010101010101
0100110010111000101111
10001010101010101010001
1100101001010000100001
11011010110011110110001
1101101100101000000001
0000001101111101000111
01011011010100100000001
11001101011010101010101
0101110100111000010111
0101111101011101111101
1101101101010100000001
1101101101010100000001

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


GEN\_C13="1,2,3,4,5,6,7,8,9,10,11,12,0"

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

GENERATORS\_HESSE\_GROUP=""

3000300030 \ 2000201230 \ 1000100111 \ 1000220200 \ 1002312010 \ 0331003211 \ 2200011331"

GENERATORS\_WEYL\_GROUP\_E8=""

-1,-1,-1,-1,0,0,0,0, \ 0,0,0,1,0,0,0,0, \ 1,0,0,0,0,0,0,0, \ 0,0,1,0,0,0,0,0, 

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

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

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

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

-1,0,-1,-1,-1,-1,-1,-1, 

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

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

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

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

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

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

0,0,0,0,0,0,1,0"

Ree\_gen1="21,5,1,6,17,1,1, 3,13,5,21,6,6,18, 21,3,21,21,22,6,14, 

14,18,1,5,13,6,7, 3,3,2,1,24,16,3, 17,3,22,10,16,24,26, 

478
21,21,6,18,20,2,5"

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

HIRSCHFELD_SURFACE_Q4_SET_OF_POINTS="0,1,2,3,4,5,6,7,8,9, 
10,11,12,13,14,23,26,27,30,31,34,35,38,39,42,47,48,51,52, 
53,54,59,60,61,62,67,68,69,70,75,76,79,80,81,82"

HYPEROVAL_16_144="0, 1, 2, 3, 52, 67, 89, 106, 126, 
141, 159, 176, 184, 199, 220, 235, 245, 262"

HYPEROVAL_16_16320="0, 1, 2, 3, 52, 70, 83, 109, 127, 
139, 156, 186, 199, 217, 229, 256, 264"

FILE_24_3_TFC_INC="24 24 72"

\n0 1 2 24 27 28 48 53 54 73 79 80 97 105 106 122 131 \
132 146 157 181 183 185 203 208 220 225 233 244 \
258 259 269 272 282 293 300 308 325 333 342 352 358 \
367 379 381 392 398 400 417 428 429 442 443 450 466 471 \
479 492 497 502 517 519 521 542 548 551 571 574 575 \
48"

\n0 1 2 24 27 28 48 53 54 73 79 80 97 105 106 122 131 \
132 146 157 181 183 185 203 208 220 225 233 244 \
258 259 269 272 281 293 301 308 324 327 342 354 357 \
367 373 378 392 400 403 417 429 442 443 447 466 472 \
479 492 500 503 518 525 526 545 549 551 571 572 574 \
48"

\n0 1 2 24 27 28 48 53 54 73 79 80 97 105 106 122 131 \
132 146 157 181 183 185 203 208 220 225 233 244 \
258 259 269 274 277 293 300 307 318 323 329 342 352 356 \
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_F7_15LINES_MCKEAN_POINTS="0,1,2,3,4,5,6,7,8,9,10,16,17,28,35,36,39,59,60,61,62,63,64,65,76,80,91,95,106,107,111,119,121,122,130,138,139,141,146,150,154,155,158,170,175,184,186,199,203,204,206,226,231,234,239,240,252,253,254,278,279,282,287,299,301,302,319,320,330,338,343,345,350,351,357,364,370,371,376,378,382,385,388,392,394,399"

SURFACE_MCKEAN_15_LINES="1,5,-1,16,-2,17,1,10,2,18,1,8,1,9,-1,11,-1,19,-1,14"
CODE_RS_6_4_7="\\n621000 \\
062100 \\
006210 \\
000621"

CODE_RS_10_8_11="\\n7,2,1,0,0,0,0,0,0,0, \n0,7,2,1,0,0,0,0,0,0, \n0,0,7,2,1,0,0,0,0,0, \n0,0,0,7,2,1,0,0,0,0, \n0,0,0,0,0,7,2,1,0,0, \\
0,0,0,0,0,0,7,2,1,0, \n0,0,0,0,0,0,0,7,2,1, \\
0,0,0,0,0,0,0,0,7,2,1 "

# Normal form for 15 lines:

F_ALPHA_BETA_GAMMA_DELTA="\n\n+ (alpha*delta - beta*gamma + alpha - beta - delta - 1)*x0*x1*x2 \n-1*(alpha*beta -alpha*delta + delta)*x0*x1*x3 \n(0-alpha*delta + alpha*gamma -beta*gamma +beta -delta -gamma)*x0*x2*x2 
-(alpha*beta -beta -delta)*x0*x2*x3 
-(delta + 1)*(alpha - 1)*x1*x1*x2 \n+(alpha*delta - alpha*gamma + beta*gamma - beta - delta + gamma)*x1*x2*x2 
alpha*beta*gamma - beta*gamma + beta - delta + gamma)*x1*x2*x3 
alpha*beta*(gamma + 1)*x1*x3*x3"

# 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*d - 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 + a*a*c*d + d - b*c)*X1*X2*X3 
- (1+1)*a*b*b*c*c + a*b*b*c*d + (1+1)*a*b*c*c*d + a*b*c*d*d 
"
\[ - b*b*c*c*d - a*a*b*c + a*a*c*d + a*a*d*d + a*b*b*c + a*b*c*c + (1+1+1+1)*a*b*c*d - a*c*c*d + a*c*d*d + b*b*c*c)X1*X2*X3 + c*a*(a*d - b*c - a + b + c - d)*(b - d)*X1*X3*X3" \]

```
KNECHT.13.1_AS_PAIRS="1,0,1,1,1,2,12,9"
KNECHT.13.1_AS_VECTOR="1,1,1,0,0,0,0,0,0,12,0,0,0,0"
KNECHT.13.2_AS_PAIRS="1,0,1,1,2,8,9,8,10,8,11"
KNECHT.13.2_AS_VECTOR="1,1,1,0,0,0,0,0,8,0,8,0,0,0"
```

```
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="
11111111
01010101
00110011
00001111"
CODE_RM.4.1_GENMA="
1111111111111111
0101010101010101
0011001100110011
0000111100001111
0000000011111111"
CODE_RS.8="
5610000 \
0561000 \
0056100 \
```

482
CODE_RS_11_REF=" \n1,0,0,0,0,0,0,7,2, 
0,1,0,0,0,0,0,8,3, 
0,0,1,0,0,0,0,1,2, 
0,0,0,1,0,0,0,8,8, 
0,0,0,0,1,0,0,10,3, 
0,0,0,0,0,1,0,1,4, 
0,0,0,0,0,0,1,0,5,4, 
0,0,0,0,0,0,0,1,5,8" 
RS_8_reduced=" \n0100011000000000000000 
0011100100000000000000 
1100110010000000000000 
0001000110000000000000 
0000111001000000000000 
0001100110010000000000 
0000000110011000000000 
0000001100110010000000 
0111100000000000100000 
11111000000000000110000 
11000100000000000100000 
10100100000000000010000 
00111000000000000100000 
11111000000000000010000 
11000100000000000010000 
10100100000000000010000 
011001000000000000010000 \nCODE_21_15_4=" \n111000100000000000000000 
110100010000000000000000 
101100010000000000000000 
011100001000000000000000 
110010000100000000000000 
101010000001000000000000 
010100000001000000000000 
001110000000001000000000 
010100000000001000000000 
001110000000000100000000 
111110000000000000010000 
110001000000000000010000 
101001000000000000010000 
011001000000000000010000 
011001000000000000010000 \n483
100101000000000000001"  

# there are 5 [15,6,6]

# ago=12
CODE_15_6_6_A="\
   111111111000000 \
   111100000010000 \
   110011000010000 \
   110101000001000 \
   101010110000010 \
   101101001000001"

# ago=12
CODE_15_6_6_B="\
   111111111000000 \
   111100000010000 \
   110011000010000 \
   110101000001000 \
   101010110000010 \
   011011001000001"

# ago=720:
CODE_15_6_6_C="\
   111111111000000 \
   111100000001000 \
   110011000001000 \
   110101000001000 \
   101101001000001 \
   100010111000001"

# ago=96:
CODE_15_6_6_D="\
   111111111000000 \
   111100000001000 \
   110011000001000 \
   110101010000100 \
   101011001000010 \
   011001011000001"

# ago=360
CODE_15_6_6_E="\
   111111111000000 \
   111100000001000 \
   110011000001000 \
   110101010000100 \
   10011010000100 \
   1001110100000100 \

484
010101110000010 \
010110101000001"
ELEMENTARY SYMMETRIC 8_2="x0*x1 + x0*x2 + x0*x3 + x0*x4 + x0*x5 + x0*x6 + x0*x7 + x1*x2 + x1*x3 + x1*x4 + x1*x5 + x1*x6 + x1*x7 + x2*x3 + x2*x4 + x2*x5 + x2*x6 + x2*x7 + x3*x4 + x3*x5 + x3*x6 + x3*x7 + x4*x5 + x4*x6 + x4*x7 + x5*x6 + x5*x7 + x6*x7"

ELEMENTARY SYMMETRIC 8_3="x0*x1*x2 + x0*x1*x3 + x0*x1*x4 + x0*x1*x5 + x0*x1*x6 + x0*x1*x7 + x0*x2*x3 + x0*x2*x4 + x0*x2*x5 + x0*x2*x6 + x0*x2*x7 + x0*x3*x4 + x0*x3*x5 + x0*x3*x6 + x0*x3*x7 + x0*x4*x5 + x0*x4*x6 + x0*x4*x7 + x0*x5*x6 + x0*x5*x7 + x0*x6*x7 + x1*x2*x3 + x1*x2*x4 + x1*x2*x5 + x1*x2*x6 + x1*x2*x7 + x1*x3*x4 + x1*x3*x5 + x1*x3*x6 + x1*x3*x7 + x1*x4*x5 + x1*x4*x6 + x1*x4*x7 + x1*x5*x6 + x1*x5*x7 + x1*x6*x7 + x2*x3*x4 + x2*x3*x5 + x2*x3*x6 + x2*x3*x7 + x2*x4*x5 + x2*x4*x6 + x2*x4*x7 + x2*x5*x6 + x2*x5*x7 + x2*x6*x7 + 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"

ELEMENTARY SYMMETRIC 8_4="x0*x1*x2*x3 + x0*x1*x2*x4 + x0*x1*x2*x5 + x0*x1*x2*x6 + x0*x1*x2*x7 + x0*x1*x3*x4 + x0*x1*x3*x5 + x0*x1*x3*x6 + x0*x1*x3*x7 + x0*x1*x4*x5 + x0*x1*x4*x6 + x0*x1*x4*x7 + x0*x1*x5*x6 + x0*x1*x5*x7 + x0*x1*x6*x7 + x0*x2*x3*x4 + x0*x2*x3*x5 + x0*x2*x3*x6 + x0*x2*x3*x7 + x0*x2*x4*x5 + x0*x2*x4*x6 + x0*x2*x4*x7 + x0*x2*x5*x6 + x0*x2*x5*x7 + x0*x2*x6*x7 + x0*x3*x4*x5 + x0*x3*x4*x6 + x0*x3*x4*x7 + x0*x3*x5*x6 + x0*x3*x5*x7 + x0*x3*x6*x7 + x0*x4*x5*x6 + x0*x4*x5*x7 + x0*x4*x6*x7 + x0*x5*x6*x7 + x0*x5*x7*x8 + x0*x6*x7*x8 + x1*x2*x3*x4 + x1*x2*x3*x5 + x1*x2*x3*x6 + x1*x2*x3*x7 + x1*x2*x4*x5 + x1*x2*x4*x6 + x1*x2*x4*x7 + x1*x2*x5*x6 + x1*x2*x5*x7 + x1*x2*x6*x7 + x1*x3*x4*x5 + x1*x3*x4*x6 + x1*x3*x4*x7 + x1*x3*x5*x6 + x1*x3*x5*x7 + x1*x3*x6*x7 + x1*x4*x5*x6 + x1*x4*x5*x7 + x1*x4*x6*x7 + x1*x5*x6*x7 + x1*x5*x7*x8 + x1*x6*x7*x8 + x2*x3*x4*x5 + x2*x3*x4*x6 + x2*x3*x4*x7 + x2*x3*x5*x6 + x2*x3*x5*x7 + x2*x3*x6*x7 + x2*x4*x5*x6 + x2*x4*x5*x7 + x2*x4*x6*x7 + x2*x5*x6*x7 + x2*x5*x7*x8 + x2*x6*x7*x8 + x3*x4*x5*x6 + x3*x4*x5*x7 + x3*x4*x6*x7 + x3*x4*x7*x8 + x3*x5*x6*x7 + x3*x5*x7*x8 + x3*x6*x7*x8 + x4*x5*x6*x7 + x4*x5*x7*x8 + x4*x6*x7*x8 + x5*x6*x7*x8 + x6*x7*x8"
+ x0*x2*x3*x4*x5*x6 + x0*x2*x3*x4*x5*x7 + x0*x2*x3*x4*x6*x7 + x0*x2*x3*x5*x6*x7 + 
  x0*x2*x4*x5*x6*x7 + x0*x3*x4*x5*x6*x7 + x1*x2*x3*x4*x5*x6*x7 + x1*x2*x3*x4*x5*x7 + 
  x1*x2*x3*x4*x5*x6*x7 + x1*x2*x3*x5*x6*x7 + x1*x2*x4*x5*x6*x7 + x1*x3*x4*x5*x6*x7 + x 
  2*x3*x4*x5*x6*x7"

481

482 ELEMENTARY_SYMMENTRIC_8_7="x0*x1*x2*x3*x4*x5*x6 + x0*x1*x2*x3*x4*x5*x7 + x0*x1*x2*
  x3*x4*x6*x7 + x0*x1*x2*x3*x5*x6*x7 + x0*x1*x2*x4*x5*x6*x7 + x0*x1*x3*x4*x5*x6*x7 + 
  x0*x2*x3*x4*x5*x6*x7 + x1*x2*x3*x4*x5*x6*x7 + x1*x2*x3*x4*x5*x6*x7"

483

484 ELEMENTARY_SYMMENTRIC_8_8="x0*x1*x2*x3*x4*x5*x6*x7"

485

486

487

488

489 PG_3_5_DESARGUESIAN_SPREAD="0, 805, 36, 108, 72, 144, \\
  581, 509, 686, 415, 639, 758, 285, 722, 332, 343, 202, \\
  592, 473, 238, 675, 379, 166, 545, 249, 451"

490

491 # elements of order 2:

492 # conjugacy class reps:

493 # elt order, class size, centralizer order

494

495 #2A: 2 48960 40320 Baer involution

496

497 #2B: 2 5355 368640 one block of 10,11

498

499 #2C: 2 64260 30720 two blocks of 10,11 (problem group)

500

501 #> pdflatex PGGL_4_4_classes_out.tex

502 #> open PGGL_4_4_classes_out.pdf

503

504 # elements of order 2:

505 # conjugacy class reps:

506 # elt order, class size, centralizer order

507

508 #2 48960 40320 Baer involution

509

510 #2 5355 368640 one block of 10,11

511

512 #2 64260 30720 two blocks of 10,11 (problem group)

513

514 CLASS_2A=-centralizer_of_element \\n
515 "1,0,0,0, 0,1,0,0, 0,0,1,0, 0,0,0,1, 1" \\n
516 -label "2A"

517

518

519
# 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 \
> "1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1, \
> 1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1, \
> 1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1, \
> 1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1, \
> 1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1, \
> 1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1, \
> 1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1, \
> 1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1, \
> 0,1,0,0,0,0,0,0,1,1,1,0,1,0,1,1,1,1"

# 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, 0"

PGGL_4_4_SUBGROUP_2C_NORMALIZER=-PGGL 4 4 \
> -subgroup_by_generators "centralizer_2C" "30720" 9 \
> "1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1, \
> 1,0,0,0,0,1,0,0,0,0,2,0,0,0,0,2,1, \
> 1,0,0,0,0,1,0,0,0,0,1,0,2,0,3,1,0, \

488
PGGL_4_4_SUBGROUP_5A=-PGGL 4 4 \\
- subgroup_by_generators "5A" 5 1 \\
"0,2,0,0, 1,1,0,0, 0,0,3,0, 0,0,0,3, 0"

PGGL_4_4_SUBGROUP_5A_NORMALIZER=-PGGL 4 4 \\
- subgroup_by_generators "normalizer_5A" "3600" 6 \\
"1,0,0,0,0,1,0,0,0,0,2,0,0,0,0,2,0, \\
1,0,0,0,0,1,0,0,0,0,1,0,0,0,2,0, \\
1,0,0,0,0,1,0,0,0,0,1,0,0,1,1,0, \\
1,0,0,0,0,1,0,0,0,0,2,0,0,2,0,0, \\
1,0,0,0,0,2,2,0,0,0,0,1,0,0,0,1,1, \\
0,1,0,0,3,3,0,0,0,0,2,0,0,0,2,0"

PGGL_4_4_SUBGROUP_5B=-PGGL 4 4 \\
- subgroup_by_generators "5B" 5 1 \\
"0,2,0,0,1,1,0,0,0,0,0,2,0,0,1,1,0"

PGGL_4_4_SUBGROUP_5B_NORMALIZER=-PGGL 4 4 \\
- subgroup_by_generators "normalizer_5B" "81600" 6 \\
"1,0,0,0,0,1,0,0,0,0,2,0,0,0,0,2,0, \\
1,0,0,0,0,1,0,0,0,0,1,0,0,0,2,0, \\
1,0,0,0,0,1,0,0,0,0,1,0,0,1,1,0, \\
1,0,0,0,0,2,2,0,0,0,0,1,0,0,0,1,1, \\
0,1,0,0,3,3,0,0,0,0,0,1,0,0,3,3,0, \\
0,0,1,0,0,0,1,2,2,0,0,0,2,3,0,0,1

PGGL_4_4_SUBGROUP_2Cx2_0=-PGGL 4 4 \\
- subgroup_by_generators "2Cx2_0" 4 2 \\
"1,0,0,0,1,1,0,0,0,0,1,0,0,0,1,1,0 \\
1,0,0,0,0,1,0,0,1,0,0,1,0,1,0,1,0"

PGGL_4_4_SUBGROUP_2Cx2_0_NORMALIZER=-PGGL 4 4 \\
- 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 \\
1,0,0,0,0,1,0,0,0,0,1,0,2,0,0,1,0, \\
1,0,0,0,0,1,0,0,0,1,0,1,0,3,1,0,1,1 \\
1,0,0,0,0,1,0,0,3,0,1,0,2,3,0,1,1"
# elementary abelian subgroups of order 4 with 3 elements of class 2C:

# nice generators, from Michael Epstein:
# subgroup of order 31 for the construction of regular packings in PG_3_5:

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 \ "1,0,0,0,0,4,0,0,0,4,0,0,0,4, \ 1,0,0,0,4,0,0,0,0,2,1,0,3,2,4, \ 1,0,0,0,0,1,0,0,0,1,0,1,1,3",

#372:
"1,0,0,0,4,0,0,0,0,4,0,0,0,4, " \ "1,0,0,0,3,0,0,0,0,3,0,0,0,3, " \ "1,0,0,0,4,0,0,0,2,1,0,3,2,4, " \ "1,0,0,0,0,1,0,0,0,1,0,1,1,3",

#Exterior square roots:

#elt of order 3:

#the exterior square root of f is X=
# elt of order 31:
# the exterior square root of g is Z=
# [1 0 0 0]
# [0 3 4 3]
# [0 3 3 4]
# [0 3 2 3]

#Michael

HAMMING_CODE_CODEWORDS="0, 67, 37, 102, 22, 85, 51, 112, 15, 76, 42, 105, 25, 90, 60, 127"

SIMPLEX_CODE_GENERATOR="1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1"

HAMMING_CODE_GENERATOR="1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1"

HAMMING_CODE_ROWS_IN_BINARY_RANKS="67, 37, 22, 15"

SIMPLEX_CODE_GENMA_CYCLIC="1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1"

CODE_GV_N15_K6="111111111100000\n111110000010000\n111001100001000\n110101010000100\n101010110000010\n10110100100001"
CODE_GV_N15_K6_CHECK="\n100000000111111\n010000000111100\n001000000111011\n000100000101010\n000010000100100\n000001000100100\n000000100100100\n000000010100100\n000000001100001"

REED_MULLER_3_1_CODEWORDS="0,255,170,85,204,51,102,\n153,240,15,90,165,60,195,150,105"

REED_MULLER_3_1_BASIS_IN_BINARY="255,170,204,240"

REED_MULLER_4_1_COLUMNS_OF_PARTITY_CHECK="1,3,5,7,9,11,13,\n15,17,19,21,23,25,27,29,31"

#-nearest_codeword "8,16,32,24,40,48,56,1,2,4,3,5,6,7,9,18,36,27,45,54,63,33,42,2\n0,11,53,62,31"

RM_6_GENERATOR_1="0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,\n22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,\n46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63"

RM_6_GENERATOR_2="1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,\n41,43,45,47,49,51,53,55,57,59,61,63"

RM_6_GENERATOR_3="2,3,6,7,18,19,22,23,10,11,14,15,26,27,30,31,34,35,38,\n39,42,43,46,47,50,51,54,55,58,59,62,63"

RM_6_GENERATOR_4="4,6,12,14,36,38,52,54,5,7,13,15,37,39,53,55,20,22,28,\n30,44,46,60,62,21,23,29,31,45,47,61,63"

RM_6_GENERATOR_5="8,9,12,13,24,25,28,29,10,11,14,15,26,27,30,31,40,41,\n44,45,56,57,60,61,42,43,46,47,58,59,62,63"

RM_6_GENERATOR_6="16,18,24,26,48,50,56,58,17,19,25,27,49,51,57,59,20,22,\n28,30,52,54,60,62,21,23,29,31,53,55,61,63"

RM_6_GENERATOR_7="32,34,48,50,33,35,49,51,36,38,52,54,37,39,53,55,40,42,\n56,58,41,43,57,59,44,46,60,62,45,47,61,63"
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
02,403,405,416,419,421,422,425,426,429,430,440,443,447,449,453,454,464,467,468,47
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"
TEST_RHS="1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1"
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=\n> -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
DELANDTSHEER\_DOYEN\_PROBLEM\_27.53 = -d1 1 -q1 27 -d2 1 -q2 53 -K 11 -DDx 2 -DDy 1
-search_control -W -end

DELANDTSHEER\_DOYEN\_PROBLEM\_27.53\_GROUP1=-subgroup \ "1,1,1,0, 1,3,1,0, 1,9,1,0, 1,0,1,1, -2,0,-4,0" "18603" -group_label "group1"

# mask 1:
# XX.
# X.X+

DELANDTSHEER\_DOYEN\_PROBLEM\_27.53\_MASK1=\`

DELANDTSHEER\_DOYEN\_PROBLEM\_3.7 = -d1 1 -q1 3 -d2 1 -q2 7 -K 5 -DDx 3 -DDy 1 -search_control -W -end

DELANDTSHEER\_DOYEN\_PROBLEM\_3.7\_GROUP1=-subgroup \ "1,1,1,0, 1,0,1,1 " "21" -group_label "group cyclic"

DELANDTSHEER\_DOYEN\_PROBLEM\_3.7\_MASK1= -mask_label "mask1" -depth 5

PENTTILA\_WILLIAMS\_PRINCE\_REG\_PACKING\_0="444,43313,154402,46682,108254,\ 
75363,27729,32139,5244,60442,142811,111115,94209,120678,89533,13798,\ 
103994,129953,82168,136838,19253,23017,145985,134996,54705,36267,\ 
55066,117542,96699,69154,72460"

PENTTILA\_WILLIAMS\_PRINCE\_REG\_PACKING\_1="616,42728,152655,48576,105431,\ 
79607,28634,32817,9799,62356,141176,110085,92557,122136,86312,13975,\ 
101942,126869,81478,139352,18028,24325,147284,130370,52074,36843,\ 
55602,118454,95973,69642,74036"

PENTTILA\_WILLIAMS\_PRINCE\_REG\_PACKING\_0\_DUAL="3938, 66740, 56555, \ 
93538, 107785, 64917, 47567, 54483, 141012, 138602, 18308, 6880, \
PG_3.5_PACKING_0_WITH_AG03="0,5201,60427,86602,11453,121452,46663,\ 
19716,32921,108680,23456,91963,68386,26921,74601,57067,36188,42312,\ 
78780,53117,118488,114700,83960,104791,126662,130960,145179,\ 
137230,150626,140216"

PG_3.5_PACKING_0_WITH_AG03_FIXP444="444,5001,12957,18194,23485,26817,\ 
34667,38299,41249,47472,50450,56601,62638,68986,71833,75369,80805,\ 
87025,92577,95676,104509,109718,114948,116333,124391,127498,133240,\ 
137711,144777,148059,150175"

# 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\n1,0,1\n1,1,0\n"

# q=17:
# 3 is p.e. mod 17.
# so we pick f=3.
# then, 2f^2=18=1
# 4f = 12
# X^4 -Y^4 -Z^4 +2f^2Y^2Z^2 +4fX^2YZ
#(1,-1,-1,0,0,0,0,0,0,0,0,2f^2,2f,0,0)
EDGE_CURVE_Q17_EQUATION="1,16,16,0,0,0,0,0,0,0,0,1,12,0,0"
EDGE_CURVE_Q17_AS_POINTS="4, 7, 16, 19, 20, 23, 32, 35, 89, 100, 244, 251"

FILE_Q17="orbit,curve,pts_on_curve,bitangents.go"
\nO0,\"$(EDGE CURLVE_Q17_EQUATION)\",\"$(EDGE CURVE Q17 AS POINTS)\",\"\",-1\n\nEND"

DESARGUES_PATH_LEX_LEAST="10 10 3\n0 15\n26 15 26 46 \n6 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="texture{ Polished.Chrome pigment{quick_color White} }

YELLOW_TRANSPARENT="texture{ pigment{ color Yellow transmit 0.7 } 
finish {diffuse 0.9 phong 0.6} }"

COLOR_RED="texture{ pigment{ color Red } 
finish {diffuse 0.9 phong 0.6} }"

COLOR_RED_SHINY="texture{ pigment{ color Red } 
finish {diffuse 0.9 phong 1}}"

COLOR_GREEN_SHINY="texture{ pigment{ color Green } 
finish { diffuse 0.9 phong 1}}"

COLOR_BLUE_SHINY="texture{ pigment{ color Blue } 
finish { diffuse 0.9 phong 1}}"

COLOR_YELLOW_SHINY="texture{ pigment{ color Yellow } 
finish { diffuse 0.9 phong 1}}"

COLOR_BLACK_SHINY="texture{ pigment{ color Black } 
finish { diffuse 0.9 phong 1}}"

COLOR_BLACK_SHINY="texture{ pigment{ color Black } 
finish { 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 specular .8} }
MONKEY_SADDLE_CUBIC="1,0,0,0,-3,0,0,0,0,0,0,0,0,0,0,0,-1,0"
ECKARDT_CUBIC_DEFORM1_LEX="0, 10, 0, -8, 10, 25, 2, 0, -20, -8, -20, -10, -24, 10
, -2, 12, 0, -8, 8, 16"
1023
1024 ECKARDT_CUBIC_DEFORM2_LEX="0, -5, 0, -5, -5, 10, -1, 0, 10, 4, 10, 5, 3, -5, 1, -6, 0, -5, -4, 1"
1025
1026 KUMMER_QUARTIC_LEX_35="-2,0,0,0,2,0,0,2,0,2,0,0,0,-2,0,2,0,-2"
1027
1028 BEAUVILLE_QUINTIC_LEX_56="-44, 228, 400, 315, -396, -852, \n1029 -512, -553, -1050, -354, 284, 504, -62, -707, -1390, -1010, \n1030 281, -167, -1644, -1024, -72, -196, 192, 373, 322, 78, 150, \n1031 966, 1540, 348, -475, -492, 1063, 1550, 390, 0, 96, 3, -337, \n1032 -426, -66, 425, 673, -156, -216, -223, -60, 1543, 1998, 618, \n1033 263, -250, -919, 557, 1800, 741"
1034
1035 ENDRASS_OCTIC_LEX_165="-93.2548,0,0,0,-309.019,0,0,527.529,0,395.647,\n1036 0,0,0,0,0,0,0,0,-687.529,0,0,1582.59,0,1186.94,0,0,0,0,-1055.06,0,\n1037 -1582.59,0,-593.47,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-309.019,\n1038 0,0,1582.59,0,1186.94,0,0,0,0,-2110.12,0,-3165.17,0,-1186.94,0,0,0,0,\n1039 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,\n1040 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,\n1041 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-1055.06,0,-1582.59,0,-593.47,0,\n1042 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,\n1043 1677.92,0,343.362,0,0,0,0,0,0,0,0,-256,0,-468.077,0,-789.019,0,\n1044 -525.726,0,0,0.941125"
1045
1046
1047
1048
1049
1050
1051 # Chapter 2 - Getting Started
1052 # Section 2.2: Orbiter Session
1053 SECTION_ORBITER_SESSION:
1054 test:
1055 $(ORBITER)
# Section 2.3: Makefiles and Shell Scripts

SECTION MAKEFILES AND SHELL SCRIPTS:

# Section 2.4: Objects and Activities

SECTION OBJECTS AND ACTIVITIES:

example set:

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

object F 2:

```
object F 2:
  $(ORBITER) -define F -finite_field -q 2 -end
```

object PG 3 2:

```
object PG 3 2:
  $(ORBITER) -define F -finite_field -q 2 -end -define P -projective_space 3 F -end
```

vector ex:

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

# Section 2.5: Mathematical Data

SECTION MATHEMATICAL_DATA:
create_BLT_5_1:
\>$\text{\textcolor{Red}{ORBITER}}$ -v 2 \ 
\>$\text{\textcolor{Red}{\define F -finite\_field -q 5 -end}}$ \ 
\>$\text{\textcolor{Red}{\define O -orthogonal\_space 0 5 F -end}}$ \ 
\>$\text{\textcolor{Red}{\with O -do \ orthogonal\_space -activity}}$ \ 
\>$\text{\textcolor{Red}{\create BLT\_set -catalogue 1 -end}}$ \ 
$\text{\textcolor{Red}{\end}}$

create_surface_4_0:
\>$\text{\textcolor{Red}{ORBITER}}$ -v 3 \ 
\>$\text{\textcolor{Red}{\define F -finite\_field -q 4 -end}}$ \ 
\>$\text{\textcolor{Red}{\define P -projective\_space 3 F -end}}$ \ 
\>$\text{\textcolor{Red}{\with P -do \ projective\_space -activity}}$ \ 
\>$\text{\textcolor{Red}{\define surface S4_0 -q 4 -catalogue 0 -end}}$ \ 
\>$\text{\textcolor{Red}{\end}}$

# Section 2.6: Set Builder

SECTION_SET_BUILDER:

set_of_primes:
\>$\text{\textcolor{Red}{ORBITER}}$ -v 2 \ 
\>$\text{\textcolor{Red}{\define S -set \-here "$2,3,5,7,11,13" \-end}}$ \ 
\>$\text{\textcolor{Red}{\print symbols}}$ \ 

set_interval:
\>$\text{\textcolor{Red}{ORBITER}}$ -v 2 -define S -set -loop 0 64 1 -end \ 
\>$\text{\textcolor{Red}{\print_symbols}}$ \ 

# Section 2.7: Vector Builder

SECTION_VECTOR_BUILDER:
vector_example1:
$\text{ORBITE}\text{R} -v 2 \ \ \ \n\text{-define F -finite_field -q 5 -end} \ \ \ \n\text{-define v -vector -field F -dense "0,1,2,3,4" -end} \ \ \ \n\text{-print_symbols}$

vector_example2:
$\text{ORBITE}\text{R} -v 2 \ \ \ \n\text{-define F -finite_field -q 5 -end} \ \ \ \n\text{-define v -vector -field F -format 2 -dense "0,1,2,3,4,0" -end} \ \ \ \n\text{-print_symbols}$

vector_example_sparse:
$\text{ORBITE}\text{R} -v 2 \ \ \ \n\text{-define F -finite_field -q 5 -end} \ \ \ \n\text{-define v -vector -field F -format 4 -sparse 20 "1,0,1,19" -end} \ \ \ \n\text{-print_symbols}$

vector_example_repeat:
$\text{ORBITE}\text{R} -v 2 \ \ \ \n\text{-define F -finite_field -q 5 -end} \ \ \ \n\text{-define v -vector -repeat "0,1,2,3" 11 -end} \ \ \ \n\text{-print_symbols}$

vector_example_all_one_11:
$\text{ORBITE}\text{R} -v 2 \ \ \ \n\text{-define v -vector -repeat 1 11 -end} \ \ \ \n\text{-print_symbols}$

matrix_example1:
$\text{ORBITE}\text{R} -v 2 \ \ \ \n\text{-define F -finite_field -q 2 -end} \ \ \ \n\text{-define v -vector -field F -format 4} \ \ \ \n\text{-dense $(\text{HAMMING\_CODE\_GENERATOR}) -end} \ \ \ \n\text{-print_symbols}$

matrix_example_co_1:
$\text{ORBITE}\text{R} -v 2 \ \ \ \n\text{-define F -finite_field -q 2 -end} \ \ \ \n\text{-define v -vector -field F -format 22} \ \ \ \n\text{-compact $(\text{CONWAY\_GEN1}) -end}$
# Chapter 3 - Basic Algebra

# Section 3.1: Basic Number Theory

PR29:
\[ \text{$(ORBITER) -v 1 -smallest\ primitive\ root\ 29} \]

PR31:
\[ \text{$(ORBITER) -v 1 -smallest\ primitive\ root\ 31} \]

PR37:
\[ \text{$(ORBITER) -v 1 -smallest\ primitive\ root\ 37} \]

PR_100:
\[ \text{$(ORBITER) -v 1 -smallest\ primitive\ root\ interval\ 2\ 100} \]

# randomized algo:

PR_915839:
\[ \text{$(ORBITER) -v 5 -primitive\ root\ 915839} \]

# a primitive root modulo 915839 is 43085

PR_915839_check:
\[ \text{$(ORBITER) -v 5 -power\ mod\ 43085\ 49842\ 915839} \]

# the power of 43085 to the 49842 mod 915839 is 487320

DL_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:
$($(ORBITER) -v 2 -square_root_mod 5 11$

sqrt_5_mod_19:
$($(ORBITER) -v 2 -square_root_mod 5 19$

sqrt_mod_20_31:
$($(ORBITER) -v 2 -square_root_mod 20 31$

order_of_2_mod_n:
$($(ORBITER) -v 3 -order_of_q_mod_n 2 3 151$

$($(ORBITER) -v 1 -csv_file_latex 1$

$($(ORBITER) -v order_of_q_mod_n_q2_3_151.csv$

pdflatex order_of_q_mod_n_q2_3_151.tex

open order_of_q_mod_n_q2_3_151.pdf

open order_of_q_mod_n_q2_3_151.pdf
Eulerfunction_150:

$\text{(ORBITER)} -v 1 -eulerfunction\_interval \text{1 150}$

$\text{(ORBITER)} -v 1 -csv\_file\_latex \text{1 \textbackslash} \text{\textbackslash}$

$\text{pdflatex table\_eulerfunction\_1\_150.tex}$

$\text{open table\_eulerfunction\_1\_150.pdf}$

PR_1000:

$\text{(ORBITER)} -v 1 -\text{smallest\_primitive\_root\_interval} \text{2 1000}$

$\text{(ORBITER)} -v 1 -csv\_file\_latex \text{1 \textbackslash} \text{\textbackslash}$

$\text{pdflatex primitive\_element\_table\_2\_1000.csv}$

$\text{open primitive\_element\_table\_2\_1000.pdf}$

PE\_number\_1000:

$\text{(ORBITER)} -v 1 -\text{number\_of\_primitive\_roots\_interval} \text{2 1000}$

$\text{(ORBITER)} -v 1 -csv\_file\_latex \text{1 table\_number\_of\_pe\_2\_1000.csv}$

$\text{pdflatex table\_number\_of\_pe\_2\_1000.tex}$

$\text{open table\_number\_of\_pe\_2\_1000.pdf}$

Eulerfunction\_10000:

$\text{(ORBITER)} -v 1 -\text{number\_of\_primitive\_roots\_interval} \text{10000 10001}$

power\_function\_2\_mod\_11:

$\text{(ORBITER)} -v 5 -\text{power\_function\_mod\_n} \text{2 11}$

$\text{(ORBITER)} -v 1 -csv\_file\_latex \text{1 power\_function\_k2\_n11.csv}$

$\text{pdflatex power\_function\_k2\_n11.tex}$

$\text{open power\_function\_k2\_n11.pdf}$

draw\_mod\_13:

$\text{(ORBITER)} -v 2 \text{\textbackslash}$

$\text{-draw\_options -embedded -end \textbackslash}$

$\text{-draw\_mod\_n -n 13 -file mod\_13 -power\_cycle 2 -end}$

$\text{pdflatex mod\_13\_draw.tex}$

$\text{open mod\_13\_draw.pdf}$

# Section 3.2: Prime Fields
SECTION_PRIME_FIELDS:

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

F_3:
$(ORBITER) -v 3 \ 
-define F -finite_field -q 3 -end \ 
-with F -do -finite_field_activity -cheat_sheet_GF -end
#pdflatex GF_3.tex
#open GF_3.pdf

F_5:
$(ORBITER) -v 3 \ 
-define F -finite_field -q 5 -end \ 
-with F -do -finite_field_activity -cheat_sheet_GF -end
pdflatex GF_5.tex
open GF_5.pdf

F_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

506
F_101_wo:

$\text{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:

$\text{ORBITER} -v 3 \$

-define $F$ -finite_field -q 1021 -without_tables -end 

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

pdflatex GF_4.tex 

open GF_4.pdf 

F_4:

$\text{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 

F_4_tables:

$\text{ORBITER} -v 3 \$

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

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

-draw_matrix -input_csv_file GF_q4_addition_table.csv 

-box_width 40 -bit_depth 24 -partition 3 4 4 -end 

-draw_matrix -input_csv_file GF_q4_multiplication_table.csv 

-box_width 40 -bit_depth 24 -partition 3 3 3 -end 

-pdflatex GF_4.tex 

-open GF_4.pdf 

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

\textbf{F$_{16}$ tables:}

\begin{enumerate}
\item $(\text{ORBITER}) -v 3$
\item -define F -finite_field -q 16 -end
\item -with F -do -finite_field_activity -cheat_sheet_GF -end
\item pdflatex GF
\end{enumerate}

\begin{enumerate}
\item -draw_matrix \input{GF_q16_addition_table.csv}
\item -box_width 40 -bit_depth 24 -partition 3 16 16 -end
\item $(\text{ORBITER}) -v 2$
\item -draw_matrix \input{GF_q16_addition_table_reordered.csv}
\item -box_width 40 -bit_depth 24 -partition 3 16 16 -end
\item $(\text{ORBITER}) -v 2$
\item -draw_matrix \input{GF_q16_multiplication_table.csv}
\item -box_width 40 -bit_depth 24 -partition 3 15 15 -end
\item $(\text{ORBITER}) -v 2$
\item -draw_matrix \input{GF_q16_multiplication_table_reordered.csv}
\item -box_width 40 -bit_depth 24 -partition 3 15 15 -end
\end{enumerate}

\begin{enumerate}
\item \textbf{SECTION LINEAR ALGEBRA:}
\item RREF:
\item $(\text{ORBITER}) -v 2$
\item -define F -finite_field -q 2 -end
\item -define v -vector -field F -format 2
\item -dense "1,1,1,1,0,1,1,0,0,1"
\item -end
\item -with F -do -finite_field_activity
\item -RREF v -normalize_from_the_right
\item -end
\end{enumerate}
RREF_V7:

\[(\text{ORBITER}) -v 2 \]
\[\text{define F} -\text{finite field} -q 7 -\text{end} \]
\[\text{define V} -\text{vector} -\text{format} 7 \]
\[\text{dense } \$(V7\_VANDERMONDE\_EXTENDED) \]
\[\text{end} \]
\[\text{with F} -\text{do} -\text{finite field activity} \]
\[\text{RREF V} \]

nullspace:

\[(\text{ORBITER}) -v 2 \]
\[\text{define F2} -\text{finite field} -q 2 -\text{end} \]
\[\text{define v} -\text{vector} -\text{field} F2 -\text{format} 2 \]
\[\text{dense } "1,1,1,0,1,1,0,0,1" \]
\[\text{end} \]
\[\text{with F2} -\text{do} \]
\[\text{finite field activity} \]
\[\text{nullspace v} \]
\[\text{normalize from the right} \]
\[\text{end} \]

eigenstuff:

\[(\text{ORBITER}) -v 6 \]
\[\text{define F} -\text{finite field} -q 5 -\text{end} \]
\[\text{eigenstuff F} 4 "0,1,0,2,0,1,2,1,4,2,3,1,2,0,4,3" \]

classes_GL3_2:

\[(\text{ORBITER}) -v 7 \]
\[\text{define F} -\text{finite field} -q 2 -\text{end} \]
\[\text{all rational normal forms} F 3 \]
\[\text{pdflatex Class_reps_GL3_2.tex} \]
\[\text{open Class_reps_GL3_2.pdf} \]

classes_GL4_2:

\[(\text{ORBITER}) -v 7 \]
\[\text{define F} -\text{finite field} -q 2 -\text{end} \]
\[\text{all rational normal forms} F 4 \]
\[\text{pdflatex Class_reps_GL4_2.tex} \]
\[\text{open Class_reps_GL4_2.pdf} \]
1538 # 252 classes
1539
1540
1541
1542 RREF.demo_4_4_q5:
1543 \$\text{ORBITER} -v 2 \$
1544 \triangleright \triangleright -\text{define } F -\text{finite\_field} -q 5 -\text{end} \$
1545 \triangleright \triangleright -\text{with } F -\text{do} \$
1546 \triangleright \triangleright -\text{finite\_field\_activity} -\text{RREF.demo 4 4 -end}
1547 pdflatex RREF.example.q5_4_4.tex
1548 \#open RREF.example.q5_4_4.pdf
1549 gs -sDEVICE=png16 -dFIXEDMEDIA \$
1550 \triangleright \triangleright -dDEVICEWIDTHPOINTS=500 -dDEVICEHEIGHTPOINTS=450 \$
1551 \triangleright \triangleright -r240 -oRREF.example.q5_4_4_page%02d.png \$
1552 \triangleright \triangleright RREF.example.q5_4_4.pdf
1553
1554 RREF.demo_4_6_q7:
1555 \$\text{ORBITER} -v 2 \$
1556 \triangleright \triangleright -\text{define } F -\text{finite\_field} -q 7 -\text{end} \$
1557 \triangleright \triangleright -\text{with } F -\text{do} \$
1558 \triangleright \triangleright -\text{finite\_field\_activity} -\text{RREF.random\_matrix 4 6 -end}
1559 pdflatex RREF.example.q7_4_6.tex
1560 gs -sDEVICE=png16 -dFIXEDMEDIA \$
1561 \triangleright \triangleright -dDEVICEWIDTHPOINTS=500 -dDEVICEHEIGHTPOINTS=450 \$
1562 \triangleright \triangleright -r240 -oRREF.example.q7_4_6_page%02d.png \$
1563 \triangleright \triangleright RREF.example.q7_4_6.pdf
1564 \triangleright \triangleright open RREF.example.q7_4_6.pdf
1565
1566 RREF.demo_4_8_q8:
1567 \$\text{ORBITER} -v 2 \$
1568 \triangleright \triangleright -\text{define } F -\text{finite\_field} -q 8 -\text{end} \$
1569 \triangleright \triangleright -\text{with } F -\text{do} \$
1570 \triangleright \triangleright -\text{finite\_field\_activity} -\text{RREF.random\_matrix 4 8 -end}
1571 pdflatex RREF.example.q8_4_8.tex
1572 \#open RREF.example.q8_4_8.pdf
1573 gs -sDEVICE=png16 -dFIXEDMEDIA \$
1574 \triangleright \triangleright -dDEVICEWIDTHPOINTS=500 -dDEVICEHEIGHTPOINTS=450 \$
1575 \triangleright \triangleright -r240 -oRREF.example.q8_4_8_page%02d.png \$
1576 \triangleright \triangleright RREF.example.q8_4_8.pdf
RREF_RS_6_4_7:

$\text{define } F = \text{finite field } -q 7 \text{ -end }$

$\text{define } v = \text{vector } -\text{field } F \text{ -format 4 }$

$\text{dense } (\text{CODE_RS_6_4_7})$

$\text{end }$

$\text{with } F \text{ -do }$

$\text{finite field activity } -\text{RREF } v \text{ -end }$

$\text{pdflatex RREF_example_q7_4_6.tex}$

$\text{gs -sDEVICE=png16 -dFIXEDMEDIA }$

$\text{-dDEVICEWIDTHPOINTS=500 -dDEVICEHEIGHTPOINTS=450 }$

$\text{-r240 -oRREF_example_q7_4_6_page%02d.png }$

$\text{RREF_example_q7_4_6.pdf}$

#############################################################################
# Section 3.5: Advanced Topics in finite fields

SECTION_ADVANCED_TOPICS_INFINITE_FIELDS:

normal_basis_2_3:

$\text{define } F = \text{finite field } -q 2 \text{ -end }$

$\text{with } F \text{ -do } \text{finite field activity }$

$\text{-normal_basis 3 -end }$

normal_basis_2_6:

$\text{define } F = \text{finite field } -q 2 \text{ -end }$

$\text{with } F \text{ -do } \text{finite field activity }$

$\text{-normal_basis 6 -end }$

F8_over_F2_field_reduction:

$\text{define } F = \text{finite field } -q 8 \text{ -end }$
F_64_over_F8_field_reduction:

$\texttt{F64\_over\_F8\_field\_reduction}$

$\texttt{\$(ORBITER) -v 2 \}$

$\texttt{\$\texttt{ORBITER} -v 2 \}$

$\texttt{-define F -finite\_field -q 64 -end}$

$\texttt{-define elts -vector -field F -loop 0 64 1 -end}$

$\texttt{-with F -do}$

$\texttt{-finite\_field\_activity -field\_reduction "F64\_over\_F8" 8 8 8}$

$\texttt{-els -end}$

$\texttt{\$(ORBITER) -v 2 -draw\_matrix \}$

$\texttt{-input\_csv\_file F64\_over\_F8.csv \}$

$\texttt{-box\_width 40 -bit\_depth 24 \}$

$\texttt{-partition 4 3 3 -end}$

$\texttt{open F64\_over\_F8\_draw.bmp}$

$\texttt{\#pdflatex field\_reduction\_Q64\_q8\_8\_8.tex}$

$\texttt{\#open field\_reduction\_Q64\_q8\_8\_8.pdf}$

$\texttt{\#open field\_reduction\_Q64\_q8\_8\_8.pdf}$

F_64_over_F4_field_reduction:

$\texttt{F64\_over\_F4\_field\_reduction}$

$\texttt{\$(ORBITER) -v 2 \}$

$\texttt{-define F -finite\_field -q 64 -end}$

$\texttt{-define elts -vector -field F -loop 0 64 1 -end}$

$\texttt{-with F -do}$

$\texttt{-finite\_field\_activity \}$

$\texttt{-field\_reduction "F64\_over\_F4" 4 8 8 elts -end}$

$\texttt{\$(ORBITER) -v 2 -draw\_matrix \}$

$\texttt{-input\_csv\_file F64\_over\_F4.csv \}$

$\texttt{-box\_width 40 -bit\_depth 24 \}$

$\texttt{-partition 4 3 3 3 3 3 3 3 -end}$

$\texttt{open F64\_over\_F4\_draw.bmp}$

$\texttt{\#pdflatex field\_reduction\_Q64\_q4\_8\_8.tex}$

$\texttt{\#open field\_reduction\_Q64\_q4\_8\_8.pdf}$

$\texttt{\#open field\_reduction\_Q64\_q4\_8\_8.pdf}$
F64_over_F2_field_reduction:
$ (ORBITER) -v 2 \n-define F -finite_field -q 64 -end \n-define elts -vector -field F -loop 0 64 1 -end \n-with F -do \n-finite_field_activity \n-field_reduction "F64_over_F2" 2 8 8 elts -end
$ (ORBITER) -v 2 -draw_matrix \n-input_csv_file F64_over_F2.csv \n-box_width 40 -bit_depth 24 \n-partition 4 "6,6,6,6,6,6,6,6" "6,6,6,6,6,6,6,6" -end
open F64_over_F2_draw.bmp
#pdflatex field_reduction_Q64_q2_8_8.tex
#open field_reduction_Q64_q2_8_8.pdf

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

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

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

Polynomial_ring:

SECTION FINITE_PROJECTIVE_SPACES:
1773
1774
1775 PG_3_2_easy:
1776 $\text{ORBITER} -v 2 \$
1777 \triangleright -\text{define } F -\text{finite_field } -q 2 -\text{-end } \$
1778 \triangleright -\text{define } P -\text{projective_space } 3 F -\text{-end }
1779
1780
1781
1782
1783 PG_1_16:
1784 $\text{ORBITER} -v 2 \$
1785 \triangleright -\text{define } F -\text{finite_field } -q 16 -\text{-end } \$
1786 \triangleright -\text{define } P -\text{projective_space } 1 F -\text{-end } \$
1787 \triangleright -\text{with } P -\text{-do } -\text{projective_space_activity } \$
1788 \triangleright -\text{cheat_sheet } \$
1789 \triangleright -\text{-end }
1790 \triangleright \text{pdflatex PG_1_16.tex}
1791 \triangleright \text{open PG_1_16.pdf}
1792
1793
1794 PG_2_4:
1795 $\text{ORBITER} -v 2 \$
1796 \triangleright -\text{define } F -\text{finite_field } -q 4 -\text{-end } \$
1797 \triangleright -\text{define } P -\text{projective_space } 2 F -\text{-end } \$
1798 \triangleright -\text{with } P -\text{-do } -\text{projective_space_activity } \$
1799 \triangleright -\text{cheat_sheet } \$
1800 \triangleright -\text{-end }
1801 \triangleright \text{pdflatex PG_2_4.tex}
1802 \triangleright \text{open PG_2_4.pdf}
1803
1804
1805
1806
1807 PG_2_13:
1808 $\text{ORBITER} -v 2 \$
1809 \triangleright -\text{define } F -\text{finite_field } -q 13 -\text{-end } \$
1810 \triangleright -\text{define } P -\text{projective_space } 2 F -\text{-end } \$
1811 \triangleright -\text{with } P -\text{-do } -\text{projective_space_activity } \$
1812 \triangleright -\text{cheat_sheet } \$
1813 \triangleright -\text{-end }
1814 \triangleright \text{pdflatex PG_2_13.tex}
1815 \triangleright \text{open PG_2_13.pdf}
1816
1817
1818
1819

515
PG_2.64:
\$(ORBITER) -v 2 \ 
\$define F -finite_field -q 64 -end \ 
\$define P -projective_space 2 F -end \ 
\$with P -do -projective_space_activity \ 
\$cheat_sheet \ 
\$end

open PG_2.64.pdf

PG_3.2:
\$(ORBITER) -v 2 \ 
\$define F -finite_field -q 2 -end \ 
\$define P -projective_space 3 F -end \ 
\$with P -do -projective_space_activity \ 
\$cheat_sheet \ 
\$end

open PG_3.2.pdf

PG_3.4:
\$(ORBITER) -v 2 \ 
\$define F -finite_field -q 4 -end \ 
\$define P -projective_space 3 F -end \ 
\$with P -do -projective_space_activity \ 
\$cheat_sheet \ 
\$end

open PG_3.4.pdf

PG_3.5:
\$(ORBITER) -v 2 \ 
\$define F -finite_field -q 5 -end \ 
\$define P -projective_space 3 F -end \ 
\$with P -do -projective_space_activity \ 
\$cheat_sheet \ 
\$end

open PG_3.5.pdf

PG_3.7:
\$(ORBITER) -v 2 \ 
\$define F -finite_field -q 7 -end \ 

516
Section 4.2: Indexing Points

PG_4_3:
$\text{define } F \text{ -finite_field } -q 3 \text{ -end }$
$\text{define } P \text{ -projective_space } 4 F \text{ -end }$
$\text{with } P \text{ -do } \text{-projective_space_activity }$
$\text{-cheat_sheet }$
$\text{-end}$
$\text{pdflatex PG_4_3.tex}$
$\text{open PG_4_3.pdf}$

PG_8_2:
$\text{define } F \text{ -finite_field } -q 2 \text{ -end }$
$\text{define } P \text{ -projective_space } 8 F \text{ -end }$
$\text{with } P \text{ -do } \text{-projective_space_activity }$
$\text{-cheat_sheet }$
$\text{-end}$
$\text{pdflatex PG_8_2.tex}$
$\text{open PG_8_2.pdf}$

# Section 4.2: Indexing Points

SECTION_INDEXING_POINTS:

PG_2_4_rank_point:
$\text{define } F \text{ -finite_field } -q 4 \text{ -end }$
$\text{with } F \text{ -do } \text{-finite_field_activity }$
$\text{-rank_point_in_PG } 2 \text{ "3,3,1" -end}$
$\text{geometry_global::do_rank_point_in_PG coeff: (3,3,1) has rank 20}$

elliptic_curve.b1.c3.q11.txt:
$\text{define } F \text{ -finite_field } -q 11 \text{ -end }$
$\text{define } P \text{ -projective_space } 2 F \text{ -end }$
$\text{define } EC \text{ -geometric_object } P$
$\text{-elliptic_curve 1 3 }$
$\text{-end }$
PG_2.2.incidence_matrix:
$\text{\texttt{ORBITER}} -v 2 \$
$\text{\texttt{define F -finite_field -q 2 \text{-end \}}}$
$\text{\texttt{define P -projective_space 2 F \text{-end \}}}$
$\text{\texttt{with P \text{-do -projective_space_activity \}}}$
$\text{\texttt{\text{-export_point_line_incidence_matrix \}}}$
$\text{\texttt{\text{-end \}}}$
$\text{\texttt{\text{-end \}}}$
$\text{\texttt{\text{-end \}}}$
$\text{\texttt{\text{-end \}}}$
PG_2.2.incidence_matrix:
$\text{\texttt{\text{-v 2 \}}}$
$\text{\texttt{\text{-define F -finite_field -q 2 \text{-end \}}}$
$\text{\texttt{\text{-define P -projective_space 2 F \text{-end \}}}$
$\text{\texttt{\text{-with P \text{-do -projective_space_activity \}}}$
$\text{\texttt{\text{-export_point_line_incidence_matrix \}}}$
$\text{\texttt{\text{-end \}}}$
PG_2.2.incidence_matrix:
$\text{\texttt{\text{-define F -finite_field -q 4 \text{-end \}}}$
$\text{\texttt{\text{-define P -projective_space 2 F \text{-end \}}}$
$\text{\texttt{\text{-with P \text{-do -projective_space_activity \}}}$
$\text{\texttt{\text{-export_point_line_incidence_matrix \}}}$
$\text{\texttt{\text{-end \}}}$
PG_2.2.incidence_matrix:
$\text{\texttt{\text{-v 2 \}}}$
$\text{\texttt{\text{-define F -finite_field -q 8 \text{-end \}}}$
2005 PG_2.8.incidence_matrix:
$\text{\texttt{\text{-v 2 \}}}$
$\text{\texttt{\text{-define F -finite_field -q 8 \text{-end \}}}$
SECTION FINITE DESARGUESIAN PROJECTIVE PLANES:

PG_2.16:

PG_2.4 with decomposition:
```
\$ -cheat_sheet_for_decomposition_by_element_PG \\
1 "0,1,0, 0,0,1, 2,1,1, 0" \\
"PG_2.4_singer" \\
-end

pdflatex PG_2.4_singer.tex
open PG_2.4_singer.pdf

#PG_2.4_singer_incma_cyclic.csv
#PG_2.4_singer_incma_subgroup_index_3.csv
#PG_2.4_singer_incma_subgroup_index_7.csv

PG_2.4_incma_cyclic:
$ (ORBITER) -v 2 \\
-list_arguments \\
-define R -vector -repeat 1 21 -end \\
-define C -vector -repeat 1 21 -end \\
-draw_matrix \\
-input_csv_file PG_2.4_singer_incma_cyclic.csv \\
-box_width 40 -bit_depth 24 \\
-partition 3 R C \\
-end
open PG_2.4_singer_incma_cyclic_draw.bmp

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

PG_2.4_incma_singer_sub_7:
$ (ORBITER) -v 2 \\
-draw_matrix \\
-input_csv_file PG_2.4_singer_incma_subgroup_index_7.csv \\
-box_width 20 -bit_depth 24 \\
-partition 3 3,3,3,3,3,3 3,3,3,3,3,3 -end
open PG_2.4_singer_incma_subgroup_index_7_draw.bmp
```
# Section 4.4: The Grassmannian

---

SECTION GRASSMANNIAN:

GR\_3\_2\_2:

```
$\texttt{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
```

rank_lines:

```
$\texttt{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 \(1,0,0,0,0,1,0,0,0,0,0,0,1,0,1,0,0,0,0,2,1)\) \
end \
define F -finite_field -q 3 -end \
define P -projective_space 3 F -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
```

planes_in_pencil:

```
$\texttt{ORBITER} -v 2 \
```
SECTION_ALGEBRAIC_SETS:

EC_11.txt:

```plaintext
define F -finite_field -q 11 -end \ndefine P -projective_space 3 F -end \nwith P -do \n-planes_through_line 0 \nend
```

Hirschfeld_surface_q4.txt:

```plaintext
define F -finite_field -q 4 -end \ndefine R -polynomial_ring -field F \nnumber_of_variables 4 \nhomogeneous_degree 3 \nend \ndefine P -projective_space 3 F -end \ndefine H4 -geometric_object P \nend
```

# Section 4.5: Algebraic Sets
"Hirschfeld\_surface\_q4" \n"Hirschfeld\_surface\_q4" \n$(HIRSCHFELD\_SURFACE\_EQUATION) \n-end \n-with H4 -do -combinatorial\_object\_activity -save \n-end

Hirschfeld\_surface\_q16.txt:

$\text{define } F \text{-finite field } q 16 \text{-end }$
$\text{-define } R \text{-polynomial ring } field F$
$\text{-number of variables 4 }$
$\text{-homogeneous of degree 3 }$
$\text{-end }$
$\text{-define } P \text{-projective space 3 } F \text{-end }$
$\text{-define } H16 \text{-geometric object } P$
$\text{-define } H16 \text{-projective variety } R$
$\text{end }$
$\text{-with } H16 -do -combinatorial\_object\_activity -save \n-end$

# the coefficient vector is given as a list of pairs.
# 165 = binomial(11,3)

Endrass\_F7.txt:

$\text{define } F \text{-finite field } q 7 \text{-end }$
$\text{-define } R \text{-polynomial ring } field F$
$\text{-number of variables 4 }$
$\text{-homogeneous of degree 8 }$
$\text{-end }$
$\text{-define } eqn \text{-vector } field F \text{-sparse 165 }$
$\text{-define } eqn \text{-sparse 165 }$
$\text{-define } P \text{-projective space 3 } F \text{-end }$
$\text{-define } Endrass\_F7 \text{-geometric object } P$
$\text{-define } Endrass\_F7 \text{-projective variety } R$
$\text{-define } Endrass\_F7 \text{-projective variety } R$
$\text{-define } Endrass\_F7 \text{-projective variety } R$
$\text{-define } eqn \text{-sparse 165 }$
$\text{-define } eqn \text{-sparse 165 }$
$\text{-define } eqn \text{-sparse 165 }$
# we created a set of 33 points, called Endrass_F7.txt

octic_prepare:

```
$(ORBITER) -v 4 \
-define A -vector -format 1 -dense "1,1,1,1" -end \
-define D -diophant \
-label octic_monomials \
-coefficient_matrix A \
-RHS "8,8,1" \
-x_min_global 0 -x_max_global 8 \
-end \
-with D -do \
-diophant_activity -solve_mckay \
-end \
-sort -r octic_monomials.sol >octic_monomials_sorted.txt
```

#Found 165 solutions with 210 backtrack steps

# 165=binomial(11,3)

```
# Section 4.6: The Klein Quadric and Pluecker coordinates

SECTION_KLEIN_QUADRIC_AND_PLUECKERCOORDINATES:
```

GR_4_2_2:

```
$(ORBITER) -v 2 \
-define F -finite_field -q 2 -end \
-with F -do -finite_field_activity \
-cheat_sheet_Gr 4 2 -end \
pdflatex Gr_4_2_2.tex \
open Gr_4_2_2.pdf
```
Section 4.7: Orthogonal spaces

Op_4.2:

```bash
$ORBITER -v 2 \
$define F -finite_field -q 2 -end \
$define O -orthogonal_space 1 4 F -without_group -end \
$with O -do -orthogonal_space_activity \
$cheat_sheet_orthogonal -end
```

```
pdflatex 0.1_4.2_report.tex
open 0.1_4.2_report.pdf
```

Op_5.2_incidence_matrix.csv:

```bash
$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
```

```
pdflatex O_5.2_incidence_matrix.csv \
box_width 20 -bit_depth 8 \
partition 2 \
all_one_r all_one_c \
end
```

```
open O_5.2_incidence_matrix_draw.bmp
```

#O(5,2) projectively = Q(4,2) = (dual of) W(3,2) = W(3,2)

# recall that W(3,2) and Q(4,q) are self dual if q is even

Op_6.2:

```bash
$ORBITER -v 2 \
$define F -finite_field -q 2 -end \
$define O -orthogonal_space 1 6 F -without_group -end \
$with O -do -orthogonal_space_activity \
$cheat_sheet_orthogonal -end
```
Op_6.2.incidence_matrix.csv:

\$(\text{ORBITER}) -v 2 \$
\> -define F -finite_field -q 2 -end \n\> -define 0 -orthogonal_space 1 6 F -without_group -end \n\> -with 0 -do -orthogonal_space_activity \n\> -export_point_line_incidence_matrix \n\> -end
\> \$(\text{ORBITER}) -v 2 \$
\> -define all_one_r -vector -repeat 1 35 -end \n\> -define all_one_c -vector -repeat 1 105 -end \n\> -draw_matrix \n\> \> -input_csv_file Op_6.2.incidence_matrix.csv \n\> \> -box_width 20 -bit_depth 8 \n\> \> -partition 2 \n\> \> \> all_one_r all_one_c \n\> \> -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 \n\> \> -define 0 -orthogonal_space 1 6 F -end \n\> \> -with 0 -do -orthogonal_space_activity \n\> \> \> -cheat_sheet_orthogonal -end
\> \> pdflatex 0.1_6.2_report.tex
\> \> open 0.1_6.2_report.pdf
\> # problem:
\> # error message:
\> #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 \n\> \> -define 0 -orthogonal_space 1 6 F -without_group -end \n\> \> -with 0 -do -orthogonal_space_activity \n\> \> \> -cheat_sheet_orthogonal \n\> \> -end
\> \> pdflatex 0.1_6.8_report.tex
\> \> open 0.1_6.8_report.pdf
Op_8.2:

```latex
$(\textsc{Orbiter})$ -v 2 \\
$\texttt{-define \ F \ finite\_field \ -q \ 2 \ -end}$ \\
$\texttt{-define \ 0 \ -orthogonal\_space \ 1 \ 8 \ \texttt{F} \ -without\_group \ -end}$ \\
$\texttt{-with \ 0 \ -do \ -orthogonal\_space\_activity}$ \\
$\texttt{-cheat\_sheet\_orthogonal \ -end}$ \\
\texttt{pdflatex \ 0.1_8_2_report.tex} \\
\texttt{open \ 0.1_8_2_report.pdf}
```

