User’s Guide
Build Number 1473

Anton Betten

November 9, 2022
Contents

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

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

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

4 Geometry
  4.1 Finite Projective Spaces ....................................... 71
  4.2 Indexing Points and Lines ..................................... 78
  4.3 Finite Desarguesian Projective Planes ......................... 83
  4.4 The Grassmannian ............................................... 87
  4.5 Algebraic Sets .................................................. 90
  4.6 The Klein Quadric and the Plücker Map ......................... 93
  4.7 Orthogonal Spaces ............................................... 95
  4.8 Hermitian Varieties ............................................. 104
  4.9 Advanced Topics ................................................. 106
  4.10 Geometric Objects .............................................. 113
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Group Theory</td>
<td></td>
</tr>
<tr>
<td>5.1</td>
<td>Permutation Groups</td>
<td>119</td>
</tr>
<tr>
<td>5.2</td>
<td>Linear Groups</td>
<td>125</td>
</tr>
<tr>
<td>5.3</td>
<td>Subgroups</td>
<td>143</td>
</tr>
<tr>
<td>5.4</td>
<td>Linear Groups, Advanced Topics</td>
<td>155</td>
</tr>
<tr>
<td>5.5</td>
<td>Induced Actions</td>
<td>159</td>
</tr>
<tr>
<td>5.6</td>
<td>Group Theoretic Activities</td>
<td>170</td>
</tr>
<tr>
<td>5.7</td>
<td>Group Theoretic Activities Based on Magma</td>
<td>178</td>
</tr>
<tr>
<td>6</td>
<td>Orbit Algorithms</td>
<td></td>
</tr>
<tr>
<td>6.1</td>
<td>Schreier Trees</td>
<td>185</td>
</tr>
<tr>
<td>6.2</td>
<td>Poset Classification</td>
<td>194</td>
</tr>
<tr>
<td>6.3</td>
<td>Orbits on Subsets</td>
<td>198</td>
</tr>
<tr>
<td>6.4</td>
<td>Orbits on Subspaces</td>
<td>202</td>
</tr>
<tr>
<td>6.5</td>
<td>Orbits on Set-Partitions</td>
<td>204</td>
</tr>
<tr>
<td>6.6</td>
<td>Arcs and Caps in Projective Spaces</td>
<td>206</td>
</tr>
<tr>
<td>6.7</td>
<td>Cubic Curves</td>
<td>211</td>
</tr>
<tr>
<td>7</td>
<td>Cubic Surfaces</td>
<td></td>
</tr>
<tr>
<td>7.1</td>
<td>Creation</td>
<td>213</td>
</tr>
<tr>
<td>7.2</td>
<td>Quartic Curves</td>
<td>221</td>
</tr>
<tr>
<td>7.3</td>
<td>Classification</td>
<td>233</td>
</tr>
<tr>
<td>7.4</td>
<td>Isomorphism Testing and Recognition</td>
<td>248</td>
</tr>
<tr>
<td>7.5</td>
<td>Dickson Surfaces</td>
<td>252</td>
</tr>
<tr>
<td>7.6</td>
<td>ATLAS and Tables</td>
<td>254</td>
</tr>
<tr>
<td>8</td>
<td>Ring Theory</td>
<td></td>
</tr>
<tr>
<td>8.1</td>
<td>Polynomials Over Finite Fields</td>
<td>255</td>
</tr>
<tr>
<td>8.2</td>
<td>Multivariate Polynomial Rings</td>
<td>264</td>
</tr>
<tr>
<td>9</td>
<td>Applications</td>
<td></td>
</tr>
<tr>
<td>9.1</td>
<td>Number Theory</td>
<td>273</td>
</tr>
<tr>
<td>9.2</td>
<td>Representation Theory</td>
<td>276</td>
</tr>
<tr>
<td>9.3</td>
<td>Cryptography</td>
<td>278</td>
</tr>
<tr>
<td>10</td>
<td>Coding Theory</td>
<td></td>
</tr>
<tr>
<td>10.1</td>
<td>Introduction</td>
<td>287</td>
</tr>
<tr>
<td>10.2</td>
<td>Linear Codes</td>
<td>294</td>
</tr>
<tr>
<td>10.3</td>
<td>Golay Codes</td>
<td>302</td>
</tr>
<tr>
<td>10.4</td>
<td>CRC Codes</td>
<td>303</td>
</tr>
<tr>
<td>10.5</td>