Op_6.64:

```latex
$(\textsc{Orbiter})$ -v 2 \\
$\texttt{-define \ F \ finite\_field \ -q \ 64 \ -end}$ \\
$\texttt{-define \ 0 \ -orthogonal\_space \ 1 \ 6 \ \texttt{F} \ -without\_group \ -end}$ \\
$\texttt{-with \ 0 \ -do \ -orthogonal\_space\_activity}$ \\
$\texttt{-cheat\_sheet\_orthogonal \ -end}$ \\
\texttt{pdflatex \ 0.1_6_64_report.tex} \\
\texttt{open \ 0.1_6_64_report.pdf}
```

# problem, because we are trying to create $\text{PGL}(6,64)$:

Op_6.64.line_rank_problem:

```latex
$(\textsc{Orbiter})$ -v 4 \\
$\texttt{-define \ F \ finite\_field \ -q \ 64 \ -end}$ \\
$\texttt{-define \ 0 \ -orthogonal\_space \ 1 \ 6 \ \texttt{F} \ -end}$ \\
$\texttt{-with \ 0 \ -do \ -orthogonal\_space\_activity}$ \\
$\texttt{-unrank\_line\_through\_two\_points \ 15447347 \ 15225451}$ \\
$\texttt{-end}$
```

# use option -without_group to skip the group. This will work:

Op_6.64.line_rank:

```latex
$(\textsc{Orbiter})$ -v 4 \\
$\texttt{-define \ F \ finite\_field \ -q \ 64 \ -end}$ \\
$\texttt{-define \ 0 \ -orthogonal\_space \ 1 \ 6 \ \texttt{F} \ -without\_group \ -end}$ \\
$\texttt{-with \ 0 \ -do \ -orthogonal\_space\_activity}$ \\
$\texttt{-unrank\_line\_through\_two\_points \ 15447347 \ 15225451}$ \\
$\texttt{-end}$
```

# this will create a basic report without the group:

Op_6.64.report:

```latex
$(\textsc{Orbiter})$ -v 4 \\
$\texttt{-define \ F \ finite\_field \ -q \ 64 \ -end}$ \\
$\texttt{-define \ 0 \ -orthogonal\_space \ 1 \ 6 \ \texttt{F} \ -without\_group \ -end}$
```
Section 4.8: Hermitian varieties

H_{2,4}:

H_{2,9}:

H_{3,4}:
# H_3.4 = the Hirschfeld surface

# Section 4.9: Projective Space Advanced Topics

SECTION_PROJECTIVE_SPACE_ADVANCED_TOPICS:

fix_structure_2A:

```bash
$ (ORBITER) \( \text{-v} \ 2 \ \text{\textbackslash \textbackslash} \\
\text{\textbackslash \textbackslash} \text{-define F -finite_field -q 4 -end \textbackslash \textbackslash} \\
\text{\textbackslash \textbackslash} \text{-define P -projective_space 3 F -end \textbackslash \textbackslash} \\
\text{\textbackslash \textbackslash} \text{-with P -do \textbackslash \textbackslash} \\
\text{\textbackslash \textbackslash} \text{-projective_space_activity \textbackslash \textbackslash} \\
\text{\textbackslash \textbackslash} \text{-cheat-sheet_for_decomposition_by_element_PG 1 \textbackslash \textbackslash} \\
\text{\textbackslash \textbackslash} \text{\"1,0,0,0, 0,1,0,0, 0,0,1,0, 0,0,0,1, 1\" \textbackslash \textbackslash} \\
\text{\textbackslash \textbackslash} \text{-fix_structure_2A \textbackslash \textbackslash} \\
\text{\textbackslash \textbackslash} \text{-end}
```

pdflatex fix_structure_2A.tex

open fix_structure_2A.pdf

fix_structure_2B:

```bash
$ (ORBITER) \( \text{-v} \ 2 \ \text{\textbackslash \textbackslash} \\
\text{\textbackslash \textbackslash} \text{-define F -finite_field -q 4 -end \textbackslash \textbackslash} \\
\text{\textbackslash \textbackslash} \text{-define P -projective_space 3 F -end \textbackslash \textbackslash} \\
\text{\textbackslash \textbackslash} \text{-with P -do \textbackslash \textbackslash} \\
\text{\textbackslash \textbackslash} \text{-projective_space_activity \textbackslash \textbackslash} \\
\text{\textbackslash \textbackslash} \text{-cheat-sheet_for_decomposition_by_element_PG 1 \textbackslash \textbackslash} \\
\text{\textbackslash \textbackslash} \text{\"1,0,0,0, 1,1,0,0, 0,0,1,0, 0,0,0,1, 0\" \textbackslash \textbackslash} \\
\text{\textbackslash \textbackslash} \text{-fix_structure_2B \textbackslash \textbackslash} \\
\text{\textbackslash \textbackslash} \text{-end}
```

pdflatex fix_structure_2B.tex

open fix_structure_2B.pdf

fix_structure_2C:

```bash
$ (ORBITER) \( \text{-v} \ 2 \ \text{\textbackslash \textbackslash} \\
\text{\textbackslash \textbackslash} \text{-define F -finite_field -q 4 -end \textbackslash \textbackslash} \\
\text{\textbackslash \textbackslash} \text{-define P -projective_space 3 F -end \textbackslash \textbackslash} \\
\text{\textbackslash \textbackslash} \text{-with P -do \textbackslash \textbackslash} \\
\text{\textbackslash \textbackslash} \text{-projective_space_activity \textbackslash \textbackslash} \\
\text{\textbackslash \textbackslash} \text{-cheat-sheet_for_decomposition_by_element_PG 1 \textbackslash \textbackslash} \\
\text{\textbackslash \textbackslash} \text{\"1,0,0,0, 1,1,0,0, 0,0,1,0, 0,0,1,1, 0\" \textbackslash \textbackslash} 
```
2524 \> \> \> \> fix_structure_2C \\
2525 \> \> -end \\
2526 \> pdflatex fix_structure_2C.tex \\
2527 \> open fix_structure_2C.pdf \\
2528 \\
2529 \\
2530 \\
2531 \\
2532 trans: \\
2533 \> $(ORBITER) -v 5 \ \\
2534 \> -define F -finite_field -q 16 -end \ \\
2535 \> -define P -projective_space 3 F -end \ \\
2536 \> -with P -do \ \\
2537 \> -projective_space_activity \ \\
2538 \> -move_two_lines_in_hyperplane_stabilizer_text \ \\
2539 \> "1,0,0,0, 0,0,0,1" "1,1,0,2, 0,0,1,0" \ \\
2540 \> "1,0,0,0, 0,0,0,1" "0,1,0,1, 0,0,1,0" \ \\
2541 \> -end \\
2542 \\
2543 del_Pezzo_F13ab_report: \\
2544 \> $(ORBITER) -v 3 \ \\
2545 \> -define F -finite_field -q 13 -end \ \\
2546 \> -define P -projective_space 3 F -end \ \\
2547 \> -define f3 -formula \ \\
2548 \> "del_Pezzo_F13a" "del\_Pezzo\_F13a" "x,y,z" \ \\
2549 \> "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" \\
2550 \> -define f4 -formula \ \\
2551 \> "del_Pezzo_F13b" "del\_Pezzo\_F13b" "x,y,z" \ \\
2552 \> "x*x*x+y*y*y+z*z+z-x*x*y*y" \\
2553 \> -define del_Pezzo13 -collection "f3,f4" \\
2554 \> -with P -do \ \\
2555 \> -projective_space_activity \ \\
2556 \> -analyze_del_Pezzo_surface del_Pezzo13 "" \ \\
2557 \> -end \\
2558 \> pdflatex del_Pezzo_F13b_report.tex \\
2559 \> open del_Pezzo_F13b_report.pdf \\
2560 \\
2561 \\
2562 \\
2563 del_Pezzo_F13a_points.txt: \\
2564 \> $(ORBITER) -v 3 \ \\
2565 \> -define F -finite_field -q 13 -end \ \\
2566 \> -define P -projective_space 3 F -end \ \\
2567 \> -define f1 -formula "del_Pezzo_F9" \\
2568 \> "del\_Pezzo\_F9" "x,y,z" \ \\
2569 \> "x*x*x+y*y*y+z*z+z*z" \\
2570
#define f2 -formula "del_Pezzo_F11" \\
"del_Pezzo\_F11" "x,y,z" \\
"x*x*x+y*y+y+z+z*z+z+x*x*y*y+x*x*z*z+y*y*z*z" \\
#define f3 -formula "del_Pezzo_F13a" \\
"del_Pezzo\_F13a" "x,y,z" \\
"x*x*x*y*y+y*y+y+z*z+z+8*x*x*y*y+8*x*x*z*z+8*y*y*z*z" \\
#define f4 -formula "del_Pezzo_F13b" \\
"del_Pezzo\_F13b" "x,y,z" \\
"x*x*x+y*y+y*z+z*z*z*z-x*x*y*y" \\
#define del_Pezzo9 -collection "f1" \\
#define del_Pezzo11 -collection "f2" \\
#define del_Pezzo13 -collection "f3,f4" \\
-with P -do \\
-projective_space_activity \\
-analyze del_Pezzo_surface del_Pezzo13 ""

pdflatex del_Pezzo_F13a_report.tex 
pdflatex del_Pezzo_F13b_report.tex 
open del_Pezzo_F13a_report.pdf 
open del_Pezzo_F13b_report.pdf 
#dot -Tpng del_Pezzo_F13a.gv >del_Pezzo_F13a.png 
#open del_Pezzo_F13a.png

# writes del_Pezzo_F13a_points.txt

del_Pezzo_169:

$(ORBITER) -v 3 \\
#define F -finite_field -q 169 -end \\
#define P -projective_space 3 F -end \\
#define f3 -formula "del_Pezzo_F169a" \\
"del_Pezzo\_F169a" "x,y,z" \\
"x*x*x+x+y*y+y*z+z*z+z+8*x*x*y*y+8*x*x*z*z+8*y*y*z*z" \\
#define f4 -formula "del_Pezzo_F169b" \\
"del_Pezzo\_F169b" "x,y,z" \\
"x*x*x+x+y*y+y+z*z+z+z-x*x*y*y" \\
#define del_Pezzo13 -collection "f3,f4" \\
-with P -do \\
-projective_space_activity \\
-analyze del_Pezzo_surface del_Pezzo13 ""

pdflatex del_Pezzo_F169a_report.tex 
pdflatex del_Pezzo_F169b_report.tex 
open del_Pezzo_F169a_report.pdf 
open del_Pezzo_F169b_report.pdf
Section 4.10: Geometric Objects

SECTION GEOMETRIC_OBJECTS:

elliptic_quadric_ovoid_q8:
$\text{ORBITER} -v 2 \$
-define F -finite_field -q 8 -end \-
-define P -projective_space 3 F -end \-
-define O -geometric_object P \-
-elliptic_quadric_ovoid \-
-end \-
-with O -do -combinatorial_object_activity -save \-
-end

#ovoid_q8.txt
# 65 points

ovoid_ST_q8:
$\text{ORBITER} -v 2 \$
-define F -finite_field -q 8 -end \-
-define P -projective_space 3 F -end \-
-define O -geometric_object P \-
-ovoid_ST \-
-end \-
-with O -do -combinatorial_object_activity -save \-
-end

#ovoid_ST_q8.txt

Baer_PG_2.4:
$\text{ORBITER} -v 2 \$
-define F -finite_field -q 4 -end \-
-define P -projective_space 2 F -end \-
-define O -geometric_object P \-
-Baer_substructure \-
-end \-
-with O -do -combinatorial_object_activity -save \-
-end

Baer_PG_3.4:
$\text{(ORBITER) -v 2 \ }
\text{-define F -finite_field -q 4 -end \ }
\text{-define P -projective_space 3 F -end \ }
\text{-define O -geometric_object P \ }
\text{-Baer_substructure \ }
\text{-end \ }
\text{-with O -do -combinatorial_object_activity -save \ }
\text{-end}

\text{BLT\_database\_5\_0:}
$\text{(ORBITER) -v 2 \ }
\text{-define F -finite_field -q 5 -end \ }
\text{-define P -projective_space 4 F -end \ }
\text{-define O -geometric_object P \ }
\text{-BLT\_database 0 \ }
\text{-end \ }
\text{-with O -do -combinatorial_object_activity -save \ }
\text{-end}

\# writes BLT\_5\_0.txt

\text{BLT\_database\_7\_0:}
$\text{(ORBITER) -v 2 \ }
\text{-define F -finite_field -q 7 -end \ }
\text{-define P -projective_space 4 F -end \ }
\text{-define O -geometric_object P \ }
\text{-BLT\_database 0 \ }
\text{-end \ }
\text{-with O -do -combinatorial_object_activity -save \ }
\text{-end}

\# writes BLT\_7\_0.txt

\text{BLT\_database\_7\_1:}
$\text{(ORBITER) -v 2 \ }
\text{-define F -finite_field -q 7 -end \ }
\text{-define P -projective_space 4 F -end \ }
\text{-define S -geometric_object P \ }
\text{-BLT\_database 1 \ }
\text{-end \ }
\text{-with S -do -combinatorial_object_activity -save \ }
\text{-end}

\# writes BLT\_7\_1.txt
```
2712 BLT_database_7_1_print:
2714 \> \$\{ORBITER\} -v 2 \n2715 \> \> -define F -finite_field -q 7 -end \n2716 \> \> -define O -orthogonal_space 0 5 F -without_group -end \n2717 \> \> -define S -set -file_orbiter_format BLT_7_1.txt -end \n2718 \> \> \> -with O -do -orthogonal_space_activity \n2719 \> \> \> -print_points S -end \n2720 \> \> pdflatex S_set_report.tex
2721 \> \> open S_set_report.pdf
2722
2723
2724 BLT_database_67_4:
2726 \> \$\{ORBITER\} -v 2 \n2727 \> \> -define F -finite_field -q 67 -end \n2728 \> \> -define P -projective_space 4 F -end \n2729 \> \> -define Obj -geometric_object P \n2730 \> \> \> -BLT_database 4 \n2731 \> \> \> -end \n2732 \> \> \> -with Obj -do -combinatorial_object_activity -save \n2733 \> \> \> -end \n2734 \> \> \> -define O -orthogonal_space 0 5 F -without_group -end \n2735 \> \> \> -define BLT_67_4 -set -file_orbiter_format BLT_67_4.txt -end \n2736 \> \> \> -with 0 -do -orthogonal_space_activity \n2737 \> \> \> \> -print_points BLT_67_4 -end
2738 \> \> \> pdflatex BLT_67_4_set_report.tex
2739 \> \> \> open BLT_67_4_set_report.pdf
2740
2741
2742 Doily_W_2:
2744 \> \$\{ORBITER\} -v 2 \n2745 \> \> -define F -finite_field -q 2 -end \n2746 \> \> -define O -orthogonal_space 0 5 F -without_group -end \n2747 \> \> -define W2_points -set -loop 0 15 1 -end \n2748 \> \> -define W2_lines -set -loop 0 15 1 -end \n2749 \> \> \> -with 0 -do \n2750 \> \> \> -orthogonal_space_activity \n2751 \> \> \> \> -print_points W2_points \n2752 \> \> \> \> \> -end \n2753 \> \> \> \> \> -with 0 -do \n2754 \> \> \> \> \> -orthogonal_space_activity \n2755 \> \> \> \> \> \> -print_lines W2_lines \n2756 \> \> \> \> \> \> \> -end
2757 \> \> \> \> \> pdflatex W2_points_set_report.tex
2758 \> \> \> \> \> open W2_points_set_report.pdf
```
# the output defines doily.csv

Doily_disjoint_sets_graph_cliques_3:

doiy.csv

Doil sets graph cliques 3:

doiy.csv

Doily_disjoint_sets_graph_cliques_5:

doiy.csv

Doil sets graph cliques 5:

doiy.csv

PG_3.2.test:
2806  $(ORBITER) -v 2 \
2807  -define F -finite_field -q 2 -end \
2808  -define P -projective_space 3 F -end \
2809  -with P -do -projective_space_activity \
2810  -cheat_sheet \
2811  -end 
2812  pdflatex PG_3_2.tex
2813  open PG_3_2.pdf
2814
2815
2816  Edge_curve_17:
2817  $(ORBITER) -v 2 \
2818  -define F -finite_field -q 17 -end \
2819  -define R -polynomial_ring -field F \
2820  -number_of_variables 3 \
2821  -homogeneous_of_degree 4 \
2822  -end \
2823  -define P -projective_space 2 F -end \
2824  -define C -geometric_object P \
2825  -projective_variety R \
2826  "Edge_q17" "Edge\_q17" \
2827  $(EDGE\_CURVE\_Q17\_EQUATION) \
2828  -end \
2829  -with C -do -combinatorial_object_activity -save \
2830  -end 
2831
2832  #Edge_q17.txt
2833  #combinatorial_object_create::init created a set of size 12
2834  #( 4, 7, 16, 19, 20, 23, 32, 35, 89, 100, 244, 251 )
2835
2836
2837
2838
2839
2840
2841
2842  Edge_curve_17_line_type:
2843  echo $(FILE_Q17) >edge_q17.csv
2844  $(ORBITER) -v 2 \
2845  -define F -finite_field -q 17 -end \
2846  -define R -polynomial_ring -field F \
2847  -number_of_variables 3 \
2848  -homogeneous_of_degree 4 \
2849  -end \
2850  -define P -projective_space 2 F -end \
2851  -define C -geometric_object P \
2852  -projective_variety R \

537
"Edge_q17" "Edge\_q17"
$(EDGE\_CURVE\_Q17\_EQUATION) 
-end 
-with C -do 
-combinatorial\_object\_activity 
-line_type 
-end 
-print\_symbols

(#( 4^6, 2^30, 1^132, 0^139 ))

Edge\_curve\_q23\_line\_type:
 $\text{def} F -\text{finite\_field} -q 23 -end 
-def P -\text{projective\_space} 2 F -end 
-def E -\text{geometric\_object} P 
-set $(EDGE\_CURVE\_Q23\_AS\_POINTS) 
-end 
-with E -do 
-combinatorial\_object\_activity 
-save 
-end 
-with E -do 
-combinatorial\_object\_activity 
-line_type 
-end 
-print\_symbols

# Chapter 5 - Group Theory
# Section 5.1: Permutation groups
SECTION_PERMUTATION_GROUPS:

Cyclic_6:

-define G -permutation_group -cyclic_group 6 -end -with G -do -group_theoretic_activity -report -end

dflaatex Perm6_report.tex
open Perm6_report.pdf

Cyclic_6_group_table:

-define G -permutation_group -cyclic_group 6 -end -with G -do -group_theoretic_activity -export group_table -end

dflaatex Perm6_report.tex
open Perm6_group_table_draw.bmp

Symmetric_3:

-define G -permutation_group -symmetric_group 3 -end -with G -do -group_theoretic_activity -report -end

dflaatex Perm3_report.tex
open Perm3_report.pdf

Symmetric_3_group_table:

-define G -permutation_group -symmetric_group 3 -end
2947 ▷▷ -with G -do \\
2948 ▷▷ -group_theoretic_activity \\
2949 ▷▷ ▷▷ -export_group_table \\
2950 ▷▷ ▷▷ -end \\
2951 ▷▷ $(ORBITER) -v 2 \\
2952 ▷▷ ▷▷ -define all_one_r -vector -repeat 1 6 -end \\
2953 ▷▷ ▷▷ -define all_one_c -vector -repeat 1 6 -end \\
2954 ▷▷ ▷▷ -draw_matrix \\
2955 ▷▷ ▷▷ ▷▷ -input_csv_file Perm3_group_table.csv \\
2956 ▷▷ ▷▷ ▷▷ -box_width 50 -bit_depth 24 \\
2957 ▷▷ ▷▷ ▷▷ -partition 3 all_one_r all_one_c \\
2958 ▷▷ ▷▷ ▷▷ -end \\
2959 ▷▷ ▷▷ open Perm3_group_table_draw.bmp \\
2960 
2961 Symmetric_3_elements: \\
2962 ▷▷ $(ORBITER) -v 3 \\
2963 ▷▷ ▷▷ -define G -permutation_group -symmetric_group 3 -end \\
2964 ▷▷ ▷▷ -with G -do \\
2965 ▷▷ ▷▷ -group_theoretic_activity \\
2966 ▷▷ ▷▷ ▷▷ -save_elements_csv "Symmetric3_elts.csv" \\
2967 ▷▷ ▷▷ ▷▷ -end \\
2968 ▷▷ $(ORBITER) -v 2 \\
2969 ▷▷ ▷▷ -define Sym3_elts -vector -load_csv_data_column \\
2970 ▷▷ ▷▷ ▷▷ Symmetric3_elts.csv 1 -end \\
2971 ▷▷ ▷▷ ▷▷ -save_matrix_csv Sym3_elts \\
2972 ▷▷ $(ORBITER) -v 2 \\
2973 ▷▷ ▷▷ -define all_one_r -vector -repeat 1 6 -end \\
2974 ▷▷ ▷▷ -define all_one_c -vector -repeat 1 3 -end \\
2975 ▷▷ ▷▷ -draw_matrix \\
2976 ▷▷ ▷▷ ▷▷ -input_csv_file Sym3_elts_matrix.csv \\
2977 ▷▷ ▷▷ ▷▷ -box_width 50 -bit_depth 8 \\
2978 ▷▷ ▷▷ ▷▷ -partition 3 \\
2979 ▷▷ ▷▷ ▷▷ ▷▷ all_one_r all_one_c \\
2980 ▷▷ ▷▷ ▷▷ ▷▷ -end \\
2981 ▷▷ ▷▷ open Sym3_elts_matrix_draw.bmp \\
2982 
2983 Symmetric_3_long: \\
2984 ▷▷ $(ORBITER) -v 3 \\
2985 ▷▷ ▷▷ -define G -permutation_group -symmetric_group 3 -end \\
2986 ▷▷ ▷▷ -with G -do \\
2987 ▷▷ ▷▷ -group_theoretic_activity \\
2988 ▷▷ ▷▷ ▷▷ -export_orbiter \\
2989 ▷▷ ▷▷ ▷▷ -end \\
2990 ▷▷ ▷▷ -with G -do \\
2991 ▷▷ ▷▷ -group_theoretic_activity \\
2992 ▷▷ ▷▷ ▷▷ -print_elements_tex \\
2993 ▷▷ ▷▷ ▷▷ -end \\

540
Symmetric 3:

```bash
$ (ORBITER) -v 3 \
-draw_options \n-nodes \n-embedded -radius 250 \n-xin 10000 -yin 10000 \n-xout 100000 -yout 600000 \n-scale 0.3 -line_width 1.0 \n-end \n-tree_draw -file Perm3_elements_tree.txt -end
```

```bash
$ (ORBITER) -v 2 \
-define M -vector -load_csv_data_column \n-Symmetric3_elts.csv 1 -end \n-save_matrix_csv M
```

```bash
$ (ORBITER) -v 2 \
-define all_one_r -vector -repeat 1 6 -end \n-define all_one_c -vector -repeat 1 3 -end \n-draw_matrix \n-input_csv_file M_matrix.csv \n-box_width 50 -bit_depth 8 \n-partition 3 \n-all_one_r all_one_c \n-end
```

```bash
pdflatex Perm3_elements_tree_draw.tex
```

```bash
open Perm3_elements_tree_draw.pdf
```

```bash
#pdflatex Perm3_report.tex
```

```bash
#open Perm3_report.pdf
```

Symmetric 4:

```bash
$ (ORBITER) -v 3 \
-define G -permutation_group -symmetric_group 4 -end \n-with G -do \n-group_theoretic_activity \n-report \n-end
```

```bash
pdflatex Perm4_report.tex
```

```bash
open Perm4_report.pdf
```

3039
3040
Symmetric_4_group_table:

$\text{define G -permutation_group -symmetric_group 4 -end}$

$\text{with G -do}$

$\text{-group_theoretic_activity}$

$\text{-export_group_table}$

$\text{-end}$

$\text{define all_one_r -vector -repeat 1 24 -end}$

$\text{define all_one_c -vector -repeat 1 24 -end}$

$\text{-draw_matrix}$

$\text{-input_csv_file Perm4_group_table.csv}$

$\text{-box_width 50 -bit_depth 24}$

$\text{-partition 3 all_one_r all_one_c}$

$\text{-end}$

$\text{open Perm4_group_table_draw.bmp}$

Symmetric_4_long:

$\text{define G -permutation_group -symmetric_group 4 -end}$

$\text{with G -do}$

$\text{-group_theoretic_activity}$

$\text{-export_orbiter}$

$\text{-end}$

$\text{with G -do}$

$\text{-group_theoretic_activity}$

$\text{-export_group_table}$

$\text{-end}$

$\text{with G -do}$

$\text{-group_theoretic_activity}$

$\text{-print_elements_tex}$

$\text{-end}$

$\text{with G -do}$

$\text{-group_theoretic_activity}$

$\text{-report}$

$\text{-end}$

$\text{with G -do}$

$\text{-group_theoretic_activity}$

$\text{-save_elements_csv "Symmetric4_elts.csv"}$

$\text{-end}$

$\text{with G -do}$

$\text{-group_theoretic_activity}$

$\text{-export_inversion_graphs "Symmetric4_inversion_graphs.csv"}$

$\text{-end}$

$\text{$(ORBITER) -v 2$}$
-draw_options \
-nodes \
-embedded -radius 175 \
-xin 10000 -yin 10000 \
-xout 1500000 -yout 600000 \
-scale 0.3 -line_width 1.0 \
-end \
-tree_draw -file Perm4_elements_tree.txt -end

$(ORBITER) -v 2 -draw_matrix \
-input_csv_file Perm4_group.table.csv \
-box_width 50 -bit_depth 24 -end 
$(ORBITER) -v 2 
-define M -vector -load_csv_data_column \
-Symmetric4_elts.csv 1 -end \
-save_matrix_csv M 
$(ORBITER) -v 2 
-define all_one_r -vector -repeat 1 24 -end \
-define all_one_c -vector -repeat 1 4 -end \
-draw_matrix \
-input_csv_file M_matrix.csv \
-box_width 50 -bit_depth 8 \n-partition 3 \
-all_one_r all_one_c \n-end 

pdflatex Perm4_elements_tree draw.tex 
open Perm4_elements_tree draw.pdf 
#pdflatex Perm4_report.tex 
#open Perm4_report.pdf

# ToDo:

Symmetric_4_stab:

$(ORBITER) -v 2 
-define G -permutation_group -symmetric_group 4 -end 
-with G -do \
-group_theoretic_activity \
-orbits_on_points \
-stabilizer_of_orbit_rep 0 \n-end 

$(ORBITER) -v 2 
-define gens -vector -file Perm4_stab_orb_0 gens.csv -end 
-define G -permutation_group \
-bsgs Perm4_stab_orb_0 "Sym3" 4 6 "0,1,2" 2 gens -end \
-define Gr -modified_group -from G \n
543
Section 5.2: Linear Groups

PGL_3_2:

PGL_4_2:

PGL_4_2_export:
$\text{define } F \text{-finite field } -q 2 \text{-end }$
$\text{define } G \text{-linear group } -\text{PGL 4 F -end }$
$\text{with } G \text{-do }$
$\text{-group_theoretic_activity }$
$\text{report }$
$\text{-end }$
$\text{with } G \text{-do }$
$\text{-group_theoretic_activity }$
$\text{export_orbiter }$
$\text{-end}$
$\text{pdflatex PGL}_4 2 \text{report.tex}$
$\text{open PGL}_4 2 \text{report.pdf}$

# created by PGL 4 2 export

PGL 4 2 generated:
$\text{define } \text{gens } -\text{vector } -\text{file PGL}_4 \_2 \text{gens.csv } -\text{end }$
$\text{define } G \text{-permutation group }$
$\text{bsgs PGL}_4 2 "\text{\{rm PGL\}}(4,2)" 15 20160 "0,1,2,3" 6 \text{gens } -\text{end }$

L 5 3:
$\text{define } F \text{-finite field } -q 3 \text{-end }$
$\text{define } G \text{-linear group } -\text{PSL 5 F -end }$
$\text{with } G \text{-do }$
$\text{-group_theoretic_activity }$
$\text{report }$
$\text{-end}$
$\text{pdflatex PSL}_5 3 \text{report.tex}$
$\text{open PSL}_5 3 \text{report.pdf}$

#PSL(5,3): Order 237783237120 = 121 * 120 * 117 * 108 * 81 * 16

L 4 5:
$\text{define } F \text{-finite field } -q 5 \text{-end }$
$\text{define } G \text{-linear group } -\text{PSL 4 F -end }$
$\text{with } G \text{-do }$
$\text{-group_theoretic_activity }$
$\text{report }$
$\text{-end}$
$\text{pdflatex PSL}_4 5 \text{report.tex}$
$\text{open PSL}_4 5 \text{report.pdf}$
#PSL(4,5): Order 7254000000

PGL_{4,5}:
\[
\text{\$\text{ORBITER} \ -v \ 2 \ -define \ F \ \text{finite}\_\text{field} \ -q \ 5 \ -end} \\
\text{-define \ G \ \text{linear}\_\text{group} \ \text{-PGL \ 4 \ F} \ -end} \\
\text{-with \ G \ -do} \\
\text{-group\_theoretic\_activity} \\
\text{\text{-report}} \\
\text{\text{-end}}
\]
pdflatex PGL_{4,5}\_report.tex
open PGL_{4,5}\_report.pdf

PGGL_{3,4}:
\[
\text{\$\text{ORBITER} \ -v \ 2 \ -define \ G \ \text{linear}\_\text{group} \ \text{-PGGL \ 3 \ 4} \ -end} \\
\text{-with \ G \ -do} \\
\text{-group\_theoretic\_activity} \\
\text{-report} \\
\text{-classes} \\
\text{-end}
\]
pdflatex PGGL_{3,4}\_report.tex
open PGGL_{3,4}\_report.pdf

PGGL_{3,8}:
\[
\text{\$\text{ORBITER} \ -v \ 2} \\
\text{-define \ G \ \text{linear}\_\text{group} \ \text{-PGGL \ 3 \ 8} \ -end}
\]

PGGL_{3,8}\_report:
\[
\text{\$\text{ORBITER} \ -v \ 3} \\
\text{-define \ G \ \text{linear}\_\text{group} \ \text{-PGGL \ 3 \ 8} \ -end} \\
\text{-with \ G \ -do} \\
\text{-group\_theoretic\_activity} \\
\text{-report} \\
\]
3276 ▷ ▷ -end
3277 ▷ pdflatex PGGL_3_8_report.tex
3278 ▷ open PGGL_3_8_report.pdf
3279
3280
3281 AGL_1_27:
3282 ▷ $(ORBITER) -v 2 \n3283 ▷ ▷ -define F -finite_field -q 27 -end \n3284 ▷ ▷ -define G -linear_group -AGL 1 F -end \n3285 ▷ ▷ -with G -do \n3286 ▷ ▷ -group_theoretic_activity \n3287 ▷ ▷ ▷ -report \n3288 ▷ ▷ -end
3289 ▷ pdflatex AGL_1_27_report.tex
3290 ▷ open AGL_1_27_report.pdf
3291
3292 #▷ ▷ -group_table \n3293
3294
3295 SP_4_2:
3296 ▷ $(ORBITER) -v 2 \n3297 ▷ ▷ -define F -finite_field -q 2 -end \n3298 ▷ ▷ -define G -linear_group -GL 4 F \n3299 ▷ ▷ ▷ -symplectic_group \n3300 ▷ ▷ -end \n3301 ▷ ▷ -with G -do \n3302 ▷ ▷ -group_theoretic_activity \n3303 ▷ ▷ ▷ -report \n3304 ▷ ▷ -end
3305 ▷ pdflatex GL_4_2_Sp_4_2_report.tex
3306 ▷ open GL_4_2_Sp_4_2_report.pdf
3307
3308 # order 720
3309
3310
3311 PSP_4_4:
3312 ▷ $(ORBITER) -v 2 \n3313 ▷ ▷ -define F -finite_field -q 4 -end \n3314 ▷ ▷ -define G -linear_group -PGL 4 F \n3315 ▷ ▷ ▷ -symplectic_group \n3316 ▷ ▷ -end \n3317 ▷ ▷ -with G -do \n3318 ▷ ▷ -group_theoretic_activity \n3319 ▷ ▷ ▷ -report \n3320 ▷ ▷ -end
3321 ▷ pdflatex PGL_4_4_Sp_4_4_report.tex
3322 ▷ open PGL_4_4_Sp_4_4_report.pdf

547
#order 979200

PGO_5_2:

\$ \$(ORBITER) -v 2 \$

\$ -define F -finite_field -q 2 -end \$

\$ -define G -linear_group -PGO 5 F -end \$

\$ -with G -do \$

\$ -group_theoretic_activity \$

\$ -report \$

\$ -end \$

\$ pdflatex PGO_5_2.report.tex \$

\$ open PGO_5_2.report.pdf \$

PGGO_5_4:

\$ \$(ORBITER) -v 2 \$

\$ -define F4 -finite_field -q 4 -end \$

\$ -define G -linear_group -PGGO 5 F4 -end \$

\$ -with G -do \$

\$ -group_theoretic_activity \$

\$ -report \$

\$ -end \$

\$ pdflatex PGGO_5_4.report.tex \$

\$ open PGGO_5_4.report.pdf \$

PGOp_6_2:

\$ \$(ORBITER) -v 2 \$

\$ -define F -finite_field -q 2 -end \$

\$ -define G -linear_group -PGOp 6 F -end \$

\$ -with G -do \$

\$ -group_theoretic_activity \$

\$ -report \$

\$ -end \$

\$ pdflatex PGOp_6_2.report.tex \$

\$ open PGOp_6_2.report.pdf \$

PGOm_6_2:

\$ \$(ORBITER) -v 2 \$

\$ -define F -finite_field -q 2 -end \$

\$ -define G -linear_group -PGOm 6 F -end \$

\$ -with G -do \$

\$ -group_theoretic_activity \$

\$ -report \$

\$ -end \$

\$ pdflatex PGOm_6_2.report.tex \$

\$ open PGOm_6_2.report.pdf \$
# the following two groups are isomorphic:

```latex
\texttt{PSP}_{6,2}:
\texttt{\$(ORBITER) -v 2 \ }
\texttt{define F -finite_field -q 2 -end \ }
\texttt{define G -linear_group -PGL 6 F \ }
\texttt{define G -linear_group -PGL 6 F \ }
\texttt{define G -linear_group -PGL 6 F \ }
\texttt{-symplectic_group \ }
\texttt{-end \ }
\texttt{with G -do \ }
\texttt{-group_theoretic_activity \ }
\texttt{-report \ }
\texttt{-end \ }
\texttt{pdflatex PGL_{6,2}Sp_{6,2}report.tex}
\texttt{open PGL_{6,2}Sp_{6,2}report.pdf}
```

# group order 1451520, acting on 63 points

```latex
\texttt{PGO}_{7,2}:
\texttt{\$(ORBITER) -v 2 \ }
\texttt{define F -finite_field -q 2 -end \ }
\texttt{define G -linear_group -PGO 7 F -end \ }
\texttt{define G -linear_group -PGO 7 F -end \ }
\texttt{-group_theoretic_activity \ }
\texttt{-report \ }
\texttt{-end}
\texttt{pdflatex PGO_{7,2}report.tex}
\texttt{open PGO_{7,2}report.pdf}
```

# group order 1451520, acting on 63 points
Section 5.3: Subgroups

**SECTION**

**SUBGROUPS:**

- **C13:**

  ```
  $(ORBITER) -v 2 \
  -define gens -vector -dense $(GEN_C13) -end \
  -define G -permutation_group \
  -bsgs C13 C_{13} 13 13 0 1 \
  gens -end \
  -with G -do \
  -group_theoretic_activity \
  -export_orbiter \
  -end \
  -with G -do \
  -group_theoretic_activity \
  -export_group_table \
  -end \
  -with G -do \
  -group_theoretic_activity \
  -report \
  -end \
  -with G -do \
  -group_theoretic_activity \
  -save_elements_csv "C13_elts.csv" \
  -end 
  pdflatex C13_report.tex 
  open C13_report.pdf 
  
  C13-generated:
  $(ORBITER) -v 2 \
  -define gens -vector -file C13_gens.csv -end \
  -define G -permutation_group \
  -bsgs C13 "C_{13}" 13 13 "0" 1 gens -end 
  
  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 \
  ```

```
3464 ▶ ▶ -end \ 
3465 ▶ ▶ -with G -do \ 
3466 ▶ ▶ -group_theoretic_activity \ 
3467 ▶ ▶ ▶ -report \ 
3468 ▶ ▶ -end \ 
3469 ▶ ▶ -with G -do \ 
3470 ▶ ▶ -group_theoretic_activity \ 
3471 ▶ ▶ ▶ -save_elements_csv "C13_elts.csv" \ 
3472 ▶ ▶ -end 
3473 ▶ #pdflatex Perm13_Subgroup_C13_13_report.tex 
3474 ▶ #open Perm13_Subgroup_C13_13_report.pdf 
3475 
3476 
3477 
3478 
3479 J1:
3480 ▶ $(ORBITER) -v 2 \ 
3481 ▶ ▶ -define G -linear_group -PGL 7 11 -Janko1 -end \ 
3482 ▶ ▶ -with G -do \ 
3483 ▶ ▶ -group_theoretic_activity \ 
3484 ▶ ▶ ▶ -report \ 
3485 ▶ ▶ -end
3486 ▶ pdflatex PGL_7_11_Subgroup_Janko1_report.tex
3487 ▶ open PGL_7_11_Subgroup_Janko1_report.pdf
3488 
3489 PGL_3_11_singer:
3490 ▶ $(ORBITER) -v 2 \ 
3491 ▶ ▶ -define G -linear_group -PGL 3 11 -singer 19 -end \ 
3492 ▶ ▶ -with G -do \ 
3493 ▶ ▶ -group_theoretic_activity \ 
3494 ▶ ▶ ▶ -report \ 
3495 ▶ ▶ -end
3496 ▶ pdflatex PGL_3_11_Singer_3_11_19_report.tex
3497 ▶ open PGL_3_11_Singer_3_11_19_report.pdf
3498 
3499 
3500 PGL_3_11_singer_and_frobenius:
3501 ▶ $(ORBITER) -v 2 \ 
3502 ▶ ▶ -define G -linear_group -PGL 3 11 -singer_and_frobenius 19 -end \ 
3503 ▶ ▶ -with G -do \ 
3504 ▶ ▶ -group_theoretic_activity \ 
3505 ▶ ▶ ▶ -report \ 
3506 ▶ ▶ -end
3507 ▶ ▶ pdflatex PGL_3_11_Singer_and_Frob3_11_19_report.tex
3508 ▶ ▶ open PGL_3_11_Singer_and_Frob3_11_19_report.pdf
3509 
3510 PG_2_4_order_21:
quaternion:

\begin{verbatim}
321 \$\text{ORBITER} -v 2 \$
322 \>$-\text{define G -linear_group -PGL 3 4 -end} \$
323 \>$-\text{with G -do} \$
324 \>$-\text{group_theoretic_activity} \$
325 \>$-\text{search_element_of_order 21} \$
326 \>$-\text{end} \$
327 \>\$
328 \>\$
3521 \text{quaternion:}
329 \>$-\text{define G -linear_group -SL 2 3} \$
330 \>$-\text{subgroup_by_generators "quaternion" "8" 3} \$
331 \>$-\text{with G -do} \$
332 \>$-\text{group_theoretic_activity} \$
333 \>$-\text{print_elements.tex} \$
334 \>$-\text{group_table} \$
335 \>$-\text{report} \$
336 \>$-\text{end} \$
337 \>$-\text{pdflatex GL_2.3_Subgroup_quaternion_8_elements.tex} \$
338 \>$-\text{open GL_2.3_Subgroup_quaternion_8_elements.pdf} \$
339 \>$-\text{pdflatex GL_2.3_Subgroup_quaternion_8_report.tex} \$
340 \>$-\text{open GL_2.3_Subgroup_quaternion_8_report.pdf} \$
341 \>\$
342 \>\$
343 \>\$
344 \>\$
345 \>\$
346 \>\$
347 \>\$
348 \>\$
349 \>\$
350 \>\$
351 \>\$
352 \>\$
353 \>\$
354 \>\$
355 \>\$
356 \>\$
357 \>\$
\end{verbatim}

\begin{verbatim}
cube_group:
348 \>$-\text{define gens -vector -dense} \$
349 \>$-\text{with G -do} \$
350 \>$-\text{group_theoretic_activity} \$
351 \>$-\text{print_elements.tex} \$
352 \>$-\text{group_table} \$
353 \>$-\text{report} \$
354 \>$-\text{end} \$
355 \>$-\text{pdflatex GL_3.3_Subgroup_cube_48_elements.tex} \$
356 \>$-\text{open GL_3.3_Subgroup_cube_48_elements.pdf} \$
\end{verbatim}
3558  ▶ open GL_3_3_Subgroup_cube_48_elements.pdf
3559
3560
tetra_group:
3562  ▶ $(ORBITER) -v 3 \
3563  ▶ ▶ -define G -linear_group -GL 3 3 \
3564  ▶ ▶ ▶ -subgroup_by_generators "tetra" "12" 2 \
3565  ▶ ▶ ▶ ▶ "0,1,0,0,0,1,1,0,0,0,0,1,2,0,0,0,2,0" \
3566  ▶ ▶ ▶ ▶ -end \
3567  ▶ ▶ ▶ -with G -do \
3568  ▶ ▶ ▶ ▶ -group_theoretic_activity \
3569  ▶ ▶ ▶ ▶ ▶ -print_elements.tex \
3570  ▶ ▶ ▶ ▶ ▶ ▶ -report \
3571  ▶ ▶ ▶ ▶ ▶ ▶ -end \
3572  ▶ ▶ ▶ ▶ ▶ ▶ ▶ pdflatex GL_3_3_Subgroup_tetra_12_report.tex
3573  ▶ ▶ ▶ ▶ ▶ ▶ ▶ open GL_3_3_Subgroup_tetra_12_report.pdf
3574  ▶ ▶ ▶ ▶ ▶ ▶ ▶ pdflatex GL_3_3_Subgroup_tetra_12_elements.tex
3575  ▶ ▶ ▶ ▶ ▶ ▶ ▶ open GL_3_3_Subgroup_tetra_12_elements.pdf
3576
3577
3578
3579  Hesse_group:
3580  ▶ $(ORBITER) -v 3 \
3581  ▶ ▶ -define gens -vector -compact \
3582  ▶ ▶ ▶ $(GENERATORS_HESSE_GROUP) \
3583  ▶ ▶ ▶ -end \
3584  ▶ ▶ ▶ -define G -linear_group -PGGL 3 4 \
3585  ▶ ▶ ▶ ▶ -subgroup_by_generators "Hesse" "432" 7 gens \
3586  ▶ ▶ ▶ ▶ -end \
3587  ▶ ▶ ▶ ▶ -with G -do \
3588  ▶ ▶ ▶ ▶ ▶ ▶ -group_theoretic_activity \
3589  ▶ ▶ ▶ ▶ ▶ ▶ ▶ -print_elements.tex \
3590  ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ -report \
3591  ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ -end \
3592  ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ pdflatex PGGL_3_4_Subgroup_Hesse_432_report.tex
3593  ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ open PGGL_3_4_Subgroup_Hesse_432_report.pdf
3594
3595  #Hesse group:
3596  #1,0,0,0,0,1,0,0,0,0,1,0,3,2,3,2,0,
3597  #1,0,0,0,0,1,0,0,3,1,2,0,1,0,1,3,0,
3598  #1,0,0,0,0,1,0,0,0,1,1,0,0,0,0,1,1,
3599  #1,0,0,0,0,2,2,0,0,2,0,0,0,0,0,1,0,
3600  #1,0,0,0,2,3,1,0,2,0,1,0,3,1,3,1,0,
3601  #0,1,1,0,2,0,0,0,1,3,2,0,2,1,1,2,1,
3602  #1,1,0,0,0,0,3,0,3,2,2,0,1,2,3,3,1,
3603
3604
Weyl_E8:

```
> $(ORBITER) -v 3
> -define gens -vector -dense
> $(GENERATORS_WEYL_GROUP_E8)
> -end
> -define G -linear_group -GL 8 3
> -subgroup_by_generators
> "Weyl_E8" "696729600" 2
> $(GENERATORS_WEYL_GROUP_E8)
> -end
> -with G -do
> -group_theoretic_activity
> -report
> -end

pdflatex GL_8_3_Subgroup_Weyl_E8_696729600_report.tex
open GL_8_3_Subgroup_Weyl_E8_696729600_report.pdf
```


test_subgroup:
```
> $(ORBITER) -v 2
> -define F -finite_field -q 2 -end
> -define G1 -linear_group -PGOp 6 F -end
> -define G2 -linear_group -PGL 6 F
> -symplectic_group
> -end
> -with G1 -and G2 -do
> -group_theoretic_activity
> -is_subgroup_of
> -end
```

coset_reps:
```
> $(ORBITER) -v 2
> -define F -finite_field -q 2 -end
> -define G1 -linear_group -PGOp 6 F -end
> -define G2 -linear_group -PGL 6 F
> -symplectic_group
> -end
> -with G1 -and G2 -do
> -group_theoretic_activity
> -coset_reps
> -end
```
SECTION LINEAR GROUPS ADVANCED TOPICS:

U_3_3:

PGL_2_3:
#Co3 from Conway et al., 1985 (ATLAS)
#order = 495766656000
#Co3 from the paper by Suleiman and Wilson 1997

Co3:

Ree_27:

# needs a lot of memory to run!
\begin{verbatim}
3745 ▼ ▼ -group_theoretic_activity \\
3746 ▼ ▼ ▼ -report \\
3747 ▼ ▼ -end
3748
3749 # needs a lot of memory to run!
3750
3751
3752 ####################################################################################################################
3753 # Section 5.5: Induced Actions
3754
3755
3756 SECTION_INDUCED_ACTIONS:
3757
3758 Symmetric_4_on_pairs:
3759 ▼ ▼ $(ORBITER) -v 3 \\
3760 ▼ ▼ ▼ -define G -permutation_group -symmetric_group 4 -end \\
3761 ▼ ▼ ▼ -define G_on_2subsets -modified_group -from G \\
3762 ▼ ▼ ▼ ▼ -on_k_subsets 2 \\
3763 ▼ ▼ ▼ -end \\
3764 ▼ ▼ ▼ -with G_on_2subsets -do \\
3765 ▼ ▼ ▼ -group_theoretic_activity \\
3766 ▼ ▼ ▼ ▼ -report \\
3767 ▼ ▼ ▼ -end
3768 ▼ -pdflatex Perm4_on_2_subsets_report.tex
3769 ▼ -open Perm4_on_2_subsets_report.pdf
3770
3771
3772 T3_on_tensors:
3773 ▼ ▼ $(ORBITER) -v 2 \\
3774 ▼ ▼ ▼ -define G \\
3775 ▼ ▼ ▼ -linear_group -GL_d_q_wr_Sym_n 2 2 3 \\
3776 ▼ ▼ ▼ ▼ -on_tensors -end \\
3777 ▼ ▼ ▼ ▼ -with G -do \\
3778 ▼ ▼ ▼ ▼ -group_theoretic_activity \\
3779 ▼ ▼ ▼ ▼ ▼ -report \\
3780 ▼ ▼ ▼ ▼ ▼ -end
3781 ▼ -pdflatex GL_2_2_wreath_Sym3_report.tex
3782 ▼ -open GL_2_2_wreath_Sym3_report.pdf
3783 ▼

3784
3785
3786 T3r1:
3787 ▼ ▼ $(ORBITER) -v 4 \\
3788 ▼ ▼ ▼ -define G \\
3789 ▼ ▼ ▼ -linear_group -GL_d_q_wr_Sym_n 2 2 3 \\
3790 ▼ ▼ ▼ ▼ -on_rank_one_tensors -end \\
3791 ▼ ▼ ▼ ▼ -with G -do \\
\end{verbatim}
T4_on_tensors:

$\text{ORBITER} -v 4 \$

T4r1:

$\text{ORBITER} -v 4 \$

PGGL_2.8_on_conic:

$\text{ORBITER} -v 4 \$
SURFACE_q13_STAB="1,0,0,0,12,0,0,0,12,0,0,0,1, \
1,0,0,0,12,0,0,0,1,0,0,0,0,12, \
1,0,0,0,0,12,0,0,0,0,0,0,1, \
0,1,0,0,1,0,0,0,0,1,0,0,0,0,1"

SURFACE_q13_stab_on_tritangents_orbits:
$(ORBITER) -v 30 \
$define F -finite_field -q 13 -end \
$define P -projective_space 3 F -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) -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 \
$define Gr -modified_group -from G_on_planes \
-restricted_action TriP \
-end \
-with Gr -do \
-group_theoretic_activity \
-report \
-end \
-with Gr -do \
-group_theoretic_activity \
-orbits_on_points \n
\\[+$\\text{ORBITER}$ -v 12 \\
\\text{-define G -linear_group -PGL 4 2 -wedge_detached -end} \\
\\text{-with G -do} \\
\\text{-group_theoretic_activity} \\
\\text{-report} \\
\\text{-end} \\
\\text{pdflatex PGL}_4_2_{-Wedge}_4_0_{-detached}_report.tex \\
\\text{open PGL}_4_2_{-Wedge}_4_0_{-detached}_report.pdf \\

\\text{SECTION GROUP THEORETIC ACTIVITIES:} \\
PGL_3_2_elements: \\
\\text{pdflatex PGL}_3_2_{-elements}.tex \\
\\text{open PGL}_3_2_{-elements}.pdf \\

\text{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 \ 
$\texttt{-draw_options}$ \ 
$\texttt{-nodes}$ \ 
$\texttt{-embedded -radius 250}$ \ 
$\texttt{-xin 10000 -yin 10000}$ \ 
$\texttt{-xout 1000000 -yout 600000}$ \ 
$\texttt{-scale 0.3 -line_width 1.0}$ \ 
$\texttt{-end}$ \ 
$\texttt{-tree_draw -file Perm3_elements_tree.txt -end}$ \ 
$\texttt{pdflatex Perm3_elements_tree_draw.tex}$ \ 
$\texttt{open Perm3_elements_tree_draw.pdf}$ \ 

$\texttt{Cycle_{13}_power:}$ \ 
$\texttt{ORBITER}$ -v 5 \ 
$\texttt{-define G -permutation_group -symmetric_group 13 -end}$ \ 
$\texttt{-with G -do}$ \ 
$\texttt{-group_theoretic_activity}$ \ 
$\texttt{-consecutive_powers "1,2,3,4,5,6,7,8,9,10,11,12,0" 13}$ \ 
$\texttt{-end}$ \ 
$\texttt{pdflatex Perm13_all_powers.tex}$ \ 
$\texttt{open Perm13_all_powers.pdf}$ \ 

$\texttt{Cycle_{12}_power:}$ \ 
$\texttt{ORBITER}$ -v 5 \ 
$\texttt{-define G -permutation_group -symmetric_group 12 -end}$ \ 
$\texttt{-with G -do}$ \ 
$\texttt{-group_theoretic_activity}$ \ 
$\texttt{-consecutive_powers "1,2,3,4,5,6,7,8,9,10,11,0" 12}$ \ 
$\texttt{-end}$ \ 
$\texttt{pdflatex Perm12_all_powers.tex}$ \ 
$\texttt{open Perm12_all_powers.pdf}$ \ 

$\texttt{PGL_{3.4}_singer:}$ \ 
$\texttt{ORBITER}$ -v 5 \ 
$\texttt{-define G -linear_group -PGL 3 4 -end}$
GL_2.8_multiply: \[ \begin{align*} &\text{define G -linear group -GL 2 8 -end} \\
&\text{with G -do} \\
&\text{group_theoretic_activity} \\
&\text{multiply "0,1,2,3" "4,5,6,7"} \\
&\text{end} \\
&\text{pdflatex GL_2.8_mult.tex} \\
&\text{open GL_2.8_mult.pdf} \end{align*} \]

GL_2.7_multiply: \[ \begin{align*} &\text{define G -linear group -GL 2 7 -end} \\
&\text{with G -do} \\
&\text{group_theoretic_activity} \\
&\text{multiply "0,1,2,3" "4,5,6,0"} \\
&\text{end} \\
&\text{pdflatex GL_2.7_mult.tex} \\
&\text{open GL_2.7_mult.pdf} \end{align*} \]

GL_2.7_inv: \[ \begin{align*} &\text{define G -linear group -GL 2 7 -end} \\
&\text{with G -do} \\
&\text{group_theoretic_activity} \\
&\text{inverse "0,1,2,3"} \\
&\text{end} \\
&\text{pdflatex GL_2.7_inv.tex} \\
&\text{open GL_2.7_inv.pdf} \end{align*} \]

GL_2.7_power: \[ \begin{align*} &\text{define G -linear group -GL 2 7 -end} \\
&\text{with G -do} \\
&\text{group_theoretic_activity} \\
&\text{raise_to_the_power "0,1,2,3" 2} \\
&\text{end} \\
&\text{pdflatex GL_2.7_power.tex} \\
&\text{open GL_2.7_power.pdf} \end{align*} \]
PGL_3_2_classes:

$(ORBITER) -v 3 \
define G -linear_group -PGL 3 2 -end \
with G -do \n-group_theoretic_activity \
\> classes_based_on_normal_form \
\> end

pdflatex PGL_3_2_classes_normal_form.tex
open PGL_3_2_classes_normal_form.pdf

#pdflatex PGL_3_2_classes_out.tex
#open PGL_3_2_classes_out.pdf

\> classes \

normal_forms_PGL_4_4:

$(ORBITER) -v 7 \
\> define G -linear_group -PGGL 4 4 -end \
\> with G -do \n\> group_theoretic_activity \
\> \> classes_based_on_normal_form \
\> \> end

pdflatex PGGL_4_4_classes_normal_form.tex
open PGGL_4_4_classes_normal_form.pdf

\> classes \

PGL_4_4_2A_rank:

$(ORBITER) -v 6 \
\> define G -linear_group -PGGL 4 4 -end \
\> with G -do \n\> group_theoretic_activity \n\> \> element_rank \n\> \> "1,0,0,0, 0,1,0,0, 0,0,1,0, 0,0,0,1, 1" \n\> \> end

PGL_4_4_2A_unrank:

$(ORBITER) -v 6 \
\> define G -linear_group -PGGL 4 4 -end \
\> with G -do \n\> group_theoretic_activity \n
563
PGL_4_5_3B_rank:
$\text{define } G \text{ -linear_group -PGL 4 5 -end }$
$\text{with } G \text{ -do }$
$\text{group_theoretic_activity }$
$\text{element_rank } "0,0,0,1, 2,3,0,1, 0,3,4,4, 0,1,2,1"$
$\text{-end}$
PGL_4_5_3B_unrank:
$\text{define } G \text{ -linear_group -PGL 4 5 -end }$
$\text{with } G \text{ -do }$
$\text{group_theoretic_activity }$
$\text{element_unrank } "701459351"$
$\text{-end}$

$\text{normal_forms_PGL_4_5:}$
$\text{define } G \text{ -linear_group -PGL 4 5 -end }$
$\text{with } G \text{ -do }$
$\text{group_theoretic_activity }$
$\text{classes_based_on_normal_form }$
$\text{-end}$
$\text{pdflatex PGL_4_5_classes_normal_form.tex}$
$\text{open PGL_4_5_classes_normal_form.pdf}$

# related to planes_in_pencil:
# we are computing the action on the planes through the line 0.

$\text{on_planes:}$
$\text{define } F \text{ -finite_field -q 8 -end }$
SECTION GROUP THEORETIC ACTIVITIES BASED ON MAGMA:

PGGL_2_4_classes:
  $(ORBITER) -v 3 \$
  define G -linear_group -PGGL 2 4 \$
  -end \$
  with G -do \$
  -group_theoretic_activity \$
  -classes \$
  -end

$((MAGMA_PATH)magma PGGL_2_4_classes.magma

PGGL_2_4_cent_2A:
  $(ORBITER) -v 3 \$

565
4168 ▶ ▶ -define G \\
4169 ▶ ▶ -linear_group -PGGL 2 4 -end \\
4170 ▶ ▶ -with G -do \\
4171 ▶ ▶ -group_theoretic_activity \\
4172 ▶ ▶ ▶ -centralizer_of_element "2A" "1,0, 0,1, 1" \\
4173 ▶ ▶ ▶ -report \\
4174 ▶ ▶ -end \\
4175 ▶ $(MAGMA_PATH)magma element_2A_centralizer.magma \\
4176 ▶ $(ORBITER) -v 6 \\
4177 ▶ ▶ -define G \\
4178 ▶ ▶ -linear_group -PGGL 2 4 -end \\
4179 ▶ ▶ -with G -do \\
4180 ▶ ▶ -group_theoretic_activity \\
4181 ▶ ▶ ▶ -centralizer_of_element "2A" "1,0, 0,1, 1" \\
4182 ▶ ▶ ▶ -report \\
4183 ▶ ▶ -end \\
4184 ▶ pdflatex PGGL_2_4_elt_2A_centralizer.tex \\
4185 ▶ open PGGL_2_4_elt_2A_centralizer.pdf \\
4186 ▶ \\
4187 \\
4188 \\
4189 \\
4190 \\
4191 \\
4192 \\
4193 PGGL_3_4_classes: \\
4194 ▶ $(ORBITER) -v 3 \\
4195 ▶ ▶ -define G \\
4196 ▶ ▶ -linear_group -PGGL 3 4 \\
4197 ▶ ▶ -end \\
4198 ▶ ▶ -with G -do \\
4199 ▶ ▶ -group_theoretic_activity \\
4200 ▶ ▶ ▶ -classes \\
4201 ▶ ▶ ▶ -end \\
4202 ▶ pdflatex PGGL_3_4_classes_out.tex \\
4203 ▶ open PGGL_3_4_classes_out.pdf \\
4204 \\
4205 \\
4206 \\
4207 \\
4208 \\
4209 \\
4210 \\
4211 classes_PGGL_4_4: \\
4212 ▶ $(ORBITER) -v 3 \\
4213 ▶ ▶ -magma_path $(MAGMA_PATH) \\
4214 ▶ ▶ -define G \\

566
group order 1974067200 = $2^{13} \ast 3^4 \ast 5^2 \ast 7 \ast 17$

# the -find_subgroup command is too specialized

subgroups_PGL_4_5:
$($ORBITER$) -v 6$
$-define G$
$-linear_group -PGL 4 5 -end$
$-with G -do$
$-group_theoretic_activity$
$-find_subgroup 3$
$-end$
pdflatex PGL_4_5_report.tex
open PGL_4_5_report.pdf

classes_PGL_4_5:
$($ORBITER$) -v 6$
$-define G$
$-linear_group -PGL 4 5 -end$
$-with G -do$
$-group_theoretic_activity$
$-classes$
$-end$
pdflatex PGL_4_5_classes_out.tex
open PGL_4_5_classes_out.pdf

# 163 classes

# two classes of elements of order 3
#Order of element = 3 Class size = 310000 Centralizer order = 93600 Normalizer order = 187200
# of order 3 and with 0 fixed points.
#0,1,0,2,0,1,2,1,4,2,3,1,2,0,4,3,

#Class size = 10075000 Centralizer order = 2880 Normalizer order = 5760
4261 # of order 3 and with 6 fixed points.
4262 #0,0,0,1,2,3,0,1,0,3,4,4,0,1,2,1,
4263
4264 PGL_4_5_3B_class_again:
4265 
4266 PGL_4_5_3B_class_again:
4267 $(ORBITER) -v 6 -define G 
4268 -linear_group -PGL 4 5 -end 
4269 -with G -do 
4270 -group_theoretic_activity 
4271 -conjugacy_class_of 
4272 -end
4273
4274 search_primitive_poly_q5_deg3:
4275 $(ORBITER) -v 6 
4276 -search_for_primitive_polynomial_in_range 5 5 3 3
4277
4278 #OK, we found an irreducible and primitive polynomial X^3 + X^2 + 2
4279
4280 GL_3_5.singer_power:
4281 $(ORBITER) -v 6 -define G 
4282 -linear_group -GL 3 5 -end 
4283 -with G -do 
4284 -group_theoretic_activity 
4285 -raise_to_the_power 
4286 "0,1,0, 0,0,1, 3,0,4" 31 
4287 -end
4288 pdflatex GL_3_5.power.tex
4289 open GL_3_5.power.pdf
4290
4291 PGL_4_5_norm_31:
4292 $(ORBITER) -v 6 -define G 
4293 -linear_group -PGL 4 5 -end 
4294 -with G -do 
4295 -group_theoretic_activity 
4296 -normalizer_of_cyclic_subgroup "31" 
4297 "2,0,0,0, 0,0,1,0, 0,0,0,1, 0,3,0,4"
4298 -end
4299 pdflatex normalizer_of_31_in_PGL_4_5.tex
4300 open normalizer_of_31_in_PGL_4_5.pdf
4301
4302
4303
4304 Normalizer_of_Z22_in_PGL_2_9:
4305 $(ORBITER) -v 2 
4306 -define G -linear_group -PGL 2 9 
4307 -subgroup_by_generators Z22 4 2 

SECTION ORBIT ALGORITHMS SCHREIER TREES:

T3r1_orbits:
$\text{(ORBITER)} -v 4 \$

\begin{verbatim}
define G 
-linear_group -GL_d_q_wr_Sym_n 2 2 3 
-on_rank_one_tensors -end 
-with G -do 
-groups_theoretic_activity 
-report 
-orbits_on_points 
-export_trees 
-end
\end{verbatim}

\begin{verbatim}
pdflatex GL_2_2_wreath_Sym3_orbits_report.tex 
open GL_2_2_wreath_Sym3_orbits_report.pdf
\end{verbatim}

T3r1_orbits_draw:
$\text{(ORBITER)} -v 3 \$

\begin{verbatim}
draw_layered_graph 
-GL_2_2_wreath_Sym3_res27_0.layered_graph 
-radius 500 -spanning_tree -embedded 
-line_width 1.1 -x_stretch 1.4 -scale 0.25 
-end
\end{verbatim}

\begin{verbatim}
#pdflatex GL_2_2_wreath_Sym3_report.tex 
#open GL_2_2_wreath_Sym3_report.pdf 
pdflatex GL_2_2_wreath_Sym3_res27_0_draw.tex 
open GL_2_2_wreath_Sym3_res27_0_draw.pdf
\end{verbatim}
2C_orbit_under_PGGL_4_4_elements_coded.csv:
$\$(ORBITER) -v 6 \n$\n-definition G -linear_group -PGGL 4 4 -end \n$\n-definition G -do \n$\n-definition group_theoretic_activity \n$\n-definition conjugacy_class_of_element \n$\n-definition "2C" "1,0,0,0, 1,1,0,0, 0,0,1,0, 0,0,1,1, 0" \n$\n-definition -end \n$\n-definition # class of size 64260
$\n-definition # creates:
$\n-definition # 2C_orbit_under_PGGL_4_4.csv
$\n-definition # 2C_orbit_under_PGGL_4_4.txt
$\n-definition # 2C_orbit_under_PGGL_4_4_elements_coded.csv
$\n-definition # 2C_orbit_under_PGGL_4_4_transporter.csv
$\n-definition # 1:33 on Mac
$\n-definition #User time: 2:59 on Mac
$\n-definition PGGL_4_4_subgroups_of_type_2C_2C: 2C_orbit_under_PGGL_4_4_elements_coded.csv
$\n-definition $\$(ORBITER) -v 6 \n$\n-definition $\n-definition -subgroup_by_generators "centralizer_2C" "30720" 9 \n$\n-definition $\n-definition $\n-definition "1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1," \n$\n-definition $\n-definition "1,0,0,0,0,1,0,0,0,0,2,0,0,0,0,2,1," \n$\n-definition $\n-definition "1,0,0,0,0,1,0,0,0,0,1,0,2,0,3,1,0," \n$\n-definition $\n-definition "1,0,0,0,0,1,0,0,0,0,1,0,1,0,3,1,0," \n$\n-definition $\n-definition "1,0,0,0,0,1,0,0,0,0,1,0,1,0,1,1,1," \n$\n-definition $\n-definition "1,0,0,0,0,1,1,0,0,0,0,1,0,0,0,1,1,0," \n$\n-definition $\n-definition "1,0,0,0,2,1,0,0,0,0,1,0,0,0,1,0,1,0," \n$\n-definition $\n-definition "1,0,0,0,1,1,2,0,0,0,1,0,0,0,0,1,0,1,0," \n$\n-definition $\n-definition "1,0,3,0,1,1,3,0,0,2,0,0,0,0,2,1," \n$\n-definition $\n-definition $\n-definition \n$\n-definition $\n-definition $\n-definition with G -do \n$\n-definition $\n-definition $\n-definition group_theoretic_activity \n$\n-definition $\n-definition $\n-definition orbits_on_group_elements_under_conjugation \n$\n-definition $\n-definition $\n-definition 2C_orbit_under_PGGL_4_4_elements_coded.csv \n$\n-definition $\n-definition $\n-definition 2C_orbit_under_PGGL_4_4_transporter.csv \n$\n-definition $\n-definition $\n-definition $\n-definition $\n-definition open subgroups_of_order_4.pdf
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 )

User time: 0:57

orbits_on_conics_q13:

```
$\text{ORBITER} -v 4 \
\texttt{-define G -linear_group -PGL 3 13 -end} \\
\texttt{-with G -do} \\
\texttt{-group_theoretic_activity} \\
\texttt{-orbits_on_polynomials 2} \\
\texttt{-end}
```

```
pdflatex poly_orbits_d2n2q13.tex
open poly_orbits_d2n2q13.pdf
```

orbits_cubic_curves_q2:

```
$\text{ORBITER} -v 4 \
\texttt{-define G -linear_group -PGL 3 2 -end} \\
\texttt{-with G -do} \\
\texttt{-group_theoretic_activity} \\
\texttt{-orbits_on_polynomials 3} \\
\texttt{-end}
```

```
pdflatex poly_orbits_d3n3q2.tex
open poly_orbits_d3n3q2.pdf
```

orbits_cubic_curves_q2_with_draw_tree:

```
$\text{ORBITER} -v 4 \
\texttt{-draw_options -yout 500000 -radius 15 -nodes_empty} \\
\texttt{-line_width 0.5 -y_stretch 0.25 -embedded -end} \\
\texttt{-define G -linear_group -PGL 3 2 -end} \\
\texttt{-with G -do} \\
\texttt{-group_theoretic_activity} \\
\texttt{-orbits_on_polynomials 3} \\
\texttt{-orbits_on_polynomials.draw_tree 6} \\
\texttt{-end}
```

```
poly_orbits_d3n3q2.csv:
$\text{ORBITER} -v 4 \
\texttt{-draw_options -yout 500000 -radius 15 -nodes_empty} \
```
poly.orbits_d3.n3.q2.get.ranks:
\$(ORBITER) -v 4 \n\$csv_file_select.cols poly.orbits_d3.n3.q2.csv 0
\$pdflatex poly.orbits_d3.n3.q2.tex
\$open poly.orbits_d3.n3.q2.pdf

T4.orbits:
\$(ORBITER) -v 4 \n\$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\$orbits.on.points \n\$end
\$pdflatex GL.2.2.wreath.Sym4.res65535.orbits.tex
\$open GL.2.2.wreath.Sym4.res65535.orbits.pdf
\$pdflatex GL.2.2.wreath.Sym4.report.tex
\$open GL.2.2.wreath.Sym4.report.pdf

T4r1.orbits:
\$(ORBITER) -v 4 \n\$define G -linear_group -GL.d.q.wr.Sym.n 2 2 4 \n\$on.rank_one.tensors -end \n\$with G -do \n\$group.theoretic.activity \n\$orbits_on.points \n\$export.trees \n\$report \n\$end
\$pdflatex GL.2.2.wreath.Sym4.orbits_report.tex
\$open GL.2.2.wreath.Sym4.orbits_report.pdf
T4r1.orbits.draw:
$\$(ORBITER) -v 3 \n-draw_layered_graph \nGL_2.2_wreath_Sym4_res81_0.layered_graph \nradius 400 -spanning_tree -embedded \n-line_width 1.1 -x_stretch 2.5 -scale 0.15 \n-end\n
#pdflatex GL_2.2_wreath_Sym3_report.tex
#open GL_2.2_wreath_Sym3_report.pdf
pdflatex GL_2.2_wreath_Sym4_res81_0.draw.tex
open GL_2.2_wreath_Sym4_res81_0_draw.pdf

T4r1.orbits.4:
$\$(ORBITER) -v 4 \n-orbiter_path $(ORBITER_PATH) \n-linear_group -GL_d_q_wr_Sym n 2 2 4 \non_rank_one_tensors -end \n-with G -do \ngroup_theoretic_activity \n-poset_classification_control -problem_label T4r1_W \ndraw_options -end -draw_poset -report -end \n-end \n-orbits_on_subsets 4 \n-report \n-end\n
pdflatex T4r1_poset.tex
open T4r1_poset.pdf

PGGL_2.8_on_conic.orbits:
$\$(ORBITER) -v 4 \n-linear_group -PGGL 2 8 -PGL20nConic -end \n-with G -do \ngroup_theoretic_activity \n-orbits_on_points\n-report \n-end\n
pdflatex PGGL_2.8_onConic_2.8_orbits_report.tex
open PGGL_2.8_onConic_2.8_orbits_report.pdf
example from the Fining manual, page 107:

```
PGGL_7_8_orbits:

$(ORBITER) -v 4 \n
-define G \n
-linear_group -PGGL 7 8 -end \n
-with G -do \n
-group_theoretic_activity \n
-report \n
-orbits_on_points \n
-end

# 1 min 31 sec on Mac

SECTION_POSET_CLASSIFICATION:

poset_of_4subsets:

$(ORBITER) -v 3 \n
-orbiter_path $(ORBITER_PATH) \n
-linear_group -PGL 2 3 -identity_group -end \n
-with G -do \n
-group_theoretic_activity \n
-poset_classification_control \n
-problem_label poset_4 \n
-\n
-\n
-depth 4 \n
-draw_options -radius 200 -end \n
-report -end \n
-draw_poset \n
-end \n
-orbits_on_subsets 4 \n
-report \n
-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 \n
-draw_layered_graph \n
-poset_4_poset_lvl_4.layered_graph 
```
poset_of_5subsets:
$(ORBITER) -v 3 \
-orbiter_path $(ORBITER_PATH) \
-define G -linear_group -PGL 2 4 -identity_group -end \
-with G -do \
-group_theoretic_activity \
-poset_classification_control \
-problem_label poset_5 \
-W -depth 5 -draw_options -radius 150 -end \
-report -end -draw_poset -end \
-orbits_on_subsets 5 \
-report \
-end
poset_of_5subsets_draw:
$(ORBITER) -v 3 \
-draw_layered_graph \
-poset_5_poset_lvl_5.layered_graph \
-radius 300 -embedded \
-line_width 1.1 -y_stretch 0.9 \
-scale 0.25 \
-end
Symmetric_4_on_pairs_poset:
$(ORBITER) -v 3 \
-orbiter_path $(ORBITER_PATH) \
-define G -permutation_group -symmetric_group 4 -end \
-define G.on.2subsets -modified_group -from G \
-on_k_subsets 2 \
-end \
-with G.on.2subsets -do \
-group_theoretic_activity \
-problem_label Sym4.on2 \

V.3.2_trivial:

```bash
$\text{orbiter}\ path\ $(\text{ORBITER}\ \text{PATH})$

#define\ G\ -linear\ group\ -PGL\ 3\ 2\ -identity\ group\ -end\

-with\ G\ -do\ 

group_theoretic_activity\

-poset_classification_control\

-problem_label\ V.3.2_trivial\

-W\ -depth\ 3\ -node_label_is_element\

-radius\ 200\ -embedded\

-end\

-report\ -end\

draw_options\

-end\

-orbits_on_subspaces\ 3\

-report\

-end\

#$\text{ORBITER}\ -v\ 5$

-draw_layered_graph\ PGL.3.2\_Identity.3.2\_poset.lvl.3.layered_graph\

-radius\ 300\ -embedded\ -line_width\ 1.1\ -y\_stretch\ 0.9\ -scale\ 0.25\

-end\

#pdflatex\ PGL.3.2\_Identity.3.2\_report.tex

#open\ PGL.3.2\_Identity.3.2\_report.pdf

#pdflatex\ PGL.3.2\_Identity.3.2\_poset.lvl.3\_draw.tex

#open\ PGL.3.2\_Identity.3.2\_poset.lvl.3\_draw.pdf

pdflatex\ PGL.3.2\_Identity.3.2\_poset.tex

open\ PGL.3.2\_Identity.3.2\_poset.pdf
```

V.4.2_trivial:

```bash
$\text{orbiter}\ path\ $(\text{ORBITER}\ \text{PATH})$

#define\ G\ -linear\ group\ -PGL\ 4\ 2\ -identity\ group\ -end\

-with\ G\ -do\ 

group_theoretic_activity\

```
DIR

  \( \text{poset\_classification\_control} \)
  \( \text{poset\_classification\_control} \)
  \( \text{problem\_label} \text{~} V_{4,2} \text{~} \text{trivial} \)
  \( \text{W} \text{~} \text{depth~} 3 \text{~} \text{node\_label\_is\_element} \)
  \( \text{radius~} 200 \text{~} \text{embedded} \)

DIR

  \( \text{draw\_options} \)
  \( \text{report} \)

DIR

  \( \text{draw\_poset} \)

DIR

  \( \text{orbits\_on\_subspaces} \)

DIR

  \#pdflatex PGL_4_2_Identity_4_2_report.tex
  \#open PGL_4_2_Identity_4_2_report.pdf
  \#pdflatex PGL_4_2_Identity_4_2_tree_lvl_4.tex
  \#open PGL_4_2_Identity_4_2_tree_lvl_4.pdf

DIR

  # Section 6.3: Orbits on Subsets
  
  SECTION_ORTBITS_ON_SUBSETS:
  
  PG_2_2_subsets:
  
  \$\text{ORBITER} -v 3 \$
  \$\text{orbiter\_path} \$\text{ORBITER\_PATH} \$
  \$\text{define F} \text{-finite\_field} -q 2 \text{-end} \$
  \$\text{define G} \text{-linear\_group} -\text{PGL~3~F} \text{-end} \$
  \$\text{with G} \text{-do} \$
  \$\text{group\_theoretic\_activity} \$
  \$\text{poset\_classification\_control} \$
  \$\text{problem\_label PGL_3_2} \$
  \$\text{depth~} 7 \$
  \$\text{radius~} 200 \text{-embedded} \$
  \$\text{report} \text{-end} \$
  \$\text{draw\_poset} \$
  \$\text{report} \$

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:

\begin{verbatim}
$\text{(ORBITER)} -v 3 \$
\end{verbatim}

\begin{verbatim}
$\text{-orbiter_path $(ORBITER_PATH)$} \$
\end{verbatim}

\begin{verbatim}
$\text{-define F -finite_field -q 2 -end}$
\end{verbatim}

\begin{verbatim}
$\text{-define G -linear_group -PGL 4 F -end}$
\end{verbatim}

\begin{verbatim}
$\text{-with G -do}$
\end{verbatim}

\begin{verbatim}
$\text{-group_theoretic_activity}$
\end{verbatim}

\begin{verbatim}
$\text{-poset_classification_control}$
\end{verbatim}

\begin{verbatim}
$\text{-problem_label PGL_4_2}$
\end{verbatim}

\begin{verbatim}
$\text{-depth 15}$
\end{verbatim}

\begin{verbatim}
$\text{-draw_options}$
\end{verbatim}

\begin{verbatim}
$\text{-radius 200 -embedded}$
\end{verbatim}

\begin{verbatim}
$\text{-end}$
\end{verbatim}

\begin{verbatim}
$\text{-report -end}$
\end{verbatim}

\begin{verbatim}
$\text{-draw_poset}$
\end{verbatim}

\begin{verbatim}
$\text{-end}$
\end{verbatim}

\begin{verbatim}
$\text{-orbits_on_subsets 15}$
\end{verbatim}

\begin{verbatim}
$\text{-report}$
\end{verbatim}

\begin{verbatim}
$\text{-end}$
\end{verbatim}

\begin{verbatim}
$\text{pdflatex PGL_4_2_poset.tex}$
\end{verbatim}

\begin{verbatim}
$\text{open PGL_4_2_poset.pdf}$
\end{verbatim}

PG(3,2) singer:

\begin{verbatim}
$\text{(ORBITER)} -v 3$
\end{verbatim}

\begin{verbatim}
$\text{-orbiter_path $(ORBITER_PATH)$}$
\end{verbatim}

\begin{verbatim}
$\text{-define G -linear_group -PGL 3 2 -singer 1 -end}$
\end{verbatim}

\begin{verbatim}
$\text{-with G -do}$
\end{verbatim}

\begin{verbatim}
$\text{-group_theoretic_activity}$
\end{verbatim}

\begin{verbatim}
$\text{-poset_classification_control}$
\end{verbatim}

\begin{verbatim}
$\text{-problem_label PGL_3_2_singer_1 -W -depth 7}$
\end{verbatim}

\begin{verbatim}
$\text{-draw_poset}$
\end{verbatim}

\begin{verbatim}
$\text{-report -end}$
\end{verbatim}
PGL\_3\_2\_singer\_1\_poset.tex

PGL\_3\_2\_lines\_poset.tex
open PGL_2_5_poset.pdf

PGL_2_7_on_subsets:
$ (ORBITER) -v 10 \
- orbiter_path $(ORBITER_PATH) \
- define G -linear_group -PGL 2 7 -end \
- with G -do \
- group_theoretic_activity \
- poset_classification_control \
- problem_label PGL_2_7 -W -depth 8 \
- draw_poset \
- draw_options -radius 200 -end \
- report -end \
- orbits_on_subsets 8 \
- report \
- end \
\pdflatex PGL_2_7_poset.tex \
open PGL_2_7_poset.pdf

PGGL_2_8_on_subsets:
$ (ORBITER) -v 10 \
- orbiter_path $(ORBITER_PATH) \
- define G -linear_group -PGGL 2 8 -end \
- with G -do \
- group_theoretic_activity \
- poset_classification_control \
- problem_label PGGL_2_8 -W -depth 9 \
- draw_poset \
- draw_options -radius 200 -end \
- report -end \
- orbits_on_subsets 9 \
- report \
- end \
\pdflatex PGGL_2_8_poset.tex \
open PGGL_2_8_poset.pdf

PGGL_2_9_on_subsets:
$ (ORBITER) -v 10 \
- orbiter_path $(ORBITER_PATH) \
- define G -linear_group -PGGL 2 9 -end \
- with G -do \
- group_theoretic_activity \
- poset_classification_control \
- problem_label PGGL_2_9 -W -depth 10 \

580
PGL_2_11_on_subsets:

$\text{(ORBITER)} -v 10 \$

- orbiter_path $(\text{ORBITER} \_\text{PATH})$

- define G -linear_group -PGL 2 11 -end

- with G -do

- group_theoretic_activity

- poset_classification_control

- problem_label PGL_2_11 -W -depth 12

- draw_poset

- draw_options -radius 200 -end

- report -end

- end

- orbits_on_subsets 12

- report

- end

- orbits_on_subsets 10

- report

- end

pdflatex PGL_2_11_poset.tex

open PGL_2_11_poset.pdf


PGGL_2_16_on_subsets:

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

- orbiter_path $(\text{ORBITER} \_\text{PATH})$

- define G -linear_group -PGGL 2 16 -end

- with G -do

- group_theoretic_activity

- poset_classification_control

- problem_label PGGL_2_16 -W -depth 10

- draw_poset

- draw_options -radius 200 -end

- report -end

- end

- orbits_on_subsets 10

- report

- end

pdflatex PGGL_2_16_poset.tex

open PGGL_2_16_poset.pdf
PGGL_2.32_on_subsets:
\$\text{ORBITER} \ -v \ 3 \$
\text{\texttt{-orbiter\_path}} \ \text{\texttt{\$(ORBITER\_PATH)\}}
\text{\texttt{-define G -linear\_group -PGGL 2 32 \-end}}
\text{\texttt{-with G \-do}}
\text{\texttt{-group\_theoretic\_activity}}
\text{\texttt{-poset\_classification\_control}}
\text{\texttt{-problem\_label PGGL_2.32 \-W \-depth 8}}
\text{\texttt{-draw\_poset}}
\text{\texttt{-report \-end}}
\text{\texttt{-end}}
\text{\texttt{-orbits\_on\_subsets 8 \-end}}
\text{\texttt{-report \-end}}
pdflatex PGGL_2.32_poset.tex
open PGGL_2.32_poset.pdf

PG_3.4_subsets:
\$\text{ORBITER} \ -v \ 3 \$
\text{\texttt{-orbiter\_path}} \ \text{\texttt{\$(ORBITER\_PATH)\}}
\text{\texttt{-define G -linear\_group -PGGL 4 4 \-end}}
\text{\texttt{-with G \-do}}
\text{\texttt{-group\_theoretic\_activity}}
\text{\texttt{-poset\_classification\_control}}
\text{\texttt{-problem\_label PGGL_4.4}}
\text{\texttt{-depth 5 \-depth 5 \-draw\_poset}}
\text{\texttt{-draw\_options \-radius 200}}
\text{\texttt{-report \-end \-end}}
\text{\texttt{-orbits\_on\_subsets 8 \-end}}
\text{\texttt{-report \-end}}
pdflatex PGGL_4.4_poset.tex
open PGGL_4.4_poset.pdf

PGGL_2.9_orbits:
\$\text{ORBITER} \ -v \ 3 \$
\text{\texttt{-orbiter\_path}} \ \text{\texttt{\$(ORBITER\_PATH)\}}
\text{\texttt{-define G -linear\_group -PGGL 2 9 \-end}}
\text{\texttt{-with G \-do}}

582
# Section 6.4: Orbits on Subspaces

SECTION ORBITS ON SUBSPACES:

subspaces_0p_4_2:
5011 $\$(\text{ORBITER})\ -v\ 5$ \\
5012 -orbiter_path $\$(\text{ORBITER}\_\text{PATH})\$
5013 -define G -linear_group -PGL 4 2 -orthogonal 1 -end \\
5014 -with G -do \\
5015 -group_theoretic_activity \\
5016 -poset_classification_control \\
5017 -node_label_is_element \\
5018 -draw_poset -draw_options -radius 200 -end \\
5019 -problem_label Op_{4,2} -W -depth 4 \\
5020 -report -end \\
5021 -end \\
5022 -orbits_on_subspaces 4 \\
5023 -report \\
5024 -end \\
5025 pdflatex PGL_{4,2}.Orthogonal\_plus_{4,2}.poset.tex \\
5026 open PGL_{4,2}.Orthogonal\_plus_{4,2}.poset.pdf \\
5027
5028
5029 PGL_{4,2}.on_subspaces: \\
5030 $\$(\text{ORBITER})\ -v\ 5$ \\
5031 -orbiter_path $\$(\text{ORBITER}\_\text{PATH})\$
5032 -define G -linear_group -PGL 4 2 -end \\
5033 -with G -do \\
5034 -group_theoretic_activity \\
5035 -poset_classification_control \\
5036 -node_label_is_element \\
5037 -draw_poset -draw_options -radius 200 -end \\
5038 -problem_label PGL_{4,2} -W -depth 4 \\
5039 -report -end \\
5040 -orbits_on_subspaces 4 \\
5041 -report \\
5042 end \\
5043 pdflatex PGL_{4,2}.report.tex \\
5044 open PGL_{4,2}.report.pdf \\
5045 pdflatex PGL_{4,2}.poset.tex \\
5046 open PGL_{4,2}.poset.pdf \\
5047
5048
5049 PGL_{4,2}.singer_on_subspaces: \\
5050 $\$(\text{ORBITER})\ -v\ 5$ \\
5051 -orbiter_path $\$(\text{ORBITER}\_\text{PATH})\$
5052 -define G -linear_group -PGL 4 2 -singer 1 -end \\
5053 -with G -do \\
5054 -group_theoretic_activity \\
5055 -node_label_is_element \\
5056 -draw_poset
Op_6_2_orbits_on_subspaces:

Op_6_2_orbits_on_subspaces:

Op_6_2_orbits_on_subspaces:

Op_6_2_orbits_on_subspaces:

Op_6_2_orbits_on_subspaces:

Op_6_2_orbits_on_subspaces:

Op_6_2_orbits_on_subspaces:

Op_6_2_orbits_on_subspaces:

Op_6_2_orbits_on_subspaces:

Op_6_2_orbits_on_subspaces:

Op_6_2_orbits_on_subspaces:

Op_6_2_orbits_on_subspaces:

Op_6_2_orbits_on_subspaces:
Op_6.3_orbits_on_subspaces:

$\text{(ORBITER) -v 5 \ }
\text{orbiter_path (ORBITER_PATH) \ }
\text{define G -linear_group -PGL 6 3 -orthogonal 1 -end \ }
\text{with G -do \ }
\text{group_theoretic_activity \ }
\text{poset_classification_control \ }
\text{node_label_is_element \ }
\text{draw_poset \ }
\text{draw_options -radius 200 -end \ }
\text{problem_label Op_6.3 -W \ }
\text{depth 6 -report -end \ }
\text{orbits_on_subspaces 6 \ }
\text{report \ }
\text{end \ }

Op_6.11_orbits_on_subspaces:

$\text{(ORBITER) -v 5 \ }
\text{orbiter_path (ORBITER_PATH) \ }
\text{draw_options -nodes_empty -end \ }
\text{define G -linear_group -PGL 6 11 -orthogonal 1 -end \ }
\text{with G -do \ }
\text{group_theoretic_activity \ }
\text{poset_classification_control \ }
\text{node_label_is_element \ }
\text{draw_poset \ }
\text{draw_options -radius 200 -end \ }
\text{problem_label Op_6.11 -W \ }
\text{depth 6 -report -end \ }
\text{end \ }
5152 ▷ ▷ -orbits_on_subspaces 6 \n5153 ▷ ▷ -report \n5154 ▷ ▷ -end
5155 ▷ ▷ pdflatex PGL_6.11_Orthogonal_plus_6.11_report.tex
5156 ▷ ▷ open PGL_6.11_Orthogonal_plus_6.11_report.pdf
5157
5158
5159 # June 3, 2020 on Mac: 12 sec
5160
5161 Op_8.2_orbits_on_subspaces:
5162 ▷ $(ORBITER) -v 5 \n5163 ▷ ▷ -orbiter_path $(ORBITER_PATH) \n5164 ▷ ▷ ▷ -define G -linear_group -PGL 8 2 -orthogonal 1 -end \n5165 ▷ ▷ ▷ -with G -do \n5166 ▷ ▷ ▷ -group_theoretic_activity \n5167 ▷ ▷ ▷ ▷ -poset_classification_control \n5168 ▷ ▷ ▷ ▷ ▷ -node_label_is_element \n5169 ▷ ▷ ▷ ▷ ▷ ▷ -draw_poset -draw_options -radius 200 -end \n5170 ▷ ▷ ▷ ▷ ▷ ▷ ▷ -problem_label Op_8.2 -W -depth 8 -report -end \n5171 ▷ ▷ ▷ ▷ ▷ ▷ ▷ -end \n5172 ▷ ▷ ▷ -orbits_on_subspaces 8 \n5173 ▷ ▷ ▷ -report \n5174 ▷ ▷ ▷ -end
5175 ▷ ▷ -end
5176 ▷ ▷ pdflatex PGL_8.2_Orthogonal_plus_8.2_poset.tex
5177 ▷ ▷ open PGL_8.2_Orthogonal_plus_8.2_poset.pdf
5178
5179
5180 PGO_7.2_on_subspaces:
5181 ▷ $(ORBITER) -v 20 \n5182 ▷ ▷ -orbiter_path $(ORBITER_PATH) \n5183 ▷ ▷ ▷ -define F -finite_field -q 2 -end \n5184 ▷ ▷ ▷ -define G -linear_group -PGL 7 F -orthogonal 0 -end \n5185 ▷ ▷ ▷ -with G -do \n5186 ▷ ▷ ▷ -group_theoretic_activity \n5187 ▷ ▷ ▷ ▷ -poset_classification_control \n5188 ▷ ▷ ▷ ▷ ▷ -node_label_is_element \n5189 ▷ ▷ ▷ ▷ ▷ ▷ -draw_poset \n5190 ▷ ▷ ▷ ▷ ▷ ▷ ▷ -draw_options -radius 200 -end \n5191 ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ -problem_label 0.7.2 \n5192 ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ -W -depth 7 \n5193 ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ -report -end \n5194 ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ -end \n5195 ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ -orbits_on_subspaces 7 \n5196 ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ -report \n5197 ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ -end
5198 ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ -end

587
# Section 6.5: Orbits on set partitions

SECTION ORBITS ON SET PARTITIONS:

C6 on partition:

```bash
$ (ORBITER) -v 5 \ 
-define G -permutation_group -cyclic_group 6 -end \ 
-with G -do \ 
-group_theoretic_activity \ 
-poset_classification_control \ 
-problem_label C6 \ 
-depth 2 \ 
-radius 200 -embedded \ 
-end \ 
-orbits_on_partition 2 \ 
-end
```

PGL_2.17 on partition:

```bash
$ (ORBITER) -v 5 \ 
-define G -linear_group -PGL 2 17 -end \ 
-with G -do \ 
-group_theoretic_activity \ 
-poset_classification_control \ 
-problem_label PGL_2.17 \ 
-depth 6 \ 
-radius 200 -embedded \ 
-end \ 
-orbits_on_partition 6 \ 
-end
```
Section 6.6: Arcs and Caps in Projective Spaces

SECTION_ARCS_AND_CAPS_IN_PROJECTIVE_SPACES:

PGL_3_27:
\$(ORBITER) -v 5 \n\$-define G \n\$-linear_group -PGL 3 27 -end \n\$-with G -do \n\$-group_theoretic_activity \n\$-report \n\$-end
\$pdflatex PGL_3_27_report.tex
\$open PGL_3_27_report.pdf

AGGL_2_27:
\$(ORBITER) -v 5 \n\$-define G \n\$-linear_group -AGGL 2 27 -end \n\$-with G -do \n\$-group_theoretic_activity \n\$-report \n\$-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 \n\$-define P -projective_space 2 F -end \n\$-with P -do \n\$-projective_space_activity \n\$-classify_arcs \n\$-poset_classification_control \n\$-problem_label hyperoval_q4 \n\$-W -depth 6 \n\$-report -end \n\$-end
hyperoval_8_classify:

$\$(ORBITER) \ -v 4 \\n\$ORBITER_PATH \end \\
-compose_field -q 8 -end \end \\
-compose_space 2 F -end \end \\
-with P -do \end \\
-compose_space_activity \end \\
-compose_arcs \end \\
-compose_classification_control \end \\
-compose_problem_label hyperoval_q8 \end \\
-W -depth 10 \end \\
-report -end \end \\
-draw_poset \end \\
-draw_options \end \\
-end \end \\
-target_size 10 \end \\
-d 2 \end \\
-end \end \\
-compose_poset \end \\
-compose_options \end \\
-end \\
-compute hyperoval_q8_poset.tex \\
-open hyperoval_q8_poset.pdf \\

frame_stabilizer_PGGL:

$\$(ORBITER) \ -v 4 \\n-compose G \end \\
-compose_linear_group -PGGL 3 8 -end \end \\
-with G -do \end \\
-compose_theoretic_activity \end \\
-compose_classification_control \end \\
-compose_problem_label frame_q8 -W -depth 4 \end \\
-draw_options -end \end \\
-report -end \end \\

590
5340 ▶ ▶ ▶ ▶ -end \n5341 ▶ ▶ ▶ -classify_arcs \n5342 ▶ ▶ ▶ ▶ -target_size 4 \n5343 ▶ ▶ ▶ ▶ -q 8 \n5344 ▶ ▶ ▶ ▶ -n 3 \n5345 ▶ ▶ ▶ ▶ -d 2 \n5346 ▶ ▶ ▶ -end \n5347 ▶ ▶ -end \n5348 ▶ pdflatex frame_q8_poset.tex \n5349 ▶ open frame_q8_poset.pdf \n5350
5351 \frame_stabilizer_PGL:\n5352 ▶ $(ORBITER) -v 4 \n5353 ▶ ▶ -define G \n5354 ▶ ▶ -linear_group -PGL 3 8 -end \n5355 ▶ ▶ -with G -do \n5356 ▶ ▶ -group_theoretic_activity \n5357 ▶ ▶ ▶ ▶ -poset_classification_control \n5358 ▶ ▶ ▶ ▶ -problem_label frame_q8 -W -depth 4 \n5359 ▶ ▶ ▶ ▶ -draw_options -end \n5360 ▶ ▶ ▶ ▶ -report -end \n5361 ▶ ▶ ▶ ▶ -end \n5362 ▶ ▶ ▶ -classify_arcs \n5363 ▶ ▶ ▶ ▶ -target_size 4 \n5364 ▶ ▶ ▶ ▶ -q 8 \n5365 ▶ ▶ ▶ ▶ -n 3 \n5366 ▶ ▶ ▶ ▶ -d 2 \n5367 ▶ ▶ ▶ -end \n5368 ▶ ▶ -end \n5369 ▶ pdflatex frame_q8_poset.tex \n5370 ▶ open frame_q8_poset.pdf \n5371
5372
5373
5374 hyperoval_16_classify:  
5375 ▶ $(ORBITER) -v 4 \n5376 ▶ ▶ -orbiter_path $(ORBITER_PATH) \n5377 ▶ ▶ -define F -finite_field -q 16 -end \n5378 ▶ ▶ -define P -projective_space 2 F -end \n5379 ▶ ▶ -with P -do \n5380 ▶ ▶ -projective_space_activity \n5381 ▶ ▶ ▶ -classify_arcs \n5382 ▶ ▶ ▶ ▶ -poset_classification_control \n5383 ▶ ▶ ▶ ▶ ▶ -problem_label_hyperoval_q16 -W -depth 18 \n5384 ▶ ▶ ▶ ▶ ▶ -report -end \n5385 ▶ ▶ ▶ ▶ ▶ -end \n5386 ▶ ▶ ▶ ▶ ▶ -target_size 18

591
We found 17028 non-conical 6 subsets.
# Eckardt point number distribution: \$13^{252}, \$9^{720}, \$5^{2304}, \$3^{13752}\$

5437 hyperoval_16_2_nonconical_type:
5438 \$\text{(ORBITER)} -v 2 \$
5439 \$\text{define F -finite_field -q 16 -end} \$
5440 \$\text{define P -projective_space 2 F -end} \$
5441 \$\text{define H_{16,2} -geometric_object P} \$
5442 \$\text{-set $(HYPEROVAL_{16,16320})} \$
5443 \$\text{-end} \$
5444 \$\text{-with H_{16,2} -do} \$
5445 \$\text{-combinatorial_object_activity} \$
5446 \$\text{-save} \$
5447 \$\text{-end} \$
5448 \$\text{-with H_{16,2} -do} \$
5449 \$\text{-combinatorial_object_activity} \$
5450 \$\text{-non_conical_type} \$
5451 \$\text{-end} \$
5452 \$\text{-print_symbols} \$
5453
5454 # We found 6188 = \{17 choose 5\} non-conical 6 subsets
5455 # Eckardt point number distribution: \$45^{68}, \$13^{2040}, \$5^{4080}\$
5456
5457 # neighbors_of_0_with_4_removed.csv
5458 # Row,C0,C1,C2,C3
5459 #0,2,3,9,10
5460 #1,1,3,7,8
5461 #2,10,12,13,15
5462 #3,1,5,10,11
5463 #4,3,5,6,13
5464 #5,8,9,11,12
5465 #6,7,11,13,17
5466 #7,7,10,14,16
5467 #8,1,9,13,16
5468 #9,2,8,13,14
5469 #10,1,2,15,17
5470 #11,6,8,10,17
5471 #12,6,7,9,15
5472 #13,2,6,11,16
5473 #14,5,9,14,17
5474 #15,5,8,15,16
5475 #16,1,6,12,14
5476 #17,2,5,7,12
5477 #18,3,12,16,17
5478 #19,3,11,14,15
5479 # END
hyperoval_16_stab_0.disjoint_sets_graph:
\$ (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
#END

clique 0:
#0,2,3,9,10
#6,7,11,13,17
#15,5,8,15,16
#16,1,6,12,14
# partition: (1,6,12,14|2,3,9,10|5,8,15,16|7,11,13,17)
# 4 is missing, it is the nucleus
# 0 is missing is the chosen point

# nonconical 6-arcs are used for classifying cubic surfaces:
nc_arcs_16:
\$\text{(ORBITER)} -v 4 \$
\$\text{-define F -finite_field -q 16 -end} \$
\$\text{-define P -projective_space 2 F -end} \$
\$\text{-with P -do} \$
\$\text{-projective_space_activity} \$
\$\text{-classify_arcs} \$
\$\text{-poset_classification_control} \$
\$\text{-problem_label nc_arcs_q16_d2} \$
\$\text{-W -depth 6} \$
\$\text{-report -end} \$
\$\text{-end} \$
\$\text{-target_size 6} \$
\$\text{-d 2} \$
\$\text{-conic_test} \$
\$\text{-end} \$
\$\text{pdflatex nc_arcs_q16_d2_poset.tex} \$
\$\text{open nc_arcs_q16_d2_poset.pdf} \$

nc_arcs_32_E13:
\$\text{(ORBITER)} -v 4 \$
\$\text{-orbiter_path $\text{(ORBITER_PATH)} \$
\$\text{-define F -finite_field -q 32 -end} \$
\$\text{-define P -projective_space 2 F -end} \$
\$\text{-with P -do} \$
\$\text{-projective_space_activity} \$
\$\text{-classify_arcs} \$
\$\text{-poset_classification_control} \$
\$\text{-problem_label nc_arcs_q32_d2} \$
\$\text{-W -depth 6} \$
\$\text{-draw_poset -draw_options -end} \$
\$\text{-report -end} \$
\$\text{-end} \$
\$\text{-target_size 6} \$
\$\text{-test_nb_Eckardt_points 13} \$
\$\text{-d 2} \$
\$\text{-conic_test} \$
\$\text{-end} \$
\$\text{-end} \$
\$\text{pdflatex nc_arcs_q32_d2_poset.tex} \$
\$\text{open nc_arcs_q32_d2_poset.pdf} \$
5574
5575  #User time: 0:00
5576
5577
5578
5579
5580  F64_work:
5581  ▶ $(ORBITER) -v 3 \
5582  ▶ ▶ -define F -finite_field -q 64 -end \ 
5583  ▶ ▶ -define f -formula "f" "f" "a*a+a" \ 
5584  ▶ ▶ -with F -do -finite_field_activity \ 
5585  ▶ ▶ ▶ -evaluate f "a=2" -end
5586
5587  F64_frob:
5588  ▶ $(ORBITER) -v 3 \
5589  ▶ ▶ -define F -finite_field -q 64 -end \ 
5590  ▶ ▶ -define f -formula "f" "f" "a*a*a*a*a*a*a*a" \ 
5591  ▶ ▶ -with F -do -finite_field_activity \ 
5592  ▶ ▶ ▶ -evaluate f "a=61" -end
5593
5594
5595  # surfaces with 13 Eckardt points have OCN=0,98,99
5596
5597  surface_64_0:
5598  ▶ $(ORBITER) -v 3 \
5599  ▶ ▶ -define F -finite_field -q 64 -end \ 
5600  ▶ ▶ -define P -projective_space 3 F -end \ 
5601  ▶ ▶ -with P -do \ 
5602  ▶ ▶ -projective_space_activity \ 
5603  ▶ ▶ ▶ -define_surface S -q 64 -catalogue 0 \ 
5604  ▶ ▶ ▶ -end \ 
5605  ▶ ▶ -end \ 
5606  ▶ ▶ -with S -do \ 
5607  ▶ ▶ -cubic_surface_activity \ 
5608  ▶ ▶ ▶ -report \ 
5609  ▶ ▶ ▶ -report_with_group \ 
5610  ▶ ▶ -end
5611  ▶ pdflatex surface_catalogue_q64_iso0_with_group.tex
5612  ▶ open surface_catalogue_q64_iso0_with_group.pdf
5613
5614
5615
5616
5617  #makes it slow:
5618  #▶ ▶ ▶ ▶ -test_nb_Eckardt_points 13 \
5619  #▶ ▶ ▶ ▶ ▶ -report -select_orbits_by_level 6 -select_orbits_by_stabilizer_order_multiple_of 24 -end \ 

596
nc_arcs_128:

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

-define \text{F} -finite_field \text{-q} 128 -end \
-define \text{P} -projective_space 2 \text{F} -use_projectivity_subgroup -end \
-with \text{P} -do \
-projective_space_activity \
-classify_arcs \
-postranization\_control \
-problem\_label nc_arcs_q128_d2 -W -depth 6 \
-select_orbits_by_stabilizer_order\_multiple_of 24 \
-target size 6 \
-conic_test \
-end \
-end \
-end \

nc_arcs_256_E13:

$\text{(ORBITER)} -v 8 $

-define \text{F} -finite_field \text{-q} 256 -end \
-define \text{P} -projective_space 2 \text{F} -use_projectivity_subgroup -end \
-with \text{P} -do \
-projective_space_activity \
-classify_arcs \
-postranization\_control \
-problem\_label nc_arcs_q256_d2 -W -depth 6 \
-target size 6 \
-conic_test \
-end \
-end \
-end \

pdflatex nc_arcs_q128_d2_poset.tex
open nc_arcs_q128_d2_poset.pdf
Example_F64:

\$(\text{ORBITER}) \ -v\ 3 \$
\$\text{-define F -finite_field -q\ 64 \ -end }\$
\$\text{-define P -projective_space 3 F -end }\$
\$\text{-with P -do }\$
\$\text{-projective_space_activity }\$
\$\text{-define_surface S64_abcd_52_8_8_52 -q\ 64 }\$
\$\text{-family_general_abcd\ 52\ 8\ 8\ 52 \ -end }\$
\$\text{-end }\$
\$\text{-with S64_abcd_52_8_8_52 -do }\$
\$\text{-cubic_surface_activity }\$
\$\text{-report }\$
\$\text{-end }\$
\$\text{pdflatex surface_family_general_abcd_q64_a52_b8_c8_d52_report.tex}\$

six_arcs_4_nbE13:

\$(\text{ORBITER}) \ -v\ 3 \$
\$\text{-define F -finite_field -q\ 4 \ -end }\$
\$\text{-define P -projective_space 2 F -end }\$
\$\text{-with P -do }\$
\$\text{-projective_space_activity }\$
\$\text{-control_six_arcs -problem_label sixarcs_q4 -end }\$
\$\text{-six_arcs_not_on_conic -filter_by_nb_Eckardt_points 13 -end}\$

six_arcs_8_nbE13:

\$(\text{ORBITER}) \ -v\ 3 \$
\$\text{-define F -finite_field -q\ 8 \ -end }\$
\$\text{-define P -projective_space 2 F -end }\$
\$\text{-with P -do }\$
\$\text{-projective_space_activity }\$
\$\text{-control_six_arcs -problem_label sixarcs_q8 -end }\$
\$\text{-six_arcs_not_on_conic -filter_by_nb_Eckardt_points 13 -end }\$
six_arcs_16_nbE13:

$\text{ORBITER} -v 3 \$
\begin{verbatim}
\$define F -finite_field -q 16 -end $
\$define P -projective_space 2 F -end $
\$with P -do $
\$projective_space_activity $
\$control_six_arcs -problem_label sixarcs.q16 -end $
\$six_arcs_not_on_conic -filter_by_nb_Eckardt_points 13 -end
\end{verbatim}

six_arcs_32_nbE13:

$\text{ORBITER} -v 3 \$
\begin{verbatim}
\$define F -finite_field -q 32 -end $
\$define P -projective_space 2 F -end $
\$with P -do $
\$projective_space_activity $
\$control_six_arcs -problem_label sixarcs.q32 -end $
\$six_arcs_not_on_conic -filter_by_nb_Eckardt_points 13 -end
\end{verbatim}

six_arcs_64_nbE13:

$\text{ORBITER} -v 3 \$
\begin{verbatim}
\$define F -finite_field -q 64 -end $
\$define P -projective_space 2 F -end $
\$with P -do $
\$projective_space_activity $
\$control_six_arcs -problem_label sixarcs.q64 -end $
\$six_arcs_not_on_conic -filter_by_nb_Eckardt_points 13 -end
\end{verbatim}

User time: 0:7
# 9 arcs: ago: 4, 8, 24^5, 48^2

six_arcs_128_nbE13:

$\text{ORBITER} -v 3 \$
\begin{verbatim}
\$define F -finite_field -q 128 -end $
\$define P -projective_space 2 F -end $
\$with P -do $
\$projective_space_activity $
\$control_six_arcs -problem_label sixarcs.q128 -end $
\$six_arcs_not_on_conic -filter_by_nb_Eckardt_points 13 -end
\end{verbatim}

# 1 min 39 sec
# 12 arcs, ago: 4^3, 24^9

six_arcs_256_nbE13:

599
\texttt{$(ORBITER) \ -v \ 3 \ \\
define F -finite_field -q \ 256 \ -end \ \\
define P -projective_space \ 2 \ F \ -end \ \\
-with \ P \ -do \ \\
-projective_space_activity \ \\
control_six_arcs -problem_label sixarcs_q256 -end \ \\
six_arcs_not_on_conic -filter_by_nb_Eckardt_points \ 13 \ -end \ \\

# 27 minutes on ripoff \nUser time: 29:11 on ripoff 7/30/21

\texttt{five_arcs_q13: \n$(ORBITER) \ -v \ 4 \ \\
define F -finite_field -q \ 13 \ -end \ \\
define P -projective_space \ 2 \ F \ -end \ \\
-with \ P \ -do \ \\
-projective_space_activity \ \\
-classify_arcs \ \\
-poset_classification_control \ \\
problem_label five_arcs_q13 -W -depth \ 5 \ \\
-report -end \ \\
-end \ \\
-target_size \ 5 \ \\
-d \ 2 \ \\
-end \ \\
pdflatex \ five_arcs_q13_poset.tex \nonumber
open \ five_arcs_q13_poset.pdf

\subsection*{Section 6.7: Cubic Curves}

\texttt{SECTION_CUBIC_CURVES: \n
cubic_curves_PG_2_4: \n$(ORBITER) \ -v \ 3 \ \\
-orbiter_path $(ORBITER_PATH) \ 

600
\define F -finite_field -q 3 -end \\
\define P -projective_space 2 F -end \\
\with P -do \\
\projective_space_activity \\
\classify_cubic_curves -q 4 \\
\target_size 9 -n 3 -d 3 \\
\poset_classification_control \\
\problem_label cc_4 -W -depth 9 \\
\draw_poset \\
\draw_options -radius 200 -embedded -end \\
\report -end \\
\end \\
pdflatex cc_4_poset.tex 
open cc_4_poset.pdf 
pdflatex cc_4_poset_lvl_9.tex 
#open cc_4_poset_lvl_9.pdf 
pdflatex Cubic_curves_q4.tex 
#open Cubic_curves_q4.pdf 

cubic_curves_PG_2_4_draw: 
$(ORBITER) -v 3 \$
\draw_layered_graph cc_4_poset_lvl_9.layered_graph 
\radius 300 -embedded -line_width 1.1 
\y_stretch 0.9 -scale 0.25 
\paths_in_between 6 4 9 0 
\end 
pdflatex cc_4_poset_lvl_9_draw.tex 
open cc_4_poset_lvl_9_draw.pdf 

cubic_curves_PG_2_8: 
$(ORBITER) -v 3 \$
\define G 
\define F -finite_field -q 8 -end 
\define P -projective_space 2 F -end 
\with P -do 
\projective_space_activity 
\classify_cubic_curves -q 8 -target_size 9 -n 3 -d 3 
\poset_classification_control 
\problem_label cc_8 -W -depth 9 
\draw_options -radius 200 -embedded -end 
\recognize "0,1,2,3,35,28" 
\recognize "1,2,3,51,28,61,46,71,40" 
\draw_poset 
\Kramer_Mesner_matrix 6 9 
\end 
\end 

cubic_curves_PG_2_4:
pdflatex cc_4_poset.tex 
open cc_4_poset.pdf 
pdflatex cc_4_poset_lvl_9.tex 
#open cc_4_poset_lvl_9.pdf 
pdflatex Cubic_curves_q4.tex 
#open Cubic_curves_q4.pdf 
cubic_curves_PG_2_8:
pdflatex cc_4_poset.tex 
open cc_4_poset.pdf 
pdflatex cc_4_poset_lvl_9.tex 
#open cc_4_poset_lvl_9.pdf 
pdflatex Cubic_curves_q4.tex 
#open Cubic_curves_q4.pdf
$(ORBITER) -v 2 -draw_matrix \n$ORBITER -v 3 -draw_layered_graph \n\n# the 6-set is orbit 7
# the 9-set is orbit 1

5857 cubic_curves_PG_2_8.draw:
$ORBITER -v 3 \
$ORBITER -draw_layered_graph \
$ORBITER -cc_8_poset_lvl_9.layered_graph \
$ORBITER -y_stretch 1.3 -scale 0.5 \
$ORBITER -paths_in_between 6 7 9 1 \n$ORBITER -end
\n5867 #cc_8_poset_lvl_9.layered_graph
5868 #cc_8_poset_detailed_lvl_9.layered_graph
5869
5870 \n\n5871 # Chapter 7 - Cubic Surfaces
5872 # Section 7.1: Cubic Surfaces Creation
5873 SECTION_CUBIC_SURFACES_CREATION:
5874 surface_4_0:
Family_general_F7:

```
$((ORBITER)) > $(ORBITER) -v 3 \n
define F -finite_field -q 7 -end \n
define P -projective_space 3 F -end \n
with P -do \n
-projective_space_activity \n
define_surface S7 -q 7 -catalogue 0 -end \n
end \n
with S7 -do \n
-cubic_surface_activity \n
report \n
-report_with_group \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

```

Family_general_F7:

```
$((ORBITER)) > $(ORBITER) -v 3 \n
define F -finite_field -q 7 -end \n
define P -projective_space 3 F -end \n
with P -do \n
-projective_space_activity \n```
\begin{verbatim}
5949 \triangleright \triangleright \texttt{-define_surface S7_abcd_2_3_3_4 -q 7 \}
5950 \triangleright \triangleright \triangleright \texttt{-family_general_abcd 2 3 3 4 -end \}
5951 \triangleright \triangleright \texttt{-end \}
5952 \triangleright \triangleright \texttt{-with S7_abcd_2_3_3_4 -do \}
5953 \triangleright \\texttt{-cubic_surface_activity \}
5954 \triangleright \triangleright \triangleright \texttt{-report \}
5955 \triangleright \\texttt{-end \}
5956 \texttt{pdflatex surface_family_general_abcd_q7_a2_b3_c3_d4_report.tex}
5957 \triangleright \texttt{open surface_family_general_abcd_q7_a2_b3_c3_d4_report.pdf}
5958 \triangleright \triangleright \texttt{
5959 \# Fermat with 18 Eckardt points
5960 \# no automorphism group, so no -report_with_group and no -all_quartic_curves
5961
5962
5963 \# Joel:
5964
5965 \texttt{eckardt_13_4_12:}
5966 \triangleright \$\texttt{(ORBITER) -v 6 \}
5967 \triangleright \triangleright \texttt{-define F -finite_field -q 13 -end \}
5968 \triangleright \triangleright \texttt{-define P -projective_space 3 F -end \}
5969 \triangleright \triangleright \texttt{-with P -do \}
5970 \triangleright \triangleright \texttt{-projective_space_activity \}
5971 \triangleright \triangleright \triangleright \texttt{-define_surface S_2_1 -q 13 \}
5972 \triangleright \triangleright \triangleright \texttt{-family_Eckardt 4 12 -end \}
5973 \triangleright \triangleright \\texttt{-end \}
5974 \triangleright \\texttt{-with S_2_1 -do \}
5975 \triangleright \\texttt{-cubic_surface_activity \}
5976 \triangleright \texttt{-report \}
5977 \triangleright \triangleright \texttt{-report_with_group \}
5978 \triangleright \\texttt{-end \}
5979
5980
5981
5982
5983
5984
5985
5986 \texttt{surface_8_0_catalogue:}
5987 \triangleright \$\texttt{(ORBITER) -v 3 \}
5988 \triangleright \triangleright \texttt{-define F -finite_field -q 8 -end \}
5989 \triangleright \triangleright \texttt{-define P -projective_space 3 F -end \}
5990 \triangleright \triangleright \texttt{-with P -do \}
5991 \triangleright \\texttt{-projective_space_activity \}
5992 \triangleright \\texttt{-define_surface S8_0 -q 8 -catalogue 0 -end \}
5993 \triangleright \\texttt{-end \}
5994 \\texttt{-with S8_0 -do \}
5995 \\texttt{-cubic_surface_activity \}
\end{verbatim}
\$(\text{ORBITER}) -v 3 \$

\$ -define F -finite_field -q 8 -end \$

\$ -define P -projective_space 3 F -end \$

\$ -with P -do \$

\$ -projective_space_activity \$

\$ -define_surface S8.0 -q 8 -catalogue 0 \$

\$ -select_double_six "15,11,22,19,24,5,16,10,23,20,25,4" \$

\$ -select_double_six "3,2,1,0,5,4,9,8,7,6,11,10" \$

\$ -transform_inverse "1,4,4,0,6,0,0,6,2,0,1,7,0,4,0,0" \$

\$ -transform "4,4,0,0,0,0,1,0,1,0,0,0,0,0,1,0" \$

\$ -transform_inverse "2,0,0,0,0,2,0,0,0,2,0,1,1,2,3,0" \$

\$ -with S8.0 -do \$

\$ -cubic_surface_activity \$

\$ -report \$

\$ -report_with_group \$

\$ -end \$

\$ -with S8.0 -do \$

\$ -define_surface S8.0 -q 8 -catalogue 0 \$

\$ -select_double_six "15,11,22,19,24,5,16,10,23,20,25,4" \$

\$ -select_double_six "3,2,1,0,5,4,9,8,7,6,11,10" \$

\$ -transform_inverse "1,4,4,0,6,0,0,6,2,0,1,7,0,4,0,0" \$

\$ -transform "4,4,0,0,0,0,1,0,1,0,0,0,0,0,1,0" \$

\$ -transform_inverse "2,0,0,0,0,2,0,0,0,2,0,1,1,2,3,0" \$

\$ -end \$

\$ -end \$

\$ -end \$

\$ -end \$

\$ -end \$

\$ -define_surface S8.0 -q 8 -catalogue 0 \$

\$ -select_double_six "15,11,22,19,24,5,16,10,23,20,25,4" \$

\$ -select_double_six "3,2,1,0,5,4,9,8,7,6,11,10" \$

\$ -transform_inverse "1,4,4,0,6,0,0,6,2,0,1,7,0,4,0,0" \$

\$ -transform "4,4,0,0,0,0,1,0,1,0,0,0,0,0,1,0" \$

\$ -transform_inverse "2,0,0,0,0,2,0,0,0,2,0,1,1,2,3,0" \$

\$ -with S8.0 -do \$

\$ -cubic_surface_activity \$

\$ -report \$

\$ -report_with_group \$

\$ -end \$

\$ -end \$

\$ -end \$
# 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 3 F -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:

$(ORBITER) -v 3 \
-define F -finite_field -q 13 -end \
-define P -projective_space 3 F -end \

\begin{verbatim}
6090  \>  \>  -with P -do \\
6091  \>  \>  -projective_space_activity \\
6092  \>  \>  \>  -define_surface S13_0 -q 13 -catalogue 0 -end \\
6093  \>  \>  -end \\
6094  \>  \>  -with S13_0 -do \\
6095  \>  \>  -cubic_surface_activity \\
6096  \>  \>  \>  -report \\
6097  \>  \>  \>  -report_with_group \\
6098  \>  \>  -end \\
6099  \>  pdflatex surface_catalogue_q13_iso0_report.tex \\
6100  \>  open surface_catalogue_q13_iso0_report.pdf \\
6101 \\
6102  # clean equation for Tekirdag-2:
6103 \\
6104  surface_16_0:
6105  \>  \>  $(\text{ORBITER}) -v 3 \$
6106  \>  \>  -define F -finite_field -q 16 -end \\
6107  \>  \>  \>  -define P -projective_space 3 F -end \\
6108  \>  \>  \>  -with P -do \\
6109  \>  \>  \>  -projective_space_activity \\
6110  \>  \>  \>  \>  -define_surface S16_0 -q 16 -catalogue 0 \\
6111  \>  \>  \>  \>  \>  -transform "1,0,0,0,0,1,0,12,0,0,0,1,0" \\
6112  \>  \>  \>  \>  \>  \>  -transform "15,11,4,0,0,12,0,0,0,0,0,1,3" \\
6113  \>  \>  \>  \>  \>  \>  \>  -end \\
6114  \>  \>  \>  \>  -with S16_0 -do \\
6115  \>  \>  \>  \>  -cubic_surface_activity \\
6116  \>  \>  \>  \>  \>  -report \\
6117  \>  \>  \>  \>  \>  \>  -report_with_group \\
6118  \>  \>  \>  \>  \>  \>  \>  -end \\
6119  \>  \>  \>  pdflatex surface_catalogue_q16_iso0_with_group.tex \\
6120  \>  \>  open surface_catalogue_q16_iso0_with_group.pdf \\
6121 \\
6122  # transform_inverse "3,0,0,0,0,1,1,0,0,0,0,0,0,0,1,0" \\
6123  # transform_inverse "13,12,1,0,12,13,1,0,0,0,1,0,0,0,0,1,0" \\
6124  # transform_inverse "1,0,0,0,0,1,0,12,12,1,0,0,0,0,1,0" \\
6125  # transform_inverse "12,0,0,0,0,12,0,0,0,0,0,1,0,0,0,1,0"
6126 
6127  # rank of lines ( 66591, 26737, 4093, 69904, 28376, 26470, 70160, 69855, 26208, 5847, 369, 32230, 529, 30293, 70068, 2178, 261, 28666, 8575, 105, 31694, 0, 51784, 25209, 22193, 49862, 274 )
6128 
6129  # 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 )
\end{verbatim}
# Tekirdag-1:

G13.8:

```latex
\$(ORBITER) -v 3 \\
-define F -finite_field -q 8 -end \\
-define P -projective_space 3 F -end \\
-with P -do \\
-projective_space_activity \\
-define_surface T1 -family_G13 2 -q 8 -end \\
-end \\
-with T1 -do \\
-cubic_surface_activity \\
-report \\
-report_with_group \\
-end \\
pdflatex surface_family_G13_q8_a2_with_group.tex \\
open surface_family_G13_q8_a2_with_group.pdf
```

# Tekirdag-2:

F13.8:

```latex
\$(ORBITER) -v 3 \\
-define F -finite_field -q 8 -end \\
-define P -projective_space 3 F -end \\
-with P -do \\
-projective_space_activity \\
-define_surface T1 -family_F13 2 -q 8 -end \\
-end \\
-with T1 -do \\
-cubic_surface_activity \\
-report \\
-report_with_group \\
-end \\
pdflatex surface_family_F13_q8_a2_with_group.tex \\
open surface_family_F13_q8_a2_with_group.pdf
```

# Tekirdag-2:

F13.16:

```latex
\$(ORBITER) -v 3 \\
```
# Tekirdag-3:

```latex
pdflatex surface_family_F13_q16_a2_with_group.tex
open surface_family_F13_q16_a2_with_group.pdf
```

# Kapadokya-1:

```latex
pdflatex surface_family_F13_q32_a2_with_group.tex
open surface_family_F13_q32_a2_with_group.pdf
```
# Kapadokya-2:

F13_64b:

```
$\$(ORBITER) -v 3
\-define F -finite_field -q 64 -end
\-define P -projective_space 3 F -end
\-with P -do
\-projective_space_activity
\-define_surface K2 -family_F13 18 -q 64 -end
\-end
\-with K2 -do
\-cubic_surface_activity
\-report
\-report_with_group
\-end
```

Colorado1:

```
$\$(ORBITER) -v 3
\-define F -finite_field -q 128 -end
\-define P -projective_space 3 F -end
\-with P -do
\-projective_space_activity
\-define_surface CO-1 -q 128 -catalogue 0
\-transform_inverse "1,0,0,0,1,0,96,0,0,1,96,0,0,0,1,0" 
\-end
\-with CO-1 -do
\-cubic_surface_activity
\-report
\-report_with_group
\-end
```

# recognize the arcs from Colorado-1,2,3:

Colorado2:

```
$\$(ORBITER) -v 3
\-define F -finite_field -q 128 -end
\-define P -projective_space 3 F -end
\-with P -do
```
-projective_space_activity \
-define_surface CO-2 -q 128 -catalogue 926 \
-transform_inverse "1,0,0,0,0,1,0,32,0,0,1,32,0,0,0,1,0" \
-end \
-with CO-2 -do \
cubic_surface_activity \
-report \
-report_with_group \
-end \

Colorado3: 
$\text{(ORBITER) -v 3} \$
-define F -finite_field -q 128 -end \
-define P -projective_space 3 F -end \
-with P -do \
-projective_space_activity \
-define_surface CO-3 -q 128 -catalogue 928 \
-transform_inverse "1,0,0,0,0,1,0,59,0,0,1,59,0,0,0,1,0" \
-end \
-with CO-3 -do \
cubic_surface_activity \
-report \
-report_with_group \
-end 

# Colorado-1: 
F13_128a: 
$\text{(ORBITER) -v 3} \$
-define F -finite_field -q 128 -end \
-define P -projective_space 3 F -end \
-with P -do \
-projective_space_activity \
-define_surface CO-1 -family F13 2 -q 128 -end \
-end \
-with CO-1 -do \
cubic_surface_activity \
-report \
-report_with_group \
-end 

# Colorado-2: 
F13_128b:
6321  $\text{(ORBITER)} -v 3 \$
6322  \text{define } F \text{ } -\text{finite_field} \text{ } -q 128 \text{ } -\text{end} \$
6323  \text{define } P \text{ } -\text{projective_space} \text{ } 3 \text{ } F \text{ } -\text{end} \$
6324  \text{with } P \text{ } -\text{do} \$
6325  \text{projective_space_activity} \$
6326  \text{define_surface CO-2 \text{-family} F13 6 \text{-q} 128 \text{-end} \$
6327  \text{-end} \$
6328  \text{with } CO-2 \text{-do} \$
6329  \text{cubic_surface_activity} \$
6330  \text{define} \text{-report} \$
6331  \text{report} \text{-report_with_group} \$
6332  \text{-end} \$
6333  \#	ext{ Colorado-3:} \$
6335  \text{F}_13\text{.}128c: \$
6336  $\text{(ORBITER)} -v 3 \$
6337  \text{define } F \text{ } -\text{finite_field} \text{ } -q 128 \text{ } -\text{end} \$
6338  \text{define } P \text{ } -\text{projective_space} \text{ } 3 \text{ } F \text{ } -\text{end} \$
6339  \text{with } P \text{ } -\text{do} \$
6340  \text{projective_space_activity} \$
6341  \text{define_surface CO-3 \text{-family} F13 14 \text{-q} 128 \text{-end} \$
6342  \text{-end} \$
6343  \text{with } CO-3 \text{-do} \$
6344  \text{cubic_surface_activity} \$
6345  \text{define} \text{-report} \$
6346  \text{report} \text{-report_with_group} \$
6347  \text{-end} \$
6348  \text{move_two_lines:} \$
6349  $\text{(ORBITER)} -v 5 \$
6350  \text{define } F \text{ } -\text{finite_field} \text{ } -q 8 \text{ } -\text{end} \$
6351  \text{with } F \text{ } -\text{do} \text{-finite_field_activity} \$
6352  \text{move_two_lines_in_hyperplane_stabilizer} \text{ } 65 \text{ } 4680 \text{ } 72 \text{ } 657 \text{-end} \$
6353  \text{F_alpha_beta_gamma_delta:} \$
6354  $\text{(ORBITER)} -v 3 \$
6355  \text{define } F \text{ } -\text{finite_field} \text{ } -q 7 \text{ } -\text{end} \$
6356  \text{with } F \text{ } -\text{do} \text{-finite_field_activity} \$
6357  \text{parse_and_evaluate} \$
6358  \text{"F_alpha_beta_gamma_delta" } \text{"x0,x1,x2,x3"} \$
6359  $\text{(F_ALPHA_BETA_GAMMA_DELTA)} \$
6360  \text{"alpha=2,beta=3,gamma=4,delta=5"} \$
612
\begin{verbatim}
6368 \[\] -end
6369 \[\] dot -Tpng F_alpha_beta_gamma_delta.gv >F_alpha_beta_gamma_delta.png
6370
6371
6372
6373 F_abcd.Eckardt_q31:
6374 \[\] $(\text{ORBITER}) -v 3 \ \$
6375 \[\] -define F -finite_field -q 31 -end \ $
6376 \[\] -define P -projective_space 3 F -end \ $
6377 \[\] -with P -do \ $
6378 \[\] -projective_space_activity \ $
6379 \[\] -define_surface F_abcd -q 31 \ $
6380 \[\] -by_equation "F_abcd" \ $
6381 \[\] -parse_and_evaluate "F\{a,b,c,d\} eqn" "a=2,b=30,c=30,d=2" \ $
6382 \[\] -end \ $
6383 \[\] -end \ $
6384 \[\] -with F_abcd -do \ $
6385 \[\] -finite_field_activity \ $
6386 \[\] -cubic_surface_activity \ $
6387 \[\] -report \ $
6388 \[\] -end \ $
6389 \[\] pdflatex surface_equation_F_abcd_q31_report.tex
6390 \[\] open surface_equation_F_abcd_q31_report.pdf
6391
6392
6393
6394
6395
6396
6397
6398 surface_F_abcd:
6399 \[\] $(\text{ORBITER}) -v 3 \ $
6400 \[\] -define F -finite_field -q 7 -end \ $
6401 \[\] -with F -do \ $
6402 \[\] -finite_field_activity \ $
6403 \[\] -parse_and_evaluate "F\{a,b,c,d\} eqn" "a=4,b=2,c=2,d=4" \ $
6404 \[\] -end \ $
6405 \[\] -end \ $
6406 \[\] dot -Tpng F_alpha_beta_gamma_delta.gv >F_alpha_beta_gamma_delta.png
6407
6408
6409
6410
6411 F_abcd_sweep_4.27.q7:
6412 \[\] $(\text{ORBITER}) -v 3 \ $
6413 \[\] -define F -finite_field -q 7 -end \ $
6414 \[\] -define P -projective_space 3 F -end \ $
\end{verbatim}
with P -do \\  
-projective\_space\_activity \\
-sweep\_4.27 sweep\_4.27_q7 -q 7 -by\_equation \"F_{abcd}\" \\
-define\(F) -finite\_field -q 7 -end \\
-define P -projective\_space 3 F -end \\
-with P -do \\
-projective\_space\_activity \\
-define\(\) surface F.2345 -q 7 \\
-by\_equation \"F_{\alpha,\beta,\gamma,\delta}\" \\
-$\{\alpha,\beta,\gamma,\delta\}$ \"x0,x1,x2,x3\" \\
\$F_{\alpha,\beta,\gamma,\delta}\$ \"alpha=2,beta=3,gamma=4,delta=5\" \\
\$D_{alpha=2,beta=3,gamma=4,delta=5}\$ \\
-override\_group 6 2 \\
-sweep \\
-end \\
-cubic\_space\_activity \\
-report \\
-report\_with\_group \\
-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 q3: \\
\$\{\text{ORBITER}\} -v 3 \$