<td>Reed-Muller Codes</td>
<td>311</td>
</tr>
<tr>
<td>10.6</td>
<td>BCH Codes</td>
<td>313</td>
</tr>
<tr>
<td>10.7</td>
<td>Reed-Solomon Codes</td>
<td>320</td>
</tr>
<tr>
<td>10.8</td>
<td>Bounds</td>
<td>325</td>
</tr>
<tr>
<td>10.9</td>
<td>Classification of Optimal Linear Codes</td>
<td>327</td>
</tr>
<tr>
<td>Chapter</td>
<td>Title</td>
<td>Pages</td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>11</td>
<td>Combinatorics</td>
<td>331-496</td>
</tr>
<tr>
<td>11.1</td>
<td>Introduction</td>
<td>331</td>
</tr>
<tr>
<td>11.2</td>
<td>Diophantine Systems</td>
<td>338</td>
</tr>
<tr>
<td>11.3</td>
<td>Combinatorial Linear Spaces</td>
<td>342</td>
</tr>
<tr>
<td>11.4</td>
<td>Classification of Configurations and Geometries</td>
<td>345</td>
</tr>
<tr>
<td>11.5</td>
<td>Design Theory</td>
<td>354</td>
</tr>
<tr>
<td>11.6</td>
<td>Design Theory – Large Sets</td>
<td>359</td>
</tr>
<tr>
<td>11.7</td>
<td>Design Theory – Delandtsheer-Doyen</td>
<td>361</td>
</tr>
<tr>
<td>11.8</td>
<td>Tactical Decompositions</td>
<td>363</td>
</tr>
<tr>
<td>12</td>
<td>Finite Geometry</td>
<td>367-467</td>
</tr>
<tr>
<td>12.1</td>
<td>Spreads</td>
<td>367</td>
</tr>
<tr>
<td>12.2</td>
<td>Translation Planes</td>
<td>381</td>
</tr>
<tr>
<td>12.3</td>
<td>Packings</td>
<td>385</td>
</tr>
<tr>
<td>12.4</td>
<td>BLT-Sets</td>
<td>388</td>
</tr>
<tr>
<td>13</td>
<td>Graph Theory</td>
<td>395-458</td>
</tr>
<tr>
<td>13.1</td>
<td>Creating Graphs</td>
<td>395</td>
</tr>
<tr>
<td>13.2</td>
<td>Graph Theoretic Activities</td>
<td>402</td>
</tr>
<tr>
<td>13.3</td>
<td>Classification of Graphs and Tournaments</td>
<td>408</td>
</tr>
<tr>
<td>13.4</td>
<td>Clique Finding</td>
<td>412</td>
</tr>
<tr>
<td>14</td>
<td>Combinatorial Objects</td>
<td>417-464</td>
</tr>
<tr>
<td>14.1</td>
<td>Combinatorial Objects</td>
<td>417</td>
</tr>
<tr>
<td>14.2</td>
<td>Encoding Combinatorial Objects</td>
<td>421</td>
</tr>
<tr>
<td>15</td>
<td>Canonical Forms with Nauty</td>
<td>425-488</td>
</tr>
<tr>
<td>15.1</td>
<td>Overview of Canonical Forms</td>
<td>425</td>
</tr>
<tr>
<td>15.2</td>
<td>Canonical Forms of Objects in Projective Space</td>
<td>427</td>
</tr>
<tr>
<td>15.3</td>
<td>Canonical Forms of Incidence Geometries</td>
<td>434</td>
</tr>
<tr>
<td>15.4</td>
<td>Canonical Forms of Objects from Design Theory</td>
<td>447</td>
</tr>
<tr>
<td>15.5</td>
<td>Canonical Forms of Linear Codes</td>
<td>451</td>
</tr>
<tr>
<td>15.6</td>
<td>Canonical Forms of General Codes</td>
<td>458</td>
</tr>
<tr>
<td>15.7</td>
<td>Canonical Forms of Graphs</td>
<td>459</td>
</tr>
<tr>
<td>15.8</td>
<td>Canonical Forms of Quartic Curves</td>
<td>464</td>
</tr>
<tr>
<td>16</td>
<td>Interfaces</td>
<td>467-488</td>
</tr>
<tr>
<td>16.1</td>
<td>Graphical Output</td>
<td>467</td>
</tr>
<tr>
<td>16.2</td>
<td>The Povray Interface</td>
<td>474</td>
</tr>
<tr>
<td>16.3</td>
<td>Creating Animations</td>
<td>487</td>
</tr>
<tr>
<td>16.4</td>
<td>Continuous Function Plotter</td>
<td>488</td>
</tr>
<tr>
<td>17</td>
<td>Miscellaneous</td>
<td>493-496</td>
</tr>
<tr>
<td>17.1</td>
<td>Miscellaneous</td>
<td>493</td>
</tr>
<tr>
<td>17.2</td>
<td>Limitations</td>
<td>496</td>
</tr>
</tbody>
</table>
18 Orbiter on Windows 497
18.1 Using Windows Subsystem Linux  . . . . . . . . . . . . . . . . . . . . 497