-define F -finite\_field -q 3 -end \\
-define P -projective\_space 3 F -end \\

with P -do \projective_space_activity \sweep_4.15_lines sweep_4.15_lines_q3 -q 3 \by_equation "F_alpha_beta_gamma_delta" \"DF_{\{\alpha,\beta,\gamma,\delta}\}D" \"x0,x1,x2,x3" \$(F_ALPHA_BETA_GAMMA_DELTA) \"alpha=2,\beta=3,\gamma=4,\delta=5\" \"D\alpha=2,\beta=3,\gamma=4,\delta=5\D" \-end \-end

# cubic surfaces with 15 lines:
surface_15lines_q7.1:
$(ORBITER) -v 3 \-define F -finite_field -q 7 -end \-define P -projective_space 3 F -end \-with P -do \-projective_space_activity \-control_six_arcs -end \-define_surface S -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=6,\beta=4,\gamma=2,\delta=2\" \"D\alpha=6,\beta=4,\gamma=2,\delta=2\D" \-end \-end \-with S -do \-cubic_surface_activity \-report \-end
pdflatex surface_equation_F_alpha_beta_gamma_delta_q7_report.tex
open surface_equation_F_alpha_beta_gamma_delta_q7_report.pdf

F_alpha_beta_gamma_delta.sweep_4.15_lines_q7:
$(ORBITER) -v 3 \-define F -finite_field -q 7 -end \-define P -projective_space 3 F -end \-with P -do \-projective_space_activity \-sweep_4.15_lines sweep_4.15_lines_q3 -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=3,\gamma=4,\delta=5\"  \
"D\{alpha=2,\beta=3,\gamma=4,\delta=5\}D"  \
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 3 F -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

surf49_recognize:

$\{ORBITER\} -v 3 \ 
-define F -finite_field -q 49 -end \ 
-define P -projective_space 3 F -end \ 
-with P -do \ 
-projective_space_activity \ 

616
> -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> -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:

$(ORBITER) -v 3 \n-define F -finite_field -q 7 -end \n-define P -projective_space 3 F -end \n-with P -do \n-projective_space_activity \n-define_surface S \n-by_coefficients $(SURFACE_MCKEAN_15_LINES) -q 7 \n-end \n-end

#pdflatex surface_by_coefficients_q7_report.tex
#open surface_by_coefficients_q7_report.pdf

# 2 Eckardt points

F_4.4.3.3.q7:

$(ORBITER) -v 3 \n-define F -finite_field -q 7 -end \n-define P -projective_space 3 F -end \n-with P -do \n-projective_space_activity \n-define_surface -q 7 -by_equation \n-by "F_alpha_beta_gamma_delta" \n-"x0,x1,x2,x3" \n-F_ALPHA_BETA_GAMMA_DELTA \n-alpha=4,beta=4, gamma=3, delta=3 \n-alpha=4,beta=4, gamma=3, delta=3\D" \n-end \n-end

#pdflatex surface_equation_F_alpha_beta_gamma_delta_q7_report.tex
#open surface_equation_F_alpha_beta_gamma_delta_q7_report.pdf
# has 4 Eckardt points

F_alpha_beta_gamma_delta_points.txt:

```
$ (ORBITER) -v 3 \
  $define F -finite_field -q 7 -end \n  $define P -projective_space 3 F -end \n  -with P -do \n  $projective_space_activity \n  -sweep nothing \n  $define_surface -q 7 -by_equation \n  "F_alpha_beta_gamma_delta" \n  "$DF_{\{\alpha,\beta,\gamma,\delta\}}" \n  "x0,x1,x2,x3" \n  $F_ALPHA_BETA_GAMMA_DELTA \n  "alpha=2,beta=3,\gamma=4,\delta=5" \n  "$D{\alpha=2,\beta=3,\gamma=4,\delta=5}\" \n  -end \n  -end
```

SECTION CUBIC SURFACES AND QUARTIC CURVES:

quartic_curve_9_0_report:
```
$ (ORBITER) -v 3 \
  $define F -finite_field -q 9 -end \n  $define P -projective_space 2 F -end \n  -with P -do \n  $projective_space_activity \n  $define_quartic_curve C -q 9 \n  $catalogue 0 -end \n  -end \n  -with C -do \n  $quartic_curve_activity \n  -report \n  -end
```

pdflatex quartic_curve_catalogue_q9_iso0_report.tex
open quartic_curve_catalogue_q9_iso0_report.pdf
quartic_curve_13_0_report:
$ (ORBITER) -v 3 \
later
\begin{verbatim}
define F -finite_field -q 13 -end \ndefine P -projective_space 2 F -end \nwith P -do \nprojective_space_activity \ndefine quartic_curve C -q 13 \ncatalogue 0 -end \nwith C -do \nquartic_curve_activity \nprojective_space_activity \ndefine quartic_curve C -q 13 \ncatalogue 1 -end \nwith C -do \nquartic_curve_activity \nprojective_space_activity \ndefine quartic_curve C -q 13 \ncatalogue 1 -end \nwith C -do \nquartic_curve_activity \nprojective_space_activity \ndefine quartic_curve C -q 13 \ncatalogue 1 -end \nwith C -do \nquartic_curve_activity \nprojective_space_activity \ndefine quartic_curve C -q 13
\end{verbatim}

quartic_curve_13_1_report:
$ (ORBITER) -v 3 \
later
\begin{verbatim}
define F -finite_field -q 13 -end \ndefine P -projective_space 2 F -end \nwith P -do \nprojective_space_activity \ndefine quartic_curve C -q 13 \ncatalogue 0 -end \nwith C -do \nquartic_curve_activity \nprojective_space_activity \ndefine quartic_curve C -q 13 \ncatalogue 1 -end \nwith C -do \nquartic_curve_activity \nprojective_space_activity \ndefine quartic_curve C -q 13
\end{verbatim}

surface_4_0_quartic_curves:
$ (ORBITER) -v 3 \
later
\begin{verbatim}
define F -finite_field -q 4 -end \ndefine P -projective_space 3 F -end \nwith P -do \nprojective_space_activity \ndefine_surface S4_0 -q 4 -catalogue 0 -end \n\end{verbatim}
# full del Pezzo surfaces:

**NB CUBIC SURFACES**

quartic curves q7:

```
 $(ORBITER) -v 3 \n
 -define F -finite_field -q 7 -end \n
 -define P -projective_space 3 F -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 \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:

```
 $(ORBITER) -v 3 \n
 -define F -finite_field -q 7 -end \n
 -define P -projective_space 2 F -end \n
 -with P -do 
```
\$\text{NB\ CUBIC\ SURFACES\ Q7} = 4$

quartic\_curves.q13:
\$\text{(ORBITER)} -v 3 \$
\$\text{-list.arguments} \$
\$\text{-define F -finite_field -q 13 -end} \$
\$\text{-define P -projective_space 3 F -end} \$
\$\text{-loop L 0 $(\text{NB\ CUBIC\ SURFACES\ Q13}) 1 \}$ \$
\$\text{-with P -do} \$
\$\text{-projective_space_activity} \$
\$\text{-define_surface S13\_\%L -q 13 -catalogue \%L -end} \$
\$\text{-end} \$
\$\text{-end\_loop} \$
\$\text{-print_symbols} \$
\$\text{-loop L 0 $(\text{NB\ CUBIC\ SURFACES\ Q13}) 1 \}$ \$
\$\text{-with S13\_\%L -do} \$
\$\text{-cubic_surface_activity} \$
\$\text{-export_all_quartic_curves} \$
\$\text{-end} \$
\$\text{-end\_loop} \$
\$\text{-print_symbols} \$
\$\text{#pdflatex surface\_catalogue.q13.iso0.quartics.tex} \$
\$\text{#open surface\_catalogue.q13.iso0.quartics.pdf} \$
%surface\_catalogue.q13.iso0.quartics.csv
quartic\_curves.q13.classify:
\$\text{(ORBITER)} -v 3 \$
\$\text{-list.arguments} \$
\$\text{-define F -finite_field -q 13 -end} \$
\$\text{-define P -projective_space 2 F -end} \$

#surface\_catalogue.q13.iso0.quartics.csv
-with P -do \-projective_space_activity \-classify_quartic_curves_with_substructure \-define F -finite_field -q 17 -end \-define P -projective_space 3 F -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 \-list_arguments \-define F -finite_field -q 17 -end \-define P -projective_space 3 F -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 \-define F -finite_field -q 17 -end \-with P -do \-projective_space_activity \-define_surface S17_%L -q 17 -catalogue %L -end \-end \-end_loop \-print_symbols

# The number of types of quartic curves is 2

#idx : ago

#0 : 24

#1 : 48

NB_CUBIC_SURFACES_Q17=7

quartic_curves_q17:

$(ORBITER) -v 3 

-list_arguments \-define F -finite_field -q 17 -end \-define P -projective_space 3 F -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 

#pdflatex surface_catalogue_q17_iso0_report.tex

#open surface_catalogue_q17_iso0.pdf
-define P -projective_space 2 F -end \n-define P -projective_space 3 F -end \n-projective_space_activity \nclassify_quartic_curves_with_substructure \nsurface_catalogue_q17.iso%d_quartics.csv \n$\text{NB}\text{\_CUBIC\_SURFACES}\text{\_Q17} 3 4 quartic\_curves_q17 \n-end \n-print_symbols \n
#User time: 2:33
#q17
#The number of types of quartic curves is 7
idx : ago
0 : 24
1 : 24
2 : 4
3 : 96
4 : 6
5 : 8
6 : 2

NB\text{\_CUBIC\_SURFACES}\text{\_Q19}=10
quartic\_curves_q19:
$\text{\(\text{\textbackslash ORBITER}\) \text\text{-v 3 \text\text{-list_arguments}}}$
-define F -finite_field -q 19 -end \n-define P -projective_space 3 F -end \n-define S19_%L -q 19 -catalogue %L -end \n-end_loop 
-print_symbols \n-loop L 0 $(\text{\text{\textbackslash NB}\text{\_CUBIC\_SURFACES}\text{\_Q19}}) 1 \n-with P -do \n-projective_space_activity \n-define_SURFACE S19_%L -q 19 -catalogue %L -end \n-end_loop 
-print_symbols \n-loop L 0 $(\text{\text{\textbackslash NB}\text{\_CUBIC\_SURFACES}\text{\_Q19}}) 1 \n-with S19_%L -do \n-cubic\_surface\_activity \n-report \n-export_all_quartic\_curves \n-end \n-end_loop 
-print_symbols
pdflatex surface\_catalogue_q19.iso0_report.tex
open surface\_catalogue_q19.iso0_report.pdf
quartic_curves_q19_classify:

-define F -finite_field -q 19 -end 
-define P -projective_space 2 F -end 
-with P -do 
-projective_space_activity 
-classify_quartic_curves_with_substructure 
surface_catalogue_q19_iso%d_quartics.csv 
$(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:

-define F -finite_field -q 13 -end 
-define P -projective_space 3 F -end 
-with P -do 
-projective_space_activity 
-set_stabilizer 4 
surface_catalogue_q19_iso%d_quartics.csv 
$(NB_CUBIC_SURFACES_Q19) "pts_on_curve" 
-end 
-print_symbols

surface_13_0_quartics:

-define F -finite_field -q 13 -end 
-define P -projective_space 3 F -end 
-with P -do 
-projective_space_activity 
-define_surface S13_0 -q 13 -catalogue 0 -end 
-end
surface_13_1_quartics:  
$\text{(ORBITER)} -v 3 \$

quartic_curve_13_2_group:  
$\text{(ORBITER)} -v 3 \$

quartic_13_2_group:  
$\text{(ORBITER)} -v 3 \$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$\text{(ORBITER)} -v 3 \$
surface_25_0:
>$\text{(ORBITER)}$ -v 3 \\
\> -define F -finite_field -q 25 -end \\
\> -define P -projective_space 3 F -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:
>$\text{(ORBITER)}$ -v 3 \\
\> -define F -finite_field -q 25 -end \\
\> -define P -projective_space 2 F -end \\
\> -loop L 0 18 1 \\
\> -with P -do \\
\> -projective_space_activity \\
\> -define_quartic_curve QC25_%L \\
\> -report \\
\> -catalogue %L -end \\
\> -end \\
\> -end_loop \\
\> -print_symbols \\
\> -loop L 0 18 1 \\
\> -with QC25_%L -do \\
\> -quartic_curve_activity \\
\> -report \\
\> -end \\
\> -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
quartic_curve_catalogue_q25_iso0_report.tex
quartic_curve_catalogue_q25_iso1_report.tex
quartic_curve_catalogue_q25_iso2_report.tex
quartic_curve_catalogue_q25_iso3_report.tex
quartic_curve_catalogue_q25_iso4_report.tex
quartic_curve_catalogue_q25_iso5_report.tex
quartic_curve_catalogue_q25_iso6_report.tex
quartic_curve_catalogue_q25_iso7_report.tex
quartic_curve_catalogue_q25_iso8_report.tex
quartic_curve_catalogue_q25_iso9_report.tex
quartic_curve_catalogue_q25_iso10_report.tex
quartic_curve_catalogue_q25_iso11_report.tex
quartic_curve_catalogue_q25_iso12_report.tex
quartic_curve_catalogue_q25_iso13_report.tex
quartic_curve_catalogue_q25_iso14_report.tex
quartic_curve_catalogue_q25_iso15_report.tex
quartic_curve_catalogue_q25_iso16_report.tex
quartic_curve_catalogue_q25_iso17_report.tex
quartic_curve_catalogue_q25_iso0_report.pdf
quartic_curve_catalogue_q25_iso1_report.pdf
quartic_curve_catalogue_q25_iso2_report.pdf
quartic_curve_catalogue_q25_iso3_report.pdf
quartic_curve_catalogue_q25_iso4_report.pdf
quartic_curve_catalogue_q25_iso5_report.pdf
quartic_curve_catalogue_q25_iso6_report.pdf
quartic_curve_catalogue_q25_iso7_report.pdf
quartic_curve_catalogue_q25_iso8_report.pdf
quartic_curve_catalogue_q25_iso9_report.pdf
quartic_curve_catalogue_q25_iso10_report.pdf
quartic_curve_catalogue_q25_iso11_report.pdf
quartic_curve_catalogue_q25_iso12_report.pdf
quartic_curve_catalogue_q25_iso13_report.pdf
quartic_curve_catalogue_q25_iso14_report.pdf
quartic_curve_catalogue_q25_iso15_report.pdf
quartic_curve_catalogue_q25_iso16_report.pdf
quartic_curve_catalogue_q25_iso17_report.pdf
quartic_curve_catalogue_q25_iso0_report.pdf
quartic_curve_13_table:
$(ORBITER) -v 3 \
$define F -finite_field -q 13 -end \n$define P -projective_space 2 F -end \n-with P -do \n-projective_space_activity \n-table_of_quartic_curves \n-end

# quartic_curves_q13_info.csv
quartic_curve_19_table:
  $(ORBITER) -v 3 \ $
  -define F -finite_field -q 19 -end \ $
  -define P -projective_space 2 F -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 2 F -end \ $
  -with P -do \ $
  -projective_space_activity \ $
  -table_of_quartic_curves \ $
  -end
  
# quartic_curves_q25_info.csv

quartic_curve_27_table:
  $(ORBITER) -v 3 \ $
  -define F -finite_field -q 27 -end \ $
  -define P -projective_space 2 F -end \ $
  -with P -do \ $
  -projective_space_activity \ $
  -table_of_quartic_curves \ $
  -end
  
# quartic_curves_q27_info.csv

quartic_curve_29_table:
  $(ORBITER) -v 3 \ $
  -define F -finite_field -q 29 -end \ $
  -define P -projective_space 2 F -end \ $
  -with P -do \ $

quartic_curve_29_table
7120 ◄ ◄ ◄ -projective_space_activity \ 
7121 ◄ ◄ ◄ ◄ -table_of_quartic_curves \ 
7122 ◄ ◄ ◄ -end 
7123 
7124 
7125 # quartic_curves_q29_info.csv 
7126 
7127 
7128 quartic_curve_31_table: 
7129 ◄ $(ORBITER) -v 3 \ 
7130 ◄ ◄ -define F -finite_field -q 31 -end \ 
7131 ◄ ◄ -define P -projective_space 2 F -end \ 
7132 ◄ ◄ -with P -do \ 
7133 ◄ ◄ ◄ -projective_space_activity \ 
7134 ◄ ◄ ◄ -table_of_quartic_curves \ 
7135 ◄ ◄ ◄ -end 
7136 
7137 
7138 # quartic_curves_q31_info.csv 
7139 
7140 
7141 
7142 surface_25_12: 
7143 ◄ $(ORBITER) -v 3 \ 
7144 ◄ ◄ -define F -finite_field -q 25 -end \ 
7145 ◄ ◄ -define P -projective_space 3 F -end \ 
7146 ◄ ◄ -with P -do \ 
7147 ◄ ◄ ◄ -projective_space_activity \ 
7148 ◄ ◄ ◄ -define_surface S25_12 -q 25 -catalogue 12 -end \ 
7149 ◄ ◄ ◄ -end \ 
7150 ◄ ◄ -with S25_12 -do \ 
7151 ◄ ◄ -cubic_surface_activity \ 
7152 ◄ ◄ ◄ -report \ 
7153 ◄ ◄ ◄ -report_with_group \ 
7154 ◄ ◄ ◄ -end 
7155 ◄ pdflatex surface_catalogue_q25_iso12_with_group.tex 
7156 ◄ open surface_catalogue_q25_iso12_with_group.pdf 
7157 
7158 
7159 surface_25_12_t1: 
7160 ◄ $(ORBITER) -v 3 \ 
7161 ◄ ◄ -define F -finite_field -q 25 -end \ 
7162 ◄ ◄ -define P -projective_space 3 F -end \ 
7163 ◄ ◄ -with P -do \ 
7164 ◄ ◄ -projective_space_activity \ 
7165 ◄ ◄ ◄ -define_surface S25_12 -q 25 -catalogue 12 \ 
7166 ◄ ◄ ◄ -transform "1,0,0,16, 0,1,0,18, 0,0,1,8, 0,0,1,1, 0" \
surface_25_12_t2:

$(\text{ORBITER})$ -v 3 \\
define F -finite_field -q 25 -end \\
define P -projective_space 3 F -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" \\
-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_t3:

$(\text{ORBITER})$ -v 3 \\
define F -finite_field -q 25 -end \\
define P -projective_space 3 F -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" \\
-end \\
-end \\
with S25_12 -do \\
-cubic_surface_activity \\
-report \\
-report_with_group \\

7214  ▶  ▶  -end
7215  ▶  ▶  pdflatex surface_catalogue_q25_iso12_with_group.tex
7216  ▶  ▶  open surface_catalogue_q25_iso12_with_group.pdf
7217
7218
7219
7220  surface_25_12_t4:
7221  ▶  ▶  $(ORBITER) -v 3 \
7222  ▶  ▶  ▶  -define F -finite_field -q 25 -end \n7223  ▶  ▶  ▶  -define P -projective_space 3 F -end \n7224  ▶  ▶  ▶  -with P -do \n7225  ▶  ▶  ▶  -projective_space_activity \n7226  ▶  ▶  ▶  ▶  -define_surface S25_12 -q 25 -catalogue 12 \n7227  ▶  ▶  ▶  ▶  ▶  -transform "1,0,0,16, 0,1,0,18, 0,0,1,8, 0,0,1,1, 0" \n7228  ▶  ▶  ▶  ▶  ▶  -transform_inverse "16,0,1,0, 3,5,1,0, 0,0,1,0, 0,0,0,1, 0" \n7229  ▶  ▶  ▶  ▶  ▶  -transform "3,0,0,0, 0,1,0,0, 0,0,1,0, 0,0,0,1, 0" \n7230  ▶  ▶  ▶  ▶  ▶  -transform_inverse "1,0,0,0, 0,1,0,0, 0,0,1,0, 13,2,2,1, 0" \n7231  ▶  ▶  ▶  ▶  -end \n7232  ▶  ▶  ▶  -end \n7233  ▶  ▶  ▶  -with S25_12 -do \n7234  ▶  ▶  -cubic_surface_activity \n7235  ▶  ▶  ▶  -report \n7236  ▶  ▶  ▶  -report_with_group \n7237  ▶  ▶  ▶  -end
7238  ▶  ▶  ▶  pdflatex surface_catalogue_q25_iso12_with_group.tex
7239  ▶  ▶  ▶  open surface_catalogue_q25_iso12_with_group.pdf
7240
7241
7242  surface_25_12_t5:
7243  ▶  ▶  $(ORBITER) -v 3 \
7244  ▶  ▶  ▶  -define F -finite_field -q 25 -end \n7245  ▶  ▶  ▶  -define P -projective_space 3 F -end \n7246  ▶  ▶  ▶  -with P -do \n7247  ▶  ▶  ▶  -projective_space_activity \n7248  ▶  ▶  ▶  ▶  -define_surface S25_12 -q 25 -catalogue 12 \n7249  ▶  ▶  ▶  ▶  ▶  -transform "1,0,0,16, 0,1,0,18, 0,0,1,8, 0,0,1,1, 0" \n7250  ▶  ▶  ▶  ▶  ▶  -transform_inverse "16,0,1,0, 3,5,1,0, 0,0,1,0, 0,0,0,1, 0" \n7251  ▶  ▶  ▶  ▶  ▶  -transform "3,0,0,0, 0,1,0,0, 0,0,1,0, 0,0,0,1, 0" \n7252  ▶  ▶  ▶  ▶  ▶  -transform_inverse "1,0,0,0, 0,1,0,0, 0,0,1,0, 13,2,2,1, 0" \n7253  ▶  ▶  ▶  ▶  ▶  -transform "1,0,0,0, 0,1,0,0, 13,1,1,0, 0,0,0,1, 0" \n7254  ▶  ▶  ▶  ▶  -end \n7255  ▶  ▶  ▶  -end \n7256  ▶  ▶  ▶  -with S25_12 -do \n7257  ▶  ▶  -cubic_surface_activity \n7258  ▶  ▶  ▶  -report \n7259  ▶  ▶  ▶  -report_with_group \n7260  ▶  ▶  -end
7261 \> pdflatex surface_catalogue_q25_iso12_with_group.tex
7262 \> open surface_catalogue_q25_iso12_with_group.pdf
7263
7264
7265 PG\_2\_25:
7266 \> $(\text{ORBITER}) \$
7267 \> \> -define F \text{-finite}\_field -q 25 -end \$
7268 \> \> -define P \text{-projective}\_space 2 F -end \$
7269 \> \> -with P -do \text{-projective}\_space\_activity -cheat\_sheet -end
7270 \> \> pdflatex PG\_2\_25.tex
7271 \> \> open PG\_2\_25.pdf
7272
7273
7274
7275 PG\_2\_25\_lines:
7276 \> $(\text{ORBITER}) -v 5 \$
7277 \> \> -oritzer\_path $(\text{ORBITER}PATH) \$
7278 \> \> -define G \text{-linear}\_group -PGGL 3 25 -end \$
7279 \> \> -define G\_on\_lines \text{-modified}\_group -from G \$
7280 \> \> \> -on_k\_subspaces 2 \$
7281 \> \> \> -end \$
7282 \> \> \> -with G\_on\_lines -do \$
7283 \> \> \> -\text{group}\_theoretic\_activity \$
7284 \> \> \> \> \> -poset\_classification\_control \$
7285 \> \> \> \> \> \> -\text{problem}\_label PGGL\_3\_25 \$
7286 \> \> \> \> \> \> \> -\text{depth} 3 -\text{draw}\_poset -\text{draw}\_options -\text{radius} 200 -end -\text{report} -end \$
7287 \> \> \> \> \> \> \> -\text{recognize} "0,25,650" \$
7288 \> \> \> \> \> \> \> \> -\text{recognize} "430,16,364" \$
7289 \> \> \> \> \> \> \> \> -\text{end} \$
7290 \> \> \> \> -\text{orbits}\_\text{on}\_\text{subsets} 3 \$
7291 \> \> \> -\text{report} \$
7292 \> \> \> -\text{end} \$
7293 \> \> \> pdflatex PGGL\_3\_25\_poset.tex
7294 \> \> \> open PGGL\_3\_25\_poset.pdf
7295
7296 surface\_25\_12\.t6:
7297 \> $(\text{ORBITER}) -v 3 \$
7298 \> \> -define F \text{-finite}\_field -q 25 -end \$
7299 \> \> -define P \text{-projective}\_space 3 F -end \$
7300 \> \> -with P -do \$
7301 \> \> -\text{projective}\_\text{activity} \$
7302 \> \> \> -\text{define}\_\text{surface} S25\_12 -q 25 -\text{catalogue} 12 \$
7303 \> \> \> -\text{transform} "1,0,0,16, 0,1,0,18, 0,0,1,8, 0,0,1,1, 0" \$
7304 \> \> \> -\text{transform}\_\text{inverse} "16,0,1,0, 3,5,1,0, 0,0,1,0, 0,0,0,1, 0" \$
7305 \> \> \> -\text{transform} "3,0,0,0, 0,1,0,0, 0,0,1,0, 0,0,0,1, 0" \$
7306 \> \> \> -\text{transform}\_\text{inverse} "1,0,0,0, 0,1,0,0, 0,0,1,0, 13,2,2,1, 0" \$
7307 \> \> \> -\text{transform}\_\text{inverse} "1,0,0,0, 0,1,0,0, 13,1,1,0, 0,0,0,1, 0" \$
7308
7309
7310
7311
7312
7313
7314
7315
7316
7317
7318
7319
7320
-transform "3,8,8,0, 22,13,22,0, 14,19,15,0, 0,0,0,1, 1"
-transform_inverse "16,0,0,0, 0,16,0,0, 21,21,21,0, 0,0,0,1, 0"
-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
PG_2.25_stab_of_triangle:
 $(ORBITER) -v 5 
-orbiter_path $(ORBITER_PATH) 
-define G -linear_group -PGGL 3 25 
-subgroup_by_generators "triangle_stab" 6912 7 
 "1,0,0,0,1,0,0,0,1,1, 
1,0,0,13,0,0,0,13,1, 
1,0,0,4,0,0,0,6,1, 
1,0,0,0,17,0,0,0,13,0, 
1,0,0,18,0,0,0,4,1, 
1,0,0,0,11,0,1,0,0, 
0,1,0,0,20,14,0,0,0"
-end 
-with G -do 
-group_theoretic_activity 
-poset_classification_control 
-problem_label PGGL_3.25 
-depth 3 -draw_poset -draw_options -radius 200 -end 
-recognize "8,44,226" 
-end 
-orbits_on_subsets 3 
-end

surface_25.12_t7:
 $(ORBITER) -v 3 
-define F -finite_field -q 25 -end 
-define P -projective_space 3 F -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" 

\begin{verbatim}
7354 \texttt{surface\_25\_12\_t8:} \\
7355 \texttt{\$\{ORBITER\} -v 3 \} \\
7356 \texttt{define F -finite_field -q 25 -end \} \\
7357 \texttt{define P -projective_space 3 F -end \} \\
7358 \texttt{with P -do \} \\
7359 \texttt{-define_surface S25\_12 -q 25 -catalogue 12 \} \\
7360 \texttt{-transform "1,0,0,16, 0,1,0,18, 0,0,1,1, 0" \} \\
7361 \texttt{-transform "3,0,0,0, 0,1,0,0, 0,0,0,1, 0" \} \\
7362 \texttt{-transform "1,0,0,10, 0,1,0,10, 0,0,0,1, 0" \} \\
7363 \texttt{-transform "1,0,0,0, 0,1,0,0, 0,0,0,1, 0" \} \\
7364 \texttt{end \} \\
7365 \texttt{end \} \\
7366 \texttt{with S25\_12 -do \} \\
7367 \texttt{-cubic_surface_activity \} \\
7368 \texttt{-report \} \\
7369 \texttt{-report_with_group \} \\
7370 \texttt{end \} \\
7371 \texttt{pdflatex surface\_catalogue\_q25\_iso12\_with\_group.tex} \\
7372 \texttt{open surface\_catalogue\_q25\_iso12\_with\_group.pdf} \\
7373
7374 \texttt{surface\_25\_12\_t9:} \\
7375 \texttt{\$\{ORBITER\} -v 3 \} \\
7376 \texttt{define F -finite_field -q 25 -end \} \\
7377 \texttt{define P -projective_space 3 F -end \} \\
7378 \texttt{with P -do \} \\
7379 \texttt{-define_surface S25\_12 -q 25 -catalogue 12 \} \\
7380 \texttt{-transform "1,0,0,16, 0,1,0,18, 0,0,1,1, 0" \} \\
7381 \texttt{-transform "3,0,0,0, 0,1,0,0, 0,0,0,1, 0" \} \\
7382 \texttt{-transform "1,0,0,10, 0,1,0,10, 0,0,0,1, 0" \} \\
7383 \texttt{-transform "1,0,0,0, 0,1,0,0, 0,0,0,1, 0" \} \\
7384 \texttt{end \} \\
7385 \texttt{-define_surface S25\_12 -q 25 -catalogue 12 \} \\
7386 \texttt{-transform "1,0,0,16, 0,1,0,18, 0,0,1,1, 0" \} \\
7387 \texttt{-transform "3,0,0,0, 0,1,0,0, 0,0,0,1, 0" \} \\
7388 \texttt{-transform "1,0,0,10, 0,1,0,10, 0,0,0,1, 0" \} \\
7389 \texttt{-transform "1,0,0,0, 0,1,0,0, 0,0,0,1, 0" \} \\
7390 \texttt{-transform "1,0,0,0, 0,1,0,0, 0,0,0,1, 0" \} \\
7391 \texttt{-end \} \\
7392 \texttt{-end \} \\
7393 \texttt{-cubic_surface_activity \} \\
7394 \texttt{-report \} \\
7395 \texttt{-report_with_group \} \\
7396 \texttt{-end \} \\
7397 \texttt{pdflatex surface\_catalogue\_q25\_iso12\_with\_group.tex} \\
7398 \texttt{open surface\_catalogue\_q25\_iso12\_with\_group.pdf} \\
7399
7400 \texttt{surface\_25\_12\_t9:}
\end{verbatim}
$\textit{ORBITER} -v 3$

> define F -finite_field -q 25 -end

> define P -projective_space 3 F -end

> with P -do

> 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, 0,0,0,1, 0"

> transform "10,0,0, 0,1,0,0, 0,0,1,0, 0,0,0,1, 0"

> transform_inverse "3,8,8,0, 22,13,22,0, 14,19,15,0, 0,0,0,1, 1"

> transform "16,0,0,0, 0,16,0,0, 21,21,21,0, 0,0,0,1, 0"

> transform "1,0,0,0, 0,5,0,0, 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 "0,1,0,0, 0,0,1,0, 0,0,0,1, 1,0,0,0, 0"

> 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_t8_quartic_curves:

$\textit{ORBITER} -v 3$

> define F -finite_field -q 25 -end

> define P -projective_space 3 F -end

> with P -do

> 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, 0,0,0,1, 0"

> transform "1,0,0,0, 0,1,0,0, 0,0,1,0, 0,0,0,1, 0"

> transform_inverse "3,8,8,0, 22,13,22,0, 14,19,15,0, 0,0,0,1, 1"

> transform "16,0,0,0, 0,16,0,0, 21,21,21,0, 0,0,0,1, 0"

> transform "10,0,0, 0,5,0,0, 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 "0,1,0,0, 0,0,1,0, 0,0,0,1, 1,0,0,0, 0"

> end

635
quartic_curve.13.0_surface:
$\text{ORBITER} -v 3 \$
\$\text{-define F -finite field -q 13 -end} \$
\$\text{-define P -projective space 2 F -end} \$
\$\text{-with P -do} \$
\$\text{-projective space activity} \$
\$\text{-define quartic_curve QC13.0 -q 13 -catalogue 0 -end} \$
\$\text{-end} \$
\$\text{-with QC13.0 -do} \$
\$\text{-quartic_curve_activity} \$
\$\text{-create_surface} \$
\$\text{-end} \$

\$\text{surface equation: 0, 0, 9, 0, 3, 0, 1, 0, 12, 7, 0, 4, 4, 0, 7, 12, 6, 2, 5, 10} \$
\$\text{#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} \$

quartic_curve.13.0_surface_create:
$\text{ORBITER} -v 3 \$
\$\text{-define F -finite field -q 13 -end} \$
\$\text{-define P -projective space 3 F -end} \$
\$\text{-with P -do} \$
\$\text{-projective space activity} \$
\$\text{-define_surface S -by_coefficients} \$
\$\text{"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"} \$
\$\text{-q 13 -end} \$
\$\text{-end} \$
\$\text{-with S -do} \$
\$\text{-cubic_surface_activity} \$
\$\text{-report} \$
\$\text{-end} \$
\$\text{pdflatex surface_by_coefficients_q13_report.tex} \$
\$\text{open surface_by_coefficients_q13_report.pdf} \$

#report_with_group \
Section 7.3: Cubic Surfaces Classification

SECTION_CUBIC_SURFACES_CLASSIFICATION:

surface_classify_q4:
$\text{define } F \text{ -finite field -q 4 -end }$
$\text{define } P \text{ -projective space 3 F -end }$
$\text{with } P \text{ -do }$
$\text{-projective space activity }$
$\text{-classify_surfaces_with_double_sixes Surf27 -W -end }$
$\text{-end }$
$\text{-with Surf27 -do }$
$\text{-classification_of_cubic_surfaces_with_double_sixes_activity }$
$\text{-report -end }$
$\text{-end }$
$\text{-print_symbols}$
$\text{pdflatex Surfaces_q4.tex}$
$\text{open Surfaces_q4.pdf}$

# time: 0:00

surface_classify_q4.arc_lifting_two_lines:
$\text{define } F \text{ -finite field -q 4 -end }$
$\text{define } P \text{ -projective space 3 F -end }$
$\text{with } P \text{ -do }$
$\text{-projective space activity }$
$\text{-control_six_arcs -problem_label sixarcs_q4 -end }$
$\text{-classify_surfaces_through_arcs_and_two_lines }$
$\text{-end }$
$\text{pdflatex surfaces_arc_lifting_4.tex}$
$\text{open surfaces.arc_lifting_4.pdf}$

surface_classify_q7:
$\text{define } F \text{ -finite field -q 4 -end }$
$\text{define } P \text{ -projective space 3 F -end }$
$\text{with } P \text{ -do }$
$\text{-projective space activity }$
$\text{-control_six_arcs -problem_label sixarcs_q7 -end }$
$\text{-classify_surfaces_with_double_sixes Surf27 -W -end }$
$\text{-end }$
$\text{-with Surf27 -do }$
$\text{-classification_of_cubic_surfaces_with_double_sixes_activity }$
$\text{-report -end }$
$\text{-end }$
$\text{-print_symbols}$
$\text{pdflatex Surfaces_q7.tex}$
$\text{open Surfaces_q7.pdf}$
-define F -finite_field -q 7 -end \\
-define P -projective_space 3 F -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 \\
-report -end \\
-end \\
-print_symbols \\
pdflatex Surfaces_q7.tex \\
open Surfaces_q7.pdf \\
surface_classify_q9: \\
$(ORBITER) -v 5 \\
-define F -finite_field -q 9 -end \\
-define P -projective_space 3 F -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 \\
-report -end \\
-end \\
-print_symbols \\
pdflatex Surfaces_q9.tex \\
open Surfaces_q9.pdf \\
surface_classify_q13: \\
$(ORBITER) -v 5 \\
-define F -finite_field -q 13 -end \\
-define P -projective_space 3 F -end \\
-with P -do \\
-projective_space_activity \\
-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 \\
pdflatex Surfaces_q13.tex
# Section 7.4: Cubic Surfaces - Isomorphism Testing and Recognition

SECTION_CUBIC_SURFACES_ISOMORPHISM_TESTING_AND_RECOGNITION:

surface_recognize_q7_abcd_2_3_3_4:

$\textsc{orbiter} -v 3 \$

-define F -finite_field -q 7 -end \n
-define P -projective_space 3 F -end \n
-with P -do \n
-projective_space_activity \n
-classify_surfaces_with_double_sixes Surf -W -end \n
-end \n
-with Surf -do \n
-classification_of_cubic_surfaces_with_double_sixes_activity \n
-recognize \n
-q 7 \n
-family_general_abcd 2 3 3 4 \n
-end \n
-end \n
-end \n
-print_symbols

surface_isomorph_16:

$\textsc{orbiter} -v 3 \$

-define F -finite_field -q 16 -end \n
-define P -projective_space 3 F -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
-end \n
-print_symbols
# 1 min 8 sec on Mac from scratch (with all data files removed)

```bash
surface_recognize_8:
  $(ORBITER) -v 3
  -define F -finite_field -q 8 -end
  -define P -projective_space 3 F -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
  -recognize
  -q 8
  -by_coefficients "1,6,1,8,1,11,1,13,1,19"
  -end
  -end
  -end
  -print_symbols

surface_recognize_F13_q4:
  $(ORBITER) -v 3
  -define F -finite_field -q 4 -end
  -define P -projective_space 3 F -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
  -identify_F13
  -end
  -print_symbols

surface_sweep_Cayley_13:
  $(ORBITER) -v 3
  -define F -finite_field -q 13 -end
  -define P -projective_space 3 F -end
  -with P -do
  -projective_space_activity
```

7684  ▷ ▷ ▷ -classify_surfaces_with_double_sixes Surf27 -W -end \n7685  ▷ ▷ -end \n7686  ▷ ▷ -with Surf27 -do \n7687  ▷ ▷ -classification_of_cubic_surfaces_with_double_sixes_activity \n7688  ▷ ▷ ▷ -sweep \n7689  ▷ ▷ -end \n7690  ▷ ▷ -print_symbols
7691
7692
7693
7694  F_sweep_15_q7:
7695  ▷ $(ORBITER) -v 20 \n7696  ▷ ▷ -define F -finite_field -q 7 -end \n7697  ▷ ▷ -define P -projective_space 3 F -end \n7698  ▷ ▷ -with P -do \n7699  ▷ ▷ ▷ -projective_space_activity \n7700  ▷ ▷ ▷ ▷ -sweep_4_15_lines sweep_4_15_lines_q7 -q 7 \n7701  ▷ ▷ ▷ ▷ ▷ -by_equation "F_alpha_beta_gamma_delta" \n7702  ▷ ▷ ▷ ▷ ▷ "\DF{\alpha,\beta,\gamma,\delta}\D" "x0,x1,x2,x3" \n7703  ▷ ▷ ▷ ▷ ▷ $(F_ALPHA_BETA_GAMMA_DELTA) \n7704  ▷ ▷ ▷ ▷ ▷ "alpha=2,beta=1,gamma=2,delta=3" \n7705  ▷ ▷ ▷ ▷ ▷ "\Dalpha=2,\beta=1,\gamma=2,\delta=3\D" \n7706  ▷ ▷ ▷ ▷ ▷ -end \n7707  ▷ ▷ ▷ -end
7708
7709
7710
7711  # 0:29
7712  #F_alpha_beta_gamma_delta_q7_sweep4_15_data.csv
7713  #F_alpha_beta_gamma_delta_q7_sweep.csv
7714  #F_alpha_beta_gamma_delta_q7_points.txt
7715
7716
7717
7718
7719  ###############################################################################
7720  # Section 7.5: Cubic Surfaces of Dickson type
7721
7722
7723  SECTION_CUBIC_SURFACES_DICKSON:
7724
7725
7726
7727  D6_q2:
7728  ▷ $(ORBITER) -v 3 \n7729  ▷ ▷ -define F -finite_field -q 2 -end \n7730  ▷ ▷ -define P -projective_space 3 F -end \n
\begin{verbatim}
7731 \triangleright \triangleright with P -do \ 
7732 \triangleright \triangleright -projective_space_activity \ 
7733 \triangleright \triangleright \triangleright -define_surface S_D6_q2 -q 2 -by_coefficients \$(D6) -end \ 
7734 \triangleright \triangleright -end \ 
7735 \triangleright \triangleright with S_D6_q2 -do \ 
7736 \triangleright \triangleright -cubic_surface_activity \ 
7737 \triangleright \triangleright \triangleright -report \ 
7738 \triangleright \triangleright -end \ 
7739 \triangleright pdflatex surface_by_coefficients_q2_report.tex 
7740 \triangleright open surface_by_coefficients_q2_report.pdf 
7741 \triangleright mv surface_by_coefficients_q2_points.txt surface_by_coefficients_q2_D6_points.txt 

7742
7743 # 1 line over GF(2)
7744
7745 D3_q4:
7746 \triangleright $(ORBITER) -v 3 \ 
7747 \triangleright \triangleright -define F -finite_field -q 4 -end \ 
7748 \triangleright \triangleright -define P -projective_space 3 F -end \ 
7749 \triangleright \triangleright with P -do \ 
7750 \triangleright \triangleright -projective_space_activity \ 
7751 \triangleright \triangleright \triangleright -define_surface S_D3_q4 -q 4 -by_coefficients $(D3) -end \ 
7752 \triangleright \triangleright -end \ 
7753 \triangleright \triangleright with S_D3_q4 -do \ 
7754 \triangleright \triangleright -cubic_surface_activity \ 
7755 \triangleright \triangleright \triangleright -report \ 
7756 \triangleright \triangleright -end \ 
7757 \triangleright pdflatex surface_by_coefficients_q4_report.tex 
7758 \triangleright open surface_by_coefficients_q4_report.pdf 
7759 \triangleright mv surface_by_coefficients_q4_points.txt surface_by_coefficients_q4_D3_points.txt 

7760 #surface_by_coefficients_q4_points.txt
7761
7762 D4_q8:
7763 \triangleright $(ORBITER) -v 3 \ 
7764 \triangleright \triangleright -define F -finite_field -q 8 -end \ 
7765 \triangleright \triangleright -define P -projective_space 3 F -end \ 
7766 \triangleright \triangleright with P -do \ 
7767 \triangleright \triangleright -projective_space_activity \ 
7768 \triangleright \triangleright \triangleright -define_surface S_D4_q8 -q 8 -by_coefficients $(D4) -end \ 
\end{verbatim}
7776  ▷  ▷  -end \
7777  ▷  ▷  -with S_D4_q8 -do \
7778  ▷  ▷  -cubic_surface_activity \
7779  ▷  ▷  ▷  -report \
7780  ▷  ▷  -end 
7781  ▷  pdflatex surface_by_coefficients_q8_report.tex 
7782  ▷  open surface_by_coefficients_q8_report.pdf 
7783  ▷  mv surface_by_coefficients_q8_points.txt surface_by_coefficients_q8_D4_points.txt 

7784
7785
7786
7787
7788
7789  D6_q4:
7790  ▷  $(ORBITER) -v 3 \n7791  ▷  ▷  -define F -finite_field -q 4 -end \n7792  ▷  ▷  -define P -projective_space 3 F -end \n7793  ▷  ▷  -with P -do \n7794  ▷  ▷  -projective_space_activity \n7795  ▷  ▷  ▷  -define_surface S_D6_q4 -q 4 -by_coefficients $(D6) -end \n7796  ▷  ▷  -end \
7797  ▷  ▷  -with S_D6_q4 -do \n7798  ▷  ▷  -cubic_surface_activity \n7799  ▷  ▷  ▷  -report \n7800  ▷  ▷  -end 
7801  ▷  pdflatex surface_by_coefficients_q4_report.tex 
7802  ▷  open surface_by_coefficients_q4_report.pdf 
7803  ▷  mv surface_by_coefficients_q4_points.txt surface_by_coefficients_q4_D6_points.txt 

7804
7805  # D6 has 7 lines over GF(4)
7806
7807
7808
7809
7810
7811  D8_q4:
7812  ▷  $(ORBITER) -v 3 \n7813  ▷  ▷  -define F -finite_field -q 4 -end \n7814  ▷  ▷  -define P -projective_space 3 F -end \n7815  ▷  ▷  -with P -do \n7816  ▷  ▷  -projective_space_activity \n7817  ▷  ▷  ▷  -define_surface S_D8_q4 -q 4 -by_coefficients $(D8) -end \n7818  ▷  ▷  -end \n7819  ▷  ▷  -with S_D8_q4 -do \n7820  ▷  ▷  -cubic_surface_activity \n
643
D1-q8:

$(\text{ORBITER})$ -v 3 \
-define F -finite_field -q 8 -end \
-define P -projective_space 3 F -end \
-with P -do \
-projective_space_activity \
-define_surface S_{D1-q8} -q 8 -by_coefficients $(D1) -end \
-end \
-with S_{D1-q8} -do \
-cubic_surface_activity \
-report \
-end \
-pdflatex surface_by_coefficients_q8_report.tex \
-open surface_by_coefficients_q8_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:

$(\text{ORBITER})$ -v 3 \
-define F -finite_field -q 4 -end \
-define P -projective_space 3 F -end \
-with P -do \
-projective_space_activity \
-define_surface S_{D1-q4} -q 4 -by_coefficients $(D1) -end \
-select_double_six "3,9,15,19,22,26,4,10,14,18,21,25" -end \
-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
D1_q4_with_select_double_six_b:
  $(ORBITER) -v 3 \\
  -define F -finite_field -q 4 -end \\
  -define P -projective_space 3 F -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:
  $(ORBITER) -v 5 -define F -finite_field -q 4 -end \\
  -move_two_lines_in_hyperplane_stabilizer_text \\
  "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 \\
  D1_q4_with_select_double_six_c:
  $(ORBITER) -v 3 \\
  -define F -finite_field -q 4 -end \\
  -define P -projective_space 3 F -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 \\
  -end \\
  -with S_D1_q4 -do \\
  -cubic_surface_activity \\
  -report \\

645
orbits_cubic_surfaces_q3:

$(ORBITER) -v 4
-define G -linear_group -PGL 4 3 -end
-with G -do
-group_theoretic_activity
-orbits_on_polynomials 3
-end
-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
-define G
-linear_group -PGL 3 2
-end
-with G -do
-group_theoretic_activity
-orbits_on_polynomials 3
-end
-pdflatex poly_orbits_d3_n2_q2.tex
-open poly_orbits_d3_n2_q2.pdf

orbits_cubic_curves_q3:

$(ORBITER) -v 4
-define G
-linear_group -PGL 3 3
-end
-with G -do
-group_theoretic_activity
-orbits_on_polynomials 3
-end
-pdflatex poly_orbits_d3_n2_q3.tex
-open poly_orbits_d3_n2_q3.pdf
Compute and analyze properties over $\mathbb{F}_2$.

```
poly_orbits_d3_n3_q2_F2.csv: poly_orbits_d3_n3_q2.csv
```

```
$\text{ORBITER} -v 4 \$
```

```
define F -finite_field -q 2 -end \n```

```
define P -projective_space 3 F -end \n```

```
with P -do \n```

```
projective_space_activity \n```

```
table_of_cubic_surfaces_compute_properties \n```

```
projective_space_activity \n```

```
cubic_surface_properties_analyze \n```

```
poly_orbits_d3_n3_q2.csv 2 0 \n```

```
end \n```

```
Dickson_q2_analyze: poly_orbits_d3_n3_q2_F2.csv
```

```
$\text{ORBITER} -v 4 \$
```

```
define F -finite_field -q 2 -end \n```

```
define P -projective_space 3 F -end \n```

```
with P -do \n```

```
projective_space_activity \n```

```
table_of_cubic_surfaces_compute_properties \n```

```
poly_orbits_d3_n3_q2.csv 2 0 \n```

```
end \n```

```
Dickson_q4_analyze: poly_orbits_d3_n3_q2_F4.csv
```

```
$\text{ORBITER} -v 4 \$
```

```
define F -finite_field -q 4 -end \n```

```
define P -projective_space 3 F -end \n```

```
with P -do \n```

```
projective_space_activity \n```

```
table_of_cubic_surfaces_compute_properties \n```

```
poly_orbits_d3_n3_q2.csv 2 0 \n```

```
end \n```

# Compute and analyze properties over $\mathbb{F}_4$

```
poly_orbits_d3_n3_q2_F4.csv: poly_orbits_d3_n3_q2.csv
```

```
$\text{ORBITER} -v 4 \$
```

```
define F -finite_field -q 4 -end \n```

```
define P -projective_space 3 F -end \n```

```
with P -do \n```

```
projective_space_activity \n```

```
table_of_cubic_surfaces_compute_properties \n```

```
poly_orbits_d3_n3_q2.csv 2 \n```

```
end \n```

```
Dickson_q4_analyze: poly_orbits_d3_n3_q2_F4.csv
```

```
$\text{ORBITER} -v 4 \$
```

```
define F -finite_field -q 4 -end \n```

```
define P -projective_space 3 F -end \n```

```
with P -do \n```

```
projective_space_activity \n```
8008 ▷ ▷ -end
8009 ▷ pdflatex poly_orbits_d3_n3_q2_F4_report.tex
8010 ▷ open poly_orbits_d3_n3_q2_F4_report.pdf
8011
8012 # compute and analyze properties over F8
8013
8014 poly_orbits_d3_n3_q2_F8.csv: poly_orbits_d3_n3_q2.csv
8015 ▷ $(ORBITER) -v 4 \n
8016 ▷ ▷ -define F -finite_field -q 8 -end \n
8017 ▷ ▷ -define P -projective_space 3 F -end \n
8018 ▷ ▷ -with P -do \n
8019 ▷ ▷ -projective_space_activity \n
8020 ▷ ▷ -table_of_cubic_surfaces_compute_properties \n
8021 ▷ ▷ ▷ poly_orbits_d3_n3_q2.csv 2 0 \n
8022 ▷ ▷ -end

8023 Dickson_q8.analyze: poly_orbits_d3_n3_q2_F8.csv
8024
8025 ▷ $(ORBITER) -v 4 \n
8026 ▷ ▷ -define F -finite_field -q 8 -end \n
8027 ▷ ▷ -define P -projective_space 3 F -end \n
8028 ▷ ▷ -with P -do \n
8029 ▷ ▷ -projective_space_activity \n
8030 ▷ ▷ -cubic_surface_properties.analyze \n
8031 ▷ ▷ ▷ poly_orbits_d3_n3_q2_F8.csv 2 \n
8032 ▷ ▷ -end

8033 pdflatex poly_orbits_d3_n3_q2_F8_report.tex
8034 open poly_orbits_d3_n3_q2_F8_report.pdf

8035
8036
8037 # compute and analyze properties over F16
8038
8039 poly_orbits_d3_n3_q2_F16.csv: poly_orbits_d3_n3_q2.csv
8040 ▷ $(ORBITER) -v 4 \n
8041 ▷ ▷ -define F -finite_field -q 16 -end \n
8042 ▷ ▷ -define P -projective_space 3 F -end \n
8043 ▷ ▷ -with P -do \n
8044 ▷ ▷ -projective_space_activity \n
8045 ▷ ▷ -table_of_cubic_surfaces_compute_properties \n
8046 ▷ ▷ ▷ poly_orbits_d3_n3_q2.csv 2 0 \n
8047 ▷ ▷ -end

8048
8049 Dickson_q16.analyze: poly_orbits_d3_n3_q2_F16.csv
8050
8051 ▷ $(ORBITER) -v 4 \n
8052 ▷ ▷ -define F -finite_field -q 16 -end \n
8053 ▷ ▷ -define P -projective_space 3 F -end \n
8054 ▷ ▷ -with P -do \n
648
+++ MAKE_TABLE_OF_CUBIC_SURFACES=-define P -projective_space 3 F -end 
   + with P -do 
   + -projective_space_activity 
   + -table_of_cubic_surfaces 
   + -end

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

cubic_surfaces_table_latex_17:
$ORBITER -v 3 -csv_file_latex 1 
$ORBITER -table_of_cubic_surfaces_q17_info.csv

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 7 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
$ORBITER -v 3 -define F -finite_field -q 8 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
$ORBITER -v 3 -define F -finite_field -q 9 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
$ORBITER -v 3 -define F -finite_field -q 11 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
$ORBITER -v 3 -define F -finite_field -q 13 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
$ORBITER -v 3 -define F -finite_field -q 16 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
cubic_surfaces_tables_19_37:

$\text{(ORBITER)} -v 3 \text{-define F finite_field -q 17 -end $(MAKE\_TABLE\_OF\_CUBIC\_SURFACES)}$

$\text{(ORBITER)} -v 3 \text{-define F finite_field -q 19 -end $(MAKE\_TABLE\_OF\_CUBIC\_SURFACES)}$

$\text{(ORBITER)} -v 3 \text{-define F finite_field -q 23 -end $(MAKE\_TABLE\_OF\_CUBIC\_SURFACES)}$

$\text{(ORBITER)} -v 3 \text{-define F finite_field -q 25 -end $(MAKE\_TABLE\_OF\_CUBIC\_SURFACES)}$

$\text{(ORBITER)} -v 3 \text{-define F finite_field -q 27 -end $(MAKE\_TABLE\_OF\_CUBIC\_SURFACES)}$

$\text{(ORBITER)} -v 3 \text{-define F finite_field -q 29 -end $(MAKE\_TABLE\_OF\_CUBIC\_SURFACES)}$

$\text{(ORBITER)} -v 3 \text{-define F finite_field -q 31 -end $(MAKE\_TABLE\_OF\_CUBIC\_SURFACES)}$

$\text{(ORBITER)} -v 3 \text{-define F finite_field -q 32 -end $(MAKE\_TABLE\_OF\_CUBIC\_SURFACES)}$

$\text{(ORBITER)} -v 3 \text{-define F finite_field -q 37 -end $(MAKE\_TABLE\_OF\_CUBIC\_SURFACES)}$

$\text{(ORBITER)} -v 3 \text{-define F finite_field -q 41 -end $(MAKE\_TABLE\_OF\_CUBIC\_SURFACES)}$

$\text{(ORBITER)} -v 3 \text{-define F finite_field -q 43 -end $(MAKE\_TABLE\_OF\_CUBIC\_SURFACES)}$

$\text{(ORBITER)} -v 3 \text{-define F finite_field -q 47 -end $(MAKE\_TABLE\_OF\_CUBIC\_SURFACES)}$

$\text{(ORBITER)} -v 3 \text{-define F finite_field -q 49 -end $(MAKE\_TABLE\_OF\_CUBIC\_SURFACES)}$

$\text{(ORBITER)} -v 3 \text{-define F finite_field -q 53 -end $(MAKE\_TABLE\_OF\_CUBIC\_SURFACES)}$

$\text{(ORBITER)} -v 3 \text{-define F finite_field -q 59 -end $(MAKE\_TABLE\_OF\_CUBIC\_SURFACES)}$

$\text{(ORBITER)} -v 3 \text{-define F finite_field -q 61 -end $(MAKE\_TABLE\_OF\_CUBIC\_SURFACES)}$

$\text{(ORBITER)} -v 3 \text{-define F finite_field -q 64 -end $(MAKE\_TABLE\_OF\_CUBIC\_SURFACES)}$

$\text{(ORBITER)} -v 3 \text{-define F finite_field -q 67 -end $(MAKE\_TABLE\_OF\_CUBIC\_SURFACES)}$

$\text{(ORBITER)} -v 3 \text{-define F finite_field -q 71 -end $(MAKE\_TABLE\_OF\_CUBIC\_SURFACES)}$

$\text{(ORBITER)} -v 3 \text{-define F finite_field -q 73 -end $(MAKE\_TABLE\_OF\_CUBIC\_SURFACES)}$
$\text{(ORBITER)} -v 3 -define F -finite_field -q 79 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$\text{(ORBITER)} -v 3 -define F -finite_field -q 81 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$\text{(ORBITER)} -v 3 -define F -finite_field -q 83 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$\text{(ORBITER)} -v 3 -define F -finite_field -q 89 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$\text{(ORBITER)} -v 3 -define F -finite_field -q 97 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$\text{(ORBITER)} -v 3 -define F -finite_field -q 101 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$\text{(ORBITER)} -v 3 -define F -finite_field -q 103 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$\text{(ORBITER)} -v 3 -define F -finite_field -q 107 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$\text{(ORBITER)} -v 3 -define F -finite_field -q 109 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$\text{(ORBITER)} -v 3 -define F -finite_field -q 113 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$\text{(ORBITER)} -v 3 -define F -finite_field -q 121 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$\text{(ORBITER)} -v 3 -define F -finite_field -q 127 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$\text{(ORBITER)} -v 3 -define F -finite_field -q 128 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$\text{(ORBITER)} -v 3 -csv -csv_file_latex 1 test.csv

$\text{(ORBITER)} -v 3 -csv -csv_file_latex 0 table_of_cubic_surfaces_q4_info.csv

$\text{(ORBITER)} -v 3 -csv -csv_file_latex 0 table_of_cubic_surfaces_q7_info.csv

$\text{(ORBITER)} -v 3 -csv -csv_file_latex 0 table_of_cubic_surfaces_q8_info.csv

$\text{(ORBITER)} -v 3 -csv -csv_file_latex 0 table_of_cubic_surfaces_q9_info.csv

$\text{(ORBITER)} -v 3 -csv -csv_file_latex 0 table_of_cubic_surfaces_q11_info.csv

$\text{(ORBITER)} -v 3 -csv -csv_file_latex 0 table_of_cubic_surfaces_q13_info.csv

$\text{(ORBITER)} -v 3 -csv -csv_file_latex 0 table_of_cubic_surfaces_q16_info.csv

$\text{(ORBITER)} -v 3 -csv -csv_file_latex 0 table_of_cubic_surfaces_q17_info.csv

$\text{(ORBITER)} -v 3 -csv -csv_file_latex 0 table_of_cubic_surfaces_q19_info.csv

$\text{(ORBITER)} -v 3 -csv -csv_file_latex 0 table_of_cubic_surfaces_q23_info.csv

$\text{(ORBITER)} -v 3 -csv -csv_file_latex 0 table_of_cubic_surfaces_q25_info.csv

$\text{(ORBITER)} -v 3 -csv -csv_file_latex 0 table_of_cubic_surfaces_q27_info.csv

$\text{(ORBITER)} -v 3 -csv -csv_file_latex 0 table_of_cubic_surfaces_q29_info.csv

$\text{(ORBITER)} -v 3 -csv -csv_file_latex 0 table_of_cubic_surfaces_q31_info.csv
surface_table:
$\texttt{\$(ORBITER) -v 3 -make_table_of_surfaces}$
$\texttt{pdflatex surfaces_report.tex}$
$\texttt{open surfaces_report.pdf}$

surface_atlas:
$\texttt{\$(ORBITER) -v 3 -create_surface_atlas 97}$
$\texttt{~/bin/tth surface_atlas.tex}$

surface_reports:
$\texttt{\$(ORBITER) -v 3 \ -orbiter_path \$(ORBITER\_PATH) -create_surface_reports 4,7,8,9,11}$

quartic_curve_tables_23:
$\texttt{\$(ORBITER) -v 3 \ -define F -finite_field -q 23 -end \ -define P -projective_space 2 F -end \ -with P -do \ -projective_space_activity \ -table_of_quartic_curves \ -end}$

quartic_curve_tables:
\begin{verbatim}
8203 \> \$\$(ORBITER) -v 3 \ 8204 \> \> -define F -finite_field -q 9 -end \ 8205 \> \> -define P -projective_space 2 F -end \ 8206 \> \> -with P -do \ 8207 \> \> \> -projective_space_activity \ 8208 \> \> \> \> -table_of_quartic_curves \ 8209 \> \> \> \> -end \ 8210 \> \$\$(ORBITER) -v 3 \ 8211 \> \> -define F -finite_field -q 13 -end \ 8212 \> \> -define P -projective_space 2 F -end \ 8213 \> \> -with P -do \ 8214 \> \> \> -projective_space_activity \ 8215 \> \> \> \> -table_of_quartic_curves \ 8216 \> \> \> \> -end \ 8217 \> \$\$(ORBITER) -v 3 \ 8218 \> \> -define F -finite_field -q 17 -end \ 8219 \> \> -define P -projective_space 2 F -end \ 8220 \> \> -with P -do \ 8221 \> \> \> -projective_space_activity \ 8222 \> \> \> \> -table_of_quartic_curves \ 8223 \> \> \> \> -end \ 8224 \ 8225 \ 8226 \> quartic_curve_tables_latex: \ 8227 \> \> \$\$(ORBITER) -v 3 -csv_file_latex 1 test.csv \ 8228 \> \> \$\$(ORBITER) -v 3 -csv_file_latex 0 quartic_curves_q9_info.csv \ 8229 \> \> \$\$(ORBITER) -v 3 -csv_file_latex 0 quartic_curves_q13_info.csv \ 8230 \> \> \$\$(ORBITER) -v 3 -csv_file_latex 0 quartic_curves_q17_info.csv \ 8231 \> \> \$\$(ORBITER) -v 3 -csv_file_latex 0 quartic_curves_q19_info.csv \ 8232 \> \> \$\$(ORBITER) -v 3 -csv_file_latex 0 quartic_curves_q25_info.csv \ 8233 \> \> \$\$(ORBITER) -v 3 -csv_file_latex 0 quartic_curves_q27_info.csv \ 8234 \> \> \$\$(ORBITER) -v 3 -csv_file_latex 0 quartic_curves_q29_info.csv \ 8235 \> \> \$\$(ORBITER) -v 3 -csv_file_latex 0 quartic_curves_q31_info.csv \ 8236 \> \#\$\$(ORBITER) -v 3 -csv_file_latex 1 quartic_curves_q9_info.csv \ 8237 \> \#pdflatex quartic_curves_q13_info.tex \ 8238 \> \#open quartic_curves_q13_info.pdf \ 8239 \> \#~/bin/tth quartic_curves_q13_info.tex \ 8240 \> \#open quartic_curves_q13_info.html \ 8241 \ 8242 \ 8243 \ 8244 \ 8245 \ 8246 \ 8247 \#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% \ 8248 \# Chapter 8 - Ring Theory \ 8249 \#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\end{verbatim}
# Section 8.1: Polynomials over Finite Fields

SECTION POLYNOMIALS:

# check which polynomials are irreducible and which are primitive:

sift polynomials deg3 q2:
  $(ORBITER) -v 2 \\  -define F -finite_field -q 2 -end \\  -with F -do \\  -finite_field_activity -sift_polynomials 8 16 -end

sift polynomials deg4 q2:
  $(ORBITER) -v 2 \\  -define F -finite_field -q 2 -end \\  -with F -do \\  -finite_field_activity -sift_polynomials 16 32 -end

poly_division:
  $(ORBITER) -v 2 \\  -define F -finite_field -q 2 -end \\  -with F -do \  -finite_field_activity \  -polynomial_division "1,0,0,0,0,0,0,0,0,0,1" "1,0,1,1" -end

poly_division2:
  $(ORBITER) -v 2 \\  -define F -finite_field -q 2 -end \\  -define A -vector -field F -sparse 11 "1,0,1,10" -end \\  -define B -vector -field F -dense "1,0,1,1" -end \\  -with F -do \  -finite_field_activity \  -polynomial_division A B -end
poly_gcd:
-define F -finite_field -q 2 -end 
-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:
-define F -finite_field -q 7 -end 
-finite_field_activity 
-polynomial_mult_mod "1,2,3" "3,4,5" "6,0,0,1" -end 

poly_mult_mod2:
-define F -finite_field -q 7 -end 
-finite_field_activity 
-polynomial_mult_mod "3,1,2" "5,3,4" "6,0,0,1" -end 

poly_mult_mod_F4:
-define F -finite_field -q 2 -end 
-finite_field_activity 
-polynomial_mult_mod "1,1" "1,1" "1,1,1" -end 

poly_mult_mod_F4:
-define F -finite_field -q 2 -end 
-finite_field_activity 
-polynomial_mult_mod "0,1" "1,1" "1,1,1" -end 

$(ORBITER) -v 2 
-polynomial_mult_mod "0,1" "0,1" "1,1,1" -end 

mult_polynomials_2,5,7:
 $(ORBITER) -v 2 \
-dfni $F$ -finite_field -q 2 -end \
-with $F$ -do \
-finite_field_activity -mult_polynomials 5 7 -end 

$\text{pdflatex polynomial\_mult\_5\_7.tex}$

$\text{open polynomial\_mult\_5\_7.pdf}$

$\text{polynomial\_division\_ranked\_2\_27\_13}$:

$\text{pdflatex polynomial\_division\_27\_13.tex}$

$\text{open polynomial\_division\_27\_13.pdf}$

$\text{mult\_polynomials\_2\_8\_15}$:

$\text{pdflatex polynomial\_mult\_8\_15.tex}$

$\text{open polynomial\_mult\_8\_15.pdf}$

$\text{polynomial\_division\_ranked\_2\_120\_25}$:

$\text{pdflatex polynomial\_division\_120\_25.tex}$

$\text{open polynomial\_division\_120\_25.pdf}$

# the answer is 5

$\text{mult\_polynomials\_2\_7\_7}$:

$\text{pdflatex mult\_polynomials\_7\_7\_7.tex}$

$\text{open mult\_polynomials\_7\_7\_7.pdf}$
mult_polynomials_2_4_6:
\[$(\text{ORBITER}) \ -v \ 2 \ \backslash\$
\begin{verbatim}
  \>$\text{-define F -finite_field -q 2 -end} \ \backslash
  \>$\text{-with F -do} \ \backslash
  \>$\text{-finite_field_activity} \ \backslash
  \>$\text{-mult_polynomials 4 6 -end} \ \backslash
  \>$\text{pdflatex polynomial_mult_4_6.tex} \ \backslash
  \>$\text{open polynomial_mult_4_6.pdf} \ \backslash
\end{verbatim}

\[\text{polynomial_division_ranked_2_24_13:}\]
\[$(\text{ORBITER}) \ -v \ 2 \ \backslash\$
\begin{verbatim}
  \>$\text{-define F -finite_field -q 2 -end} \ \backslash
  \>$\text{-with F -do} \ \backslash
  \>$\text{-finite_field_activity} \ \backslash
  \>$\text{-polynomial_division_ranked 24 13} \ \backslash
  \>$\text{-end} \ \backslash
  \>$\text{pdflatex polynomial_division_24_13.tex} \ \backslash
  \>$\text{open polynomial_division_24_13.pdf} \ \backslash
\end{verbatim}

\[\text{mult_polynomials_1024_999_997:}\]
\[$(\text{ORBITER}) \ -v \ 2 \ \backslash\$
\begin{verbatim}
  \>$\text{-define F -finite_field -q 2 -end} \ \backslash
  \>$\text{-with F -do} \ \backslash
  \>$\text{-finite_field_activity} \ \backslash
  \>$\text{-mult_polynomials 999 997} \ \backslash
  \>$\text{-end} \ \backslash
  \>$\text{pdflatex polynomial_mult_999_997.tex} \ \backslash
  \>$\text{open polynomial_mult_999_997.pdf} \ \backslash
\end{verbatim}

\[\text{polynomial_division_ranked_2_349147_1033:}\]
\[$(\text{ORBITER}) \ -v \ 2 \ \backslash\$
\begin{verbatim}
  \>$\text{-define F -finite_field -q 2 -end} \ \backslash
  \>$\text{-with F -do} \ \backslash
  \>$\text{-finite_field_activity} \ \backslash
  \>$\text{-polynomial_division_ranked 349147 1033} \ \backslash
  \>$\text{-end} \ \backslash
  \>$\text{pdflatex polynomial_division_349147_1033.tex} \ \backslash
  \>$\text{open polynomial_division_349147_1033.pdf} \ \backslash
\end{verbatim}
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
# evaluates to 61

test_mult_polynomials_17_12:
  $ (ORBITER) -v 2 \n  -define F -finite_field -q 2 -end \n  -with F -do \n  -finite_field_activity \n  -mult_polynomials 17 12 -end
  pdflatex polynomial_mult_17_12.tex
  open polynomial_mult_17_12.pdf
# gives 204

test_polynomial_division_ranked_2_204_37:
  $ (ORBITER) -v 2 \n  -define F -finite_field -q 2 -end \n  -with F -do \n  -finite_field_activity \n  -polynomial_division_ranked 204 37 \n  -end
  pdflatex polynomial_division_204_37.tex
  open polynomial_division_204_37.pdf
# answer is 18

test_crc32:
  $ (ORBITER) -v 3 \n  -crc32 "123456789"
Berlekamp_matrix_crc32:

```
$ (ORBITER) -v 2 \n  -define F -finite_field -q 2 -end \n  -define v -vector -field F -sparse 33 $(CRC32_SPARSE) -end \n  -with F -do \n  -finite_field_activity \n  -Berlekamp_matrix v -end
```

```
# N = 2^32-1 = 3 * 5 * 17 * 257 * 65537
# N / 3 = 1431655765
# N / 5 = 858993459
# N / 17 = 252645135
# N / 257 = 16711935
# N / 65537 = 65535
```

```
TWO_TO_THE_32_MINUS_2=4294967294
```

```
power_mod_inverse:
$ (ORBITER) -v 2 \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
```

```
INVERSE_SPARSE="1,31,1,25,1,22,1,21,1,15,\n 1,11,1,10,1,9,1,7,1,6,1,4,1,3,1,1,1,0"
```

```
A(X)=X^{31} + X^{25} + X^{22} + X^{21} + X^{15} + X^{11} + X^{10} + X^{9} + X^{7} + X^{6} + X^{4} + X^{3} + X + 1
```

```
mult_mod_to_get_one:
$ (ORBITER) -v 2 \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
```
Berlekamp matrix 2:

```plaintext
$ \text{define } F \text{ -finite_field -q 2 -end} \\
$ \text{define } v \text{ -vector -field F -dense } "1,1,0,1" \text{ -end} \\
$ \text{with F } \text{-do } \\
$ \text{finite_field.activity} \\
$ \text{-Berlekamp_matrix } v \text{-end} \\
```

# the polynomial $X^3+X+1$ is irreducible over GF(2) because the rank of the Berlekamp matrix is 2.

Berlekamp matrix 4:

```plaintext
$ \text{define } F \text{ -finite_field -q 4 -end} \\
$ \text{define } v \text{ -vector -field F -dense } "1,3,0,1" \text{ -end} \\
$ \text{with F } \text{-do } \\
$ \text{finite_field.activity} \\
$ \text{-Berlekamp_matrix } v \text{-end} \\
```

# the polynomial $X^4+X+1$ is irreducible over GF(2) because the rank of the Berlekamp matrix is 3.
find_roots_a:

find_roots_b:

find_roots_c:

find_roots_d:

find_roots_e:

roots_over_F2:
define v -vector -field F -dense "0,1,0,1,1,1" -end
with F -do 
finite_field_activity 
polynomial_find_roots v -end

roots_over_F8:
$\text{ORBITER} -v 2 \$
$\text{define F -finite_field -q 8 -override_polynomial 11 -end} \$
$\text{define v -vector -field F -dense "0,1,0,1,1,1" -end} \$
$\text{with F -do} \$
finite_field_activity 
polynomial_find_roots v -end

# degree and then order of the field of coefficients:
irred_3_2:
$\text{ORBITER} -v 3 \$
$\text{define F -finite_field -q 2 -end} \$
$\text{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:
$\text{ORBITER} -v 3 \$
$\text{define F -finite_field -q 2 -end} \$
$\text{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:
$\text{ORBITER} -v 3 \$
$\text{define F -finite_field -q 2 -end} \$
$\text{with F -do} \$
finite_field_activity 
make_table_of_irreducible_polynomials 5 -end

662
8671 \> pdflatex Irred_q2_d5.tex  
8672 \> open Irred_q2_d5.pdf  
8673  
8674 \> # 6 polys  
8675  
8676 \> irred_6_2:  
8677 \> $(\text{ORBITER}) -v 3 \ \  
8678 \> \> -define F -finite_field -q 2 -end \ \  
8679 \> \> -with F -do \ \  
8680 \> \> -finite_field_activity \ \  
8681 \> \> -make_table_of_irreducible_polynomials 6 -end  
8682 \> pdflatex Irred_q2_d6.tex  
8683 \> open Irred_q2_d6.pdf  
8684  
8685 \> # 9 polys  
8686  
8687 \> irred_7_2:  
8688 \> $(\text{ORBITER}) -v 3 \ \  
8689 \> \> -define F -finite_field -q 2 -end \ \  
8690 \> \> -with F -do \ \  
8691 \> \> -finite_field_activity \ \  
8692 \> \> -make_table_of_irreducible_polynomials 7 -end  
8693 \> pdflatex Irred_q2_d7.tex  
8694 \> open Irred_q2_d7.pdf  
8695  
8696 \> # 18 polys  
8697  
8698 \> irred_8_2:  
8699 \> $(\text{ORBITER}) -v 3 \ \  
8700 \> \> -define F -finite_field -q 2 -end \ \  
8701 \> \> -with F -do \ \  
8702 \> \> -finite_field_activity \ \  
8703 \> \> -make_table_of_irreducible_polynomials 8 -end  
8704 \> pdflatex Irred_q2_d8.tex  
8705 \> open Irred_q2_d8.pdf  
8706  
8707 \> # 30 polys  
8708  
8709 \> irred_9_2:  
8710 \> $(\text{ORBITER}) -v 3 \ \  
8711 \> \> -define F -finite_field -q 2 -end \ \  
8712 \> \> -with F -do \ \  
8713 \> \> -finite_field_activity \ \  
8714 \> \> -make_table_of_irreducible_polynomials 9 -end  
8715 \> pdflatex Irred_q2_d9.tex  
8716 \> open Irred_q2_d9.pdf  
8717
irred_10_2:

PDFLATEX Irred_q2_d10.tex

open Irred_q2_d10.pdf

irred_2_4:

PDFLATEX Irred_q4_d2.tex

open Irred_q4_d2.pdf

irred_3_4:

PDFLATEX Irred_q4_d3.tex

open Irred_q4_d3.pdf

searchPrimitivePoly_2:

PDFLATEX Irred_q4_d2.tex

open Irred_q4_d2.pdf

# stuck in factoring 2^61-1 (which is prime)
8765  search_primitive_poly_3:
8766  $(ORBITER) -v 6 \n8767  $$ -search_for_primitive_polynomial_in_range 3 3 2 60
8768  $
8769
8770  search_primitive_poly_4:
8771  $(ORBITER) -v 6 \n8772  $$ -search_for_primitive_polynomial_in_range 4 4 2 30
8773  $
8774  search_primitive_poly_5:
8775  $(ORBITER) -v 6 \n8776  $$ -search_for_primitive_polynomial_in_range 5 5 2 30
8777  $
8778
8779  search_primitive_poly_7:
8780  $(ORBITER) -v 6 \n8781  $$ -search_for_primitive_polynomial_in_range 7 7 2 20
8782  $
8783
8784  search_primitive_poly_8:
8785  $(ORBITER) -v 6 \n8786  $$ -search_for_primitive_polynomial_in_range 8 8 2 20
8787  $
8788
8789  search_primitive_poly_9:
8790  $(ORBITER) -v 6 \n8791  $$ -search_for_primitive_polynomial_in_range 9 9 2 15
8792  $
8793
8794  search_primitive_poly_11:
8795  $(ORBITER) -v 6 \n8796  $$ -search_for_primitive_polynomial_in_range 11 11 2 15
8797  $
8798
8799  search_primitive_poly_13:
8800  $(ORBITER) -v 6 \n8801  $$ -search_for_primitive_polynomial_in_range 13 13 2 15
8802  $
8803
8804  search_primitive_poly_degree_16:
8805  $(ORBITER) -v 6 \n8806  $$ -search_for_primitive_polynomial_in_range 2 2 16 16
8807  $
8808
8809
8810
8811
Section 8.2: Multivariate Polynomials

SECTION
MULTIVARIATE
POLYNOMIALS:

CREMONA MAP
Y0 = "3*y0*y0*y0*y0*y2+4*y0*y0*y1*y2*y1*y2\n+6*y0*y1*y2*y2*y2+9*y0*y2*y2*y2*y2*y2"

CREMONA MAP
Y1 = "y0*y0*y0*y0*y2+y0*y2*y2\n+3*y0*y1*y1*y1*y1*y1+y0*y1*y1*y1*y1*y1"

CREMONA MAP
Y2 = "10*y0*y0*y0*y0*y0*y0+11*y0*y0*y0*y2*y2\n+4*y0*y0*y1*y1*y1*y1+y0*y1*y1*y1*y1*y1"

CREMONA MAP
Y3 = "0"

Cremona map:
-define F -finite_field -q 13 -end 
-define P -projective_space 2 F -end 
-define R -polynomial_ring 
-define Y0 -formula 
"y0" "y_0" "y0,y1,y2" 
-define Y1 -formula 
"y1" "y_1" "y0,y1,y2" 
-define Y2 -formula 
"y2" "y_2" "y0,y1,y2" 
-define Cremona -collection "Y0,Y1,Y2" 
-with P -do 

8859 ▶ ▶ -projective_space_activity \n8860 ▶ ▶ ▶ -map R Cremona "" \n8861 ▶ ▶ -end
8862
8863
8864
8865
8866 arcs_5_2_q11:
8867 ▶ $(ORBITER) -v 4 \n8868 ▶ ▶ -define F -finite_field -q 11 -end \n8869 ▶ ▶ -define P -projective_space 2 F -end \n8870 ▶ ▶ -with P -do \n8871 ▶ ▶ -projective_space_activity \n8872 ▶ ▶ ▶ -classify.arcs \n8873 ▶ ▶ ▶ ▶ -poset_classification_control \n8874 ▶ ▶ ▶ ▶ ▶ -problem_label arcs_5_2_q11 \n8875 ▶ ▶ ▶ ▶ ▶ -W -depth 5 \n8876 ▶ ▶ ▶ ▶ ▶ -report -end \n8877 ▶ ▶ ▶ ▶ -end \n8878 ▶ ▶ ▶ ▶ -target_size 5 \n8879 ▶ ▶ ▶ ▶ -d 2 \n8880 ▶ ▶ ▶ -end \n8881 ▶ ▶ -end
8882 ▶ pdflatex arcs_5_2_q11_poset.tex
8883 ▶ open arcs_5_2_q11_poset.pdf
8884
8885
8886 # 2 orbits:
8887 # 0 1 2 3 37
8888 # 0 1 2 3 49
8889
8890
8891 arcs_5_2_q11_ideal:
8892 ▶ $(ORBITER) -v 2 \n8893 ▶ ▶ -define F -finite_field -q 11 -end \n8894 ▶ ▶ -define R -polynomial_ring \n8895 ▶ ▶ ▶ -field F \n8896 ▶ ▶ ▶ -number_of_variables 3 \n8897 ▶ ▶ ▶ -homogeneous_of_degree 2 \n8898 ▶ ▶ ▶ -monomial_ordering_lex \n8899 ▶ ▶ ▶ -variables "x0,x1,x2" "x_0,x_1,x_2" \n8900 ▶ ▶ ▶ -end \n8901 ▶ ▶ -define C -combinatorial_objects \n8902 ▶ ▶ ▶ -file_of_points arcs_5_2_q11_lvl_5 \n8903 ▶ ▶ ▶ -end \n8904 ▶ ▶ -with C -do \n8905 ▶ ▶ -combinatorial_object_activity \n
667
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, 41, 49, 58, 77, 83, 95, 109, 130 )

The ideal has dimension 2
generators for the ideal:
#0 1 0 0 2 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 0
#x0*x0*x1 + 2*x0*x1*x1 + 2*x0*x1*x3
#2*x2*x2*x3 + 2*x2*x3*x3
SURFACE_F.9 = "x0*x0*x1 - x0*x1*x1 - x0*x1*x3 - x2*x2*x3 - x2*x3*x3"

F.9.q7:

-define F -finite_field -q 7 -end 
-define P -projective_space 3 F -end 
-with P -do 
-projective_space_activity 
-define_surface F.9 -q 7 
-by_equation "F.9" 
"DF.9" "x0,x1,x2,x3" 
\"" 
\"\Dno parameters\D" 
-\-end 
-\-end 
-with F.9 -do 
-cubic_surface_activity 
-reported 
-end 
pdflatex surface_equation_F.9_q7_report.tex
open surface_equation_F.9_q7_report.pdf
# we create 20 5-subsets of PG(2,11) at random. Note that PG(2,11) has 133 points .
random_k_subsets:
\$\text{ORBITER} -v 4 
\$\text{ORBITER} -v 4
-create_random_k_subsets 133 5 20

# We compute the line intersections:
line_type_in_PG.2.11:
\$\text{ORBITER} -v 3 
-orbiter_path $(\text{ORBITER\_PATH}) 
-define F -finite_field -q 11 -end 
-define P -projective_space 2 F -end 
-define C -combinatorial_objects 

669
-file_of_points random_k_subsets_n133_k5_nb20.csv

-define F -finite_field -q 11 -end

-define R -polynomial_ring

-number_of_variables 3

-homogeneous_of_degree 2

-monomial_ordering_lex

-variables "x0,x1,x2" "x_0,x_1,x_2"

-end

-define C -combinatorial_objects

-set_of_points "3,33,40,83,102"

-end

-with C -do

-combinatorial_object_activity

-ideal R

-end

# the second one is an arc: 3,33,40,83,102

# we compute the ideal:

random_arc_5_2_q11_ideal:

$\text{(ORBITER)} -v 2$

-define F -field F

-number_of_variables 3

-homogeneous_of_degree 2

-monomial_ordering_lex

-variables "x0,x1,x2" "x_0,x_1,x_2"

-end

-define C -combinatorial_objects

-set_of_points "3,33,40,83,102"

-end

-with C -do

-combinatorial_object_activity

-ideal R

-end

#generator 0 / 1 is 10*x0*x0 + 3*x0*x1 + 8*x0*x2 + 2*x1*x1 + 10*x2*x2

#We found 12 points on the generator of the ideal

#They are : ( 3, 15, 19, 33, 40, 42, 46, 50, 83, 88, 102, 108 )

# Chapter 9 - Applications

# Section 9.1: Number Theory
SECTION_NUMBER_THEORY:

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

jacobi_35_41:
> $(ORBITER) -v 5 -jacobi 35 41
> pdflatex jacobi_35_41.tex
> open jacobi_35_41.pdf

jacobi_33_41:
> $(ORBITER) -v 5 -jacobi 33 41
> pdflatex jacobi_33_41.tex
> open jacobi_33_41.pdf

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

jacobi_5_19:
> $(ORBITER) -v 5 -jacobi 5 19

sqrt_mod_7817:
> $(ORBITER) -v 2 -sqrt_mod 2221 7817

# Section 9.2: Representation Theory
SECTION_REPRESENTATION_THEORY:

representation_on_polynomials_of_degree_3:
representation_tetrahedral_group_on_polynomials_of_degree_3;
$$(\text{ORBITER}) -v 2 \$
$$(\text{ORBITER}) -v 4 \$
SECTION_CRYPTOGRAPHY:
EC_add:
EC_cyclic_subgroup:
9140 ▶ $(ORBITER) -v 2 \
9141 ▶ ▶ -define F -finite_field -q 11 -end \
9142 ▶ ▶ -with F -do \n9143 ▶ ▶ -finite_field_activity \n9144 ▶ ▶ -EC_cyclic_subgroup 1 3 "1,4" -end
9145
9146
9147 EC_points_13:
9148 ▶ $(ORBITER) -v 2 \
9149 ▶ ▶ -define F -finite_field -q 13 -end \
9150 ▶ ▶ -with F -do \n9151 ▶ ▶ -finite_field_activity \n9152 ▶ ▶ -EC_points "EC_2_5_q13" 2 5 -end
9153 ▶ $(ORBITER) -v 2 -draw_matrix \
9154 ▶ ▶ -input_csv_file EC_2_5_q13_points_xy.csv \
9155 ▶ ▶ -box_width 20 -bit_depth 24 \
9156 ▶ ▶ -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" -end
9157
9158
9159
9160
9161 EC_points_199:
9162 ▶ $(ORBITER) -v 2 \
9163 ▶ ▶ -define F -finite_field -q 199 -end \
9164 ▶ ▶ -with F -do \n9165 ▶ ▶ -finite_field_activity \n9166 ▶ ▶ -EC_points "EC_5_7_q199" 5 7 -end
9167 ▶ $(ORBITER) -v 2 \
9168 ▶ ▶ -draw_matrix -input_csv_file EC_5_7_q199_points_xy.csv \
9169 ▶ ▶ -box_width 10 -bit_depth 24 \
9170 ▶ ▶ -partition 2 199 199 -end
9171
9172 EC_Koblitz_encoding:
9173 ▶ $(ORBITER) -v 6 -seed 17 \
9174 ▶ ▶ -define F -finite_field -q 199 -end \
9175 ▶ ▶ -with F -do \n9176 ▶ ▶ -finite_field_activity \n9177 ▶ ▶ -EC_Koblitz_encoding 5 7 67 "147,164" "DEADBEEF" \
9178 ▶ ▶ -end
9179
9180 EC-bsgs:
9181 ▶ $(ORBITER) -v 2 \
9182 ▶ ▶ -define F -finite_field -q 199 -end \
9183 ▶ ▶ -with F -do \n9184 ▶ ▶ -finite_field_activity \n9185 ▶ ▶ -EC-bsgs 5 7 "147,164" 212 \
9186 ▶ ▶ "172,158,45,195,50,22,10,103,55,33,50,22,145,105,31,74,73,155,67,60,25,6" \

673
```bash
EC_bsgs_decode:
  $(ORBITER) -v 2 \
  -define F -finite_field -q 199 -end \n  -with F -do \n  -finite_field_activity \n  -EC_bsgs_decode 5 7 "129,176" 212 \n  -end

"127,188,51,141,85,29,106,90,41,105,179,71,171,2,16,197,183,72,27,129,37,10" \n"50,179,169,13,153,169,115,116,188,110,176" \n- -end

NTRU_N=7
NTRU_P=3
NTRU_Q=41
NTRU_D=2
NTRUE_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"

#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"
```

674
ALICE2:

\[
\text{ORBITER} -v 2 -\text{define } F -\text{finite_field -q } (NTRU_P) -\text{end -with } F -\text{do -extended_gcd_for_polynomials -with } F -\text{do -finite_field_activity\}} \]

\[
\text{ALICE2:} \\
F_p(x) = x^6 + 2x^5 + x^3 + x^2 + x + 1
\]

ALICE\_PRIVATE\_FP="1,1,1,0,2,1"

BOB\_MESSAGE="1,-1,1,1,0,-1"

BOB\_ONE\_TIME\_KEY="-1,1,0,0,0,-1,1"

NTRU\_Alice\_public\_key:

\[
\text{ALICE}\_PUBLIC\_KEY="30,26,8,38,2,40,20"
\]

NTRU\_encrypt:

\[
\text{BOB\_ENCRYPT = "25,3,40,2,4,19,31"}
\]

NTRU\_decrypt1:
$\text{NTRU decrypt2:}$
$\text{$ORBITER$ -v 2 \$
}\text{\indent -define F -finite_field -q $(NTRUE_Q)$ -end \$
}\text{\indent -with F -do \$
}\text{\indent -finite_field_activity \$
}\text{\indent -polynomial_center_lift $(ALICE_C1)$ -end \$
}\text{\indent -end}$

$\text{C(X)=X^{6} + 10X^{5} + 33X^{4} + 40X^{3} + 40X^{2} + X + 40}$

$ALICE_C1={"40,1,40,40,33,10,1"}$

$\text{NTRU decrypt3:}$
$\text{$ORBITER$ -v 2 \$
}\text{\indent -define F -finite_field -q $(NTRUE_P)$ -end \$
}\text{\indent -with F -do \$
}\text{\indent -finite_field_activity \$
}\text{\indent -polynomial_reduce_mod_p $(ALICE_C2)$ -end \$
}\text{\indent -end}$

$\text{A(X)=X^{6} + 10X^{5} - 8X^{4} - X^{3} - X^{2} + X - 1}$

$ALICE_C2={"-1,1,-1,-1,-8,10,1"}$

$\text{NTRU decrypt4:}$
$\text{$ORBITER$ -v 2 \$
}\text{\indent -define F -finite_field -q $(NTRUE_Q)$ -end \$
}\text{\indent -with F -do \$
}\text{\indent -finite_field_activity \$
}\text{\indent -polynomial_mult_mod $(ALICEPRIVATE_F)$ \$
}\text{\indent -end}$

$\text{C(X)=2X^{5} + X^{3} + X^{2} + 2X + 1}$

$ALICE_C4={"1,2,1,1,0,2"}$

$\text{NTRU decrypt5:}$
$\text{$ORBITER$ -v 2 \$
}\text{\indent -define F -finite_field -q $(NTRUE_P)$ -end \$
}\text{\indent -with F -do \$
}\text{\indent -finite_field_activity \$
}\text{\indent -polynomial_center_lift $(ALICE_C4)$ -end \$
}\text{\indent -end}$
#A(X) = - X^5 + X^3 + X^2 - X + 1
plaintext BOB_MESSAGE

inv_59_mod:
$ (ORBITER) -v 2 -inverse_mod 59 10200

# the inverse of 59 mod 10200 is 2939

RSA_e:
$ (ORBITER) -v 2 \ -RSA 59 10403 2 "1921,1605,1804,2116,0518"

RSA_d:
$ (ORBITER) -v 2 \ -RSA 2939 10403 2 "902,3509,9833,3548,5181"

im1:
$ (ORBITER) -v 2 -inverse_mod 869 1843488

# the inverse of 869 mod 1843488 is 386093

# FUNFACTOR:
RSA_e1:
$ (ORBITER) -v 2 \ -RSA 386093 1846303 3 "62114,60103,201518"

RSA_d1:
$ (ORBITER) -v 2 \ -RSA 869 1846303 3 "1248407,345776,317846"

# 5503*4603 = 25330309
# 5502*4602 = 25320204
im1061:
- $(ORBITER) -v 2 \
- -inverse_mod 1061 25320204
# 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"

# 7253 * 8171 = 59264263
# 7252 * 8170 = 59248840

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
# 35270141, 9642524, 49091707

RSA_d3:
- $(ORBITER) -v 2 \
- -RSA 4358629 59264263 3 "35270141, 9642524, 49091707"
# 51403, 182516, 200504 = encrypted

# 7879 * 7901 = 62251979
# 7878 * 7900 = 62236200
9421 9422  # e = 9423 9424  im4: 9425  \\
9426 9427  # the inverse of 583 mod 62236200 is 32559247 9428 9429  RSA_e4: 9430  \\
9431 9432 9433  #-RSA_encrypt_text 583 62251979 venividivici 9434 9435 9436  RSA_d4: 9437  \\
9438 9439 9440 9441 9442  # 7369 * 7127 = 52518863 9443 9444 9445 9446 9447  im5: 9448  \\
9449 9450  # the inverse of 173 mod 52504368 is 38543669 9451  \\
9452  RSA_e5: 9453  \\
9454 9455 9456 9457 9458 9459 9460  RSA_d5: 9461  \\
9462 9463 9464 9465 9466 9467
smooth:

```bash
$ (ORBITER) -v 2 \n$ (ORBITER) -v 2 -sift_smooth 100000 100 "2,3,5,7,11,13,17,19"
```

```
# 1999 * 7907 = 15806093
# 1998 * 7906 = 15796188
```

```bash
im7:

```bash
$ (ORBITER) -v 2 -inverse_mod 3221 15796188
```

```
# the inverse of 3221 mod 15796188 is 10048553
```

```bash
im8:

```bash
$ (ORBITER) -v 2 \n$ (ORBITER) -v 2 -RSA_encrypt_text 10048553 15806093 3 beachandfun
```

```
# 7853 * 7673 = 60256069
# 7852 * 7672 = 60240544
```

```bash
RSA_e7:

```bash
$ (ORBITER) -v 2 \n$ (ORBITER) -v 2 -RSA_encrypt_text 9017 60240544 3 beachandfun
```

```
# the inverse of 9017 mod 60240544 is 14430473
```

```bash
RSA_e8:

```bash
$ (ORBITER) -v 2 \n$ (ORBITER) -v 2 -RSA_encrypt_text 9017 60256069 3 strawberry
```

```bash
sqrt_big:

```bash
$ (ORBITER) -v 2 -square_root 1002001
```

```bash
sqrt_mod_33.41:

```bash
$ (ORBITER) -v 2 -square_root_mod 33 41
```

```bash
quadratic_sieve:
```

```bash
$ (ORBITER) -v 5 -quadratic_sieve 31 500 1
```
pseudoprime3:
$\texttt{(ORBITER) -v 5 \n-seed 2531011 -find_pseudoprime 3 5 0 0}$
$\texttt{pdflatex pseudoprime_3.tex}$
$\texttt{open pseudoprime_3.pdf}$

pseudoprime10:
$\texttt{(ORBITER) -v 5 \n-seed 2531011 -find_pseudoprime 10 5 5 5}$
$\texttt{pdflatex pseudoprime_10.tex}$
$\texttt{open pseudoprime_10.pdf}$

# 4460190157

PR10_test1:
$\texttt{(ORBITER) -v 5 -power_mod 1293 2230095078 4460190157}$
$\texttt{(ORBITER) -v 5 -power_mod 9865 2230095078 4460190157}$
$\texttt{(ORBITER) -v 5 -power_mod 19645 2230095078 4460190157}$
$\texttt{(ORBITER) -v 5 -power_mod 974586571 2230095078 4460190157}$
$\texttt{(ORBITER) -v 5 -power_mod 974586571 1486730052 4460190157}$
$\texttt{(ORBITER) -v 5 -power_mod 974586571 15222492 4460190157}$
$\texttt{(ORBITER) -v 5 -power_mod 974586571 284796 4460190157}$

pseudoprime11:
$\texttt{(ORBITER) -v 5 \n-seed 2531011 -find_pseudoprime 11 5 5 5}$
$\texttt{pdflatex pseudoprime_11.tex}$
$\texttt{open pseudoprime_11.pdf}$

# 63814633367

# product is 284625399616057168619

pseudoprime20:
$\texttt{(ORBITER) -v 5 \n-seed 2531011 -find_pseudoprime 20 5 5 5}$
$\texttt{pdflatex pseudoprime_20.tex}$
$\texttt{open pseudoprime_20.pdf}$
PR10:
$\text{$(ORBITER) -v 5 -primitive\_root 4460190157}$

# mistake! long integer overflow
# a primitive root modulo 165222861 is 1293

pseudoprime50:
$\text{$(ORBITER) -v 5 }$
$\text{\$seed 2531011 -find\_pseudoprime 50 5 0 0}$
$\text{pdflatex pseudoprime\_50.tex}$
$\text{open pseudoprime\_50.pdf}$

#91322792878581218181431392170986926262336688354473

pseudoprime51:
$\text{$(ORBITER) -v 5 }$
$\text{\$seed 2531011 -find\_pseudoprime 51 5 5 5}$
$\text{pdflatex pseudoprime\_51.tex}$
$\text{open pseudoprime\_51.pdf}$

#754600727746834470214089702490004944659715367045417

pseudoprime30:
$\text{$(ORBITER) -v 5 }$
$\text{\$seed 2531011 -find\_pseudoprime 30 5 5 5}$
$\text{pdflatex pseudoprime\_30.tex}$
$\text{open pseudoprime\_30.pdf}$

# 286525565474504516914595596387

pseudoprime31:
$\text{$(ORBITER) -v 5 }$
$\text{\$seed 2531011 -find\_pseudoprime 31 5 5 5}$
$\text{pdflatex pseudoprime\_31.tex}$
$\text{open pseudoprime\_31.pdf}$

#8777266765422645523724129853331

# 2514911323283298698837184692002835573476743643265896783515097
9608  # maybe 2 seconds
9609
9610 pseudoprime33:
9611 $\triangleright$ $(\text{ORBITER}) -v 5$
9612 $\triangleright$ $\triangleright$ -seed 2531011 -find_pseudoprime 33 5 5 5
9613 $\triangleright$ pdflatex pseudoprime_33.tex
9614 $\triangleright$ open pseudoprime_33.pdf
9615
9616
9617 #371674199498295345543363004459891
9618
9619
9620 pseudoprime34:
9621 $\triangleright$ $(\text{ORBITER}) -v 5$
9622 $\triangleright$ $\triangleright$ -seed 2531011 -find_pseudoprime 34 5 5 5
9623 $\triangleright$ pdflatex pseudoprime_34.tex
9624 $\triangleright$ open pseudoprime_34.pdf
9625
9626 #9309708224110488378214945245346817
9627
9628 # 3460178351758962531912872979731874528849142238619677890786061016947
9629 # 18 sec
9630
9631
9632
9633 pseudoprime35:
9634 $\triangleright$ $(\text{ORBITER}) -v 5$
9635 $\triangleright$ $\triangleright$ -seed 2531011 -find_pseudoprime 35 5 5 5
9636 $\triangleright$ pdflatex pseudoprime_35.tex
9637 $\triangleright$ open pseudoprime_35.pdf
9638
9639 #81329557792505271120435930267680203
9640
9641 pseudoprime36:
9642 $\triangleright$ $(\text{ORBITER}) -v 5$
9643 $\triangleright$ $\triangleright$ -seed 2531011 -find_pseudoprime 36 5 5 5
9644 $\triangleright$ pdflatex pseudoprime_36.tex
9645 $\triangleright$ open pseudoprime_36.pdf
9646
9647 #16262468089199340433363207561599139
9648
9649 # 13226193383093105242537919350220354135219441323641636665484262532145217
9650 # factoring takes 46 seconds
9651
9652
9653 MATH360_hw2:
$(\text{ORBITER}) -v 3 \$

-define F -finite_field -q 16 -end \\
-with F -do -finite_field_activity \\
-parse_and_evaluate "test" "" "a+b" "a=8,b=14" -end

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

-define F -finite_field -q 16 -end \\
-with F -do -finite_field_activity \\
-parse_and_evaluate "test" "" "a*b" "a=9,b=13" -end

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

-define F -finite_field -q 16 -end \\
-with F -do -finite_field_activity \\
-parse_and_evaluate "test" "" "a*a*a*a*a" "a=9" -end

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

-define F -finite_field -q 16 -end \\
-with F -do -finite_field_activity \\
-parse_and_evaluate "test" "" "(a+b)*(a+b)" "a=5,b=7" -end

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

-define F -finite_field -q 16 -end \\
-with F -do -finite_field_activity \\
-parse_and_evaluate "test" "" "a*a+b*b" "a=5,b=7" -end

F_{256} \text{Rijndahl:}

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

-define F -finite_field -q 256 -override_polynomial 283 -end \\
-with F -do -finite_field_activity -cheat_sheet_GF -end

all_square_roots_mod_n_{1549411}:

$(\text{ORBITER}) -v 3 -all_square_roots_mod_n 1075922 1549411

power_mod_{211}:

$(\text{ORBITER}) -v 3 -power_mod_n 2 211

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

-plot_function power_mod_n_{a2,n211}.csv

$(\text{ORBITER}) -v 2 -draw_matrix \$

-input_csv_file power_mod_n_{a2,n211_graph}.csv \$

-box_width 10 -bit_depth 8 -partition 3 211 211 -end

power_mod_{2,31}:
# Chapter 10 - Coding Theory

### Section 10.1: Coding Theory

**Introduction:**

Hamming space 4 2 distance matrix:

```bash
$ (ORBITER) -v 2 -Hamming_space_distance_matrix 4 2
$ (ORBITER) -v 2 -draw_matrix
$ -input_csv_file Hamming_n4_q2.csv
$ -box_width 20 -bit_depth 24 -partition 4 16 16 -end
open Hamming_n4_q2.draw.bmp
```

Code 5 2 3 diagram:

```bash
$ (ORBITER) -v 2 -code_diagram "code.5_2_3" 
$ $ (CODE_5_2_3_CODEWORDS) 5 -metric_balls 1
$ (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:

-define G -graph -Hamming 5 2 -end

-with G -do

-graph_theoretic_activity -export_csv -end

-with G -do

-graph_theoretic_activity -export_graphviz -end

-with G -do

-graph_theoretic_activity -save -end

$(ORBITER) -v 2 -draw_matrix

-input_csv_file Hamming_5_2.csv

-box_width 8 -bit_depth 24 -partition 4 32 32 -end

dot -Tpng Hamming_5_2.gv >Hamming_5_2.png

Hamming_5_2_with_5_2_3_code:

$($ORBITER) -v 2

-define G -graph -Hamming 5 2

-subset ".\_code\_5\_2\_3\" "\\\_code\\\_5\\\_2\\\_3\" \n
$(CODE_5_2_3_CODEWORDS) -end

-with G -do

-graph_theoretic_activity -export_csv -end

-with G -do

-graph_theoretic_activity -export_graphviz -end

-with G -do

-graph_theoretic_activity -save -end

-with G -do

-graph_theoretic_activity -automorphism_group -end

pdflatex Hamming_5_2_code_5_2_3_report.tex

open Hamming_5_2_code_5_2_3_report.pdf

# group has order 32

code_6:

$($ORBITER) -v 2

-general_code_binary 6 "0,60,50,41,14,21,27,39"

$(ORBITER) -v 2 -draw_matrix

-input_csv_file code_matrix_8_8.csv

-box_width 20 -bit_depth 24

-partition 2 "1,1,1,1,1,1,1,1" "1,1,1,1,1,1,1,1" -end

pdflatex code_6_8.tex
Hamming generator:

```bash
$ (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  -with F -do \n  -finite_field_activity \n  -nullspace v \n  -end
```

Hamming code words:

```bash
$ (ORBITER) -v 2 \n  -define v -vector -dense $(HAMMING_CODE_ROWS_IN_BINARY_RANKS) -end \n  -linear_code_through_basis 7 v
```

```bash
pdflatex code_n7_k4_q2.tex
open code_n7_k4_q2.pdf
```

Hamming weight enumerator:
Hamming code diagram:

- $\$(ORBITER) \ -v 2 \ -define F -finite_field -q 2 -end \n- $\$(ORBITER) \ -v 2 \ -define v -vector -field F -format 4 \n- $\$(ORBITER) \ -v 2 \ -dense $\$(HAMMING_CODE_GENERATOR) \n- $\$(ORBITER) \ -v 2 \ -with F -do \n- $\$(ORBITER) \ -v 2 \ -finite_field_activity \n- $\$(ORBITER) \ -v 2 \ -weight Enumerator v -end

Hamming code systematic:

- $\$(ORBITER) \ -v 2 \ -define v -vector -dense $\$(HAMMING_CODE_ROWS_IN_BINARY_RANKS) -end \n
688
-linear_code_through_basis 7 v

$(ORBITER) -v 2 -draw_matrix \
-input_csv_file code_matrix_16_8.csv \\n-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_RREF:

$(ORBITER) -v 2 \
-define F -finite_field -q 2 -end \
-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:

$(ORBITER) -v 2 \
-define F2 -finite_field -q 2 -end \
-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

pdflatex nullspace_4_7.tex
open nullspace_4.7.pdf

#check equations of the Hamming code:
# a4+a5+a6+a7 = 1+0+1+0 = 0 mod 2 OK.
# a2+a3+a6+a7 = 0+1+1+0 = 0 mod 2 OK.
# a1+a3+a5+a7 = 1+1+0+0 = 0 mod 2 OK.

#1010101
#0110011
#0001111

Hamming_long:
$(ORBITER) -v 2 -long_code 7 4
"0,5,6"
"1,4,6"
"2,4,5"
"3,4,5,6"
$(ORBITER) -v 2 -loop L 0 16 1 -draw_matrix
"input_csv_file long_code_genma_n7_k4_codeword_0L.csv"
"box_width 25 -bit_depth 8 -partition 3 4 2 -end"
"end_loop"

#long_code_genma_n7_k4_codeword_0.csv
#long_code_genma_n7_k4_codeword_15.csv
#Weight distribution: ( 0, 3^7, 4^7, 7 )

Hamming_code_macwilliams:
$(ORBITER) -v 2
"make_macwilliams_system 7 4 2"
pdflatex MacWilliams_n7_k4_q2.tex
open MacWilliams_n7_k4_q2.pdf

Hamming_singer:
$(ORBITER) -v 3
"define G -linear_group -PGL 3 2 -singer 1 -end"
"-with G -do"
"-group_theoretic_activity"
"-report"
"-orbits_on_points"
"-end"
pdflatex PGL_3_2_Singer_3_2_1_report.tex
9984  ▶ open PGL_3_2_Singer_3_2_1_report.pdf
9985
9986  # cycle is 0,1,2,5,3,4,6
9987
9988  #1001110
9989  #0100111
9990  #0011101
9991
9992
9993
9994  Hamming_cyclic_generator:
9995  ▶ $(ORBITER) -v 2 \n9996  ▶ ▶ -define F -finite_field -q 2 -end \n9997  ▶ ▶ -define v -vector -format 3 -field F \n9998  ▶ ▶ ▶ -dense $(SIMPLEX_CODE_GENMA_CYCLIC) \n9999  ▶ ▶ -end \n10000 ▶ ▶ -with F -do -finite_field_activity \n10001 ▶ ▶ -nullspace v \n10002 ▶ ▶ -end
10003 ▶ pdflatex nullspace_3_7.tex
10004 ▶ open nullspace_3_7.pdf
10005
10006 Hamming_cyclic_long:
10007  ▶ $(ORBITER) -v 2 -long_code 7 4 \n10008  ▶ ▶ "0,4,6" \n10009  ▶ ▶ "1,4,5,6" \n10010  ▶ ▶ "2,4,5" \n10011  ▶ ▶ "3,5,6"
10012  ▶ $(ORBITER) -v 2 -loop L 0 16 1 -draw_matrix \n10013  ▶ ▶ -input_csv_file long_code_genma_n7_k4_codeword_%L.csv \n10014  ▶ ▶ -box_width 25 -bit_depth 8 -partition 3 4 2 -end \n10015  ▶ ▶ -end_loop
10016
10017
10018
10019 Hamming_cyclic:
10020  ▶ $(ORBITER) -v 2 \n10021  ▶ ▶ -define v -vector -dense "69,39,22,11" -end \n10022  ▶ ▶ -linear_code_through_basis 7 v
10023  ▶ ▶ $(ORBITER) -v 2 -draw_matrix \n10024  ▶ ▶ -input_csv_file code_matrix_16_8.csv \n10025  ▶ ▶ -box_width 25 -bit_depth 8 -partition 2 16 8 -end
10026  ▶ open code_matrix_16_8.draw.bmp
10027  ▶ pdflatex code_7_16.tex
10028  ▶ open code_7_16.pdf
10029
10030
Hamming cyclic clean:

```bash
$(ORBITER) -v 2
```

```bash
define F -finite_field -q 2 -end
```

```bash
define v -vector -format 3 -field F
```

```bash
dense $(SIMPLEX_CODE_GENMA_CYCLIC)
```

```bash
-end
```

```bash
-with F -do -finite_field_activity
```

```bash
-nullspace v
```

```bash
-normalize_from_the_right
```

```bash
-end
```

```bash
nullspace
```

```bash
open
```

Hamming cyclic clean:

```bash
$(ORBITER) -v 2
```

```bash
define v -vector -dense "88,44,22,11" -end
```

```bash
linear_code_through_basis 7 v
```

```bash
$(ORBITER) -v 2 -draw_matrix
```

```bash
-input_csv_file code_matrix_16_8.csv
```

```bash
box_width 25 -bit_depth 8 -partition 2 16 8 -end
```

```bash
open code_matrix_16_8.draw.bmp
```

```bash
pdflatex code_7_16.tex
```

```bash
open code_7_16.pdf
```

Hamming cyclic clean_long:

```bash
$(ORBITER) -v 2 -long_code 7 4
```

```bash
"0,2,3"
```

```bash
"1,3,4"
```

```bash
"2,4,5"
```

```bash
"3,5,6"
```

```bash
$(ORBITER) -v 2
```

```bash
-loop L 0 16 1 -draw_matrix
```

```bash
-input_csv_file
```

```bash
long_code_genma_n7_k4_codeword_%L.csv
```

```bash
box_width 25 -bit_depth 8
```

```bash
partition 3 4 2 -end
```

```bash
-end_loop
```

#11111111 = 255

#01010101 = 170

#00110011 = 204
10078 #00001111 = 240
10079
10080
10081
10082 # Section 10.3: Coding Theory - Golay codes
10083
10084 SECTION CODING THEORY GOLAY CODES:
10085
10086
10087
10088 Golay23 code words:
10089 ▶ $(ORBITER) -v 2 \n10090 ▶ ▶ -define v -vector -dense $(GOLAY_23_COLUMN_RANKS_PROJECTIVELY) -end \n10091 ▶ ▶ -linear code through columns of parity check projectively 12 v
10092 ▶ pdflatex code_n23_k12_q2.tex
10093 ▶ open code_n23_k12_q2.pdf
10094
10095
10096
10097 Golay23 code diagram:
10098 ▶ $(ORBITER) -v 2 \n10099 ▶ ▶ -code_diagram_from_file "Golay_23" \n10100 ▶ ▶ codewords_n23_k12_q2.csv 23 -enhance 4
10101 ▶ ▶
10102 ▶ ▶ #-metric_balls 3
10103
10104
10105 Golay23 code diagram draw:
10106 ▶ $(ORBITER) -v 2 \n10107 ▶ ▶ -draw_matrix \n10108 ▶ ▶ ▶ -input_csv_file Golay_23_diagram_01_23_4096.csv \n10109 ▶ ▶ ▶ -box_width 4 -bit_depth 8 \n10110 ▶ ▶ ▶ -partition 20 4096 2048 \n10111 ▶ ▶ -end
10112
10113
10114
10115
10116
10117
10118 # Section 10.4: Coding Theory - CRC codes
10119
10120 SECTION CODING THEORY CRC CODES:
10121
10122
10123
10124
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,\n1,5,1,4,1,2,1,1,1,0"

CRC32CASTAGNOLI="1,32,1,28,1,27,1,26,1,25,1,23,1,22,1,20,1,19,1,\n1,18,1,14,1,13,1,11,1,10,1,9,1,8,1,6,1,0"

CRC64ECMA182="1,64,1,62,1,57,1,55,1,54,1,53,1,52,1,47,1,46,1,45,\n1,40,1,39,1,38,1,37,1,36,1,33,1,32,1,31,1,29,1,27,1,24,1,23,1,22,\n1,21,1,19,1,17,1,13,1,12,1,10,1,9,1,7,1,4,1,1,1,0"

CRC64ROCKSOFT="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,\n1,19,1,16,1,13,1,12,1,10,1,9,1,6,1,4,1,3,1,0"

encode text 5bits:

encode_text_5bits:

encode_text_5bits_check:

encode_text_5bits_1error:
10172 ▷ $(ORBITER) -encode_text_5bits \\
10173 ▷ "Hithere" "text.csv"
10174 ▷ $(ORBITER) -v 2 \\
10175 ▷ -define F -finite_field -q 2 -end \\
10176 ▷ -with F -do \\
10177 ▷ -finite_field_activity \\
10178 ▷ -polynomial_division_from_file_all_k_bit_error_patterns \\
10179 ▷ text.csv 13 1 -end
10180 ▷ pdflatex polynomial_division_file_all_1_error_patterns_13.tex
10181 ▷ open polynomial_division_file_all_1_error_patterns_13.pdf
10182
10183
10184
10185 CRC_3_128_10:
10186 ▷ $(ORBITER) -v 1 \\
10187 ▷ -define F -finite_field -q 2 -end \\
10188 ▷ -with F -do -finite_field_activity \\
10189 ▷ -find_CRC_polynomials 3 128 10 \\
10190 ▷ -end
10191
10192
10193
10194
10195 crc32_test:
10196 ▷ $(ORBITER) -v 3 \\
10197 ▷ -crc32 "123456789" 
10198
10199 crc32_test_hexdata:
10200 ▷ $(ORBITER) -v 3 \\
10201 ▷ -crc32_hexdata "7BD11C4010" 
10202
10203 crc32_Berlekamp_matrix:
10204 ▷ $(ORBITER) -v 2 \\
10205 ▷ -define F -finite_field -q 2 -end \\
10206 ▷ -define v -vector -field F -sparse 33 $(CRC32_Ethernet) -end \\
10207 ▷ -with F -do \\
10208 ▷ -finite_field_activity \\
10209 ▷ -Berlekamp_matrix v -end
10210
10211
10212 CRC_F256_roots_771:
10213 ▷ $(ORBITER) -v 3 \\
10214 ▷ -define F -finite_field -q 256 -end \\
10215 ▷ -with F -do -finite_field_activity \\
10216 ▷ -nth_roots 771 \\
10217 ▷ -end
10218
\texttt{\textbackslash pdflatex \textit{BCH} codes}\texttt{\textbackslash q256\textbackslash n771\textbackslash d2.tex}

\texttt{open BCH_codes\textunderscore q256\textunderscore n771\textunderscore d2.pdf}

\texttt{\textbackslash g++ \textit{check\textunderscore q256\textunderscore n771\textunderscore r2.cpp} -o \textit{check\textunderscore q256\textunderscore n771\textunderscore r2.out}}

\texttt{./check\textunderscore q256\textunderscore n771\textunderscore r2.out}

\texttt{\textit{F256\_BCH\_code\_d16:}}

\texttt{\textbackslash pdflatex \textit{BCH\_codes}\textunderscore q256\textunderscore n771\textunderscore d16.tex}

\texttt{open BCH\_codes\textunderscore q256\textunderscore n771\textunderscore d16.pdf}

\texttt{#generator polynomial is \texttt{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,
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,\n150,24,169,192,212,112,144,97,109,174,253,1"

F256
BCH
write
code
for
division

$(ORBITER) -v 2 \
(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 \nwrite_code_for_division check_q256_n771_r30.cpp A B -end

$g++ check_q256_n771_r30.cpp -o check_q256_n771_r30.out

./check_q256_n771_r30.out

F256
BCH
code
d16
division:

$(ORBITER) -v 2 \
(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 \npolynomial_division A B -end

F256
BCH
code
d16
error:

$(ORBITER) -v 2 \
(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 \npolynomial_division A B -end

F256
BCH
code
d16
error:
# Section 10.5: Coding Theory - Reed-Muller codes

SECTION CODING THEORY REED MULLER CODES:

RM\_3\_1 code words:

```
$\text{ORBITER} -v 2 \$

$\text{ORBITER} -v 2 -vector -dense \text{(REED_MULLER_3_1_BASIS_IN_BINARY)} -end \$

$\text{ORBITER} -v 2 -linear\text{code\_through\_basis} 8 v$
```

```
pdflatex code\_n8\_k4\_q2.tex
```

```
open code\_n8\_k4\_q2.pdf
```

```
#Codewords: (0,255,170,85,204,51,102,153,240,15,90,165,60,195,150,105)
```

```
RM\_3\_1 Hamming space diagram:
```

```
$\text{ORBITER} -v 2 -code\_diagram \text{"RM\_3\_1"}$
```

```
$\text{ORBITER} -v 2 -metric\_balls 1$
```

```
$\text{ORBITER} -v 2 -draw\_matrix$
```

```
$\text{ORBITER} -v 2 -draw\_matrix$
```

```
$\text{ORBITER} -v 2 -box\_width 25 -bit\_depth 8 -partition 4 16 16 -end$
```

```
$\text{ORBITER} -v 2 -box\_width 25 -bit\_depth 8 -partition 4 16 16 -end$
```

```
open \text{RM\_3\_1\_diagram\_8\_16\_draw.bmp}$
```

```
RM\_3\_1 split:
```

```
$\text{ORBITER} -split\_by\_values \text{RM\_3\_1\_holes\_8\_16.csv}$
```

```
RM\_3\_1 holes draw:
```

```
$\text{ORBITER} -v 2$
```

```
$\text{ORBITER} -loop L 0 3 1$
```

```
$\text{ORBITER} -draw\_matrix$
```

```
$\text{ORBITER} -input\_csv\_file \text{RM\_3\_1\_holes\_8\_16\_value\_L.csv}$
```

```
$\text{ORBITER} -box\_width 25 -bit\_depth 8 -partition 5 16 16 -end$
```

```
$\text{ORBITER} -end\_loop$
```

698
\[ E_0 = E_1 + E_2 + E_3 + E_4 \]
\[8^4 + X_1X_2X_4X_5X_8^4 + X_1X_2X_4X_6X_8^4 + X_1X_2X_5X_7X_8^4 + X_1X_2X_6X_7X_8^4 + \]
\[X_1X_3X_4X_5X_8^4 + X_1X_3X_4X_7X_8^4 + X_1X_3X_5X_6X_8^4 + X_1X_3X_6X_7X_8^4 + X_1X_4X_5X_6X_8^4 + \]
\[X_1X_4X_5X_7X_8^4 + X_1X_4X_6X_7X_8^4 + X_2X_3X_4X_5X_8^4 + X_2X_3X_4X_7X_8^4 + X_2X_3X_5X_6X_8^4 + \]
\[X_2X_3X_5X_7X_8^4 + X_2X_4X_5X_6X_8^4 + X_2X_4X_5X_7X_8^4 + X_2X_4X_6X_7X_8^4 + X_2X_5X_6X_7X_8^4 + \]
\[X_3X_4X_5X_6X_8^4 + X_3X_4X_5X_7X_8^4 + X_3X_4X_6X_7X_8^4 + X_3X_5X_6X_7X_8^4 + X_3X_5X_6X_7X_8^4 + \]
\[\text{10380} \]
\[\text{10381} \] # E_2 + E_3 + E_4
\[\text{10382} \]
\[\text{10383} \]
\[\text{10384} \text{RM}_4\!\!1:\]
\[\text{10385} \>] \$(\text{ORBITER}) -v 2 \]
\[\text{10386} \>] \-linear\_code\_through\_columns\_of\_parity\_check 5 \]
\[\text{10387} \>] \$(\text{REED}\_\text{MULLER}\_4\!\!1.\text{COLUMNS}\_OF\_\text{PARTITY}\_\text{CHECK})\]
\[\text{10388} \] pdf\_latex\_code\_n16\_k5\_q2.tex
\[\text{10389} \] open code\_n16\_k5\_q2.pdf
\[\text{10390} \]
\[\text{10391} \]
\[\text{10392} \] # codewords\_n16\_k5\_q2.csv
\[\text{10393} \]
\[\text{10394} \]
\[\text{10395} \]
\[\text{10396} \text{RM}_4\!\!1\_\text{diagram}:\]
\[\text{10397} \>] \$(\text{ORBITER}) -v 2 \]
\[\text{10398} \>] \-code\_diagram\_from\_file "\text{RM}_4\!\!1" \]
\[\text{10399} \>] \codewords\_n16\_k5\_q2.csv 16
\[\text{10400} \>] \#-enhance 4
\[\text{10401} \>] \#-metric\_balls 3
\[\text{10402} \]
\[\text{10403} \text{RM}_4\!\!1\_\text{diagram}\_\text{draw}:\]
\[\text{10404} \>] \$(\text{ORBITER}) -v 2 -\text{draw\_matrix} \]
\[\text{10405} \>] \-\text{input\_csv\_file} \text{RM}_4\!\!1\_\text{diagram}_01\_16\_32.csv \]
\[\text{10406} \>] \-box\_width 25 -\text{bit\_depth} 8 -\text{partition} 10 256 256 -\text{end}\]
\[\text{10407} \] open RM\_4\!\!1\_diagram\_01\_16\_32\_draw.bmp
\[\text{10408} \]
\[\text{10409} \]
\[\text{10410} \text{RM}_4\!\!1\_\text{split}:\]
\[\text{10411} \>] \$(\text{ORBITER}) -\text{split\_by\_values} \text{RM}_4\!\!1\_\text{holes\_16\_32.csv}\]
\[\text{10412} \]
\[\text{10413} \text{RM}_4\!\!1\_\text{diagram}\_\text{draw}\_\text{holes}:\]
\[\text{10414} \>] \$(\text{ORBITER}) -v 2 -\text{draw\_matrix} \]
\[\text{10415} \>] \-\text{input\_csv\_file} \text{RM}_4\!\!1\_\text{holes\_16\_32.csv} \]
\[\text{10416} \>] \-box\_width 25 -\text{bit\_depth} 8 -\text{partition} 10 256 256 -\text{end}\]
\[\text{10417} \] $(\text{ORBITER}) -v 2 -\text{loop} L 0 7 1 -\text{draw\_matrix} \]
\[\text{10418} \>] \-\text{input\_csv\_file} \text{RM}_4\!\!1\_\text{holes\_16\_32\_value\%L.csv} \]
\[\text{10419} \>] \-box\_width 25 -\text{bit\_depth} 8 -\text{partition} 10 256 256 -\text{end} -\text{end\_loop}\]
\[\text{10420} \]
RM_4.1_diagram_metric_balls:
$ $(ORBITER) -v 2 -code_diagram_from_file "RM_4.1" \ 
  codewords_n16_k5_q2.csv 16 -metric_balls 3 
$ $(ORBITER) -v 2 -draw_matrix \ 
-input_csv_file RM_4.1_diagram_16_32.csv \ 
-box_width 25 -bit_depth 8 -partition 10 256 256 -end 

RM_4.1_hole0:
$ $(ORBITER) -v 3 -define F -finite_field -q 2 -end \ 
  -with F -do -finite_field.activity \ 
  -algebraic_normal_form RM_4.1_holes_16_32_value0.csv 16 -end 

Reed_Muller_6:
$ $(ORBITER) -v 2 -long_code 64 7 \ 
  -set_builder -loop 0 64 1 -end \ 
  -set_builder -loop 0 32 1 -affine_function 2 1 -end \ 
  -set_builder -loop 0 16 1 -affine_function 4 2 -clone_with_affine_function 4 
    3 -end \ 
  -set_builder -set_builder -set_builder -loop 0 4 1 -affine_function 1 4 \ 
  -clone_with_affine_function 1 12 -end -clone_with_affine_function 1 16 \ 
  -end -clone_with_affine_function 1 32 -end\ 
  -set_builder -set_builder -loop 0 8 1 -affine_function 1 8 \ 
  -clone_with_affine_function 1 24 -end -clone_with_affine_function 1 32 -end 
  \ 
  -set_builder -loop 0 16 1 -affine_function 1 16 -clone_with_affine_function 1 
    48 -end \ 
  -set_builder -loop 0 32 1 -affine_function 1 32 -end \ 
$ $(ORBITER) -v 2 -draw_matrix \ 
-input_csv_file long_code_genma_n64_k7.csv \ 
-box_width 25 -bit_depth 8 -partition 3 7 64 -end 
open long_code_genma_n64_k7.draw.bmp
10465 \> $(ORBITER) -v 2 -long_code 64 7 \\
10466 \> $\{(RM_6}_{\text{GENERATOR}}_{1}\) \\
10467 \> $\{(RM_6}_{\text{GENERATOR}}_{2}\) \\
10468 \> $\{(RM_6}_{\text{GENERATOR}}_{3}\) \\
10469 \> $\{(RM_6}_{\text{GENERATOR}}_{4}\) \\
10470 \> $\{(RM_6}_{\text{GENERATOR}}_{5}\) \\
10471 \> $\{(RM_6}_{\text{GENERATOR}}_{6}\) \\
10472 \> $\{(RM_6}_{\text{GENERATOR}}_{7}\) \\
10473 \> $(ORBITER) -v 2 -draw_matrix \\
10474 \> -input_csv_file long_code_genma_n64_k7.csv \\
10475 \> -box_width 25 -bit_depth 8 -partition 3 7 64 -end \\
10476 \> $(ORBITER) -v 2 -draw_matrix \\
10477 \> -input_csv_file long_code_genma_n64_k7_codeword_0.csv \\
10478 \> -box_width 25 -bit_depth 8 -partition 3 8 8 -end \\
10479 \> $(ORBITER) -v 2 -draw_matrix \\
10480 \> -input_csv_file long_code_genma_n64_k7_codeword_1.csv \\
10481 \> -box_width 25 -bit_depth 8 -partition 3 8 8 -end \\
10482 \> $(ORBITER) -v 2 -draw_matrix \\
10483 \> -input_csv_file long_code_genma_n64_k7_codeword_2.csv \\
10484 \> -box_width 25 -bit_depth 8 -partition 3 8 8 -end \\
10485 \\
10486 \\
10487 \\
10488 \\
10489 RM6words:
10490 \> - mkdir RM6 \\
10491 \> #$(ORBITER) -v 2 -draw_matrix -input_csv_file long_code_genma_n64_k7_codeword_0 .csv -box_width 25 -bit_depth 8 -partition 4 8 8 -end \\
10492 \> $(ORBITER) -v 2 -loop L 0 128 1 \\
10493 \> -draw_matrix -input_csv_file \\
10494 \> -box_width 25 -bit_depth 8 -partition 4 8 8 -end \\
10495 \> -mv long_code_genma_n64_k7_codeword_%L.csv \\
10496 \> -frame 8 RM6/RM_6_1_codeword_%L.png \\
10497 \> -end_loop \\
10498 \\
10500 \\
10501 \\
10502 RM6_convert: \\
10503 \> - mkdir RM6_PNG \\
10504 \> convert RM6/RM_6_1_codeword_0.png -frame 8 RM6_PNG/000.png \\
10505 \> convert RM6/RM_6_1_codeword_1.png -frame 8 RM6_PNG/001.png \\
10506 \> convert RM6/RM_6_1_codeword_2.png -frame 8 RM6_PNG/002.png \\
10507 \> convert RM6/RM_6_1_codeword_3.png -frame 8 RM6_PNG/003.png \\
10508 \> convert RM6/RM_6_1_codeword_4.png -frame 8 RM6_PNG/004.png \\
10509 \> convert RM6/RM_6_1_codeword_5.png -frame 8 RM6_PNG/005.png \\
10510 \> convert RM6/RM_6_1_codeword_6.png -frame 8 RM6_PNG/006.png
convert RM6/RM_6_1_codeword_101.bmp -frame 8 RM6_PNG/101.png
convert RM6/RM_6_1_codeword_102.bmp -frame 8 RM6_PNG/102.png
convert RM6/RM_6_1_codeword_103.bmp -frame 8 RM6_PNG/103.png
convert RM6/RM_6_1_codeword_104.bmp -frame 8 RM6_PNG/104.png
convert RM6/RM_6_1_codeword_105.bmp -frame 8 RM6_PNG/105.png
convert RM6/RM_6_1_codeword_106.bmp -frame 8 RM6_PNG/106.png
convert RM6/RM_6_1_codeword_107.bmp -frame 8 RM6_PNG/107.png
convert RM6/RM_6_1_codeword_108.bmp -frame 8 RM6_PNG/108.png
convert RM6/RM_6_1_codeword_109.bmp -frame 8 RM6_PNG/109.png
convert RM6/RM_6_1_codeword_110.bmp -frame 8 RM6_PNG/110.png
convert RM6/RM_6_1_codeword_111.bmp -frame 8 RM6_PNG/111.png
convert RM6/RM_6_1_codeword_112.bmp -frame 8 RM6_PNG/112.png
convert RM6/RM_6_1_codeword_113.bmp -frame 8 RM6_PNG/113.png
convert RM6/RM_6_1_codeword_114.bmp -frame 8 RM6_PNG/114.png
convert RM6/RM_6_1_codeword_115.bmp -frame 8 RM6_PNG/115.png
convert RM6/RM_6_1_codeword_116.bmp -frame 8 RM6_PNG/116.png
convert RM6/RM_6_1_codeword_117.bmp -frame 8 RM6_PNG/117.png
convert RM6/RM_6_1_codeword_118.bmp -frame 8 RM6_PNG/118.png
convert RM6/RM_6_1_codeword_119.bmp -frame 8 RM6_PNG/119.png
convert RM6/RM_6_1_codeword_120.bmp -frame 8 RM6_PNG/120.png
convert RM6/RM_6_1_codeword_121.bmp -frame 8 RM6_PNG/121.png
convert RM6/RM_6_1_codeword_122.bmp -frame 8 RM6_PNG/122.png
convert RM6/RM_6_1_codeword_123.bmp -frame 8 RM6_PNG/123.png
convert RM6/RM_6_1_codeword_124.bmp -frame 8 RM6_PNG/124.png
convert RM6/RM_6_1_codeword_125.bmp -frame 8 RM6_PNG/125.png
convert RM6/RM_6_1_codeword_126.bmp -frame 8 RM6_PNG/126.png
convert RM6/RM_6_1_codeword_127.bmp -frame 8 RM6_PNG/127.png

to a0

to a1

to a2

to a3

to a4

to a5
# Section 10.6: Coding Theory - BCH Codes

## F_8 BCH Code d_3:

```bash
$(ORBITER) -v 3
-define F -finite_field -q 8 -override_polynomial 11 -end
```

## 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
```
#generator polynomial is $X^4 + 4X^3 + 4X^2 + 3X + 4$

F_8 BCH code d4:

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

-define F -finite_field -q 8 -override_polynomial 11 -end

-with F -do -finite_field_activity -make_BCH_code 21 4 -end

#generator polynomial is $X^5 + 6X^4 + 7X^3 + 2X + 3$

F_8 BCH code d5:

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

-define F -finite_field -q 8 -override_polynomial 11 -end

-with F -do -finite_field_activity -make_BCH_code 21 5 -end

#generator polynomial is $X^7 + 3X^6 + 3X^5 + 2X^4 + X^3 + 2X^2 + X + 2$

F_8 BCH code d6:

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

-define F -finite_field -q 8 -override_polynomial 11 -end

-with F -do -finite_field_activity -make_BCH_code 21 6 -end

#generator polynomial is $X^9 + 5X^8 + 5X^6 + 4X^3 + 5X + 4$

F_8 BCH code d7:

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

-define F -finite_field -q 8 -override_polynomial 11 -end

-with F -do -finite_field_activity -make_BCH_code 21 7 -end

#generator polynomial is $X^{10} + X^9 + 2X^8 + 5X^7 + 2X^6 + 4X^4 + 6X^3 + 5X^2 + 6X + 6$

F_8 BCH code d8:

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

-define F -finite_field -q 8 -override_polynomial 11 -end

-with F -do -finite_field_activity -make_BCH_code 21 8 -end
#generator polynomial is $X^{12} + 2X^{10} + 6X^{9} + 5X^{8} + 7X^{7} + 6X^{6} + 2 X^{5} + 7X^{4} + 5X^{3} + 5X^{2} + 6$

# after reduction: [63,27] $r=36$

F2_BCH_code_n21:

10728 $(ORBITER) -v 3 \$
10729 $define F -finite_field -q 2 -end \$
10730 $with F -do -finite_field_activity \$
10731 $make_BCH_code 21 3 \$
10732 $end$
10733
10734
10735 F7_RS_code_n6:

10736 $(ORBITER) -v 30 \$
10737 $define F -finite_field -q 7 -end \$
10738 $with F -do -finite_field_activity \$
10739 $make_BCH_code 6 3 \$
10740 $end$
10741
10742
10743 F_64_again:

10744 $(ORBITER) -v 3 \$
10745 $define F -finite_field -q 64 -end \$
10746 $with F -do -finite_field_activity \$
10747 $cheat_sheet_GF \$
10748 $end$
10749 $pdflatex GF_64.tex$
10750 $open GF_64.pdf$
10751
10752
10753 BCH_255_5_evaluate_elementary_symmetric_functions_1:

10754 $(ORBITER) -v 3 -define F -finite_field -q 256 -end \$
10755 $define e1 -formula "e1" "e1" "$(ELEMENTARY_Symmetrical_S.1) \$
10756 $define e2 -formula "e2" "e2" "$(ELEMENTARY_Symmetrical_S.2) \$
10757 $define e3 -formula "e3" "e3" "$(ELEMENTARY_Symmetrical_S.3) \$
10758 $define e4 -formula "e4" "e4" "$(ELEMENTARY_Symmetrical_S.4) \$
10759 $define e5 -formula "e5" "e5" "$(ELEMENTARY_Symmetrical_S.5) \$
10760 $define e6 -formula "e6" "e6" "$(ELEMENTARY_Symmetrical_S.6) \$
10761 $define e7 -formula "e7" "e7" "$(ELEMENTARY_Symmetrical_S.7) \$
10762 $define e8 -formula "e8" "e8" "$(ELEMENTARY_Symmetrical_S.8) \$
10763 $define E8 -collection "e1,e2,e3,e4,e5,e6,e7,e8" \$
10764 $with F -do -finite_field_activity \$
10765 $evaluate E8 "x0=2,x1=4,x2=16,x3=29,x4=76,x5=157,x6=95,x7=133" -end
evaluate elementary symmetric functions 2:

```plaintext
-define e1 -formula "e1" "e1" "" $(ELEMENTARY_SYMMETRIC_8_1) \\
-define e2 -formula "e2" "e2" "" $(ELEMENTARY_SYMMETRIC_8_2) \\
-define e3 -formula "e3" "e3" "" $(ELEMENTARY_SYMMETRIC_8_3) \\
-define e4 -formula "e4" "e4" "" $(ELEMENTARY_SYMMETRIC_8_4) \\
-define e5 -formula "e5" "e5" "" $(ELEMENTARY_SYMMETRIC_8_5) \\
-define e6 -formula "e6" "e6" "" $(ELEMENTARY_SYMMETRIC_8_6) \\
-define e7 -formula "e7" "e7" "" $(ELEMENTARY_SYMMETRIC_8_7) \\
-define e8 -formula "e8" "e8" "" $(ELEMENTARY_SYMMETRIC_8_8) \\
-define E8 -collection "e1,e2,e3,e4,e5,e6,e7,e8" \\
-with F -do -finite_field -q 256 -end \\
-evaluate E8 "x0=8,x1=64,x2=205,x3=143,x4=70,x5=217,x6=130,x7=23" -end
```

BCH15:

```plaintext
-BCH 15 2 3
-BCH 15 2 5
-BCH 15 2 7
-BCH 15 2 9
```

BCH11:

```plaintext
-BCH 11 2 3
-BCH 11 2 5
```

BCH13:

```plaintext
-BCH 13 2 3
-BCH 13 2 5
```

BCH7:

```plaintext
-BCH 7 2 3
-BCH 7 2 5
```

BCH21:

```plaintext
-BCH 21 2 3
-BCH 21 2 5
-BCH 21 2 7
-BCH_dual 21 2 7
```

BCH93:

```plaintext
-BCH 93 2 3
```

709
BCH255:
$\text{(ORBITER)} \ -BCH \ 255 \ 2 \ 4$
$\text{(ORBITER)} \ -v \ 2 \ -\text{draw_matrix} \ \$
$\text{input_csv_file} \ BCH_{255}.\text{csv} \ \$
$\text{box_width} \ 40 \ -\text{bit_depth} \ 24 \ \$
$\text{partition} \ 10 \ "239" \ "255" \ -\text{end}$

#BCH_{255}.\text{csv}

BCH273:
$\text{(ORBITER)} \ -BCH \ 273 \ 2 \ 4$
$\text{(ORBITER)} \ -v \ 2 \ \$
$\text{draw_mod} \ 31:$$
$\text{(ORBITER)} \ -v \ 2 \ \$
$\text{-draw_options \ -embedded \ -end} \ \$
$\text{-draw_mod_n \ 31 \ mod.31 \ -draw_mod_n.power_cycle \ 2}$
$\text{pdflatex mod.31_draw.tex}$
$\text{open mod.31_draw.pdf}$

PR127:
$\text{(ORBITER)} \ -v \ 5 \ -\text{primitive_root} \ 127$
$\text{(ORBITER)} \ -v \ 2 \ \$
$\text{draw_mod.127_power:}$
$\text{(ORBITER)} \ -v \ 2 \ \$
$\text{-draw_options \ -scale \ 0.4 \ -embedded \ -end} \ \$
$\text{-draw_mod_n \ 127 \ mod.127 \ -draw_mod_n.power_cycle \ 3}$
$\text{pdflatex mod.127_draw.tex}$
$\text{open mod.127_draw.pdf}$

draw_mod.251:
$\text{(ORBITER)} \ -v \ 2 \ \$
$\text{-draw_options \ -nodes_empty \ -radius \ 10 \ -embedded \ -end} \ \$
$\text{-draw_mod_n \ 251 \ mod.251}$
$\text{pdflatex mod.251_draw.tex}$
$\text{open mod.251_draw.pdf}$

#-\text{draw_mod_n_inverse}$
draw_mod_255_cyclotomic_1:
  $(ORBITER) -v 2 \ 
  -draw_options -nodes_empty -radius 10 \ 
  -line_width 0.4 -embedded -end \ 
  -draw_mod_n -n 255 -file mod_255_cyclotomic_1 \ 
  -cyclotomic_sets 2 "1" -end
  pdflatex mod_255_cyclotomic_1_draw.tex
  open mod_255_cyclotomic_1_draw.pdf

draw_mod_255_cyclotomic_3:
  $(ORBITER) -v 2 \ 
  -draw_options -nodes_empty -radius 10 \ 
  -line_width 0.4 -embedded -end \ 
  -draw_mod_n -n 255 -file mod_255_cyclotomic_3 \ 
  -cyclotomic_sets 2 "3" -end
  pdflatex mod_255_cyclotomic_3_draw.tex
  open mod_255_cyclotomic_3_draw.pdf

draw_mod_255_cyclotomic_1_and_3:
  $(ORBITER) -v 2 \ 
  -draw_options -nodes_empty -radius 10 \ 
  -line_width 0.4 -embedded -end \ 
  -draw_mod_n -n 255 -file mod_255_cyclotomic_1_and_3 \ 
  -cyclotomic_sets 2 "1,3" -end
  pdflatex mod_255_cyclotomic_1_and_3_draw.tex
  open mod_255_cyclotomic_1_and_3_draw.pdf

draw_mod_63_4_cyclotomic_3_6:
  $(ORBITER) -v 2 \ 
  -draw_options -radius 20 \ 
  -line_width 0.1 -embedded -end \ 
  -draw_mod_n -n 63 -file mod_63_4_cyclotomic_3_6 \ 
  -cyclotomic_sets 4 "3,6" \ 
  -cyclotomic_sets_thickness 50 \ 
  -end
  pdflatex mod_63_4_cyclotomic_3_6_draw.tex
  open mod_63_4_cyclotomic_3_6_draw.pdf

BCH_F_64:
  $(ORBITER) -v 3 \ 
  -define F -finite_field -q 64 -end \ 
  -with F -do -finite_field_activity -cheat_sheet_GF -end
  pdflatex GF_64.tex

BCH_elementary_symmetric_functions_3:
  $(ORBITER) -make_elementary_symmetric_functions 3 3
BCH_63_4.evaluate_elementary_symmetric_functions_1:

```
$(ORBITER) -v 3 -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=8,x1=62,x2=15" -end
```

#The values of the formulae are:

```
#0 : 57
#1 : 0
#2 : 1
```

# poly: 1,0,2,1

BCH_63_4.evaluate_elementary_symmetric_functions_2:

```
$(ORBITER) -v 3 -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:

```
$(ORBITER) -v 2 \ 
(poly_mult_mod "1,0,2,1" "1,0,3,1" \ 
"1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1" \ 
-end
```
#C(X)=X^6 + X^5 + X^4 + X^2 + 1

poly 1,0,1,0,1,1,1

BCH
poly division a:
#ORBITER
-v 2
-def 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
poly division b:
#ORBITER
-v 2
-def 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
poly division ab:
#ORBITER
-v 2
-def 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
generator matrix:
#ORBITER
-v 2
-def F -finite_field -q 4 -end
-with F -do
-finite_field_activity
-generator_matrix_cyclic_code 21 "1,0,1,0,1,1,1" -end

BCH
15 weight enumerator:
#ORBITER
-v 2
-def F -finite_field -q 4 -end
\begin{verbatim}
11001  >  -define v -vector -field F -format 15 \\
11002  >  -dense $(BCH_{21\_15\_GENERATOR\_MATRIX}) \\
11003  >  -end \ \\
11004  >  -with F -do \ \\
11005  >  -finite_field_activity -weight_enumerator v -end \\
11006  \\
11007  # too slow! \\
11008  \\
11009  BCH_{21\_15\_dual}: \\
11010  >  $(ORBITER) -v 2 \ \\
11011  >  -define F -finite_field -q 4 -end \ \\
11012  >  -define v -vector -field F -format 15 \ \\
11013  >  -dense $(BCH_{21\_15\_GENERATOR\_MATRIX}) -end \ \\
11014  >  -with F -do -finite_field_activity \ \\
11015  >  -nullspace v \ \\
11016  >  -normalize_from_the_right \ \\
11017  >  -end \\
11018  \\
11019  \\
11020  BCH_{21\_6\_weight\_enumerator}: \\
11021  >  $(ORBITER) -v 2 \ \\
11022  >  -define F -finite_field -q 4 -end \ \\
11023  >  -define v -vector -field F -format 6 \ \\
11024  >  -dense $(BCH_{21\_6\_GENERATOR\_MATRIX}) \ \\
11025  >  -end \ \\
11026  >  -with F -do \ \\
11027  >  -finite_field_activity -weight_enumerator v -end \\
11028  \\
11029  # 1y^{21} + 63x^{8}y^{13} + 294x^{12}y^{9} + 756x^{14}y^{7} + 1890x^{16}y^{5} + 1092x^{18}y^{3} \\
11030  \\
11031  #( 1, 0, 0, 0, 0, 63, 0, 0, 0, 294, 0, 756, 0, 1890, 0, 1092, 0, 0, 0 ) \\
11032  \\
11033  \\
11034  \\
11035  BCH_{21\_6\_4\_macwilliams}: \\
11036  >  $(ORBITER) -v 2 -make_macwilliams_system 21 6 4 \\
11037  >  pdflatex MacWilliams_n21_k6_q4.tex \\
11038  >  open MacWilliams_n21_k6_q4.pdf \\
11039  \\
11040  \\
11041  \\
11042  #ww := [1, 0, 0, 84, 252, 1575, 10080, 58032, 319662, 1411116, 5133744, 15282792, 37951620, 79336530, 135622080, 190615824, 21373081, 188911548, 125744304, 59721732, 17767512, 2580255] \\
11043  \\
11044  \\
\end{verbatim}
11045
11046
11047
11048
11049
11050
11051
11052
11053
11054
11055
11056
11057
11058
11059
11060
11061
11062
11063
11064
11065
11066
11067
11068
11069
11070
11071
11072
11073
11074
11075
11076
11077
11078
11079
11080
11081
11082
11083
11084
11085
11086
11087
11088
11089
11090

BCH 21 15 4 field reduction:
. $(ORBITER) -v 2 \
. . -define F -finite field -q 4 -end \
. . -with F -do \
. . -finite field activity \
. . -field reduction "BCH 21 15 4" 2 15 21 $(BCH 21 15) -end
. $(ORBITER) -v 2 \
. . -draw matrix -input csv file BCH 21 15 4.csv \
. . -box width 20 -bit depth 24 \
. . -partition 4 "30" "42" -end
. pdflatex field reduction Q4 q2 15 21.tex
. open BCH 21 15 4 draw.bmp
. #open field reduction Q4 q2 15 21.pdf
# poly of degree 12: 1,0,1,0,1,0,0,0,1,0,0,0,1
BCH 21 poly division c:
. $(ORBITER) -v 2 \
. . -define F -finite field -q 2 -end \
. . -with F -do \
. . -finite field activity \
. . -polynomial division "1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1" "1,0,1,0,1,0,0,0,1,0,0,0,1" -end

F16 roots 5:
. $(ORBITER) -v 3 \
. . -define F -finite field -q 2 -end \
. . -with F -do -finite field activity -nth roots 5 -end

F64 roots 21:
. $(ORBITER) -v 3 \
. . -define F -finite field -q 2 -end \
. . -with F -do -finite field activity -nth roots 21 -end

BCH F256 roots 771:
. $(ORBITER) -v 3 \
. . -define F -finite field -q 256 -end \
. . -with F -do -finite field activity -nth roots 771 -end

BCH F256 BCH code d16:

715


#generator polynomial is $X^{30} + 253X^{29} + 174X^{28} + 109X^{27} + 97X^{26} + 144X^{25} + 112X^{24} + 212X^{23} + 192X^{22} + 169X^{21} + 24X^{20} + 150X^{19} + 110X^{18} + 248X^{17} + 3X^{16} + 193X^{15} + 194X^{14} + 205X^{13} + 9X^{12} + 56X^{11} + 95X^{10} + 199X^{9} + 108X^{8} + 58X^{7} + 160X^{6} + 148X^{5} + 138X^{4} + 24X^{3} + 210X^{2} + 26X + 1$

# Section 10.7: Coding Theory - Reed Solomon codes

SECTION CODING THEORY REED SOLOMON CODES:

# ToDo:

F_7_BCH_code_n6:

RREF_RS_6_4_7_weight_enumerator:

compact $(CODE_RS_6_4_7)$
\begin{verbatim}
Code_RS_11:
\$(ORBITER) -v 2 \\
  -define F -finite_field -q 11 -end  \\
  -define v -vector -format 8 -field F  \\
  -compact \$(CODE_RS_10_8_11)  \\
  -end  \\
  -with F -do  \\
  -finite_field_activity -RREF v -end
\end{verbatim}

\begin{verbatim}
pdflatex RREF_example_q11_8_10.tex
\end{verbatim}

\begin{verbatim}
#gs -sDEVICE=png16 -dFIXEDMEDIA -dDEVICEWIDTHPOINTS=500 -dDEVICEHEIGHTPOINTS=45 0 \\
#> -r240 -oRREF_example_q11_8_10_page%02d.png \\
#RREF_example_q11_8_10.pdf
\end{verbatim}

\begin{verbatim}
open RREF_example_q11_8_10.pdf
\end{verbatim}

\begin{verbatim}
Code_RS_11_weight Enumerator:
\$(ORBITER) -v 2 \\
  -define F -finite_field -q 11 -end  \\
  -define v -vector -format 8 -field F  \\
  -compact \$(CODE_RS_11_RREF)  \\
  -end  \\
  -with F -do  \\
  -finite_field_activity  \\
  -weight Enumerator v  \\
  -end
\end{verbatim}

\begin{verbatim}
#1*y^(10) + 1200*x^3*y^7 + 16800*x^4*y^6 + 209160*x^5*y^5 + 1734600*x^6*y^4 + 991800*x^7*y^3 + 37189800*x^8*y^2 + 82644700*x^9*y + 82644620*x^(10)
\end{verbatim}

\begin{verbatim}
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  \\
  -finite_field_activity  \\
  -weight Enumerator v  \\
  -end
\end{verbatim}
the group cannot be computed

RS_8_field_reduction:

`$\text{(ORBITER)} -v 2 \$

- define F -finite_field -q 8 -end \
- with F -do \
- finite_field_activity \
- field_reduction "RS_8_red_2" \

- 2 5 7 $(CODE_RS_8) \
- end \

- $(ORBITER) -v 2 \$

- draw_matrix -input_csv_file RS_8_red_2.csv \
- box_width 40 -bit_depth 24 \
- partition 4 "3,3,3,3,3" "3,3,3,3,3,3,3" -end

- pdflatex field_reduction_Q8_q2.5.7.tex \
- open field_reduction_Q8_q2.5.7.pdf 

RREF_RS_8_reduced_weight Enumerator:

- $(ORBITER) -v 2 \$
- define F -finite_field -q 2 -end \
- define v -vector -format 15 -field F \
- compact $(RS_8_reduced) \
- end \
- with F -do \
- finite_field_activity \
- weight Enumerator v \
- end

CODE_21_15_4_store:

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

CODE_21_15_4_weight Enumerator:

- $(ORBITER) -v 2 \$
- define F -finite_field -q 2 -end \
- define v -vector -format 15 -field F \
- compact $(CODE_21_15_4) \

718
Reed_solomon_F8_work:

```bash
$(ORBITER) -v 3 -define F -finite_field -q 8 -end \
with F -do -finite_field_activity \n-weight Enumerator v \n-end
```

---

# Section 10.8: Coding Theory - Bounds

**SECTION CODING THEORY BOUNDS:**

**bounds for d given n6 k4 q7:**

```bash
$(ORBITER) -v 2 \
-make_bounds_for_d_given_n_and_k_and_q 6 4 7
```

**bounds for d given n15 k6 q2:**

```bash
$(ORBITER) -v 2 \
-make_bounds_for_d_given_n_and_k_and_q 15 6 2
```

```bash
# n = 15 k=6 q=2
#_GV = 5
#_singleton = 10
#_hamming = 6
#_plotkin = 7
#_griesmer = 6
```
11273  >  >  -define P -projective_space 8 F -end \
11274  >  >  -with P -do \
11275  >  >  -projective_space_activity \
11276  >  >  >  -make_gilbert_varshamov_code 15 5 \
11277  >  >  >  -end 
11278
11279
11280  # [15,6] code created 
11281
11282
11283
11284  bounds_for_d_given_n12_k4_q13: 
11285  >  $(ORBITER) -v 2 \
11286  >  >  -make.bounds_for_d_given_n.and.k.and.q 12 4 13 
11287
11288
11289
11290
11291  GV_n15_k6_d5.weight_enumerator: 
11292  >  $(ORBITER) -v 2 \
11293  >  >  -define F -finite_field -q 2 -end \
11294  >  >  -define v -vector -format 6 -field F \ 
11295  >  >  >  -compact $(CODE_GV_N15_K6) \ 
11296  >  >  >  -end \ 
11297  >  >  -with F -do \ 
11298  >  >  -finite_field_activity \ 
11299  >  >  >  -weight_enumerator v \ 
11300  >  >  >  -end 
11301
11302  #1y^{15} + 27x^6y^9 + 24x^8y^7 + 9x^{10}y^5 + 3x^{12}y^3 
11303  # surprise: d = 6 
11304
11305
11306  code_n15_k6_d6_a_we: 
11307  >  $(ORBITER) -v 2 \
11308  >  >  -define F -finite_field -q 2 -end \
11309  >  >  -define v -vector -format 6 -field F \ 
11310  >  >  >  -compact $(CODE_15_6_6_A) \ 
11311  >  >  >  -end \ 
11312  >  >  -with F -do \ 
11313  >  >  -finite_field.activity -weight_enumerator v -end 
11314
11315  #1y^{15} + 27x^6y^9 + 24x^8y^7 + 9x^{10}y^5 + 3x^{12}y^3 
11316
11317
11318
11319  # weight enumerator 

720
#1y^{15} + 28x^6y^9 + 21x^8y^7 + 12x^{10}y^5 + 2x^{12}y^3

code_n15_k6_d6_RREF:

```
$(ORBITER) -v 2 \\
define F -finite_field -q 2 -end \\
define v -vector -format 6 -field F \\
\> \> -compact $(CODE_GV_N15_K6) \\
\> \> -end \\
\> \> -with F -do -finite_field_activity \\
\> \> -RREF v -normalize_from_the_right \\
\> \> -end \\
```

code_n15_k6_d6_check_RREF:

```
$(ORBITER) -v 2 \\
define F -finite_field -q 2 -end \\
define v -vector -format 9 -field F \\
\> \> -compact $(CODE_GV_N15_K6_CHECK) \\
\> \> -end \\
\> \> -with F -do -finite_field_activity \\
\> \> -RREF v -normalize_from_the_right \\
\> \> -end \\
```

Section 10.9: Coding Theory - Classification

SECTION_CODING_THEORY_CLASSIFICATION:

code classification:

codes.8_4.4:

```
define G \\
linear_group -PGL 4 2 -end \\
with G -do \\
\> -group_theoretic_activity \\
```
-poset_classification_control \
-problem_label codes_8_4_4 \
-draw_poset \
-draw_options -embedded -radius 250 \
-line_width 1.0 -spanning_tree -end \
-report -end \
draw: 
$(ORBITER) -v 3 \
-draw_layered_graph \
codes_8_4_4_poset_lvl_8.layered_graph \
-radius 250 -embedded -line_width 1.0 \
-y_stretch 1.0 -scale 0.5 \
-end 
-pdflatex codes_8_4_4_poset_lvl_8.tex 
-open codes_8_4_4_poset_lvl_8.pdf 
-pdflatex codes_8_4_4_poset.tex 
-open codes_8_4_4_poset.pdf 
codes_8_4_4.draw: 
$(ORBITER) -v 6 \
-define G \
-linear_group -PGL 10 3 -end \
-with G -do \
-group_theoretic_activity \
-poset_classification_control \
-problem_label codes_14_4_9_3 \
-draw_poset \
-draw_options \
-embedded -radius 250 \
-end \
-end 
-pdflatex codes_14_4_9_3_poset_lvl_13.tex 
-open codes_14_4_9_3_poset_lvl_13.pdf 
codes_15_6_6_2: 
$(ORBITER) -v 6 \

```latex
\begin{verbatim}
define G  
linear_group -PGL 9 2 -end  
with G -do  
group_theoretic_activity  
poet_classification_control  
problem_label codes_15_6_6_2  
draw_poset  
draw_options  
\end

linear_codes 6 15  
\end

dflatex codes_15_6_6_2.posetlvl15.tex  
open codes_15_6_6_2.posetlvl15.pdf  

# 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
```

11461 codes_d4:
11462 ▶ $(ORBITER) -v 3 \n11463 ▶ ▶ -define G -linear_group -PGL 4 2 -end \n11464 ▶ ▶ -with G -do \n11465 ▶ ▶ -group_theoretic_activity \n11466 ▶ ▶ -poset_classification_control -W \n11467 ▶ ▶ -problem_label codes_r4_d4 -draw_poset \n11468 ▶ ▶ -embedded -linear_codes 4 100 \n11469 ▶ ▶ -end \n11470 ▶ ▶ -end
11471
11472
11473 codes_24_12_8:
11474 ▶ $(ORBITER) -v 6 \n11475 ▶ ▶ -orbiter_path $(ORBITER_PATH) \n11476 ▶ ▶ -define G \n11477 ▶ ▶ -linear_group -PGL 12 2 -end \n11478 ▶ ▶ -with G -do \n11479 ▶ ▶ -group_theoretic_activity \n11480 ▶ ▶ ▶ -poset_classification_control \n11481 ▶ ▶ ▶ ▶ -problem_label codes_24_12_8 \n11482 ▶ ▶ ▶ ▶ -draw_poset \n11483 ▶ ▶ ▶ ▶ -draw_options -embedded -radius 250 \n11484 ▶ ▶ ▶ ▶ -line_width 1.0 -spanning_tree -end \n11485 ▶ ▶ ▶ ▶ -report -end \n11486 ▶ ▶ ▶ -end \n11487 ▶ ▶ -linear_codes 8 24 \n11488 ▶ ▶ -end
11489 ▶ pdflatex codes_24_12_8_poset.tex
11490 ▶ open codes_24_12_8_poset.pdf
11491
11492 #codes_24_12_8_poset_lvl_24.layered_graph
11493
11494 codes_24_12_8.draw:
11495 ▶ $(ORBITER) -v 3 \n11496 ▶ -draw_layered_graph \n11497 ▶ ▶ codes_24_12_8_poset_lvl_24.layered_graph \n11498 ▶ ▶ -radius 100 -spanning_tree -embedded \n11499 ▶ ▶ -line_width 0.5 -x_stretch 1.4 \n11500 ▶ ▶ -scale 0.25 -nodes_empty \n11501 ▶ ▶ -end
11502 ▶ pdflatex codes_24_12_8_poset_lvl_24_draw.tex
11503 ▶ open codes_24_12_8_poset_lvl_24_draw.pdf
11504
11505 glynna_arc:
11506 ▶ $(ORBITER) -v 5 \n11507 ▶ ▶ -orbiter_path $(ORBITER_PATH) 

724
five points in general:

```bash
$(ORBITER) -v 5 \
-orbiter_path $(ORBITER_PATH) \
-define G \
-linear_group -PGL 4 2 -end \
-with G -do \
-group_theoretic_activity \
-poset_classification_control \
Enter the number of objects to classify and its dimensions.
```
Section 11.1: Combinatorics

Sym4_conj_classes:

$$(\text{ORBITER}) -v 2 -\text{conjugacy\_classes\_Sym\_n 4}$$

Sym10_conj_classes:

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

Sym15_conj_classes:

$$(\text{ORBITER}) -v 2 -\text{conjugacy\_classes\_Sym\_n 15}$$

Char_Sym4:

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

Char_Sym5:

$$(\text{ORBITER}) -v 2 -\text{character\_table\_symmetric\_group 5}$$

Char_Sym6:

$$(\text{ORBITER}) -v 2 -\text{character\_table\_symmetric\_group 6}$$

all_subsets_10_3:

$$(\text{ORBITER}) -v 2 -\text{tree\_of\_all\_k\_subsets 10 3}$$
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 sets:
GENERATORS_H5="1,2,3,4,0,6,7,8,9,5,10,11,12"
# (0, 1, 2, 3, 4)(5, 6, 7, 8, 9)

GENERATORS_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)

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"
# (0, 6, 1, 8)(2, 9, 3)(4, 7, 11)(5, 10),

LARGE_SET_S2="2,0,1,6,3,4,11,5,7,8,10,9,12"
#(0, 2, 1)(3, 6, 11, 9, 8, 7, 5, 4),

LARGE_SET_S3="12,5,6,11,3,7,10,8,9,1,4,2,0"
#(0, 12)(1, 5, 7, 8, 9)(2, 6, 10, 4, 3, 11),

LARGE_SET_S4="5,8,10,3,11,0,2,1,12,4,6,7,9"
#(0, 5)(1, 8, 12, 9, 4, 11, 7)(2, 10, 6),

LARGE_SET_S5="10,11,0,7,12,2,3,1,4,5,8,6,9"
#(0, 10, 8, 4, 12, 9, 5, 2)(1, 11, 6, 3, 7),

LARGE_SET_S6="3,4,1,9,5,6,8,2,7,11,12,10,0"
#(0, 3, 9, 11, 10, 12)(1, 4, 5, 6, 8, 7, 2),

LARGE_SET_S7="9,11,0,6,1,3,5,4,2,12,8,7,10"
#(0, 9, 12, 10, 8, 2)(1, 11, 7, 4)(3, 6, 5),

LARGE_SET_S8="10,2,12,8,0,3,4,1,5,6,9,7,11"
#(0, 10, 9, 6, 4)(1, 2, 12, 11, 7)(3, 8, 5),

LARGE_SET_S9="1,3,4,10,5,6,9,7,8,11,0,12,2"
#(0, 1, 3, 10)(2, 4, 5, 6, 9, 11, 12),
11696
11697 LARGE_SET_S10="7,12,1,6,0,4,5,2,3,10,9,8,11"
11698 #(0, 7, 2, 1, 12, 11, 8, 3, 6, 5, 4)(9, 10).
11699
11700
11701 file_S:
11702 ‡ echo ROW,C0"\n"0,"$(LARGE_SET_S0)"\n"1,"$(LARGE_SET_S1)"\n"2,"$(LARGE_SET_S2)"\n"3,"$(LARGE_SET_S3)"\n"4,"$(LARGE_SET_S4)""5,"$(LARGE_SET_S5)"\n"6,"$(LARGE_SET_S6)"\n"7,"$(LARGE_SET_S7)"\n"8,"$(LARGE_SET_S8)""9,"$(LARGE_SET_S9)"\n"10,"$(LARGE_SET_S10)""n"END"n" >S.csv
11703
11704 Large_set_H5:
11705 ‡ $(ORBITER) -v 10 \\n11706 ‡ ‡ -define G -permutation_group -symmetric_group 13 \\
11707 ‡ ‡ ‡ -subgroup_by_generators H5 5 1 $(GENERATORS_H5) -end \\
11708 ‡ ‡ -with G -do \\
11709 ‡ ‡ -group_theoretic_activity \\
11710 ‡ ‡ ‡ -report \\
11711 ‡ ‡ -end \\
11712 ‡ ‡ -with G -do \\
11713 ‡ ‡ -group_theoretic_activity \\
11714 ‡ ‡ ‡ -save_elements_csv "H5_elts.csv" \\
11715 ‡ ‡ -end
11716 ‡ pdflatex Perm13_Subgroup_H5_5_report.tex
11717 ‡ open Perm13_Subgroup_H5_5_report.pdf
11718
11719 Large_set_C13:
11720 ‡ $(ORBITER) -v 10 \\n11721 ‡ ‡ -define G -permutation_group -symmetric_group 13 \\
11722 ‡ ‡ ‡ -subgroup_by_generators C13 13 1 $(GENERATORS_C13) -end \\
11723 ‡ ‡ -with G -do \\
11724 ‡ ‡ -group_theoretic_activity \\
11725 ‡ ‡ ‡ -export_orbiter \\
11726 ‡ ‡ -end \\
11727 ‡ ‡ -with G -do \\
11728 ‡ ‡ -group_theoretic_activity \\
11729 ‡ ‡ ‡ -report \\
11730 ‡ ‡ -end \\
11731 ‡ ‡ -with G -do \\
11732 ‡ ‡ -group_theoretic_activity \\
11733 ‡ ‡ ‡ -save_elements_csv "C13_elts.csv" \\
11734 ‡ ‡ -end
11735 ‡ pdflatex Perm13_Subgroup_C13_13_report.tex
11736 ‡ open Perm13_Subgroup_C13_13_report.pdf
11737
11738
11739 ## the following lines were created using -export_orbiter:
11740 GENERATORS_Perm13_Subgroup_C13_13 = 
11742 "11,0,10,12,5,3,7,4,2,8,6,9,1"
11743 Perm13_Subgroup_C13_13:
11745 $(ORBITER) -v 2 \
11746 -define G -permutation_group -symmetric_group 13 \
11747 -subgroup_by_generators Perm13_Subgroup_C13_13 13 1 \
11748 $(GENERATORS_Perm13_Subgroup_C13_13) \
11749 -end
11750 ###
11752
11753 Large_set_mult_C13xS:
11755 $(ORBITER) -v 10 \
11756 -define G -permutation_group -symmetric_group 13 -end \
11757 -with G -do \
11758 -group_theoretic_activity \
11759 -multiply_elements_csv_column_major_ordering \ 
11760 - C13_elts.csv S.csv C13xS.csv \ 
11761 -end
11762 Large_set_mult_C13xSxH5:
11764 $(ORBITER) -v 10 \
11765 -define G -permutation_group -symmetric_group 13 -end \
11766 -with G -do \
11767 -group_theoretic_activity \
11768 -multiply_elements_csv_column_major_ordering \
11769 - C13xS.csv H5_elts.csv C13xSxH5.csv \ 
11770 -end
11771
11772 Large_set_mult_C13xSxH5_apply:
11773 $(ORBITER) -v 10 \
11774 -define G -permutation_group -symmetric_group 13 -end \
11775 -with G -do \
11776 -group_theoretic_activity \
11777 -apply_elements_csv_to_set \
11778 - C13xSxH5.csv C13xSxH5_images.csv "0,1,2,3" \ 
11779 -end
11780
11781
11782 domino_portrait:
11784 $(ORBITER) -v 3 -domino_portrait 7 4 anton_28x32 -end
11785
11786 domino_portrait_input:
11787 \> \> $(ORBITER) -v 2 \$
11788 \> \> \> -define all_one_r -vector -repeat 1 28 -end \$
11789 \> \> \> -define all_one_c -vector -repeat 1 32 -end \$
11790 \> \> \> -draw_matrix \$
11791 \> \> \> \> -grayscale \$
11792 \> \> \> \> -invert_colors \$
11793 \> \> \> \> -input_csv_file anton\_28x32\_m.csv \$
11794 \> \> \> \> -box_width 20 -bit_depth 8 \$
11795 \> \> \> \> -partition 3 \$
11796 \> \> \> \> \> all_one_c all_one_r \$
11797 \> \> \> \> -end \$
11798 \> \> \> open anton\_28x32\_m\_draw.bmp
11799
11800
11801 # Section 11.2: Diophantine Systems
11803
11804 SECTION\_DIOPHANT:
11805
11806
11807
11808 part10:
11809 \> $(ORBITER) -v 4 \$
11810 \> \> -define A -vector -dense "10,9,8,7,6,5,4,3,2,1" -end \$
11811 \> \> -define D -diophant \$
11812 \> \> \> -label part10 \$
11813 \> \> \> \> -coefficient_matrix A \$
11814 \> \> \> \> -RHS "10,10,1" \$
11815 \> \> \> \> -x_min_global 0 -x_max_global 10 \$
11816 \> \> \> -end \$
11817 \> \> \> -with D -do \$
11818 \> \> \> \> diophant\_activity -solve_mckay \$
11819 \> \> \> -end \$
11820 \> \>
11821
11822
11823 \> \>
11824 # Finds 42 solutions with 67 backtrack steps
11825
11826
11827 octic\_monomials:
11828 \> $(ORBITER) -v 4 \$
11829 \> \> -define A -vector -dense "1,1,1,1" -end \$
11830 \> \> -define D -diophant \$
11831 \> \> \> -label octic\_monomials \$
11832 \> \> \> \> -coefficient_matrix A \$
11833 \> \> \> \> -RHS "8,8,1" \$

731
11834  ▶ ▶ ▶ -x_min_global 0 -x_max_global 8 \n11835  ▶ ▶ -end \n11836  ▶ ▶ -with D -do \n11837  ▶ ▶ ▶ -diophant_activity -solve_mckay \n11838  ▶ ▶ -end \n11839  ▶ sort -r octic_monomials.sol >octic_monomials_sorted.txt
11840
11841  #Found 165 solutions with 210 backtrack steps
11842  # 165=binomial(11,3)
11843
11844
11845  solve_test_system:
11846  ▶ $(ORBITER) -v 4 \n11847  ▶ ▶ -define A -vector -format 7 -dense $(TEST_SYSTEM) -end \n11848  ▶ ▶ -define D -diophant \n11849  ▶ ▶ ▶ -label test_system \n11850  ▶ ▶ ▶ -coefficient_matrix A \n11851  ▶ ▶ ▶ -RHS $(TEST_RHS) \n11852  ▶ ▶ ▶ -x_min_global 0 -x_max_global 1 \n11853  ▶ ▶ -end \n11854
11855
11856  McKay_test:
11857  ▶ $(ORBITER) -v 4 \n11858  ▶ ▶ -define A -vector -format 7 -dense $(TEST_SYSTEM) -end \n11859  ▶ ▶ -define D -diophant \n11860  ▶ ▶ ▶ -label test_system \n11861  ▶ ▶ ▶ -coefficient_matrix A \n11862  ▶ ▶ ▶ -RHS $(TEST_RHS) \n11863  ▶ ▶ ▶ -x_min_global 0 -x_max_global 1 \n11864  ▶ ▶ -end \n11865  ▶ ▶ -with D -do \n11866  ▶ ▶ ▶ -diophant_activity -solve_mckay \n11867  ▶ ▶ -end \n11868
11869  DLX_test:
11870  ▶ $(ORBITER) -v 4 \n11871  ▶ ▶ -define A -vector -format 7 -dense $(TEST_SYSTEM) -end \n11872  ▶ ▶ -define D \n11873  ▶ ▶ -diophant -label test_system \n11874  ▶ ▶ ▶ -coefficient_matrix A \n11875  ▶ ▶ ▶ -RHS $(TEST_RHS) \n11876  ▶ ▶ ▶ -x_min_global 0 -x_max_global 1 \n11877  ▶ ▶ -end \n11878  ▶ ▶ -with D -do \n11879  ▶ ▶ ▶ -diophant_activity -solve_DLX \n11880  ▶ ▶ -end

732
# Section 11.3: Combinatorial Linear Spaces

**SECTION COMBINATORIAL_LINEAR_SPACES:**

---

**linsp6:**

```
$(ORBITER) -v 4 \
  -define A -vector -format 1 -dense "15,10,6,3,1" -end \
  -define D -diophant -label linsp6 \
  -coefficient_matrix A \
  -RHS "15,15,1" \
  -x_min_global 0 \
  -x_max_global 15 \
  -end \
  -with D -do \
  -diophant_activity -solve_mckay \
  -end
```

# Found 15 solutions with 22 backtrack steps

---

**linsp7:**

```
$(ORBITER) -v 4 \
  -define A -vector -format 1 -dense "21,15,10,6,3,1" -end \
  -define D -diophant -label linsp7 \
  -coefficient_matrix A \
  -RHS "21,21,1" \
  -x_min_global 0 \
  -x_max_global 21 \
  -end \
  -with D -do \
  -diophant_activity -solve_mckay \
  -end
```

# 32 solutions in 45 backtrack steps
geometry builder:
11975 \>$($ORBITER) -v 8 \$
11976 \>$\text{-define Test_lines -set -loop 1 7 1 -end} \$
11977 \>$\text{-define Geo -geometry_builder} \$
11978 \>$\text{-V 6 -B 4 -TDO 2 -fuse 1} \$
11979 \>$\text{-fname GEO pasch} \$
11980 \>$\text{-test Test_lines} \$
11981 \>$\text{-end} \$
11982
11983
11984 geo_petersen:
11985 \>$($ORBITER) -v 8 \$
11986 \>$\text{-define Test_lines -set -loop 3 11 1 -end} \$
11987 \>$\text{-define Geo -geometry_builder} \$
11988 \>$\text{-V 10 -B 15 -TDO 3 -fuse 1} \$
11989 \>$\text{-fname GEO petersen -girth 5} \$
11990 \>$\text{-search_tree} \$
11991 \>$\text{-test Test_lines} \$
11992 \>$\text{-end} \$
11993
11994 geo_7_3:
11995 \>$($ORBITER) -v 2 \$
11996 \>$\text{-define Test_lines -set -loop 3 8 1 -end} \$
11997 \>$\text{-define Geo -geometry_builder} \$
11998 \>$\text{-V 7 -B 7 -TDO 3} \$
11999 \>$\text{-fuse 1 -fname GEO 7_3} \$
12000 \>$\text{-test Test_lines} \$
12001 \>$\text{-end} \$
12002
12003 geo_7_3_no_square_test:
12004 \>$($ORBITER) -v 2 \$
12005 \>$\text{-define Test_lines -set -loop 3 8 1 -end} \$
12006 \>$\text{-define Geo -geometry_builder} \$
12007 \>$\text{-V 7 -B 7 -TDO 3} \$
12008 \>$\text{-fuse 1 -fname GEO 7_3_nst} \$
12009 \>$\text{-test Test_lines} \$
12010 \>$\text{-no_square_test} \$
12011 \>$\text{-end} \$
12012
12013 geo_7_3_no_square_test_draw:
12014 \>$($ORBITER) -v 10 \$
12015 \>$\text{-draw incidence.structure.description} \$
12016 \>$\text{-width 60 -with 10 6 -end} \$
12017 \>$\text{-define C -combinatorial_objects} \$
12018 \>$\text{-file_of_incidence_geometries 7_3_nst.inc 7 7 21} \$
12019 \>$\text{-end} \$
12020 \>$\text{-with C -do} \$
12021 \>$\text{-combinatorial_object_activity} \$

735
draw_incidence_matrices 
7_3_nst 
-end 
pdflatex 7_3_nst_incma.tex
open 7_3_nst_incma.pdf

gemo_7_3_orderly:
$(ORBITER) -v 200 
-define Test_lines -set -loop 3 8 1 -end 
-define Geo -geometry_builder 
-V 7 -B 7 -TDO 3 
-fuse 1 -fname GEO 7_3 
-test Test_lines 
-search_tree 
-orderly 
-end 
gemo_7_3_orderly_draw:
$(ORBITER) -v 20 
-draw_options -embedded -radius 50 
-xin 10000 -yin 10000 
-xout 1000000 -yout 1000000 
-nodes_empty 
-scale 0.5 -line_width 0.3 
-end 
-tree_draw -file 7_3_tree.txt -end 
pdflatex 7_3_tree_draw.tex
open 7_3_tree_draw.pdf

gemo_7_3_orderly_mem_debug:
$(ORBITER) -v 20 
-memory_debug 2 
-define Test_lines -set -loop 3 8 1 -end 
-define Geo -geometry_builder 
-V 7 -B 7 -TDO 3 
-fuse 1 -fname GEO 7_3 
-test Test_lines 
-search_tree 
-orderly 
-end 
gemo_8_3:
$(ORBITER) -v 2 
-define Test_lines -set -loop 3 9 1 -end 
-define Geo -geometry_builder 

#-print at line 4
# 1 geo: 0 11 18 29 30 38 44 54
# ago=48

geo 9.3:
$(ORBITER) -v 2 
-define Test_lines -set -loop 3 10 1 -end 
-define Geo -geometry_builder 
-V 9 -B 9 -TDO 3 
-fuse 1 -fname_GEO 9.3 
-test Test_lines 
-end

geo 10.3:
$(ORBITER) -v 2 
-define Test_lines -set -loop 4 11 1 -end 
-define Geo -geometry_builder 
-V 10 -B 10 -TDO 3 -fuse 1 
-fname_GEO 10.3 
-test Test_lines 
-end

# 10 geos
# 8/26/2021: 0 sec on Mac

calib_e 3:
$(ORBITER) -v 10 
-draw.incidence_structure.description 
-width 60 -with_10 6 -end 
-define C -combinatorial_objects 
-file.of.incidence.geometries 
-10.3.inc 10 10 30 
-end 
-with C -do 

12116 \   \ -combinatorial_object_activity
12117 \   \ -draw_incidence_matrices
12118 \   \ -end
12119 \   \ 10.3.inc
12120 \  \-draw incidence matrices
12121 \  \open 10.3.inc.incma.pdf
12122
12123
12124 geo_10.3_orderly:
12125 \ $(ORBITER) -v 20
12126 \ -define Test_lines -set -loop 4 11 1 -end
12127 \ -define Geo -geometry_builder
12128 \ -V 10 -B 10 -TDO 3 -fuse 1
12129 \ -fname_geo 10.3
12130 \ -test Test_lines
12131 \ -orderly
12132 \ -end
12133
12134 geo_10.3_orderly_mem_debug:
12135 \ $(ORBITER) -v 2
12136 \ -memory_debug 2
12137 \ -define Test_lines -set -loop 4 11 1 -end
12138 \ -define Geo -geometry_builder
12139 \ -V 10 -B 10 -TDO 3 -fuse 1
12140 \ -fname_geo 10.3
12141 \ -test Test_lines
12142 \ -orderly
12143 \ -end
12144
12145
12146 geo_10.3_tree:
12147 \ $(ORBITER) -v 20
12148 \ -define Test_lines -set -loop 0 11 1 -end
12149 \ -define GEO -geometry_builder
12150 \ -V 10 -B 10 -TDO 3 -fuse 1
12151 \ -fname_geo 10.3
12152 \ -search_tree
12153 \ -test Test_lines
12154 \ -end
12155 \ $(ORBITER) -v 20
12156 \ -draw_options -embedded -radius 20
12157 \ -paperheight 220
12158 \ -paperwidth 330
12159 \ -xin 10000 -yin 10000
12160 \ -xout 1000000 -yout 500000
12161 \ -scale 2 -line_width 0.3
12162 \ -nodes_empty

738
12163 \> \> -end \ 
12164 \> \> -tree_draw \ 
12165 \> \> \> -file 10_3.tree.txt \ 
12166 \> \> -end \ 
12167 \> pdflatex 10_3.tree_draw.tex \ 
12168 \> open 10_3.tree_draw.pdf \ 
12169 \ 
12170 \ 
12171 \ 
12172 \ 
12173 geo_10_3.tree_path: \ 
12174 \> $(ORBITER) -v 20 \ 
12175 \> \> -define Test_lines -set -loop 0 11 1 -end \ 
12176 \> \> -define GEO -geometry_builder \ 
12177 \> \> \> -V 10 -B 10 -TDO 3 -fuse 1 \ 
12178 \> \> \> -fname_GEO 10_3 \ 
12179 \> \> \> -search_tree \ 
12180 \> \> \> -test Test_lines \ 
12181 \> \> -end \ 
12182 \> $(ORBITER) -v 20 \ 
12183 \> \> -draw_options -embedded -radius 20 \ 
12184 \> \> \> -paperheight 220 \ 
12185 \> \> \> -paperwidth 330 \ 
12186 \> \> \> -xin 10000 -yin 10000 \ 
12187 \> \> \> -xout 1000000 -yout 500000 \ 
12188 \> \> \> -scale 2 -line_width 0.3 \ 
12189 \> \> -end \ 
12190 \> \> -tree_draw \ 
12191 \> \> \> -restrict 2 \ 
12192 \> \> \> -file 10_3.tree.txt \ 
12193 \> \> \> -select_path "0,0,15,26,46,56,72,80,93,106,119" \ 
12194 \> \> -end \ 
12195 \> pdflatex 10_3.tree_draw.tex \ 
12196 \> open 10_3.tree_draw.pdf \ 
12197 \ 
12198 #> \> \> -nodes_empty \ 
12199 #>-sideways \ 
12200 \ 
12201 \ 
12202 Desargues_path_lex_least_draw: \ 
12203 \> echo $(DESARGUES_PATH.LEX.LEAST) >Desargues_path_lex_least.inc \ 
12204 \> $(ORBITER) -v 10 \ 
12205 \> \> -draw_incidence_structure_description \ 
12206 \> \> \> -width 60 -with 10 6 -end \ 
12207 \> \> \> -define C -combinatorial_objects \ 
12208 \> \> \> -file_of_incidence_geometries_by_row_ranks \ 
12209 \> \> \> Desargues_path_lex_least.inc 10 10 3 \ 

739
12210 ▶ ▶ -end \n12211 ▶ ▶ -with C -do \n12212 ▶ ▶ -combinatorial_object_activity \n12213 ▶ ▶ ▶ -draw_incidence_matrices \n12214 ▶ ▶ ▶ Desargues_path_lex_least \n12215 ▶ ▶ -end
12216 ▶ pdflatex Desargues_path_lex_least_incma.tex
12217 ▶ open Desargues_path_lex_least_incma.pdf
12218
12219 DESARGUES_PATH_CANONICAL_ANCESTOR="10 10 3\n0\n1 0\n2 112 119\n3 89 112 119\n4 1 118 89 82\n5 106 114 69 107 111\n6 85 105 112 99 83 61\n7 94 105 113 85 35 83 6\n8 26 119 55 105 92 79 48\n9 119 93 106 15 26 79 55 73 47\n10 0 119 93 106 1\n-1"
12220
12221 Desargues_path_can_anc_draw:
12222 ▶ echo $(DESARGUES_PATH_CANONICAL_ANCESTOR) >Desargues_path_can_anc.inc
12223 ▶ $(ORBITER) -v 10 \n12224 ▶ ▶ -draw_incidence_structure_description \n12225 ▶ ▶ ▶ -width 60 -with_10 6 -end \n12226 ▶ ▶ -define C -combinatorial_objects \n12227 ▶ ▶ ▶ -file_of_incidence_geometries_by_row_ranks Desargues_path_can_anc.inc 10 10 \n12228 ▶ ▶ ▶ 3 \n12229 ▶ ▶ -end \n12230 ▶ ▶ -with C -do \n12231 ▶ ▶ -combinatorial_object_activity \n12232 ▶ ▶ ▶ -draw_incidence_matrices \n12233 ▶ ▶ ▶ Desargues_path_can_anc \n12234 ▶ ▶ -end
12235 ▶ pdflatex Desargues_path_can_anc_incma.tex
12236 ▶ open Desargues_path_can_anc_incma.pdf
12237
12238
12239 geo_11.3:
12240 ▶ $(ORBITER) -v 2 \n12241 ▶ ▶ -define Test_lines -set -loop 4 12 1 -end \n12242 ▶ ▶ -define Geo -geometry_builder \n12243 ▶ ▶ ▶ -V 11 -B 11 -TDO 3 \n12244 ▶ ▶ ▶ -fuse 1 -fname_GEO 11_3 \n12245 ▶ ▶ ▶ -test Test_lines \n12246 ▶ ▶ -end
12247
12248 # 31 geos
12249 # 8/26/2021: 0 sec on Mac
12250
12251 geo_12.3:
12252 ▶ $(ORBITER) -v 2 \n
12253  ▶ ▶ -define Test_lines -set -loop 4 13 1 -end \n12254  ▶ ▶ -define Geo -geometry_builder \n12255  ▶ ▶ ▶ -V 12 -B 12 -TDO 3 \n12256  ▶ ▶ ▶ -fuse 1 -fname_GEO 12_3 \n12257  ▶ ▶ ▶ -test Test_lines \n12258  ▶ ▶ -end
12259
12260 # 229 geos
12261 #User time: 0.45 of a second, dt=45 tps = 100
12262 #nb_calls_to_densenauty=24586
12263
12264
12265 geo_12_3_orderly:
12266  ▶ $(ORBITER) -v 2 \n12267  ▶ ▶ -define Test_lines -set -loop 4 13 1 -end \n12268  ▶ ▶ -define Geo -geometry_builder \n12269  ▶ ▶ ▶ -V 12 -B 12 -TDO 3 \n12270  ▶ ▶ ▶ -fuse 1 -fname_GEO 12_3 \n12271  ▶ ▶ ▶ -test Test_lines \n12272  ▶ ▶ ▶ -f_orderly \n12273  ▶ ▶ -end
12274
12275
12276
12277 geo_13_3:
12278  ▶ $(ORBITER) -v 2 \n12279  ▶ ▶ -define Test_lines -set -loop 4 14 1 -end \n12280  ▶ ▶ -define Geo -geometry_builder \n12281  ▶ ▶ ▶ -V 13 -B 13 -TDO 3 \n12282  ▶ ▶ ▶ -fuse 1 -fname_GEO 13_3 \n12283  ▶ ▶ ▶ -test Test_lines \n12284  ▶ ▶ -end
12285
12286 # 2036 geos, 96, 39, 13, 12^4, 8^3, 6^16, 4^30, 3^20, 2^190, 1^1770
12287 #User time: 0:4
12288 #nb_calls_to_densenauty=216777
12289
12290 geo_13_3_orderly:
12291  ▶ $(ORBITER) -v 2 \n12292  ▶ ▶ -define Test_lines -set -loop 4 14 1 -end \n12293  ▶ ▶ -define Geo -geometry_builder \n12294  ▶ ▶ ▶ -V 13 -B 13 -TDO 3 \n12295  ▶ ▶ ▶ -fuse 1 -fname_GEO 13_3 \n12296  ▶ ▶ ▶ -test Test_lines \n12297  ▶ ▶ ▶ -f_orderly \n12298  ▶ ▶ -end
12299
geo_14_3:

```
$(ORBITER) -v 2 \
  -define Test_lines -set -loop 4 15 1 -end \
  -define Geo -geometry_builder \
  -V 14 -B 14 -TDO 3 \
  -fuse 1 -fname GEO 14_3 \
  -test Test_lines \
  -end
```

# 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

#User time: 0:55
#nb_calls_to_densenauty=2089344

geo_14_3.orderly:

```
$(ORBITER) -v 2 \
  -define Test_lines -set -loop 4 15 1 -end \
  -define Geo -geometry_builder \
  -V 14 -B 14 -TDO 3 \
  -fuse 1 -fname GEO 14_3 \
  -test Test_lines \
  -f orderly \
  -end
```

#User time: 0:50

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 \
  -test Test_lines \
  -end
```

# 245342 geos, 8064, 720, 192^2, 128, 72, 48^-6, 32, 30^-2, 24^-2, 20^-2, 18, 16^-10, 15^-2, 12^-11, 10^-3, 8^-34, 6^-59, 5^-5, 4^-180, 3^-69, 2^-3709, 1^-241240

# 8 min 11 sec on Mac
# running out of memory

geo_15_3.g4:

```
$(ORBITER) -v 2 \
```
12345 \>$\text{-define Test_lines -set -loop 4 16 1 -end} \$
12346 \>$\text{-define Geo -geometry\_builder} \$
12347 \>$\text{-V 15 -B 15 -TDO 3} \$
12348 \>$\text{-fuse 1 -fname\_GEO 15\_3\_g4} \$
12349 \>$\text{-girth 4} \$
12350 \>$\text{-search\_tree} \$
12351 \>$\text{-test Test\_lines} \$
12352 \>$\text{-end} \$
12353 \>$\text{ORBITER} -v 2 \$
12354 \>$\text{-draw\_options -embedded -radius 50} \$
12355 \>$\text{-nodes\_empty} \$
12356 \>$\text{-scale 0.5 -line\_width 0.3 -end} \$
12357 \>$\text{-tree\_draw -file 15\_3\_g4\_tree.txt -end} \$
12358 \>$\text{pdflatex 15\_3\_g4\_tree\_draw.tex} \$
12359 \>$\text{open 15\_3\_g4\_tree\_draw.pdf} \$
12360 \>\text{# The unique Cremona Richmond configuration with group of order 720} \$
12361 \>\text{User time: 0 of a second, dt=0 tps = 100} \$
12362 \>\text{nb\_calls\_to\_densenauty=23} \$
12363 \>\text{-sideways} \$
12364 \>\text{-sideways} \$
12366
12367
12368
12369
12370 \text{geo\_17\_3\_g4\_orderly:} \$
12371 \>$\text{ORBITER} -v 2 \$
12372 \>$\text{-memory\_debug 2} \$
12373 \>$\text{-define Test\_lines -set -loop 4 18 1 -end} \$
12374 \>$\text{-define Geo -geometry\_builder} \$
12375 \>$\text{-V 17 -B 17 -TDO 3} \$
12376 \>$\text{-fuse 1 -fname\_GEO 17\_3\_g4} \$
12377 \>$\text{-girth 4} \$
12378 \>$\text{-test Test\_lines} \$
12379 \>$\text{-orderly} \$
12380 \>$\text{-end} \$
12381
12382 \>\text{# 1 sol} \$
12383
12384 \text{geo\_18\_3\_g4:} \$
12385 \>$\text{ORBITER} -v 2 \$
12386 \>$\text{-define Test\_lines -set -loop 4 19 1 -end} \$
12387 \>$\text{-define Geo -geometry\_builder} \$
12388 \>$\text{-V 18 -B 18 -TDO 3} \$
12389 \>$\text{-fuse 1 -fname\_GEO 18\_3\_g4} \$
12390 \>$\text{-girth 4} \$
12391 \>$\text{-search\_tree} \$

743
geo_19.3_g4:

$ (ORBITER) -v 2 \\
$define Test_lines -set -loop 4 20 1 -end \\
$define Geo -geometry_builder \\
-V 19 -B 19 -TDO 3 \\
-fuse 1 -fname_GEO 19.3_g4 \\
girth 4 \\
-test Test_lines \\
-end \\

# 4 sol, 1 sec

geo_20.3_g4:

$ (ORBITER) -v 2 \\
$define Test_lines -set -loop 4 21 1 -end \\
$define Geo -geometry_builder \\
-V 20 -B 20 -TDO 3 \\
-fuse 1 -fname_GEO 20.3_g4 \\
girth 4 \\
-test Test_lines \\
-end \\

# 14 sol, 10 sec on Mac

geo_21.3_g4:

$ (ORBITER) -v 2 \\
$define Test_lines -set -loop 4 22 1 -end \\
$define Geo -geometry_builder \\
-V 21 -B 21 -TDO 3 \\
-fuse 1 -fname_GEO 21.3_g4 \\
girth 4 \\
-test Test_lines \\
-end \\

# 162 sol, User time: 2:5 on Mac

geo_15.4:

$ (ORBITER) -v 2 \\
$define Test_lines -set -loop 4 16 1 -end \\
$define Geo -geometry_builder \\
-V 15 -B 15 -TDO 4 \\
-fuse 1 -fname_GEO 15.4 \\

# 4 sol, 1 sec

12439 \times \times \times \ -\text{search\_tree} \\
12440 \times \times \times \ -\text{test Test\_lines} \\
12441 \times \times \times \ -\text{end} \\
12442 \times \times \times \ -$(\text{ORBITER}) -v 2 \\
12443 \times \times \times \ -\text{draw\_options} -\text{embedded} -\text{radius} 50 \\
12444 \times \times \times \ -\text{nodes\_empty} \\
12445 \times \times \times \ -\text{scale} 0.5 -\text{line\_width} 0.3 -\text{end} \\
12446 \times \times \times \ -\text{tree\_draw} -\text{file} 15_4\_tree\_txt -\text{end} \\
12447 \times \times \times \ -\text{pdflatex} 15_4\_tree\_draw\_tex \\
12448 \times \times \times \ -\text{open} 15_4\_tree\_draw\_pdf \\
12449 \\
12450 \# 4 \text{ objects} \\
12451 \# \text{ago=360, 30, 24, 15} \\
12452 \#\text{User time: 0.15 of a second, dt=15 tps = 100} \\
12453 \#\text{nb\_calls\_to\_densenauty=6767} \\
12454 \\
12455 \\
12456 12457 \text{geo\_16\_4\_g4:} \\
12458 \times \times \times \ -$(\text{ORBITER}) -v 2 \\
12459 \times \times \times \ -\text{define Test\_lines} -\text{set} -\text{loop} 4 17 1 -\text{end} \\
12460 \times \times \times \ -\text{define Geo} -\text{geometry\_builder} \\
12461 \times \times \times \ -\text{V 16} -\text{B 16} -\text{TDO 4} \\
12462 \times \times \times \ -\text{fuse 1} -\text{fname\_GEO 16\_4\_g4} \\
12463 \times \times \times \ -\text{girth 4} \\
12464 \times \times \times \ -\text{test Test\_lines} \\
12465 \times \times \times \ -\text{end} \\
12466 \\
12467 \# \text{none} \\
12468 \\
12469 12470 \text{40\_4\_g4\_inc:} \\
12471 \times \times \times \ -$(\text{ORBITER}) -v 2 \\
12472 \times \times \times \ -\text{define Test\_lines} -\text{set} -\text{loop} 0 41 1 -\text{end} \\
12473 \times \times \times \ -\text{define Geo} -\text{geometry\_builder} \\
12474 \times \times \times \ -\text{V 40} -\text{B 40} -\text{TDO 4} \\
12475 \times \times \times \ -\text{fuse 1} -\text{fname\_GEO 40\_4\_g4} \\
12476 \times \times \times \ -\text{girth 4} \\
12477 \times \times \times \ -\text{search\_tree} \\
12478 \times \times \times \ -\text{test Test\_lines} \\
12479 \times \times \times \ -\text{end} \\
12480 \times \times \times \ -$(\text{ORBITER}) -v 2 \\
12481 \times \times \times \ -\text{draw\_options} -\text{embedded} -\text{radius} 50 \\
12482 \times \times \times \ -\text{xin 10000} -\text{yin 10000} \\
12483 \times \times \times \ -\text{xout 1000000} -\text{yout 1000000} \\
12484 \times \times \times \ -\text{nodes\_empty} \\
12485 \times \times \times \ -\text{scale} 0.5 -\text{line\_width} 0.3 -\text{end} \\

745
geo_63_3_g6:
$\$(ORBITER) -v 2 \\
define Test_lines -set -loop 4 64 1 -end \\
define Geo -geometry_builder \\
-V 63 -B 63 -TDO 3 \\
fuse 1 -fname_GEO 63_3_g6 \\
girth 6 \\
test Test_lines \\
-end

geo_LSQ6:
$\$(ORBITER) -v 2 \\
define Test_lines -set -loop 1 19 1 -end \\
define Geo -geometry_builder \\
-V 6,6,6 -B 1,1,1,36 -TDO "1,0,0,6, 0,1,0,6, 0,0,1,6" \\
fuse 3 -fname_GEO LSQ6 \\
test Test_lines \\
-end

geo_16:
$\$(ORBITER) -v 2 \\
define Test_lines -set -loop 3 17 1 -end \\
define Geo -geometry_builder \\
-V 16 -B 20 -TDO 5 \\
fuse 1 -fname_GEO geo_16 \\
test Test_lines \\
-end

# Section 11.5: Design Theory

SECTION DESIGN THEORY:
design_PG_2.3:
$(ORBITER) -v 8 \n-define D -design -q 3 -family PG.2.q -end \n-with D -do \n-design_activity \n-export_inc \n-end

# writes PG.2.3_inc.txt

design_PG_2.4:
$(ORBITER) -v 8 \n-define D -design -q 4 -family PG.2.q -end \n-with D -do \n-design_activity \n-export_inc \n-end

GENERATORS
H5="1,2,3,4,0,6,7,8,9,5,10,11,12"

GENERATORS
N5="
0,1,2,3,4,9,5,6,7,8,10,11,12, 
0,1,2,3,4,5,6,7,8,9,10,12,11, 
0,4,3,2,1,5,9,8,7,6,10,11,12, 
0,2,4,1,3,5,7,9,6,8,10,11,12, 
0,1,2,3,4,5,6,7,8,9,11,10,12, 
1,2,3,4,0,6,7,8,9,5,10,11,12, 
5,9,8,7,6,0,4,3,2,1,10,11,12"

design_PG_2.3_table_create:
$(ORBITER) -v 2 \n-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

#User time: 7:30

747
design_PG_2_3_group_5:

-define D -design -q 3 -family PG_2_q -end \ 
-define T -design_table D "PG_2.13" Sym13 -end \ 
-define LSW -large_set_with_symmetry_assumption T \ 
-H "5" $(GENERATORS_H5) \ 
-N "1200" $(GENERATORS_N5) \ 
-prefix "H5" \ 
-selected_orbit_length 5 \ 
-end \ 
-with LSW -do \ 
-large_set_with_symmetry_assumption_activity \ 
-normalizer_on_orbits_of_a_given_length 5 \ 
-end

#H5_N_orbit_reps.csv
678 orbits
User time: 2:39

wreath_product_designs_n4_k2_inc.txt:

-design_activity \ 
-export.inc \ 
-end

wreath_product_designs_n8_k6_inc.txt:

$ORBITER -v 8 \ 
-design_activity \ 
-export.inc \ 
-end
# wreath_product_designs_n8_k6_inc.txt

The design with have 16 points and 3920 blocks of size 6.

# KM_cyclic_7:

```
$ (ORBITER) -v 3 \\
-define gens -vector -dense "1,2,3,4,5,6,0" -end \\
-define G -permutation group -symmetric group 7 \\
-subgroup by generators "C7" 7 1 gens \\
-define G -end \\
-define D -design activity \\
-export inc \\
-end \\
```

# to create simple 7-designs on 33 points with block size 8 and lambda = 10 invariant under PGGL(2,32):

```
$ (ORBITER) -v 3 \\
```

749
define G -linear_group -PGGL 2 32 -end 
-with G -do 
-group_theoretic_activity 
-poset_classification_control 
-problem_label KM_PGGL_2.32 -W -depth 8 
-Kramer_Mesner_matrix 7 8 
-draw_poset 
-draw_options -embedded -sideways -radius 50 
-scale 0.5 -line_width 0.3 -end 
-end 
-orbits_on_subsets 8 
-end 
$\text{ORBITER} -v 2 -draw_matrix 
-input.csv.file KM_PGGL_2.32_KM_7_8.csv 
-box_width 20 -bit_depth 24 
-partition 3 32 97 -end 
pdflatex KM_PGGL_2.32_poset lvl_8.tex 
open KM_PGGL_2.32_poset lvl_8.pdf 
open KM_PGGL_2.32_KM_7_8.draw.bmp 
$\text{ORBITER} -v 4 
-define A -vector -file KM_PGGL_2.32_KM_7_8.csv -end 
-define D -diophant 
-label "KM_PGGL_2.32_KM_7_8_system" 
-coefficient_matrix A 
-RHS_constant "10,10,1" 
-x_min_global 0 -x_max_global 1 
-end 
-with D -do 
-diophant_activity -solve_mckay 
-end 

KM_PSL_3.5: 
$\text{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 

AG_2.3.inc:

```
$(ORBITER) -v 2 -define Geo -geometry_builder

-define Geo -geometry_builder

-V 9 -B 12

-TDO 4 -fuse 1

-fname GEO AG_2.3

-test 3,4,5,6,7,8,9

-end

#9 12 3

#0 13 22 27 35 41 47 53 55 59 71 76

#-1 1

#432

LS AG_2.3 design_table_create:
```
 ORBITER -v 20
-define D -design -list_of_blocks
-define Sym9 -permutation_group -symmetric_group 9 -end
-define T -design_table D "AG_2_3" Sym9

# creates AG_2_3_design_table.csv
# and AG_2_3_makefile

#0,0,13,22,27,35,41,47,53,55,59,71,76
# is the first design in AG_2_3_design_table.csv

#poset_orbit_node::init_root_node storing strong generators for a group of order 362880
# stabilizer order 432
# 840 designs

#User time: 0.13 of a second, dt=13 tps = 100

AG_2_3_on_designs:
-define gens -vector -file AG_2_3_gens.csv -end
-define G -permutation_group
-bsgs AG_2_3 "AG_2_3" 840 362880 "0,1,2,3,4,5,6,7" 8 gens -end
-with G -do
-group_theoretic_activity
-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:
-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" 840 432 "0,1,2,3,4,5,6,7" 5 gens
-end
-define Gr -modified_group -from G
-restricted_action $(LARGE_SET_AG_2_3_NEIGHBOR_SET)
-end

752
AG_2_3_stab_orb_0_Perm840_res192:

```bash
$ (ORBITER) -v 2 \
define gens -vector -file Perm840_res192_gens.csv -end \
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 \
```

```bash
pdflatex Perm840_res192_report.tex
open Perm840_res192_report.pdf
```

LS_AG_2_3_disjoint_sets_graph_and_cliques:

```bash
$ (ORBITER) -v 2 \
define Gamma -graph \
disjoint_sets_graph \
AG_2_3_design_table.csv \
end \
with Gamma -do \
graph_theoretic_activity \
save \
end \
with Gamma -do \
graph_theoretic_activity \
find_cliques -target_size 7 -end \
end \
```

```bash
symbols
```

```bash
```

#AG_2_3_design_table_disjoint_sets_colored_graph
#User time: 0.66 of a second, dt=66 tps = 100
#AG_2_3_design_table_disjoint_sets_sol.txt
#AG_2_3_design_table_disjoint_sets_sol.csv
#15360 solutions
12860 \$ (ORBITER) -v 4 \
12861 \$ (ORBITER) -v 4 \
12862 -define Gamma -graph -load \
12863 \$ (ORBITER) -v 4 \
12864 -with Gamma -do \
12865 -graph_theoretic_activity \
12866 -split_by_clique "0" "0" \
12867 -end \
12868 
12869 
12870 #AG_2_3_design_table_disjoint_sets_0.graph 
12871 #AG_2_3_design_table_disjoint_sets_0_subset.txt 
12872 
12873 
12874 
12875 LS_AG_2_3_export_solutions: 
12876 \$ (ORBITER) -v 20 \
12877 -define D -design -list_of_blocks 9 3 \
12878 -define Sym9 -permutation_group -symmetric_group 9 -end \
12879 -define T -design_table D "AG_2_3" Sym9 \
12880 -with D -do \
12881 -design_activity \
12882 -extract_solutions_by_index "AG_2_3" Sym9 \
12883 -solutions.csv \
12884 -solutions.csv \
12885 -end \
12886 
12887 
12888 
12889 
12890 #User time: 0.39 of a second, dt=39 tps = 100 
12891 # solutions.csv 
12892 
12893 
12894 
12895 
12896 ####################################################################################################################
12897 # Section 11.7: Design Theory - Delandtsheer Doyen 
12898 
12899 SECTION_DESIGN_THEORY_DELANDTHSHEER_DOYEN: 
12900 
12901 
12902 DD_PP4: 
12903 \$ (ORBITER) -v 6 \
12904 -Delandtsheer_Doyen $(PP4) $(PP4_GROUP1) $(PP4_MASK1) \
12905 -end \
12906
\[
\text{ORBITER} -v 4 \\
\text{define D -diophant -label PP4} \\
\text{problem_of_Steiner_type 10 PP4_pair_covering.csv} \\
\text{has_sum 1} \\
\text{end} \\
\text{with D -do} \\
\text{diophant_activity -solve_mckay} \\
\text{end}
\]

\[
\text{ORBITER} -v 6 \\
\text{Delandtsheer_Doyen -search_wrt_subgroup} \\
\text{DELANDTSHEER DOYEN PROBLEM COLBOURN COLBOURN 7 13} \\
\text{DELANDTSHEER DOYEN PROBLEM COLBOURN COLBOURN 7 13 GROUP1} \\
\text{DELANDTSHEER DOYEN PROBLEM COLBOURN COLBOURN 7 13 MASK1} \\
\text{end}
\]

#target level: 6
#k2: 15
#number of k-orbits at target level: 1774964

# creates DD_CC_7_13_pair_covering.csv

\[
\text{ORBITER} -v 4 \\
\text{define D -diophant -label DD_CC_7_13} \\
\text{problem_of_Steiner_type 45 DD_CC_7_13_pair_covering.csv} \\
\text{has_sum 3} \\
\text{end} \\
\text{with D -do} \\
\text{diophant_activity -solve_mckay} \\
\text{end}
\]

# no solution

# 18603 = 27 * 53 * 13

755
# Section 11.8: Tactical Decompositions

SECTION TACTICAL DECOMPOSITIONS:

12954 DD_M1_G1:
12955 ➢ $(ORBITER) -v 4 \\n12956 ➢ ➢ -Delandtsheer_Doyen \\
12957 ➢ ➢ ➢ $(DELANDTSHEER_DOYEN_PROBLEM_27_53) \\
12958 ➢ ➢ ➢ $(DELANDTSHEER_DOYEN_PROBLEM_27_53_GROUP1) \\
12959 ➢ ➢ ➢ $(DELANDTSHEER_DOYEN_PROBLEM_27_53_MASK1) \\
12960 ➢ ➢ ➢ -end \\
12961 \\
12962 DD_M1_G1_S:
12963 ➢ $(ORBITER) -v 4 \\n12964 ➢ ➢ -Delandtsheer_Doyen \\
12965 ➢ ➢ ➢ $(DELANDTSHEER_DOYEN_PROBLEM_27_53) \\
12966 ➢ ➢ ➢ $(DELANDTSHEER_DOYEN_PROBLEM_27_53_GROUP1) \\
12967 ➢ ➢ ➢ $(DELANDTSHEER_DOYEN_PROBLEM_27_53_MASK1) \\
12968 ➢ ➢ ➢ -singletons \\
12969 ➢ ➢ ➢ -end \\
12970 \\
12971 DD_PG_2_4_M1_G1:
12972 ➢ $(ORBITER) -v 4 \\n12973 ➢ ➢ -Delandtsheer_Doyen \\
12974 ➢ ➢ ➢ $(DELANDTSHEER_DOYEN_PROBLEM_3_7) \\
12975 ➢ ➢ ➢ $(DELANDTSHEER_DOYEN_PROBLEM_3_7_GROUP1) \\
12976 ➢ ➢ ➢ $(DELANDTSHEER_DOYEN_PROBLEM_3_7_MASK1) \\
12977 ➢ ➢ ➢ -end \\
12978 ➢ ➢ ➢ -end \\
12979 \\
12980 PG_2_27_special:
12981 ➢ $(ORBITER) -v 2 \\n12982 ➢ ➢ -define F -finite_field -q 27 -override_polynomial 46 -end \\
12983 ➢ ➢ -define P -projective_space 2 F -end \\
12984 ➢ ➢ -with P -do -projective_space_activity \\
12985 ➢ ➢ ➢ -cheat_sheet \\
12986 ➢ ➢ ➢ -end \\
12987 ➢ pdflatex PG_2_27.tex \\
12988 ➢ open PG_2_27.pdf \\
12989 \\
12990 \\
12991 \\
12992 \\
12993 \\
12994 \\
12995 \\
12996 \\
12997 ################################################################################################################
12998 # Section 11.8: Tactical Decompositions
12999 
13000 SECTION_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 \
$ $ (ORBITER) -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

# Chapter 12 - Finite Geometry

SECTION SPREADS:

desarguesian_spread_in_PG_3_2:
$ (ORBITER) -v 3 \
$ $ (ORBITER) -define FQ -finite_field -q 4 -end \$ 
$ $ (ORBITER) -define Fq -finite_field -q 2 -end \$ 
$ $ (ORBITER) -with FQ -and Fq -do -finite_field_activity \$ 

-des cheat_sheet_desarguesian_spread 2 -end
$ pdflatex Desarguesian_Spread_3_2.tex
$ open Desarguesian_Spread_3_2.pdf

desarguesian_spread_in_PG_5_2:
$ (ORBITER) -v 3 \
$ $ (ORBITER) -define FQ -finite_field -q 8 -end \$ 
$ $ (ORBITER) -define Fq -finite_field -q 2 -end \$ 
$ $ (ORBITER) -with FQ -and Fq -do -finite_field_activity \$ 

-des cheat_sheet_desarguesian_spread 2 -end
$ pdflatex Desarguesian_Spread_5_2.tex
$ open Desarguesian_Spread_5_2.pdf

757
desarguesian_spread_in_PG_3.4:
$\textsc{(Orbiter)} -v 3 \$
-define FQ -finite_field -q 16 -end 
-define Fq -finite_field -q 4 -end 
-with FQ -and Fq -do -finite_field_activity 
- cheat_sheet_desarguesian_spread 2 -end 
pdflatex Desarguesian_Spread_3.4.tex
open Desarguesian_Spread_3.4.pdf

desarguesian_spread_in_PG_3.5:
$\textsc{(Orbiter)} -v 3 \$
-define FQ -finite_field -q 25 -end 
-define Fq -finite_field -q 5 -end 
-with FQ -and Fq -do -finite_field_activity 
- cheat_sheet_desarguesian_spread 2 -end 
pdflatex Desarguesian_Spread_3.5.tex
open Desarguesian_Spread_3.5.pdf

spreads4:
- mkdir SPREADS_4
- rm live_points.txt
$\textsc{(Orbiter)} -v 10 \$
-define F -finite_field -q 2 -end 
-define P -projective_space 3 F -end 
-with P -do 
-projective_space_activity 
-spread_classify 2 
-problem_label spreads_2.2 -depth 5 
-draw_poset 
-end

spreads16.4:
$\textsc{(Orbiter)} -v 6 \$
-orbiter_path $\textsc{(Orbiter\_Path)} \$
-define F -finite_field -q 4 -end 
-define P -projective_space 3 F -end 
-with P -do 
-projective_space_activity 
-spread_classify 2 
-problem_label spreads_4.2 
-W -depth 17 -draw_poset 
-draw_options -radius 20 
-nodes_empty -line_width 0.2 -embedded 
-end 
-report \$

758
# Section 12.2: Translation planes

SECTION_TRANSLATION_PLANES:

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 -partition 6 91 91 -end

open TP9-0_incma_draw.bmp

pdflatex TP9-0_report.tex
open TP9-0_report.pdf

TP_9.1:
$(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 1 "TP9-1" \\
> -end

$(ORBITER) -v 2 -draw_matrix \\
> -input_csv_file TP9-1_incma.csv \\
> -box_width 6 -bit_depth 8 -partition 6 91 91 -end
open TP9-1_incma_draw.bmp
pdflatex TP9-1_report.tex
open TP9-1_report.pdf

TP_16.4:
$(ORBITER) -v 3 \\
> -define F -finite_field -q 4 -end \\
> -define PGGL4 -linear_group -PGGL 4 F -end \\
> -define PGGL5 -linear_group -PGGL 5 F -end \\
> -with PGGL4 -and PGGL5 -do \\
> -group_theoretic_activity \\
> -Andre_Bruck_Bose_construction 0 "TP16-4-HALL" \\
> -end
$(ORBITER) -v 2 -draw_matrix \\
> -input_csv_file TP16-4-HALL_incma.csv \\
> -box_width 6 -bit_depth 8 \\
> -partition 6 273 273 \\
> -end
open TP16-4-HALL_incma_draw.bmp
pdflatex TP16-4-HALL_report.tex
open TP16-4-HALL_report.pdf

#0 : "1200", // Hall spread
#1 : "81600", // Desarguesian spread
#2 : "576", // Semifield spread

TP_16.2.0:
$(ORBITER) -v 3 \\

760
-define F -finite_field -q 2 -end \
-define PGL8 -linear_group -PGL 8 F -end \
-define PGL9 -linear_group -PGL 9 F -end \
-with PGL8 -and PGL9 -do \
-group_theoretic_activity \
-Andre_Bruck_Bose_construction 0 "TP16-0-1008" \
-end \
pdflatex TP16-0-1008_report.tex 
open TP16-0-1008_report.pdf 

#0 : "1008", 
#1 : "1008", 
#2 : "1728", 
#3 : "216", 
#4 : "360", 
#5 : "288", 
#6 : "3600", 
#7 : "244800", 

# 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 3 F -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:
13229
13230  \text{BLT}_5:1:
13231 \texttt{\$\{ORBITER\} -v 2 \ \}
13232 \texttt{\> \> \> \text{-define F -finite_field -q 5 -end \}
13233 \texttt{\> \> \> \text{-define 0 -orthogonal_space 0 5 F -end \}
13234 \texttt{\> \> \> \text{-with 0 -do -orthogonal_space_activity \}
13235 \texttt{\> \> \> \> \> \text{-create_BLT_set -catalogue 1 -end \}
13236 \texttt{\> \> \> \> \> \text{-end}
13237 \texttt{\> \> \text{pdflatex catalogue_q5_iso1.tex}
13238 \texttt{\> \> \text{open catalogue_q5_iso1.pdf}
13239
13240  \text{BLT}_5\text{Linear:}
13241 \texttt{\> \$\{ORBITER\} -v 2 \ \}
13242 \texttt{\> \> \> \text{-define F -finite_field -q 5 -end \}
13243 \texttt{\> \> \> \text{-define 0 -orthogonal_space 0 5 F -end \}
13244 \texttt{\> \> \> \text{-with 0 -do -orthogonal_space_activity \}
13245 \texttt{\> \> \> \> \> \text{-create_BLT_set -family "Linear" -end \}
13246 \texttt{\> \> \> \> \> \text{-end}
13247 \texttt{\> \> \text{pdflatex BLT_Linear_q5.tex}
13248 \texttt{\> \> \text{open BLT_Linear_q5.pdf}
13249
13250  \text{BLT}_9\text{K1:}
13251 \texttt{\> \$\{ORBITER\} -v 2 \ \}
13252 \texttt{\> \> \> \text{-define F -finite_field -q 9 -end \}
13253 \texttt{\> \> \> \text{-define 0 -orthogonal_space 0 5 F -end \}
13254 \texttt{\> \> \> \text{-with 0 -do -orthogonal_space_activity \}
13255 \texttt{\> \> \> \> \> \text{-create_BLT_set -family "K1" -end \}
13256 \texttt{\> \> \> \> \> \text{-end}
13257 \texttt{\> \> \text{pdflatex BLT_K1_q9.tex}
13258 \texttt{\> \> \text{open BLT_K1_q9.pdf}
13259
13260
13261
13262  \text{BLT}_{11,0:}
13263 \texttt{\> \$\{ORBITER\} -v 2 \ \}
13264 \texttt{\> \> \> \text{-define F -finite_field -q 11 -end \}
13265 \texttt{\> \> \> \text{-define 0 -orthogonal_space 0 5 F -end \}
13266 \texttt{\> \> \> \text{-with 0 -do -orthogonal_space_activity \}
13267 \texttt{\> \> \> \> \> \text{-create_BLT_set -catalogue 0 -end \}
13268 \texttt{\> \> \> \> \> \text{-end}
13269 \texttt{\> \> \text{pdflatex 0_1_6_2_report.tex}
13270 \texttt{\> \> \text{open 0_1_6_2_report.pdf}
13271
13272
13273
13274  \text{BLT}_{11,\text{Fisher:}}
13275 \texttt{\> \$\{ORBITER\} -v 2 \ \}

762
-define F -finite_field -q 11 -end \\
-define O -orthogonal_space 0 5 F -end \\
-with 0 -do -orthogonal_space_activity \\
-create_BLT_set -family "Fisher" -end \\
-end

dfflatex BLT_Fisher_q11.tex
open BLT_Fisher_q11.pdf

BLT_11_Mondello:
$\text{ORBITER} -v 2$
-define F -finite_field -q 11 -end \\
-define O -orthogonal_space 0 5 F -end \\
-with 0 -do -orthogonal_space_activity \\
-create_BLT_set -family "Mondello" -end \\
-end

dfflatex BLT_Mondello_q11.tex
open BLT_Mondello_q11.pdf

BLT_13_FTWKB:
$\text{ORBITER} -v 2$
-define F -finite_field -q 11 -end \\
-define O -orthogonal_space 0 5 F -end \\
-with 0 -do -orthogonal_space_activity \\
-create_BLT_set -family "FTWKB" -end \\
-end

dfflatex BLT_FTWKB_q11.tex
open BLT_FTWKB_q11.pdf

# for K2, q must be congruent to 2 or 3 mod 5
BLT_13_K2:
$\text{ORBITER} -v 2$
-define F -finite_field -q 13 -end \\
-define O -orthogonal_space 0 5 F -end \\
-with 0 -do -orthogonal_space_activity \\
-create_BLT_set -family "Kantor2" -end \\
-end

dfflatex BLT_K2_q13.tex
open BLT_K2_q13.pdf

BLT_13_starter_5:
$\text{ORBITER} -v 2$
-define F -finite_field -q 13 -end \\
-define O -orthogonal_space 0 5 F -end \\
-with 0 -do -orthogonal_space_activity \\
# Section 13.1: Creating Graphs

SECTION CREATING GRAPHS:

Cycle_graph_13:
```
$ (ORBITER) -v 2 \
$ -define Gamma -graph \ 
$ -cycle 13 \ 
$ -end
```

make_triangle_graph:
```
$ echo $(TRIANGLE_GRAPH) >triangle_graph.csv 
$ $ (ORBITER) -v 6 \
$ $ -define G -graph \ 
$ $ -load_csv_no-border \ 
$ $ -triangle_graph.csv \ 
$ $ -end
```

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 \ 
```

triheiral_pair_graph:
```
$ (ORBITER) -v 2 \
```
# 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).
HJ315_orbital_graph_3:
-define gens -vector -file halljanko315.gens.csv -end
-define G -permutation_group
-bsgs halljanko315 "File\haliljanko315" 315 1209600 "0,1,2" 6 gens
-end
-define Gamma -graph
-orbital_graph G 3
-end

HJ_d2_graph:
-define G -graph
-load_csv_no_border halljanko315.csv
-distance_2
-end

cayley.Z11_1mod3:
-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
-end
-define Gamma -graph
-Cayley_graph G S
-end

cayley.Sym4_coxeter:
-define S -vector -dense "1,0,2,3, 0,2,1,3, 0,1,3,2" -end
-define G -permutation_group -symmetric_group 4
-end
-define Gamma -graph
-Cayley_graph G S
-end

cayley.Sym4_star:
-define S -vector -dense "1,0,2,3, 0,2,1,3, 0,1,3,2" -end
-define G -permutation_group -symmetric_group 4
-end
-define Gamma -graph
-Cayley_graph G S
-end

cayley.Sym4_cayley:
-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
-end
-define Gamma -graph
-Cayley_graph G S
-end
# Section 13.2: Graphs Theoretic Activities

SECTION_GRAPH_THEORETIC_ACTIVITIES:

triangle_graph_properties:
  echo $(TRIANGLE_GRAPH) >triangle_graph.csv
  $(ORBITER) -v 6 \
  -define G -graph \
  -load_csv_no_border \
  triangle_graph.csv \
  -end \
  -with G -do \
  -graph_theoretic_activity -properties \
  -end

Cycle_13_draw:
  $(ORBITER) -v 2 \
  -define Gamma -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

Cycle_9_eigenvalues:
Paley_13_draw:

```bash
$ (ORBITER) -v 2 \
   > -define Gamma -graph \n   >   > -cycle 9 \n   > -end \n   > -with Gamma -do \n   > -graph_theoretic_activity -eigenvalues -end
```

```bash
pdflatex Cycle_9_eigenvalues.tex
open Cycle_9_eigenvalues.pdf
```

Paley_13_eigenvalues:

```bash
$ (ORBITER) -v 2 \
   > -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
```

```bash
$ (ORBITER) -v 2 -draw_matrix \
   > -input_csv_file Paley_13.csv \
   > -box_width 20 -bit_depth 8 -partition 4 13 13 -end
```

```bash
dot -Tpng Paley_13.gv >Paley_13.png
open Paley_13.yaml
```

Paley_13_eigenvalues:

```bash
$ (ORBITER) -v 2 \
   > -define Gamma -graph \n   >   > -Paley 13 \n   > -end \n   > -with Gamma -do \n   > -graph_theoretic_activity -eigenvalues -end
```

```bash
pdflatex Paley_13_eigenvalues.tex
open Paley_13_eigenvalues.pdf
```

Cayley_Z11_1mod3_eigenvalues_and_draw:

```bash
$ (ORBITER) -v 2 \
   > -draw_options -xin 2000000 \n   > -yin 2000000 -embedded -radius 20000 -end \n   > -define F -finite_field -q 11 -end \n   > -define S -vector -dense \n   > "1,1, 1,4, 1,7, 1,10" -end \n   > -define G -linear_group -AGL 1 F \n   > -subgroup_by_generators "Z11" 11 1 "1,1" \n   > -end \n   > -define Gamma -graph \n   > -Cayley_graph G S \n```

769
Cayley_sym4_coxeter_draw:

\$(ORBITER) -v 2 \\
\$draw_options -xin 2000000 -yin 2000000 \\
$draw -radius 20000 -embedded -nodes_empty -end \\
$define S -vector -dense \\
$define G -permutation_group -symmetric_group 4 \\
$define Gamma -graph \\
$end \\
$with Gamma -do \\
$graph_theoretic_activity -draw -end \\
pdflatex Cayley_graph_AGL_1_11_draw.tex \\
open Cayley_graph_AGL_1_11_draw.pdf \\

Cayley_sym5_coxeter_draw:

\$(ORBITER) -v 2 \\
$draw_options -xin 1000000 -yin 1000000 \\
$embedded -radius 10000 -nodes_empty -end \\
$define S -vector -dense \\
"1,0,2,3, 0,2,1,3, 0,1,3,2" -end \\
$define G -permutation_group -symmetric_group 5 \\
$end \\
$with Gamma -do \\
$graph_theoretic_activity -draw -end \\
pdflatex Cayley_graph_Perm4_draw.tex \\
open Cayley_graph_Perm4_draw.pdf \\

Cayley_sym4_star_eigenvalues_and_draw:

\$(ORBITER) -v 2 \\
$draw_options -xin 1000000 -yin 1000000 -embedded -end \\
$define S -vector -dense "1,0,2,3, 2,1,0,3, 3,1,2,0" -end \\
$define G -permutation_group -symmetric_group 4 \\

13650 ▶ ▶ -end \n13651 ▶ ▶ -define Gamma -graph \n13652 ▶ ▶ ▶ -Cayley_graph G S \n13653 ▶ ▶ -end \n13654 ▶ ▶ -with Gamma -do \n13655 ▶ ▶ -graph_theoretic_activity -eigenvalues -end \n13656 ▶ ▶ -with Gamma -do \n13657 ▶ ▶ -graph_theoretic_activity -draw -end
13658 ▶ pdflatex Cayley_graph_Perm4_draw.tex
13659 ▶ open Cayley_graph_Perm4_draw.pdf
13660 ▶ pdflatex Cayley_graph_Perm4_eigenvalues.tex
13661 ▶ open Cayley_graph_Perm4_eigenvalues.pdf
13662
13663
13664
13665
13666 graph_v5_e7.colored_graph:
13667 ▶ $(ORBITER) -v 2 \n13668 ▶ ▶ -define G -graph -edges_as_pairs 5 \n13669 ▶ ▶ ▶ "0,1,0,2,0,3,0,4,1,3,1,4,2,4" \n13670 ▶ ▶ -end \n13671 ▶ ▶ -with G -do \n13672 ▶ ▶ -graph_theoretic_activity -save -end
13673
13674
13675
13676 small_graph_draw:
13677 ▶ $(ORBITER) -v 2 \n13678 ▶ ▶ -define G -graph -edges_as_pairs 5 \n13679 ▶ ▶ ▶ "0,1,0,2,0,3,0,4,1,3,1,4,2,4" \n13680 ▶ ▶ -end \n13681 ▶ ▶ -with G -do \n13682 ▶ ▶ -graph_theoretic_activity -export_csv -end \n13683 ▶ ▶ -with G -do \n13684 ▶ ▶ -graph_theoretic_activity -export_graphviz -end \n13685 ▶ ▶ -with G -do \n13686 ▶ ▶ -graph_theoretic_activity -save -end
13687 ▶ $(ORBITER) -v 2 -draw_matrix \n13688 ▶ ▶ -input_csv_file graph_v5_e7.csv \n13689 ▶ ▶ -box_width 40 -bit_depth 24 \n13690 ▶ ▶ -partition 4 "1,1,1,1" "1,1,1,1" -end
13691 ▶ dot -Tpng graph_v5_e7.gv >graph_v5_e7.png
13692
13693 # creates graph_v5_e7.csv
13694 # creates graph_v5_e7.colored_graph
13695
13696
13697 petersen_draw:
13698 $(ORBITER) -v 2 \ 
13699 define G -graph -Johnson 5 2 0 -end \ 
13700 with G -do \ 
13701 graph_theoretic_activity -export_csv -end \ 
13702 with G -do \ 
13703 graph_theoretic_activity -export_graphviz -end \ 
13704 with G -do \ 
13705 graph_theoretic_activity -save -end
13706 $(ORBITER) -v 2 -draw_matrix \ 
13707 -input_csv_file Johnson_5_2_0.csv \ 
13708 -box_width 40 -bit_depth 24 -partition 4 "10" "10" -end
13709 dot -Tpng Johnson_5_2_0.gv >Johnson_5_2_0.png
13710
13711
13712 Johnson_6_2_0_draw:
13713 $(ORBITER) -v 2 \ 
13714 define G -graph -Johnson 6 2 0 -end \ 
13715 with G -do \ 
13716 graph_theoretic_activity -export_csv -end \ 
13717 with G -do \ 
13718 graph_theoretic_activity -export_graphviz -end \ 
13719 with G -do \ 
13720 graph_theoretic_activity -save -end
13721 $(ORBITER) -v 2 -draw_matrix \ 
13722 -input_csv_file Johnson_6_2_0.csv \ 
13723 -box_width 40 -bit_depth 24 -partition 4 "10" "10" -end
13724 dot -Tpng Johnson_6_2_0.gv >Johnson_6_2_0.png
13725
13726
13727
13728 Hamming_graph_3_draw:
13729 $(ORBITER) -v 2 \ 
13730 define G -graph -Hamming 3 2 -end \ 
13731 with G -do \ 
13732 graph_theoretic_activity -export_csv -end \ 
13733 with G -do \ 
13734 graph_theoretic_activity -export_graphviz -end \ 
13735 with G -do \ 
13736 graph_theoretic_activity -save -end
13737 $(ORBITER) -v 2 -draw_matrix \ 
13738 -input_csv_file Hamming_3_2.csv \ 
13739 -box_width 40 -bit_depth 24 \ 
13740 -partition 4 "1,1,1,1,1,1,1,1" "1,1,1,1,1,1,1,1" -end
13741 dot -Tpng Hamming_3_2.gv >Hamming_3_2.png
13742
13743

772
Hamming_graph_7_draw:

13744 $(ORBITER) -v 2 \$
13745 \> -define G -graph -Hamming 7 2 -end \$
13746 \> \> -with G -do \$
13747 \> \> -graph_theoretic_activity -export_csv -end \$
13748 \> \> -with G -do \$
13749 \> \> -graph_theoretic_activity -export_graphviz -end \$
13750 \> \> -with G -do \$
13751 \> \> -graph_theoretic_activity -save -end
13752 \> $(ORBITER) -v 2 -draw_matrix \$
13753 \> -input_csv_file Hamming_7_2.csv \$
13754 \> \> -box_width 8 -bit_depth 24 -partition 4 128 128 -end
13755 \> \> dot -Tpng Hamming_7_2.gv >Hamming_7_2.png
13756
13757
13758
13759
13760
13761
13762
13763 Chain_232_properties:
13764 \> $(ORBITER) -v 2 \$
13765 \> \> -define P1 -vector -dense 2,3,2 -end \$
13766 \> \> -define P2 -vector -dense 2,3,2 -end \$
13767 \> \> -define Gamma -graph \$
13768 \> \> \> -chain_graph P1 P2 \$
13769 \> \> \> -end \$
13770 \> \> \> -with Gamma -do \$
13771 \> \> \> \> -graph_theoretic_activity -export_csv \$
13772 \> \> \> \> -end \$
13773 \> \> \> \> -with Gamma -do \$
13774 \> \> \> \> -graph_theoretic_activity -properties \$
13775 \> \> \> \> -end
13776
13777 Chain_232_eigen:
13778 \> $(ORBITER) -v 2 \$
13779 \> \> -define P1 -vector -dense 2,3,2 -end \$
13780 \> \> -define P2 -vector -dense 2,3,2 -end \$
13781 \> \> -define Gamma -graph \$
13782 \> \> \> -chain_graph P1 P2 \$
13783 \> \> \> -end \$
13784 \> \> \> -with Gamma -do \$
13785 \> \> \> \> -graph_theoretic_activity \$
13786 \> \> \> \> -eigenvalues \$
13787 \> \> \> \> -end
13788 \> pdflatex chain_graph_eigenvalues.tex
13789 \> open chain_graph_eigenvalues.pdf
13790
# need the file halljanko315.csv

HJ properties:
$(ORBITER) -v 6 \
-define G -graph \
-load_csv_no_border \
halljanko315.csv \
-end \
-with G -do \
-graph_theoretic_activity -properties \
-end

#Degree type: (10^{315} )

HJ.d2.properties:
$(ORBITER) -v 6 \
-define G -graph \
-load_csv_no_border \
halljanko315.csv \
-distance_2 \
-end \
-with G -do \
-graph_theoretic_activity \
-properties \
-end

#Degree type: (80^{315} )

PGO_5.2_collinearity_graph: 0_5.2_incidence_matrix.csv
$(ORBITER) -v 3 \
-define Inc -vector -file 0_5.2_incidence_matrix.csv -end \
-define Gamma -graph -collinearity_graph Inc -end \
-with Gamma -do \
-graph_theoretic_activity \
-properties \
-end

trihedral.pair_graph_draw:
Section 13.3: Graph Theory: Classification

SECTION_GRAPH THEORY CLASSIFICATION:

graph_classify_5:

$($ORBITER) \ -v 2 \ -define Gamma \ 
- graph - trihedral_pair_disjointness_graph - end \\
- with Gamma - do \\
- graph_theoretic_activity - export.csv - end \\
$($ORBITER) \ -v 2 \ - draw_matrix \\
- input_csv_file trihedral_pair_disjointness.csv \\
- box_width 20 - depth 8 - end \\
open trihedral_pair_disjointness draw.bmp \\
- end \\
- end \\
- with GC - do \\
- 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 \\
pdflatex graphs_v5_level_5_reps.tex
13885 ▷ open_graphs_v5_level_5_reps.pdf
13886 ▷ pdflatex graphs_v5_poset.tex
13887 ▷ open_graphs_v5_poset.pdf
13888
13889
13890 tournament_classify_4:
13891 ▷ $(ORBITER) -v 2 \
13892 ▷ ▷ -define GC -graph_classification \n13893 ▷ ▷ ▷ -n 4 -tournament \n13894 ▷ ▷ ▷ -poset_classification_control \n13895 ▷ ▷ ▷ ▷ -problem_label tournament_4 -depth 6 -draw_poset \n13896 ▷ ▷ ▷ ▷ ▷ -draw_options -radius 250 -embedded -end \n13897 ▷ ▷ ▷ ▷ -end \n13898 ▷ ▷ -end \n13899 ▷ ▷ -with GC -do \n13900 ▷ ▷ -graph_classification_activity \n13901 ▷ ▷ ▷ -draw_options -radius 400 \n13902 ▷ ▷ ▷ ▷ -line_width 2 -scale 0.10 -end \n13903 ▷ ▷ ▷ ▷ -draw_graphs_at_level 6 \n13904 ▷ ▷ ▷ -end \n13905 ▷ ▷ -print_symbols
13906 ▷ pdflatex tournament_4_level_6_reps.tex
13907 ▷ open tournament_4_level_6_reps.pdf
13908 ▷
13909
13910
13911
13912
13913 graph_classify_8_r3:
13914 ▷ $(ORBITER) -v 3 \
13915 ▷ ▷ -define GC -graph_classification \n13916 ▷ ▷ ▷ -n 8 -regular 3 \n13917 ▷ ▷ ▷ -poset_classification_control \n13918 ▷ ▷ ▷ ▷ -problem_label graphs_v8_r3 -depth 12 -draw_poset \n13919 ▷ ▷ ▷ ▷ ▷ -draw_options -radius 250 \n13920 ▷ ▷ ▷ ▷ ▷ ▷ -line_width 0.2 -embedded -end \n13921 ▷ ▷ ▷ ▷ ▷ -end \n13922 ▷ ▷ ▷ -end \n13923 ▷ ▷ -with GC -do \n13924 ▷ ▷ -graph_classification_activity \n13925 ▷ ▷ ▷ -draw_options -radius 400 \n13926 ▷ ▷ ▷ ▷ -line_width 2 -scale 0.10 -end \n13927 ▷ ▷ ▷ ▷ -draw_graphs_at_level 12 \n13928 ▷ ▷ ▷ -end \n13929 ▷ ▷ -print_symbols
13930 ▷ #pdflatex graphs_v8_r3_poset_lvl_12.tex
13931 ▷ #open graphs_v8_r3_poset_lvl_12.pdf

776
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 \
$\$ -print_symbols 

Symmetric 5 inversion graph recognize:
$\$ (ORBITER) -v 10 \
$\$ -define G -permutation_group -symmetric_group 5 -end \
$\$ -with G -do \
$\$ -group_theoretic_activity \
$\$ -export_inversion_graphs "Symmetric5_inversion_graphs.csv" \
$\$ -end \
$\$ $(ORBITER) -v 2 \
$\$ -orbiter_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 \
$\$ -with GC -do \
$\$ -graph_classification_activity \
$\$ -recognize_graphs_from_adjacency_matrix_csv Symmetric5_inversion_graphs.csv
\begin{verbatim}
13978 \end{verbatim}
13979 \end{verbatim}
13980 \end{verbatim}
13981 \end{verbatim}
13982 \end{verbatim}
13983 \end{verbatim}
13984 \end{verbatim}
13985 \end{verbatim}
13986 \end{verbatim}
13987 \end{verbatim}
13988 \end{verbatim}
13989 \end{verbatim}
13990 small_graph_cliques: graph_v5.e7.colored_graph
13991 \end{verbatim}
13992 \end{verbatim}
13993 \end{verbatim}
13994 \end{verbatim}
13995 \end{verbatim}
13996 \end{verbatim}
13997 \end{verbatim}
13998 # nb_sol = 3
13999
14000 small_graph_cliques_Sajeeb:
14001 \end{verbatim}
14002 \end{verbatim}
14003 \end{verbatim}
14004 \end{verbatim}
14005 \end{verbatim}
14006 \end{verbatim}
14007 \end{verbatim}
14008 # nb_sol = 3
14009
14010 Paley_13.aut:
14011 \end{verbatim}
14012 \end{verbatim}
14013 \end{verbatim}
14014 \end{verbatim}
14015 \end{verbatim}
14016 \end{verbatim}
14017 \end{verbatim}
14018 \end{verbatim}
14019 \end{verbatim}
14020 \end{verbatim}
14021
14022
14023
778
Paley_13:

$\{(ORBITER) -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 \$

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 \$

Paley_13_cliques_all:

$\{(ORBITER) -v 10 \$
$> > -define Gamma -graph -Paley 13 -end \$
$> > -with Gamma -do \$
$> > -graph_theoretic_activity \$
$> > > -find_cliques -target_size 3 \$
$> > > -end \$

PGO_5_2_cliques: 0_5_2_incidence_matrix.csv

$\{(ORBITER) -v 3 \$
$> > -define Inc -vector -file 0_5_2_incidence_matrix.csv -end \$
$> > -define Gamma -graph -collinearity_graph Inc -end \$
$> > -with Gamma -do \$
$> > -graph_theoretic_activity \$

#User time: 0.01 of a second, dt=1 tps = 100

PGO_5_2_cliques: 0_5_2_incidence_matrix.csv
14071 > > > -find_cliques -target_size 3 -end \
14072 > > -end
14073
14074
14075 HJ_d2_c5:
14076 > $(ORBITER) -v 6 \
14077 > > -define G -graph \
14078 > > > -load_csv_no_border \
14079 > > > halljanko315.csv \
14080 > > > -distance_2 \
14081 > > -end \
14082 > > -with G -do \
14083 > > -graph_theoretic_activity \
14084 > > > -find_cliques -target_size 5 -end \
14085 > > -end
14086
14087
14088
14089 #graph_theoretic_activity::perform_activity Gr->label=halljanko315 nb_sol = 26208 0
14090
14091
14092 HJ64_cliques5:
14093 > $(ORBITER) -v 6 \
14094 > > -define Gamma -graph \
14095 > > > -load \
14096 > > > Group_perm315_orbital_3.colored_graph \
14097 > > -end \
14098 > > -with Gamma -do \
14099 > > -graph_theoretic_activity \
14100 > > > -find_cliques -target_size 5 -end \
14101 > > -end
14102
14103 #graph_theoretic_activity::perform_activity Gr->label=Group_perm315_orbital_3 nb_sol = 1008
14104 #Group_perm315_orbital_3_sol.csv
14105
14106
14107
14108 HJ64_cliques5_classify:
14109 > $(ORBITER) -v 6 \
14110 > > -define Gamma -graph \
14111 > > > -load \
14112 > > > Group_perm315_orbital_3.colored_graph \
14113 > > -end \
14114 > > -define gens -vector \
14115 > > > -file halljanko315_gens.csv \}
# Chapter 14 - Combinatorial Objects

SECTION COMBINATORIAL OBJECTS:

Hirschfeld $q_4$ from set:

```bash
$(ORBITER) -v 4 \n
$\text{define } H \text{ -set -here} \n
$\text{define } C \text{ -combinatorial_objects} \n```

```plaintext
#HJ64_cliques_reps_lvl_5.csv

# 1 orbit
#ROW,REP,AGO,OL
#0,"0,8,31,110,283",1200,1008
#END
```

HEXEND
hyperoval_16_create:

-define C -combinatorial_objects \\
-set_of_points $(HYPEROVAL_16_16320) \ 
-set_of_points $(HYPEROVAL_16_144) \ 
-end \\

EC_read: elliptic_curve_b1_c3_q11.txt

-define C -combinatorial_objects \\
-file_of_points elliptic_curve_b1_c3_q11.txt \\
-end \\

PG_3_5_assume_31_read:

-define C -combinatorial_objects \\
-file_of_packings_through_spread_table \\
-H31_packings.csv \\
-SPREAD_TABLES_5_REG/spread_25_spreads.csv \\
-5 \\
-end \\

LS_AG_2_3_read:

-define C -combinatorial_objects \\
-file_of_designs \\
-solutions.csv 9 84 3 12 \\
-end \\

geo_7_3_read:

-file_of_incidence_geometries \
Desargues_path_lex_least_read:
  echo $(DESARGUES_PATH_LEX_LEAST) >Desargues_path_lex_least.inc
$(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

geo_pasch_read:
  $(ORBITER) -v 10 \
  -define C -combinatorial_objects \
  -file_of_incidence_geometries \
  pasch.inc 6 4 12 \
- end

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

# Section 14.2: File Formats

geo_pasch_read:
  $(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:

SECTION OBJECTS_IN_PROJECTIVE_SPACE:

EC_canon: elliptic_curve_b1_c3_q11.txt

$($ORBITER) -v 40 \ 
$define C -combinatorial_objects \ 
$file_of_points elliptic_curve_b1_c3_q11.txt \ 
$end \ 
$define F -finite_field -q 11 -end \ 
$define P -projective_space 2 F -end \ 
$with C -do \ 
$combinatorial_object_activity \ 
$canonical_form PG P \ 
$classification_prefix EC \ 
$label EC \ 
$save ago \ 
$max TDO_depth 4 \ 
$end \ 
$report \ 
$prefix EC \ 
$export_flag_orbits \ 
$show_TDO \ 
$show_TDA \ 
$dont_show_incidences_matrices \ 
$export_group \ 
$end \ 
$pdflatex EC_classification.tex

open EC_classification.pdf

$($ORBITER) -v 2 -draw_matrix \ 
$input_csv_file EC_object0_TDA_flag_orbits.csv \ 
$secondary_input_csv_file EC_object0_TDA.csv \ 
$box_width 20 -bit_depth 24 \ 
$end

open EC_object0_TDA_flag_orbits_draw.bmp
Hirschfeld_q4.c: Hirschfeld_surface_q4.txt

$ (ORBITER) -v 6 \n$-define C -combinatorial_objects \n-file_of_points Hirschfeld_surface_q4.txt \n-end \n$-define F -finite_field -q 4 -end \n$-define P -projective_space 3 F -end \n-with C -do \n$-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
pdflatex Hirschfeld_surface_q4_classification.tex
open Hirschfeld_surface_q4_classification.pdf

# group order is 51840

Hirschfeld_q4_set_c:

$ (ORBITER) -v 4 \n$-define H -set -here \n$(HIRSCHFELD_SURFACE_Q4_SET_OF_POINTS) \n-end \n$-define C -combinatorial_objects \n-set_of_points H \n-end \n$-define F -finite_field -q 4 -end \n$-define P -projective_space 3 F -end \n-with C -do \n$-combinatorial_object_activity \n$-canonical_form_PG P \n$-classification_prefix Hirschfeld_surface_q4 \n
Dickson_sets_stabilizer:

\$(\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 "0,1,2,5,6,7,8" \\
-set_of_points "0,1,2,3,5,6,7" \\
-set_of_points "0,1,2,3,5,6,9" \\
-set_of_points "0,1,2,3,5,6,10" \\
-set_of_points "0,1,2,3,5,6,4" \\
-set_of_points "0,1,2,3,8,11,13" \\
-set_of_points "3,6,9,7,10,12,8,11,13,14,4" \\
-set_of_points "3,5,6,9,7,10,12,11,13,14,4" \\
-set_of_points "0,1,2,3,5,6,9,7,10,12,4" \\
-end \\
-define F -finite_field -q 2 -end \\
-define P -projective_space 3 F -end \\
-with C -do \\
-combinatorial_object_activity \\
-canonical_form_PG P \\
-classification_prefix Dickson_sets \\
-save_ago \\
-end \\
-report \\
-end \\
pdflatex Dickson_sets_classification.tex \\
open Dickson_sets_classification.pdf \\

Endrass_7c: Endrass_F7.txt 

\$(\text{ORBITER}) -v 2 \$

-define C -combinatorial_objects \\
-file_of_points Endrass_F7.txt \\
-end \\
-define F -finite_field -q 7 -end \\
-define P -projective_space 3 F -end \\
-with C -do \\

786
hyperoval_16_canonical_form:
$\text{ORBITER} -v 2 \$

-define C -combinatorial_objects 
-set_of_points $(HYPEROVAL_16_16320) 
-set_of_points $(HYPEROVAL_16_164) 
-end 

-define F -finite_field -q 16 -end 
-define P -projective_space 2 F -end 
-with C -do 
-define P -combinatorial_object_activity 
-canonical_form PG P 
-classification_prefix hyperoval_q16 
-label hyperoval_q16 
-save ago 
-save_transversal 
-max_TDO_depth 10 
-end 
-report 
-prefix hyperoval_q16 
-export_flag_orbits 
-show_TDO 
-show_TDA 
-dont_show_incidence_matrices 
-export_group 
-end 

pdflatex hyperoval_q16_classification.tex
open hyperoval_q16_classification.pdf
$(ORBITER) -v 2 -draw_matrix \
-input_csv_file hyperoval_q16_object1_TDA_flag_orbits.csv \
-box_width 4 -bit_depth 24 \
-end

$(ORBITER) -v 6 \
-define C -combinatorial_objects \n-set_of_points "2,3,28,46,51,61,40,71" \n-end \
-define F -finite_field -q 8 -end \
-define P -projective_space 2 F -end \
-with C -do \
-combinatorial_object_activity \
-canonical_form_PG P \
-classification_prefix cc_8 \
-save_ago \
-max_TDO_depth 10 \
-end \
-report \
-end

pdflatex cc_8_classification.tex
open cc_8_classification.pdf

F_alpha_beta_gamma_delta_classify_q7.nauty: F_alpha_beta_gamma_delta_q7_points.txt

ds Oriental_Airlines
#4:38

#User time: 4:12 on Mac

6 orbits

ovoid_q8.canon: ovoid_q8.txt

$ (ORBITER) -v 6 \n
define C -combinatorial_objects \n
-define C -combinatorial_objects -file_of_points ovoid_q8.txt \n
-end \n
-define F -finite_field -q 8 -end \n
-define P -projective_space 3 F -end \n
-define F -finite_field -q 8 -end \n
-define P -projective_space 3 F -end \n
-define C -combinatorial_objects -file_of_points ovoid_q8.txt \n
-end \n
-canonical_form_PG P \n
-classification_prefix ovoid \n
-label ovoid \n
-save_ago \n
-max_TDO_depth 4 \n
-end \n
-report \n
-prefix ovoid \n
-show_TDO \n
-show_TDA \n
-dont_show_incidence_matrices \n
-export_group \n
-end \n
-report \n
-prefix ovoid \n
-export_flag_orbits \n
-show_TDO \n
-show_TDA \n
-dont_show_incidence_matrices \n
-export_group \n

14538 #> > > -end \n14539
14540
14541 ovoid_ST.q8_canon: ovoid_ST.q8.txt
14542 > $(ORBITER) -v 6 \n14543 > > -define C -combinatorial_objects \n14544 > > > -file_of_points ovoid_ST.q8.txt \n14545 > > > -end \n14546 > > > > -define F -finite_field -q 8 -end \n14547 > > > > -define P -projective_space 3 F -end \n14548 > > > > -with C -do \n14549 > > > > -combinatorial_object_activity \n14550 > > > > > -canonical_form_PG P \n14551 > > > > > > -classification_prefix ovoid_ST \n14552 > > > > > > > -label ovoid_ST \n14553 > > > > > > > > -save_ago \n14554 > > > > > > > > > -max_TDO_depth 4 \n14555 > > > > > > > > > > -end \n14556 > > > > > > > > > > > -report \n14557 > > > > > > > > > > > > -prefix ovoid_ST \n14558 > > > > > > > > > > > > > -show_TDO \n14559 > > > > > > > > > > > > > > -show_TDA \n14560 > > > > > > > > > > > > > > > -dont_show_incidence_matrices \n14561 > > > > > > > > > > > > > > > > -export_group \n14562 > > > > > > > > > > > > > > > > > -end \n14563 > > > > > > > > > > > > > > > > > > -end
14564 > pdflatex ovoid_ST_classification.tex
14565 > open ovoid_ST_classification.pdf
14566
14567 # group order 87360 = 3 * 29120
14568 SUZUKI_8_GENERATORS="\n14569 1,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,1,1, \n14570 1,0,0,0,0,6,0,0,0,0,2,0,0,0,0,3,0, \n14571 1,0,0,0,1,1,1,0,0,0,1,0,1,0,0,1,0,1,0,0,1,0,1,0,1,0,1,0,2"
14572 0,1,0,0,1,0,0,0,0,0,0,1,0,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,2"
14573
14574
14575 Suzuki_8:
14576 > $(ORBITER) -v 6 \n14577 > > -define F -finite_field -q 8 -end \n14578 > > -define gens -vector -field F \n14579 > > > -compact $(SUZUKI_8_GENERATORS) -end \n14580 > > > -define G -linear_group -PGGL 4 8 \n14581 > > > > -subgroup_by_generators "Sz8" "87360" 5 gens \n14582 > > > > > -end \n14583 > > > > > > -with G -do \n14584 > > > > > > -group_theoretic_activity \n
790
# Section 15.3: Incidence Geometries

SECTION INCIDENCE GEOMETRIES:

geo_7_3.c:

```
geo_7_3.c:
```
geo_10_3.c:

```c
$(ORBITER) -v 10
-define Test_lines -set -loop 4 11 1 -end
-define C -combinatorial_objects
-file_of_incidence_geometries 10.3.inc 10 10 30
-end
-define C -do
-define C -combinatorial_object_activity
-define C -canonical_form
-define C -classification_prefix 10.3
-label 10.3
-save_ago
-save_transversal
-end
-report
-prefix 10.3
-export_flag_orbits
-show_incidence_matrices
-export_group
-end
-end
```

```
$(ORBITER) -v 2 -draw_matrix
-input_csv_file 10.3.object7.TDA_flag_orbits.csv
-secondary_input_csv_file 10.3.object7.TDA.csv
-box_width 16 -bit_depth 24
-end
 $(ORBITER) -v 2 -draw_matrix
-input_csv_file 10.3.object7.INP_flag_orbits.csv
-secondary_input_csv_file 10.3.object7.INP.csv
-box_width 16 -bit_depth 24
-end
```

```
pdflatex 10_3_classification.tex
open 10_3_classification.pdf
```
14679 geo_10_3_c_lex_least:
14680 $(ORBITER) -v 10 \
14681 -draw.incidence.structure.description \  
14682 -width 60 -with_10 6 -end \ 
14683 -define Test_lines -set -loop 4 11 1 -end \ 
14684 -define Geo -geometry_builder \ 
14685 -V 10 -B 10 -TDO 3 -fuse 1 \ 
14686 -define Geo -fname GEO 10_3 \ 
14687 -test Test_lines \ 
14688 -end \ 
14689 -define C -combinatorial_objects \ 
14690 -file_of_incidence_geometries 10_3.inc 10 10 30 \ 
14691 -end \ 
14692 -with C -do \ 
14693 -combinatorial.object.activity \ 
14694 -canonical_form \ 
14695 -classification_prefix 10_3 \ 
14696 -label 10_3 \ 
14697 -save_ag \ 
14698 -save_transversal \ 
14699 -end \ 
14700 -report \ 
14701 -prefix 10_3 \ 
14702 -export.flag.orbits \ 
14703 -show.incidence.matrices \ 
14704 -export.group \ 
14705 -show.TDO \ 
14706 -show.TDA \ 
14707 -lex.least Geo \ 
14708 -end \ 
14709 -end
14710 pdfflatex 10_3_classification.pdf
14711 open 10_3_classification.pdf
14712 $(ORBITER) -v 2 -draw.matrix \
14713 -input.csv_file 10_3.object7.TDA_flag.orbits.csv \
14714 -secondary.input.csv_file 10_3.object7.TDA.csv \
14715 -box.width 16 -bit.depth 24 \ 
14716 -end \ 
14717 $(ORBITER) -v 2 -draw.matrix \
14718 -input.csv_file 10_3.object7.INP_flag.orbits.csv \
14719 -secondary.input.csv_file 10_3.object7.INP.csv \
14720 -box.width 16 -bit.depth 24 \
14721 -end
14722 #10_3_object7.TDA_flag.orbits.csv
14724 #0 1 2 10 13 14 20 25 26 31 33 35 41 44 46 52 53 56 62 67 68 74 77 79 85 88 89
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 \\
-canoncal.form \\
-classification.prefix 14_3 \\
-label 14_3 \\
-save_ago \\
-save_transversal \\
-end \\
-end

#-
-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 \\
-canoncal.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 \
  -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

g4_40.g4.c:
  $(ORBITER) -v 2 \
  -draw_incidence_structure_description \
  -width 50 -with_10 5 -end \
  -define C -combinatorial_objects \
  -file_of_incidence_geometries 40_4_g4.inc 40 40 160 \
  -end \
  -end
  -with C -do \
  -combinatorial_object_activity \
  -canonical_form \
  -classification_prefix 40_4_g4 \
  -label 40_4_g4 \
  -save_ago \
  -end \
  -report \
  -prefix 40_4_g4 \
  -end
  -end

795
\begin{verbatim}
14820  \texttt{-export_flag_orbits \}
14821  \texttt{-show_TDO \}
14822  \texttt{-show_TDA \}
14823  \texttt{-show_incidence_matrices \}
14824  \texttt{-end \}
14825  \texttt{-end \}
14826  \texttt{pdflatex 40_4_g4_classification.tex \}
14827  \texttt{open 40_4_g4_classification.pdf \}
14828  \texttt{geo_17_3_g4.c: \}
14829  \texttt{$(ORBITER) -v 2 \}
14830  \texttt{-draw_incidence_structure_description \}
14831  \texttt{-width 50 -with_10 5 -end \}
14832  \texttt{-define C -combinatorial_objects \}
14833  \texttt{-file_of_incidence_geometries 17_3_g4.inc 17 17 51 \}
14834  \texttt{-end \}
14835  \texttt{-with C -do \}
14836  \texttt{-combinatorial_object_activity \}
14837  \texttt{-canonical_form \}
14838  \texttt{-classification_prefix 17_3_g4 \}
14839  \texttt{-label 17_3_g4 \}
14840  \texttt{-save_ago \}
14841  \texttt{-end \}
14842  \texttt{-report \}
14843  \texttt{-prefix 17_3_g4 \}
14844  \texttt{-export_flag_orbits \}
14845  \texttt{-show_TDO \}
14846  \texttt{-show_TDA \}
14847  \texttt{-show_incidence_matrices \}
14848  \texttt{-end \}
14849  \texttt{-end \}
14850  \texttt{pdflatex 17_3_g4_classification.tex \}
14851  \texttt{open 17_3_g4_classification.pdf \}
14852  \texttt{AG_2_3_c: AG_2_3.inc \}
14853  \texttt{$(ORBITER) -v 2 \}
14854  \texttt{-define C -combinatorial_objects \}
14855  \texttt{-file_of_incidence_geometries \}
14856  \texttt{AG_2_3.inc 9 12 36 \}
14857  \texttt{-end \}
14858  \texttt{-with C -do \}
14859  \texttt{-combinatorial_object_activity \}
14860  \texttt{-canonical_form \}
14861  \texttt{-classification_prefix AG_2_3 \}
14862  \texttt{-label AG_2_3 \}
14863  \texttt{-save_ago \}
\end{verbatim}
\begin{verbatim}
pdflatex AG_2_3.classification.tex
open AG_2_3.classification.pdf
$\texttt{(ORBITER)} -v 2 -draw_matrix \backslash
\texttt{input.csv_file AG_2_3.object0.INP_flag_orbits.csv} \backslash
\texttt{secondary_input.csv_file AG_2_3.object0.INP.csv} \backslash
\texttt{box_width 40 -bit_depth 24} \backslash
\texttt{end} \backslash
open AG_2_3.object0.INP_flag_orbits.draw.bmp
geo_LSQ6.c:
$\texttt{(ORBITER)} -v 10 \backslash
\texttt{draw_incidence_structure_description} \backslash
\texttt{width 60 -with_10 6 -end} \backslash
\texttt{define C -combinatorial_objects} \backslash
\texttt{file_of_incidence_geometries} \backslash
\texttt{LSQ6.inc 18 39 126} \backslash
\texttt{end} \backslash
\texttt{with C -do} \backslash
\texttt{combinatorial_object_activity} \backslash
\texttt{canonical_form} \backslash
\texttt{classification_prefix LSQ6} \backslash
\texttt{label LSQ6} \backslash
\texttt{save_ago} \backslash
\texttt{save_transversal} \backslash
\texttt{end} \backslash
\texttt{report} \backslash
\texttt{prefix LSQ6} \backslash
\texttt{export_flag_orbits} \backslash
\texttt{show_incidence_matrices} \backslash
\texttt{end} \backslash
\texttt{report} \backslash
\texttt{prefix LSQ6} \backslash
\texttt{export_flag_orbits} \backslash
\texttt{show_incidence_matrices} \backslash
\texttt{end} \backslash
pdflatex LSQ6.classification.tex
#open LSQ6.classification.pdf
\end{verbatim}
$(ORBITER) -v 2 -draw 
\input{csv_file LSQ6_object0_TDA_flag_orbits.csv} 
\input{csv_file LSQ6_object0_TDA_flag_orbits.csv} 
\box{width 32 -bit_depth 24} 
\end 
$(ORBITER) -v 2 -draw 
\input{csv_file LSQ6_object0_INP_flag_orbits.csv} 
\input{csv_file LSQ6_object0_INP_flag_orbits.csv} 
\box{width 32 -bit_depth 24} 
\end 
open LSQ6_object0_INP_flag_orbits.draw.bmp 

# ToDo:

quartic_curve_25_0_0_canonical: 
$(ORBITER) -v 3 
\define F -finite_field -q 25 -end 
\define P -projective_space 2 F -end 
\with P \do 
\projective_space_activity 
\canonical_form_PG 
\input 
\set_of_points "10,11,59,63,124,135,136,170,206,257,275,284,285,367,378,393,433,619,641,644" 
\set_of_points "9, 24, 62, 67, 77, 84, 87, 89, 125, 130, 158, 172, 197, 219, 266, 271, 325, 356, 391, 392, 400, 429, 454, 458, 470, 503, 531, 553, 605, 625, 627, 646" 
\set_of_points "2, 12, 48, 65, 87, 120, 189, 246, 305, 323, 354, 375, 434, 435, 455, 482, 496, 557, 586, 595" 
\end 
\classification_prefix quartic_25_0_0 
\report 
\end 
\end
SECTION OBJECTS FROM DESIGN THEORY:
15000
15001
15002
15003 \texttt{LS\_AG\_2\_3\_solutions.classify:}
15004 \texttt{\>$\$(ORBITER) -v 2 \textbackslash}
15005 \texttt{\textbackslash \textbackslash -draw.incidence.structure.description \textbackslash}
15006 \texttt{\textbackslash \textbackslash \textbackslash -width 20 -width10 2 -end \textbackslash}
15007 \texttt{\textbackslash \textbackslash -define C -combinatorial.objects \textbackslash}
15008 \texttt{\textbackslash \textbackslash -file_of_designs \textbackslash}
15009 \texttt{\textbackslash \textbackslash -solutions.csv 9 84 3 12 \textbackslash}
15010 \texttt{\textbackslash \textbackslash -end \textbackslash}
15011 \texttt{\textbackslash \textbackslash -with C -do \textbackslash}
15012 \texttt{\textbackslash \textbackslash -combinatorial.object_activity \textbackslash}
15013 \texttt{\textbackslash \textbackslash -canonical.form \textbackslash}
15014 \texttt{\textbackslash \textbackslash \textbackslash -save.ago \textbackslash}
15015 \texttt{\textbackslash \textbackslash \textbackslash -save.transversal \textbackslash}
15016 \texttt{\textbackslash \textbackslash \textbackslash -classification.prefix LS\_AG\_2\_3 \textbackslash}
15017 \texttt{\textbackslash \textbackslash \textbackslash -label LS\_AG\_2\_3 \textbackslash}
15018 \texttt{\textbackslash \textbackslash \textbackslash -max.TDO.depth 10 \textbackslash}
15019 \texttt{\textbackslash \textbackslash \textbackslash -end \textbackslash}
15020 \texttt{\textbackslash \textbackslash -report \textbackslash}
15021 \texttt{\textbackslash \textbackslash \textbackslash -prefix LS\_AG\_2\_3 \textbackslash}
15022 \texttt{\textbackslash \textbackslash \textbackslash -export.flag.orbits \textbackslash}
15023 \texttt{\textbackslash \textbackslash \textbackslash -show.TDO \textbackslash}
15024 \texttt{\textbackslash \textbackslash \textbackslash -end \textbackslash}
15025 \texttt{\textbackslash \textbackslash -end \textbackslash}
15026 \texttt{pdfflatex LS\_AG\_2\_3.classification.tex}
15027 \texttt{open LS\_AG\_2\_3.classification.pdf}
15028 \texttt{\>$\$(ORBITER) -v 2 -draw.matrix \textbackslash}
15029 \texttt{\textbackslash \textbackslash -input.csv.file LS\_AG\_2\_3.object0.INP.flag.orbits.csv \textbackslash}
15030 \texttt{\textbackslash \textbackslash -secondary.input.csv.file LS\_AG\_2\_3.object0.INP.csv \textbackslash}
15031 \texttt{\textbackslash \textbackslash -box.width 12 -bit.depth 24 \textbackslash}
15032 \texttt{\textbackslash \textbackslash -end \textbackslash}
15033 \texttt{open LS\_AG\_2\_3.object0.INP.flag.orbits.draw.bmp}
15034 \texttt{\>$\$(ORBITER) -v 2 -draw.matrix \textbackslash}
15035 \texttt{\textbackslash \textbackslash -input.csv.file LS\_AG\_2\_3.object1.INP.flag.orbits.csv \textbackslash}
15036 \texttt{\textbackslash \textbackslash -secondary.input.csv.file LS\_AG\_2\_3.object1.INP.csv \textbackslash}
15037 \texttt{\textbackslash \textbackslash -box.width 12 -bit.depth 24 \textbackslash}
15038 \texttt{\textbackslash \textbackslash -end \textbackslash}
15039 \texttt{open LS\_AG\_2\_3.object1.INP.flag.orbits.draw.bmp}
15040
15041
15042
15043
15044
15045 \texttt{design\_27c:}
15046 \texttt{\>$\$(ORBITER) -v 4 \textbackslash}

-define C -combinatorial_objects \
-define F -finite_field -q 27 -override_polynomial 46 -end \
-define P -projective_space 2 F -end \
-with C -do \
-combinatorial_object_activity \
-canonical_form_PG P \
-classification_prefix design \
-end \
-report \
-end \
pdflatex design_classification.tex \
open design_classification.pdf 

design_PG_2_3_canonical: 
$(ORBITER) -v 3 \
-define D -design -q 3 -family PG_2.q -end \
-with D -do \
-design_activity \
-export_inc \
-end \
-end 

$(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 \
-saveago \
-save_transversal \
-end \
-report \
-prefix PG_2.3 \
-export_flag_orbits \
-show_incidence_matrices \
-export_group \
-end \
-end 

pdflatex PG_2.3_classification.tex
wreath_product_designs_n4_k2.c: wreath_product_designs_n4_k2.inc.txt

wreath_product_designs_n8_k6.c: wreath_product_designs_n8_k6.inc.txt
# Section 15.5: Linear Codes

SECTION_CANONICAL_FORMS_OF_LINEAR_CODES:

```
15171 code_3_2_aut:
15172  $\text{ORBITER} -v 20 \ 
15173  -define F -finite_field -q 2 -end \ 
15174  -define genma -vector -field F -format 2 \ 
15175  -dense $(\text{CODE}_3\_2\_Q2\_GENMA) \ 
15176  -end \ 
15177  -define P -projective_space 1 F -end \ 
15178  -with P -do \ 
15179  -projective_space_activity \ 
15180  -canonical_form_of_code \ 
15181  "3.2" genma -save_ago -label "3.2" \ 
15182  -classification_prefix "3.2" \ 
15183  -end \ 
15184  -end \ 
15185  pdflatex 3_2_classification.tex
15186  open 3_2_classification.pdf
15187  $\text{ORBITER} -v 2 -draw_matrix \ 
```
15188 \  \  -input_csv_file 3_2_object0_TDA_flag_orbits.csv \\
15189 \  \  -secondary_input_csv_file 3_2_object0_TDA.csv \\
15190 \  \  -box_width 16 -bit_depth 24 \\
15191 \  \  -end \\
15192 \  \  open 3_2_object0_TDA_flag_orbits_draw.bmp \\
15193 \\
15194 \\
15195 \\
15196 \\
15197  code_6_3_aut: \\
15198  \  \  $(ORBITER) -v 20 \\
15199  \  \  -define F -finite_field -q 2 -end \ 
15200  \  \  -define genma -vector -field F -format 3 \ 
15201  \  \  \  \  -compact $(CODE_N6_K3_Q2_GENMA) \ 
15202  \  \  \  \  -end \ 
15203  \  \  \  \  -define P -projective_space 2 F -end \ 
15204  \  \  \  \  \  -with P -do \ 
15205  \  \  \  \  \  \  -projective_space_activity \ 
15206  \  \  \  \  \  \  \  -canonical_form_of_code \ 
15207  \  \  \  \  \  \  \  \  \  "6_3" genma -save_ago -label "6_3" \ 
15208  \  \  \  \  \  \  \  \  \  -classification_prefix "6_3" \ 
15209  \  \  \  \  \  \  \  \  \  \  -end \ 
15210  \  \  \  \  -end \\
15211  \  \  \  \  \  \  \  \  \  \  \  pdflatex 6_3_classification.tex \\
15212  \  \  \  \  \  \  \  \  \  \  \  open 6_3_classification.pdf \\
15213  \  \  \  \  \  \  \  \  \  \  \  $(ORBITER) -v 2 -draw_matrix \ 
15214  \  \  \  \  \  \  \  \  \  \  \  -input_csv_file 6_3_object0_TDA_flag_orbits.csv \ 
15215  \  \  \  \  \  \  \  \  \  \  \  -secondary_input_csv_file 6_3_object0_TDA.csv \ 
15216  \  \  \  \  \  \  \  \  \  \  \  -box_width 16 -bit_depth 24 \ 
15217  \  \  \  \  \  \  \  \  \  \  \  -end \\
15218  \  \  \  \  \  \  \  \  \  \  \  open 6_3_object0_TDA_flag_orbits_draw.bmp \\
15219 \\
15220  # group of order 24 \\
15221 \\
15222 \\
15223  RM_3_1_group: \\
15224  \  \  $(ORBITER) -v 2 \ 
15225  \  \  -define F -finite_field -q 2 -end \ 
15226  \  \  -define genma -vector -field F -format 4 \ 
15227  \  \  \  \  -compact $(CODE_RM_3_1_GENMA) \ 
15228  \  \  \  \  \  -end \ 
15229  \  \  \  \  \  -define P -projective_space 3 F -end \ 
15230  \  \  \  \  \  \  -with P -do \ 
15231  \  \  \  \  \  \  \  -projective_space_activity \ 
15232  \  \  \  \  \  \  \  \  -canonical_form_of_code \ 
15233  \  \  \  \  \  \  \  \  \  \  \  "RM_3_1" genma -save_ago -label "RM_3_1" \ 
15234  \  \  \  \  \  \  \  \  \  \  \  \  -classification_prefix "RM_3_1" \ 

804
15235 \> \> \> -end \\
15236 \> \> -end \\
15237 \> pdflatex RM_3.1_classification.tex \\
15238 \> open RM_3.1_classification.pdf \\
15239 \\
15240 # group order 1344 \\
15241 #RM_3.1_object0_INP_flag_orbits.csv \\
15242 \\
15243 RM_3.1_group_and_diagram: \\
15244 \> $(ORBITER) -v 2 \\
15245 \> \> -define F -finite_field -q 2 -end \\
15246 \> \> -define genma -vector -field F -format 4 \\
15247 \> \> \> -compact $(CODE_RM_3.1_GENMA) \\
15248 \> \> -end \\
15249 \> \> -define P -projective_space 3 F -end \\
15250 \> \> -with P -do \\
15251 \> \> -projective_space_activity \\
15252 \> \> \> -canonical_form_of_code \\
15253 \> \> \> \> "RM_3.1" genma -save_ago -label "RM_3.1" \\
15254 \> \> \> -classification_prefix "RM_3.1" \\
15255 \> \> \> -end \\
15256 \> \> -end \\
15257 \> pdflatex RM_3.1_classification.tex \\
15258 \> open RM_3.1_classification.pdf \\
15259 \> $(ORBITER) -v 2 -draw_matrix \\
15260 \> \> -input_csv_file RM_3.1_object0_INP_flag_orbits.csv \\
15261 \> \> -secondary_input_csv_file RM_3.1_object0_INP.csv \\
15262 \> \> -box_width 16 -bit_depth 24 \\
15263 \> \> -end \\
15264 \> $(ORBITER) -v 2 -draw_matrix \\
15265 \> \> -input_csv_file RM_3.1_object0_TDA_flag_orbits.csv \\
15266 \> \> -secondary_input_csv_file RM_3.1_object0_TDA.csv \\
15267 \> \> -box_width 16 -bit_depth 24 \\
15268 \> \> -end \\
15269 \> open RM_3.1_object0_INP_flag_orbits_draw.bmp \\
15270 \> open RM_3.1_object0_TDA_flag_orbits_draw.bmp \\
15271 \\
15272 \\
15273 \\
15274 RM_4.1_group: \\
15275 \> $(ORBITER) -v 2 \\
15276 \> \> -define F -finite_field -q 2 -end \\
15277 \> \> -define genma -vector -field F -format 5 \\
15278 \> \> \> -compact $(CODE_RM_4.1_GENMA) \\
15279 \> \> -end \\
15280 \> \> -define P -projective_space 4 F -end \\
15281 \> \> -with P -do \
805
-projective_space_activity \  
-canonical_form_of_code \  "RM_4_1" genma -save_ago -label "RM_4_1" \  
-classification_prefix "RM_4_1" \  
-end \  
pdflatex RM_4_1_classification.tex  
open RM_4_1_classification.pdf  
$\text{ORBITER} -v 2 -draw_matrix \  
-input_csv_file RM_4_1_object0_INP_flag_orbits.csv \  
-secondary_input_csv_file RM_4_1_object0_INP.csv \  
-box_width 16 -bit_depth 24 \  
-end \  
$\text{ORBITER} -v 20 \  
-define F -finite_field -q 7 -end \  
-genma -vector -field F -format 4 \  
-compact $(\text{CODE_RS}_6_4_7) \  
-end \  
(define P -projective_space 3 F -end \  
-with P -do \  
-projective_space_activity \  
-canonical_form_of_code \  "RS_6" genma -save_ago -label "RS_6" \  
-classification_prefix "RS_6" \  
-end \  
-end  

GV_n15_k6_d5_group:  
$\text{ORBITER} -v 20 \  
-define F -finite_field -q 2 -end \  
-genma -vector -field F -format 6 \  
-compact $(\text{CODE_GV_N15_K6}) \  
-end \  

806
\begin{verbatim}
15329 \triangleright \triangleright \texttt{-define P-projective_space 5 F -end \}
15330 \triangleright \triangleright \texttt{-with P -do \}
15331 \triangleright \triangleright \texttt{-projective_space_activity \}
15332 \triangleright \triangleright \texttt{-canonical_form_of_code \}
15333 \triangleright \triangleright \texttt{ "GV_n15_k6_d5" genma -save_ago -label "GV_n15_k6_d5" \}
15334 \triangleright \triangleright \texttt{ -classification_prefix "GV_n15_k6_d5" \}
15335 \triangleright \triangleright \texttt{ -end \}
15336 \triangleright \triangleright \texttt{ -end \}
15337 \triangleright \texttt{ pdflatex GV_n15_k6_d5_classification.tex \}
15338 \triangleright \texttt{ open GV_n15_k6_d5_classification.pdf \}
15339 \texttt{ #ago=12 \}
15340 15341 15342 15343 15344 code_n15_k6_d6_a_group: 
15345 \triangleright \texttt{ $(ORBITER) -v 20 \}
15346 \triangleright \texttt{ -define F-finite_field -q 2 -end \}
15347 \triangleright \texttt{ -define genma -vector -field F -format 6 \}
15348 \triangleright \texttt{ -compact $(CODE_15_6_6_A) \}
15349 \triangleright \texttt{ -end \}
15350 \triangleright \texttt{ -define P-projective_space 5 F -end \}
15351 \triangleright \texttt{ -with P -do \}
15352 \triangleright \texttt{ -projective_space_activity \}
15353 \triangleright \texttt{ -canonical_form_of_code \}
15354 \triangleright \texttt{ "n15_k6_d6_a" genma -save_ago -label "n15_k6_d6_a" \}
15355 \triangleright \texttt{ -classification_prefix "n15_k6_d6_a" \}
15356 \triangleright \texttt{ -end \}
15357 \triangleright \texttt{ -end \}
15358 \triangleright \texttt{ pdflatex n15_k6_d6_a_classification.tex \}
15359 \triangleright \texttt{ open n15_k6_d6_a_classification.pdf \}
15360 15361 15362 code_n15_k6_d6_b_group: 
15363 \triangleright \texttt{ $(ORBITER) -v 20 \}
15364 \triangleright \texttt{ -define F-finite_field -q 2 -end \}
15365 \triangleright \texttt{ -define genma -vector -field F -format 6 \}
15366 \triangleright \texttt{ -compact $(CODE_15_6_6_B) \}
15367 \triangleright \texttt{ -end \}
15368 \triangleright \texttt{ -define P-projective_space 5 F -end \}
15369 \triangleright \texttt{ -with P -do \}
15370 \triangleright \texttt{ -projective_space_activity \}
15371 \triangleright \texttt{ -canonical_form_of_code \}
15372 \triangleright \texttt{ "n15_k6_d6_b" genma -save_ago -label "n15_k6_d6_b" \}
15373 \triangleright \texttt{ -classification_prefix "n15_k6_d6_b" \}
15374 \triangleright \texttt{ -end \}
15375 \triangleright \texttt{ -end \}
\end{verbatim}
# Section 15.6: General Codes

SECTION_CANONICAL_FORMS_OF_GENERAL CODES:

Hamming graph 7 with Hamming code:

```bash
graph $\input{Hamming_7} -end
```

# group of order 2688 = 16 * 168

# Section 15.7: Graphs

SECTION_CANONICAL_FORMS_OF_GRAPHS:

Cycle 13 aut:

```bash
graph $\input{Cycle_13} -end
```

808
15423       \> \> \> -graph_theoretic_activity -automorphism_group \\
15424       \> \> -end \\
15425
15426
15427       inversion_graph:
15428       \> $(ORBITER) -v 6 \\
15429       \> \> -define G -graph \\
15430       \> \> \> -inversion_graph "1,0,2,3" \\
15431       \> \> -end \\
15432       \> \> -with G -do \\
15433       \> \> \> -graph_theoretic_activity -properties \\
15434       \> \> -end \\
15435       \> \> -with G -do \\
15436       \> \> \> -graph_theoretic_activity -automorphism_group \\
15437       \> \> -end \\
15438
15439
15440
15441       Chain\_232\_aut:
15442       \> $(ORBITER) -v 2 \\
15443       \> \> -define P1 -vector -dense 2,3,2 -end \\
15444       \> \> -define P2 -vector -dense 2,3,2 -end \\
15445       \> \> -define Gamma -graph \\
15446       \> \> \> -chain_graph P1 P2 \\
15447       \> \> -end \\
15448       \> \> -with Gamma -do \\
15449       \> \> \> -graph_theoretic_activity -automorphism_group \\
15450       \> \> -end \\
15451       \> pdflatex chain\_graph\_report.tex \\
15452       \> open chain\_graph\_report.pdf \\
15453
15454
15455
15456       JK\_graph\_pp16\_1:
15457       \> $(ORBITER) -v 2 \\
15458       \> \> -define Gamma -graph -load_dimacs \\
15459       \> \> ..../JUNTTILA\_KASKI/benchmarks/pp/pp16-1 \\
15460       \> \> -end \\
15461       \> \> -with Gamma -do \\
15462       \> \> \> -graph_theoretic_activity -save -end \\
15463       \> \> \> -with Gamma -do \\
15464       \> \> \> -graph_theoretic_activity -automorphism_group -end \\
15465
15466 # go=34217164800
15467 #nauty\_interface\_with\_group::create\_automorphism\_group\_of\_graph\_with\_partition\_and\_labeling: nb\_backtrack1 = 6
15468 #nauty\_interface\_with\_group::create\_automorphism\_group\_of\_graph\_with\_partition\_an
d_labeling: nb_backtrack2 = 134
15469
15470 JK_graph_pp16_2:
15471 $ (ORBITER) -v 2 \
15472 "define Gamma -graph -load_dimacs " \
15473 "../JUNTTILA_KASKI/benchmarks/pp/pp16-2 " \
15474 " -end " \
15475 " -with Gamma -do " \
15476 " -graph_theoretic_activity -save -end " \
15477 " -with Gamma -do " \
15478 " -graph_theoretic_activity -automorphism_group -end " \
15479
15480 # does not finish
15481
15482 JK_graph_pp16_9:
15483 $ (ORBITER) -v 2 \
15484 "define Gamma -graph -load_dimacs " \
15485 "../JUNTTILA_KASKI/benchmarks/pp/pp16-9 " \
15486 " -end " \
15487 " -with Gamma -do " \
15488 " -graph_theoretic_activity -save -end " \
15489 " -with Gamma -do " \
15490 " -graph_theoretic_activity -automorphism_group -end " \
15491
15492
15493 JK_graph_grid_3_3:
15494 $ (ORBITER) -v 2 \
15495 "define Gamma -graph -load_dimacs " \
15496 "../JUNTTILA_KASKI/benchmarks/grid/grid-w-3-3 " \
15497 " -end " \
15498 " -with Gamma -do " \
15499 " -graph_theoretic_activity -save -end " \
15500 " -with Gamma -do " \
15501 " -graph_theoretic_activity -automorphism_group -end " \
15502
15503
15504 #nauty_interface_with_group::create_automorphism_group_of_graph_with_partition_an
d labeling: nb_backtrack1 = 4
15505 #nauty_interface_with_group::create_automorphism_group_of_graph_with_partition_an
d labeling: nb_backtrack2 = 9
15506 #Written file grid-w-3-3_group.makefile of size 579
15507 #User time: 0 of a second, dt=0 tps = 100
15508 #nb_calls_to_densenauty=1
15509
15510
15511 JK_graph_sts_13:
15512 $ (ORBITER) -v 2 \
15513
define Gamma -graph -load_dimacs \
define Gamma -graph -load ../JUNTTILA_KASKI/benchmarks/srg/sts-13 \
define Gamma -graph -load \
define Gamma -graph -load 

make ORBITER_PATH=$(ORBITER_PATH) -f sts-13_group.makefile sts-13

HJ aut:

HJ group and orbits:

HJ group and orbits:
HJ_orbital_graph_3:

$(ORBITER) -v 2 \\
-define gens -vector -file halljanko315.gens.csv -end \\
-define G -permutation_group \\
-bsgs halljanko315 "File\haljanko315" \\
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 \\

#Group_Per531_Orbital_3.colored_graph

Degree type: \(64^{315}\) 

PGO_5_2_graph_group: 0_5_2_incidence_matrix.csv

\(\text{\textbackslash}$(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 \
# Section 15.8: Quartic Curves

SECTION_CANONICAL_FORMS_OF_QUARTIC_CURVES:

Edge_curve_17_nauty:

Edge_curve_17_nauty:

Edge_curve_17_nauty:

Edge_curve_17_nauty:

Edge_curve_17_nauty:
pdflatex Edge_curve_q17_classification.tex
open Edge_curve_q17_classification.pdf

$\text{ORBITER} -v 2 -draw\_matrix$

-input\_csv\_file Edge_curve_q17\_object0\_TDA\_flag\_orbits.csv
-Secondary\_input\_csv\_file Edge_curve_q17\_object0\_TDA.csv
-box\_width 4 -bit\_depth 24
-end

open Edge_curve_q17\_object0\_TDA\_flag\_orbits\_draw.bmp

# 9 backtrack nodes total

# aut = 24
# User time: 0.04 of a second, dt=4 tps = 100

# generators for a group of order 24:

#1,0,0,0,13,0,0,0,4,
#1,0,0,0,0,16,0,16,0,
#0,1,16,2,4,4,15,4,4,

Edge_curve_17\_group:

$\text{ORBITER} -v 3$
-define G -linear\_group -PGL 3 17
-subgroup\_by\_generators "Stab\_Edge" "24" 3

"1,0,0,0,13,0,0,0,4"
"1,0,0,0,0,16,0,16,0"
"0,1,16,2,4,4,15,4,4"
-end

-with G -do

-group\_theoretic\_activities
-print\_elements\_tex

-group\_table

-report

814
# Chapter 16 - Interfaces

# Section 16.1: Graphical Output

**SECTION GRAPHICAL OUTPUT:**

```
F_7_tables:
$\text{(ORBITER) -v 3 \}
$\text{-define F -finite_field -q 7 -end \}
$\text{-with F -do -finite_field_activity \}
$\text{-cheat sheet_GF \}
$\text{-end \}
$\text{(ORBITER) -v 2 \}
$\text{-draw_matrix \}
$\text{-input_csv_file GF_q7_addition_table.csv \}
$\text{-box_width 40 \}
$\text{-bit_depth 24 \}
$\text{-partition 3 7 7 \}
$\text{-end \}
$\text{open GF_q7_addition_table_draw.bmp \}

PG_2.4_cyclic_incma:
$\text{(ORBITER) -v 2 \}
$\text{-define F -finite_field -q 4 -end \}
$\text{-define P -projective_space 2 F -end \}
$\text{-with P -do -projective_space_activity \}
$\text{-cheat_sheet_for_decomposition_by_element_PG \}
$\text{1 "0,1,0, 0,0,1, 2,1,1, 0" "PG_2.4_singer" \}
$\text{-end \}
$\text{(ORBITER) -v 4 \}
$\text{-list_arguments \}
$\text{-define R -vector -repeat 1 21 -end \}
```
15746 $>$ $>$ -define C -vector -repeat 1 21 -end \\
15747 $>$ $>$ -draw matrix \\
15748 $>$ $>$ -input_csv_file PG_2_4_singer_incma_cyclic.csv \\
15749 $>$ $>$ -box_width 40 -bit_depth 24 \\
15750 $>$ $>$ -partition 3 R C \\
15751 $>$ $>$ -end \\
15752 open PG_2_4_singer_incma_cyclic_draw.bmp \\
15753 \\
15754 \\
15755 \\
15756 PGL_4_2_Wedge_4_0_graphical_output: \\
15757 $>$ $(ORBITER)$ -v 4 \\
15758 $>$ $>$ -define G -linear_group -PGL 4 2 \\
15759 $>$ $>$ $>$ -wedge_detached \\
15760 $>$ $>$ $>$ -end \\
15761 $>$ $>$ -with G -do \\
15762 $>$ $>$ -group_theoretic_activity \\
15763 $>$ $>$ $>$ -report \\
15764 $>$ $>$ -end \\
15765 $>$ $>$ -end \\
15766 $>$ pdflatex PGL_4_2_Wedge_4_2_detached_report.tex \\
15767 $>$ open PGL_4_2_Wedge_4_2_detached_report.pdf \\
15768 \\
15769 $>$ $>$ $>$ -draw_options -radius 200 -end \\
15770 \\
15771 schreier_tree_graphical_output: \\
15772 $>$ $(ORBITER)$ -v 4 \\
15773 $>$ $>$ -draw_options \\
15774 $>$ $>$ $>$ -yout 500000 \\
15775 $>$ $>$ $>$ -radius 15 -nodes_empty \\
15776 $>$ $>$ $>$ -line_width 0.5 -y_stretch 0.25 \\
15777 $>$ $>$ -end \\
15778 $>$ $>$ -define G -linear_group -PGL 4 2 -end \\
15779 $>$ $>$ -with G -do \\
15780 $>$ $>$ -group_theoretic_activity \\
15781 $>$ $>$ $>$ -orbits_on_polynomials 3 \\
15782 $>$ $>$ $>$ -orbits_on_polynomials_draw_tree 6 \\
15783 $>$ $>$ -end \\
15784 $>$ pdflatex poly_orbits_d3_n3_q2.tex \\
15785 $>$ open poly_orbits_d3_n3_q2.pdf \\
15786 \\
15787 Queens_graph: \\
15788 $>$ $(ORBITER)$ -v 2 \\
15789 $>$ $>$ -define G -graph -non_attacking_queens_graph 8 -end \\
15790 $>$ $>$ -with G -do \\
15791 $>$ $>$ -group_theoretic_activity -export_csv -end \\
15792 $>$ $>$ $>$ -end
Section 16.2: The Povray Interface

SECTION POVRAY:

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 
-end 
-scene_objects 
-obj_file cube_centered.obj 
-edge "0, 1" 
-edge "0, 2" 
-edge "0, 4" 
-edge "1, 3" 
-edge "1, 5" 
-edge "2, 3" 
-edge "2, 6" 
-edge "3, 7" 
-edge "4, 5" 
-edge "4, 6" 
-edge "5, 7" 
-edge "6, 7" 
-group_of_things_as_interval 0 8 
-spheres 0 0.3 $(POLISHED_CHROME_WHITE) 
-group_of_things_as_interval 0 6 

math261_test:
  $(ORBITER) -v 2 -povray
  -round 0 -nb_frames_default 30
  -output_mask math261_%d%03d.pov
  -video_options -W 1024 -H 768
  -global_picture_scale 0.1
  -default_angle 75
  -clipping_radius 2.7
  -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 cube_0_*.*.pov POV
  mv makefile_animation POV
mv makefile_animation POV

plane1:
$(ORBITER) -v 2 -povray \
  -round 0 -nb_frames_default 30 \n  -output_mask plane1_%d_%03d.pov \n  -video_options -W 1024 -H 768 \n  -global_picture_scale 0.40 \n  -default_angle 75 \n  -clipping_radius 5 -omit_bottom_plane \n  -camera 0 "0,0,1" "5,5,3" "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  -point "-2.25,0,0" \n  -point "0,-1.8,0" \n  -point "0,0,9" \n  -face "0,1,2,0" \n  -group_of_things "0" \n  -group_of_things "1" \n  -group_of_things "2" \n  -lines 0 0.15 $(COLOR_RESHINY) \n  -lines 1 0.15 $(COLOR_GREEN_SHINY) \n  -lines 2 0.15 $(COLOR_BLUE_SHINY) \n  -group_of_things_as_interval 3 39 \n  -lines 3 0.05 $(COLOR_BLACK_SHINY) \n  -group_of_things "0" \n  -planes 0 $(COLOR_BLUE_SEE_THROUGH) \n  -group_of_things "1" \n  -group_of_things "2" \n  -group_of_things "0" \n  -prisms 0 0.05 $(COLOR_YELLOW_THICK) \n  -scene_objects_end \n  -povray_end
  -rm -rf POV
mkdir POV
mv plane1_0_*.pov POV
mv makefile_animation POV
15934
15935 plane2:
15936 |> $(ORBITER) -v 2 -povray \
15937 |>  -round 0 -nb_frames_default 30 \
15938 |>  -output_mask plane2_%d_%03d.pov \
15939 |>  -video_options -W 2560 -H 1920 \
15940 |>  -global_picture_scale 0.40 \
15941 |>  -default_angle 75 \
15942 |>  -clipping_radius 5 -omit_bottom_plane \
15943 |>  > -camera 0 "0,0,1" "6,6,2" "0,0,0" \
15944 |>  > -rotate_about_z_axis \
15945 |>  > -boundary_box \
15946 |>  > -end \
15947 |>  > -scene_objects \
15948 |>  >  -line_through_two_points_recentered_from_csv_file coordinate_grid.csv \
15949 |>  >  -plane_by_dual_coordinates "0,0,1,0" \
15950 |>  >  -plane_by_dual_coordinates "0,1,0,0" \
15951 |>  >  -plane_by_dual_coordinates "1,0,0,0" \
15952 |>  >  -plane_by_dual_coordinates "4,5,-1,9" \
15953 |>  >  -group_of_things "0"
15954 |>  >  -group_of_things "1"
15955 |>  >  -group_of_things "2"
15956 |>  >  -group_of_things_as_interval 3 39
15957 |>  >  -lines 0 0.15 $(COLOR_RED_SHINY)
15958 |>  >  -lines 1 0.15 $(COLOR_GREEN_SHINY)
15959 |>  >  -lines 2 0.15 $(COLOR_BLUE_SHINY)
15960 |>  >  -lines 3 0.05 $(COLOR_BLACK_SHINY)
15961 |>  >  -group_of_things "0"
15962 |>  >  -planes 4 $(COLOR_BLUESEE_THROUGH)
15963 |>  >  -group_of_things "3"
15964 |>  >  -scene_objects_end \
15965 |>  > -povray_end
15966 |>  - rm -rf POV
15967 |>  mkdir POV
15968 |>  mv plane2_0_*_.pov POV
15969 |>  mv makefile_animation POV
15970
15971 #>  > -planes 5 "texture{ pigment{ color Yellow transmit 0.5 } finish{ diffuse 0.9 phong 1}}" 
15972
15973
15974
15975 analytic_geo_1:
15976 |> $(ORBITER) -v 2 -povray \
15977 |>  -round 0 -nb_frames_default 30 \
15978 |>  -output_mask analytic_geo_1_%d_%03d.pov \
15979 |>  -video_options -W 2560 -H 1920 \

820
-global_picture_scale 0.80 \
-default_angle 75 \n-clipping_radius 5 -omit_bottom_plane \n-camera 0 "0,0,1" "6,6,2" "0,0,0" \
-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-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 0.05 $(COLOR_RED_SHINY) \n-lines 1 0.05 $(COLOR_GREEN_SHINY) \n-lines 2 0.05 $(COLOR_BLUE_SHINY) \n-lines 3 0.04 $(COLOR_BLACK_SHINY) \n-group_of_things "0" \n-group_of_things "1" \n-group_of_things "2" \n-planes 4 $(COLOR_BLUE_SEE_THROUGH) \n-planes 5 $(COLOR_GREEN_SEE_THROUGH) \n-planes 6 $(COLOR_RED_SEE_THROUGH) \n-point "0,0,0" \n-point "1,0,0" \n-point "1,2,0" \n-point "1,2,3" \n-edge "84,85" \n-edge "85,86" \n-edge "86,87" \n-edge "84,87" \n-group_of_things "84,85,86" \n-spheres 7 0.1 $(POLISHED_CHROME_WHITE) \n-group_of_things "87" \n-spheres 8 0.10 $(COLOR_YELLOW_SHINY) \n-group_of_things "0,1,2" \n-cylinders 9 0.075 $(POLISHED_CHROME_WHITE) \n-group_of_things "3" \n-cylinders 10 0.075 $(COLOR_YELLOW_SHINY) \n-scene_objects_end \npovray_end
- rm -rf POV
mkdir POV
mv analytic_geo.1_0.*.pov POV
mv makefile

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 \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 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 \
  -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 \

822
-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

-label 0 "a1"

-label 2 "a2"

-label 4 "a3"

-label 6 "a4"

-label 8 "a5"

-label 10 "a6"

-label 12 "b1"

-label 14 "b2"

-label 16 "b3"

-label 18 "b4"

-label 20 "b5"

-label 22 "b6"

-label 24 "c12"

-label 26 "c13"

-label 30 "c15"

-label 32 "c16"

-label 34 "c23"

-label 36 "c24"

-label 40 "c26"

-label 42 "c34"

-label 44 "c35"

-label 48 "c45"

-label 50 "c46"

-label 52 "c56"

-group_of_things_as_interval 0 6

-texts 4 0.2 0.15 $(COLOR_BLACK_NO_SHADOW)

-texts 5 0.2 0.15 $(COLOR_BLACK_NO_SHADOW)

-group_of_things_as_interval 12 12
16121 ▶ ▶ ▶ -texts 6 0.2 0.15 $(COLOR_BLACK_NO_SHADOW) \
16122 ▶ ▶ -scene_objects_end \
16123 ▶ ▶ -povray_end 
16124 ▶ - rm -rf POV 
16125 ▶ mkdir POV 
16126 ▶ mv Eckardt_0_*.pov POV 
16127 ▶ mv makefile_animation POV 
16128 
16129 
16130 #"-3,2.333,4" * 1.5 = "-4.5,3.5,6" 
16131 \n16132 KM := Matrix([[[-4.5, 3.5, 6], [1, 1, 1]]]) 
16133 #NullSpace(M) 
16134 #=0.186080731891197,-0.781539073943026,0.595458342051830 
16135 ▶ ▶ ▶ -rotate.about.custom_axis "0.186080731891197,-0.781539073943026,0.595458342051830" \
16136 #W 1024 -H 768 
16137 #W 2560 -H 1920 
16138 #W 4096 -H 3072 
16139 
16140 
16141 
16142 Eckardt_deform: 
16143 ▶ $(ORBITER) -v 2 -povray \
16144 ▶ -round 0 -nb_frames_default 93 \ 
16145 ▶ -output_mask Eckardt_deform_%d_%03d.pov \ 
16146 ▶ -video_options -W 1024 -H 768 \ 
16147 ▶ -global_picture_scale 0.9 \ 
16148 ▶ -default_angle 75 \ 
16149 ▶ -clipping_radius 2.4 \ 
16150 ▶ -camera 0 "1,1,1" "-3,1,3" "0.12,0.12,0.12" \ 
16151 ▶ -end \ 
16152 ▶ -scene_objects \ 
16153 ▶ -Hilbert.Cohn_Vossen_surface \ 
16154 ▶ -hilbert_Cohn_Vossen_surface \ 
16155 ▶ -deformation_of_cubic_lex 93 1.107148718 1.570796327 0 \ 
16156 ▶ -ECKARDT_CUBIC_DEFORM1_LEX \ 
16157 ▶ -ECKARDT_CUBIC_DEFORM2_LEX \ 
16158 ▶ -group_of_things_as_interval 0 93 \ 
16159 ▶ -group_is_animated 1 \ 
16160 ▶ -cubics 1 $(SURFACE_COLOR_SEETHROUGH) \ 
16161 ▶ -scene_objects_end \ 
16162 ▶ -povray_end 
16163 ▶ - rm -rf POV 
16164 ▶ mkdir POV 
16165 ▶ mv Eckardt_deform_0_*.pov POV 
16166 ▶ mv makefile_animation POV 

824
Eckardt_deform_2:

```
$(ORBITER) -v 2 -povray \
-round 0 -nb_frames_default 30 \
-output_mask Eckardt_deform_%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 \n-scene_objects \n   -Hilbert_Cohn_Vossen_surface \n   -group_of_things "0" \n   -deformation_of_cubic_lex 93 1.107148718 1.570796327 0 \n   -group_of_things $(ECKARDT_CUBIC_DEFORM1_LEX) \n   -group_of_things $(ECKARDT_CUBIC_DEFORM2_LEX) \n   --group_of_things_as_interval 0 93 \n   --group_is_animated 1 \n   -group_of_things "0" \n   -cubics 1 $(SURFACE_COLOR_SEETHROUGH) \n   -cubics 2 $(COLOR_RED) \n   -cubics 3 $(COLOR_BLUE) \n-scene_objects_end \n-povray_end
```

Clebsch:

```
$(ORBITER) -v 2 -povray \
-round 0 -nb_frames_default 30 \
-output_mask Clebsch_%d.%03d.pov \
-video_options -W 1024 -H 768 \
-global_picture_scale 0.9 \
-default_angle 80 \
-clipping_radius 2.4 \
-camera 0 "1,1,1" "-4.5,3.5,6" "0,0,0" \
```
end

-end \n-scene_objects \n-Clebsch_surface \n-group_of_things "0" \ncubics 0 $(SURFACE_COLOR) \n-group_of_things_as_interval 0 6 \n-group_of_things_as_interval 6 6 \n-group_of_things_as_interval 12 15 \n-lines 1 0.02 $(COLOR_RED_SHINY) \n-lines 2 0.02 $(COLOR_BLUE_SHINY) \n-lines 3 0.02 $(COLOR_YELLOW_SHINY) \n-group_of_things_as_interval 0 12 \n-spheres 4 0.08 $(COLOR_TURQUOISE) \n-scene_objects_end \n-povray_end

rmdir -rf POV
mkdir POV
mv Clebsch_0*.pov POV
mv makefile_animation POV

endrass8:
$(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_bottomplane \ncamera 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) \noctic_lex_165 $(ENDRASS_OCTIC_LEX_165) \n-plane_by_dual_coordinates "0,0,1,0" \n
16280 SECTION_ANIMATIONS:

16284 dode:
16285 $(ORBITER) -v 2 \n16286 -povray \n16287 -round 0 -nb_frames_default 30 \n16288 -output_mask dode_%d_%03d.pov \n16289 -video_options -W 1024 -H 768 \n16290 -global_picture_scale 0.50 \n16291 -default_angle 45 \n16292 -clipping_radius 5 \n16293 -camera 0 "1,1,1" "-2,2,4" "0,0,0" \n16294 -rotate_about_111 \n16295 -end \n16296 -scene_objects \n16297 -dodecahedron \n16298 -group_of_things_as_interval 0 20 \n16299 -spheres 0 0.075 $(POLISHED_CHROME_WHITE) \n16300 -group_of_things_as_interval 0 30 \n16301 -cylinders 1 0.05 $(COLOR_RED_SHINY) \n16302 -group_of_things_as_interval 0 12 \n16303 -prisms 2 0.02 $(YELLOW_TRANSPARENT) \n16304 -scene_objects_end \n16305 -povray_end
16306 -rm -rf POV
16307 mkdir POV
mv dode_0_*.pov POV
mv makefile_animation POV

16308
dode_video:
16309 rm -r FRAMES
16310 mkdir FRAMES
16311 rm dode.mp4
16312 $(ORBITER) \ 
16313 \prepare_frames \ 
16314 \ \i 0 30 DODE/dode_0_%03d.png \ 
16315 \ \output_starts_at 0 \ 
16316 \ \o FRAMES/frame%04d.png \ 
16317 \-end
16318 ffmpeg -r 5 -f image2 -i FRAMES/frame%04d.png \ 
16319 \-f mp4 -q:v 0 -vcodec mpeg4 dode.mp4

monkey_video:
16326 rm -r FRAMES
16327 mkdir FRAMES
16328 rm monkey.mp4
16329 $(ORBITER) \ 
16330 \prepare_frames \ 
16331 \ \i 0 30 monkey_0_%03d.png \ 
16332 \ \output_starts_at 0 \ 
16333 \ \o FRAMES/frame%04d.png \ 
16334 \-end
16335 ffmpeg -r 5 -f image2 -i FRAMES/frame%04d.png \ 
16336 \-f mp4 -q:v 0 -vcodec mpeg4 monkey.mp4

Eckardt_deform_video:
16339 $(ORBITER) \ 
16340 \prepare_frames \ 
16341 \ \i 0 93 Eckardt_deform_0/Eckardt_deform_0_%03d.png \ 
16342 \ \output_starts_at 0 \ 
16343 \ \o FRAMES/frame%04d.png \ 
16344 \-end
16345 ffmpeg -r 5 -f image2 -i FRAMES/frame%04d.png \ 
16346 \-f mp4 -q:v 0 -vcodec mpeg4 Eckardt_deform.mp4

Eckardt_surface:
16353 $(ORBITER) -v 2 -povray \

828
# Maple:
#Kummer := expand((x0^2 + x1^2 + x2^2 + x3^2)^2 - 3*(x0^4 + x1^4 + x2^4 + x3^4))
Kummer_video:

- rm -r FRAMES
- mkdir FRAMES
- rm Kummer.mp4

$(ORBITER) \\n-prepare_frames \n- i 0 30 KUMMER/Kummer_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 
-f mp4 -q:v 0 -vcodec mpeg4 Kummer.mp4

Beauville_surface:

$(ORBITER) -v 2 -povray \\n-round 0 -nb_frames_default 30 \\n-output_mask Beauville_%d.%03d.pov \\n-video_options -W 1024 -H 768 \\n-global_picture_scale 0.3 \\n-default_angle 75 \\n-clipping_radius 2.4 \\ncamera 0 "1,1,1" "-3,1,3" "0.12,0.12,0.12" \\n-end \n-scene_objects \\n-quintic_lex_56 $(BEAUVILLE_QUINTIC_LEX_56) \\n-group_of_things "0" \\n-quintics 0 $(SURFACE_COLOR_SEETHROUGH) \\n-scene_objects_end \\npovray_end
- rm -rf POV
mkdir POV
mv Beauville_0.*.pov POV
mv makefile_animation POV

# Clebsch map up for surface created using arc lifting
# We take a circle of radius r centered at the origin in the affine real plane
# and map it up on the surface.
# The Clebsch surface has
# a = d = 2.618033988 = (3+sqrt(5))/2
# b = c = 1.618033988 = (1+sqrt(5))/2

CLEBSCH_A=2.618033988

CLEBSCH_D=2.618033988

CLEBSCH_B=1.618033988

CLEBSCH_C=1.618033988

TWO_PI=6.283185308

# to go from the arclifting surface to the defining equation:
Matrix(4, 4, 
[-0.4472136021531273, 1.1708204000530853, 1.1708204000530853, -0.4472135957999158], 
[-1.1708204000530853, 0.4472136021531272, 1.4472136021531272, 0.4472135957999158], 
[4.2360680044124255, -4.2360680044124255, -4.2360680044124255, -4.2360680044124255, 5, 0.], 
[1.6180340022062127, -2.6180340022062127, -1.6180340022062127, 0.]])

#T00=-0.44721360215312733, 1.1708204000530853, 1.1708204000530853, -0.4472135957999158

#T10=-1.1708204000530853, 0.4472136021531272, 1.4472136021531272, 0.4472135957999158

#T20=4.2360680044124255, -4.2360680044124255, -4.2360680044124255, -4.2360680044124255, 0.

T00=-0.4472136021531273

T01=1.1708204000530853

T02=1.1708204000530853

T03=-0.4472135957999158

T04=-1.1708204000530853

T05=0.4472136021531272

T06=0.4472135957999158

T07=1.4472136021531272

T08=-1.6180340022062127

T09=-2.6180340022062127

T10=-1.6180340022062127

T11=0.4472136021531272

T12=1.4472136021531272

T13=0.4472135957999158

T14=4.2360680044124255

T15=-2.6180340022062127

T16=-4.2360680044124255

T17=-4.2360680044124255

T18=0.

T19=0.

T20=4.2360680044124255

T21=-4.2360680044124255

T22=-4.2360680044124255

T23=0.
$T_{23} = 0.$

$T_{30} = 1.6180340022062127$

$T_{31} = -2.6180340022062127$

$T_{32} = -1.6180340022062127$

$T_{33} = 0.$

CLEBSCH_CUBICS =

```
push b push b mult push d push c push m mult add mult \\
push b push c push d push d push m mult mult add mult \\
push a push d push d push m add mult mult add add \\
push a push c push m mult add mult \\
store c001 \\
push b push d mult \\
push b push 1 push m push c mult add mult \\
push d push a push 1 push m mult add mult add \\
push m push a mult add push c add \\
push c push m push a mult add \\
push m push m mult \\
store c002 \\
push b \\
push d push c push a push m mult add mult \\
push c push a push m push 1 mult add mult add mult \\
push a push d mult push c push 1 push m mult add mult \\
push m push m mult add \\
push a push c push m mult add mult \\
store c011 \\
push b push b push c mult mult \\
push 1 push d push m mult add mult \\
push a push b mult push c push d push d push m mult mult add mult \\
push m mult add \\
push a push d mult push c push d push m mult add mult add \\
push a push c push m mult add mult \\
store c012 \\
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 mult \\
push b push d push m mult add mult \\
store d001 \\
push m \\
push d push c push m mult add push a push a mult mult \\
push c push c mult push d push m mult add push a mult add \\
```
push m push b push c mult mult push c push 1 push m mult add mult add mult push m push d push m mult add mult \store d011 \push m \push c push d mult push d push m mult add push a push a mult mult \push c 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 mult \push a push a mult 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 mult \store m002 \push m \push d push c 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 d112 \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 a push c mult mult \store m022 \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 push c push 1 push m mult add mult add \
Clebsch up_create_points:

```plaintext
16608 $(ORBITER) -v 2 \
16609 -smooth_curve "Clebsch_map_of_circle_to_defining_eqn_r2" \
16610 0.07 1000 5 0 $(TWO_PI) \
16611 -const a $(CLEBSCH_A) b $(CLEBSCH_B) c $(CLEBSCH_C) d $(CLEBSCH_D) \
16612 t00 $(T00) t01 $(T01) t02 $(T02) t03 $(T03) \
16613 t10 $(T10) t11 $(T11) t12 $(T12) t13 $(T13) \
16614 t20 $(T20) t21 $(T21) t22 $(T22) t23 $(T23) \
16615 t30 $(T30) t31 $(T31) t32 $(T32) t33 $(T33) \
16616 r 2 one 1 m -1 \
16617 -const_end \
16618 -var t \
16619 c001 c002 c011 c012 \
16620 d001 d011 d012 d112 \
16621 m002 m012 m022 m122 \
16622 n002 n012 n112 n022 n122 \
16623 y0 y1 y2 \
16624 834
```
push t cos push r mult store y0 \  
push t sin push r mult store y1 \  
push one store y2 \  
push y0 push y0 push y1 mult mult store y001 \  
push y0 push y0 push y2 mult mult store y002 \  
push y0 push y1 push y1 mult mult store y011 \  
push y0 push y1 push y2 mult mult store y012 \  
push y0 push y2 push y2 mult mult store y022 \  
push y1 push y1 push y2 mult mult store y112 \  
push y1 push y2 push y2 mult mult store y122 \  
$(CLEBSCH_CUBICS)  
push c001 push y001 mult \  
push c002 push y002 mult add \  
push c011 push y011 mult add \  
push c012 push y012 mult add \  
store x0 \  
push d001 push y001 mult \  
push d011 push y011 mult add \  
push d012 push y012 mult add \  
push d112 push y112 mult add \  
store x1 \  
push m002 push y002 mult \  
push m012 push y012 mult add \  
push m022 push y022 mult add \  
push m122 push y122 mult add \  
store x2 \  
push n002 push y002 mult \  
push n012 push y012 mult add \  
push n022 push y022 mult add \  
push n112 push y112 mult add \  
push n122 push y122 mult add \  
store x3 \  
push x0 push t00 mult \  
push x1 push t10 mult add \  
push x2 push t20 mult add \  
push x3 push t30 mult add \  
return \  
push x0 push t01 mult \  
push x1 push t11 mult add \  
push x2 push t21 mult add \  
push x3 push t31 mult add \  
return \  
push x0 push t02 mult \
16672 push x1 push t12 mult add \n
16673 push x2 push t22 mult add \n
16674 push x3 push t32 mult add \n
16675 return \n
16676 push x0 push t03 mult \n
16677 push x1 push t13 mult add \n
16678 push x2 push t23 mult add \n
16679 push x3 push t33 mult add \n
16680 return \n
16681 -code_end \n
16682 \n
16683 \n
16684 Clebsch_surface: \n
16685 $(ORBITER) -v 2 -povray \n
16686 -round 0 -nb_frames_default 30 \n
16687 -output_mask Clebsch_%d_%03d.pov \n
16688 -video_options -W 1024 -H 768 \n
16689 -global_picture_scale 0.9 \n
16690 -default_angle 75 \n
16691 -clipping_radius 2.4 \n
16692 -camera 0 "1,1,1" "-3,1,3" "0.12,0.12,0.12" \n
16693 -end \n
16694 -scene_objects \n
16695 -cubic_orbiter "0,0,0,0,0,-4.236067972,\n
16696 0,0.4.236067972,4.236067972,17.94427188,\n
16697 -17.94427188,0,0,- 9.472135941,0,0,5.236067971,\n
16698 8.472135938,- 27.41640782" \n
16699 -group_of_things "0" \n
16700 -cubics 0 $(SURFACE_COLOR_SEETHROUGH) \n
16701 -point_list_from_csv_file \n
16702 function_Clebsch_map_of_circle_N1000_points.csv \n
16703 -group_of_things_as_interval 0 954 \n
16704 -spheres 1 0.07 $(COLOR_RED) \n
16705 -scene_objects_end \n
16706 -povray_end \n
16707 - rm -rf PO\n
16708 mkdir PO\n
16709 mv Clebsch_0_*.pov PO\n
16710 mv makefile_animation PO\n
16711 \n
16712 \n
16713 Clebsch_surface_defining_equation: \n
16714 $(ORBITER) -v 2 -povray \n
16715 -round 0 -nb_frames_default 30 \n
16716 -output_mask Clebsch_%d_%03d.pov \n
16717 -video_options -W 1024 -H 768 \n
16718 -global_picture_scale 0.6 \n

836
Clebsch_surface_defining_equation_and_curves:

```
$ORBITER $v 2 -povray \
-round 0 -nb_frames_default 30 \
-output_mask Clebsch_2curves_%d%03d.pov \
-video_options -W 1024 -H 768 \
-global_picture_scale 0.6 \
-default_angle 75 \
-clipping_radius 1.6 \
-camera 0 "1,1,1" "-2,0,2" "0,0,0" \
-end \
-scene_objects \
-cubic_orbiter "0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,2,2,2,2" \
-group_of_things "0" \
-cubics 0 $(SURFACE_COLOR_SEETHROUGH) \
-scene_objects_end \
-povray_end
```

rm -rf POV
mkdir POV
mv Clebsch_0_*.pov POV
mv makefile.animation POV

Clebsch_surface_defining_equation_and_curves:

```
\$ORBITER $v 2 -povray \
-round 0 -nb_frames_default 30 \
-output_mask Clebsch_2curves_%d%03d.pov \
-video_options -W 1024 -H 768 \
-global_picture_scale 0.6 \
-default_angle 75 \
-clipping_radius 1.6 \
-camera 0 "1,1,1" "-2,0,2" "0,0,0" \
-end \
-scene_objects \
-cubic_orbiter "0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,2,2,2,2" \
-group_of_things "0" \
-cubics 0 $(SURFACE_COLOR_SEETHROUGH) \
-point_list_from_csv_file \
-function_Clebsch_map_of_circle_to_defininig_eqn_N1000.points.csv \
-group_of_things_as_interval 0 656 \
-spheres 1 0.07 $(COLOR_RED) \
-point_list_from_csv_file \
-function_Clebsch_map_of_circle_to_defininig_eqn_r2_N1000.points.csv \
-group_of_things_as_interval 656 1042 \
-spheres 2 0.07 "texture{ pigment{ color Blue } \nfinish { diffuse 0.9 phong 1}}" \
-scene_objects_end \
povray_end
```

rm -rf POV
mkdir POV
mv Clebsch_2curves_0_*.pov POV
mv makefile_animation POV
-point_list_from_csv_file function_Clebsch_map_of_circle_N1000_points.csv

-group_of_things_as_interval 0 954

-spheres 1 0.07 "texture{ pigment{ color Red } finish { diffuse 0.9 phong 1}}"

F7_povray:

$(ORBITER) -v 2 -povray

-round 0 -nb_frames_default 30

-output_mask F7_15_lines_%d_%03d.pov

-video_options -W 1024 -H 768

-global_picture_scale 1.5

default_angle 80

-clipping_radius 4.4

-omit_bottom_plane

-camera 0 "1,1,1" 

-end

-scene_objects

-cubic_lex "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" 

-group_of_things "0" 

-cubics 0 $(SURFACE_COLOR_SEETHROUGH) 

-line_through_point_with_direction "0, 0, 0, 1, 0, 0" 

-line_through_point_with_direction "0, 0, -1, 0, 1, 0" 

-line_through_point_with_direction "0, 0, 0, 0, -1" 

-line_through_point_with_direction "1, 0, 0, 1, 1, 1" 

-line_through_point_with_direction ",-1.414213562, 0, 0, 4.146264370, 1.732050808, 0 , 0, 0"

-line_through_point_with_direction "0, 1.732050808, -1, 2.414213562, -0.317837246, 2.414213562" 

-line_through_point_with_direction ",-2.133352390, 0, -1, 1.674708020, 1, 0"

-line_through_point_with_direction ",-2.539058015, 0, 0, 2.211360755, 1, 0"

-line_through_point_with_direction ",0, 1.48188060, 0, 0, -0.9435440612, 1"

-line_through_point_with_direction ",-0.971197117, 0, 0, 1.162155272, 0, 1"

-line_through_point_with_direction ",2.096037870, 2.096037870, 0, -1.065851905, -1.065851905, 1"

-line_through_point_with_direction ",3.921555783, 2.921555781, 0, -1.722456585, -1.722456585, 1"

-group_of_things_as_interval 0 12
-lines 1 0.04 $(COLOR_YELLOW) \n-scene_objects_end \npovray_end
- rm -rf POV
mkdir POV
mv F7.15_lines_0.*.pov POV
mv makefile_animation POV

F7_video:
- rm -r FRAMES
mkdir FRAMES
rm fifteen_with_lines.mp4
$(ORBITER) 
-prepare_frames 
-i 0 30 F7b/F7.15_lines_0_%03d.png 
-output_starts_at 0 
-o FRAMES/frame%04d.png 
-end
ffmpeg -r 5 -f image2 -i FRAMES/frame%04d.png 
-f mp4 -q:v 0 -vcodec mpeg4 fifteen_with_lines.mp4

McKean_povray:
$(ORBITER) -v 2 -povray 
-round 0 -nb_frames_default 30 
-output_mask McKean_%d_%03d.pov 
-video_options -W 1024 -H 768 
-global_picture_scale 1.5 
-default_angle 80 
-clip_radius 4.4 
-omit_bottom_plane 
-camera 0 "i,1,1" "-4.5,3.5,6" "0,0,0" 
-end 
-scene_objects 
-cubic_lex "0, 0, 1, 0, 0, -1, -2, 1, \n2, 0, 0, 1, 1,-1, -1, -1, 0, 0, 0, 0" 
-group_of_things "0" 
-cubics 0 $(SURFACE_COLOR_SEETHROUGH) 
-scene_objects_end 
povray_end
- rm -rf POV
mkdir POV
mv McKean_*.pov POV
mv makefile_animation POV

McKean.video:
- rm -r FRAMES
- mkdir FRAMES
- rm McKean.mp4
$(ORBITER) \
  -prepare_frames \
  -i 0 30 MCKEAN/McKean_%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 McKean.mp4

# Section 16.4: Continuous Function Plotter
SECTION CONTINUOUS_FUNCTION_PLOTTER:
lissajous:
$(ORBITER) -v 2 \
  -smooth_curve "lissajous" 0.07 2000 15 0 18.85 \
  -const a 3 b 2 c 1.57 r 7 -const_end \
  -var t -var_end \
  -code \
  push t push a mult push c add sin push r mult return \
  push t push b mult sin push r mult return \
  -code_end \
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 \

#function_lissajous_N2000_points.csv
-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 \ 
-lines 4 0.1 "texture{ pigment{ color Red } \ 
finish { diffuse 0.9 phong 1} }" \ 
-plane_by_dual_coordinates "0,0,1,0" \ 
-group_of_things "0" \ 
-planes 5 "texture{ pigment{ color Blue*0.5 \ 
-transmit 0.5 } }" \ 
-scene_objects_end \ 
-povray_end \ 
-rm -rf POV \ 
mkdir POV \ 
-mv lissajous_0_*.pov POV \ 
-mv makefile_animation POV \ 

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
-camera 0 "0,0,1" "7,7,5" "0,0,1" 
-rotate_about_z_axis
-end
-scene_objects
-line_through_two_points_recentered_from_csv_file
-coordinate_grid.csv
-group_of_things "0"
-group_of_things "1"
-group_of_things "2"
-lines 0 0.09 "texture{ pigment{ color Yellow } }"
-lines 1 0.09 "texture{ pigment{ color Yellow } }"
-lines 2 0.09 "texture{ pigment{ color Yellow } }"
-lines 3 0.02 "texture{ pigment{ color Black } }"
-point_list_from_csv_file
-function_lissajous_3d_N2000_points.csv
-group_of_things_as_interval 3 39
-spheres 4 0.1 "texture{ pigment{ color Red } finish { diffuse 0.9 phong 1}}"
-plane_by_dual_coordinates "0,0,1,0"
-group_of_things "0"
-scene_objects_end
-povray_end
-rm -rf POV
-mkdir POV
-mv lissajous_3d_0.*.pov POV
-mv makefile_animation POV

# Chapter 17 - Miscellaneous
# Section 17.1: Miscellaneous
SECTION_MISCELLANEOUS:
# Section 17.2: Limitations

SECTION LIMITATIONS:

####

# unclassified:
extract:

```bash
$ (ORBITER) -v 3
```

```bash
~/bin/a2tex.out -numbers -text_width 80 <make_Cremona_map.txt >make_Cremona_map.tex
```

draw eigenvalue diag23:

```bash
$ (ORBITER) -v 2
```

```bash
~draw_options
```

```bash
~radius 10
```

```bash
~line_width 1.5 -embedded
```

```bash
~draw_mod_n -n 20
```

```bash
~file ev_diag23
```

```bash
~eigenvalues 2 0 0 3
```

```bash
~end
```

```bash
pdflatex ev_diag23_draw.tex
```

```bash
open ev_diag23_draw.pdf
```
Bibliography

Mathematical Software; James H. Davenport, Manuel Kauers, George Labahn, Josef

[2] Awss Al-ogaidi and Anton Betten. Large Arcs in Small Planes, Congressus Numeran-


[4] Laura Bader, Guglielmo Lunardon, and Joseph A. Thas. Derivation of flocks of


com/abetten/orbiter.

2008.

2013—Proceedings of the 38th International Symposium on Symbolic and Algebraic

Proceedings of the 45th International Symposium on Symbolic and Algebraic Compu-


[57] L. Schläfli. 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.