19 The Makefile 507
19.1 The Makefile  . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 507
References  . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 903
Chapter 1

Introduction

1.1 What is Orbiter

Orbiter is a computer algebra system for the classification of combinatorial objects. Orbiter contributes to the knowledge base of combinatorial structures, and to provide useful tools to investigate structures from various points of view, including their symmetry properties. Orbiter is optimized for efficiency in terms of memory and execution speed. Orbiter is a library of C++ classes, together with a command line driven front end. There is no graphical user interface. The system offers two modes of use, programming or command line interface. This manual is about the command line interface. Readers who are interested in the Orbiter C++ class library should consult the programmer’s guide. A makefile with all commands used in this guide can be found in the examples subdirectory. For background on Orbiter, see [9].
Chapter 2

Getting Started

2.1 Running and Installing Orbiter

There are two ways to run Orbiter: Native and Docker. Native means that Orbiter is compiled from scratch, using the source code from the github repository (cf. [10]). Docker [25] is a system to run preconfigured software in an encapsulated way on various platform, including Windows. We describe using Orbiter through unix makefiles, which are run through the tool make (cf. [28]). This is a software tool that allows collecting short command snippets in the form of text files that can easily be handled. However, the conventions in the tool involve some subtleties regarding the use of whitespace, which can cause problems to novice users. We will point out possible pitfalls along the way. Note that it is not necessary to use makefiles. Another possibility would be to use shell scripts. Ultimately, it would be possible to type out all commands into a terminal window. This could be a little tedious though, considering the fact that most Orbiter commands expect lengthy parameters from the command line.

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

```
make
```

Once compiled, the Orbiter executable is

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

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

```
|--- orbiter
|   |--- work
```

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

test:
  $(ORBITER_PATH)orbiter.out

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

make test

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

test:
  $(ORBITER_PATH)orbiter.out

is the makefile target “test.” It is important that the indentation after the makefile target is done using tab characters (no spaces). There can be multiple targets in one makefile, as long as they are separated by an empty line. For more information about the syntax of makefiles, see [28].

A second way to run Orbiter is through Docker [25]. This does not require a compile environment. However, it comes at a small performance cost when running Orbiter commands that are computationally heavy. Orbiter has already been precompiled (by the Orbiter developer) into an image, which is a completely self-sustained copy of a unix-environment that can run by the user under the docker front-end. The image is stored on a docker server under the name abetten/orbiter. Docker will receive the name of the image from the command line, pull a local copy of the image, and run the image in an encapsulated environment called a container. A copy of the image is stored locally, so that subsequent calls to Orbiter can be satisfied using the local copy, which increases turnaround speed. For instance, the following bare-bones makefile sets up Orbiter for use through Docker:

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

test:
  $(ORBITER_PATH)orbiter.out

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

make test

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

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

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

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

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

ORBITER_PATH=~/DEV.22/orbiter/
ORBITER=$(ORBITER_PATH)orbiter.out
SANDBOX=$(ORBITER_PATH)sandbox/sandbox.out
```

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

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

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

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

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

This will produce an output similar to the following:

```
sh-3.2$ docker run -it --volume $PWD:/mnt -w /mnt abetten/orbiter orbiter.out
Unable to find image 'abetten/orbiter:latest'
latest: Pulling from abetten/orbiter
004f1ee8d7df: Pull complete
5d6f1e8117db: Pull complete
48c2f6f66abe: Pull complete
234b70d0479d: Pull complete
6fa07e00e2f0: Pull complete
9187bd98e241: Pull complete
ae87b7ef500b: Pull complete
260a2765fa99: Pull complete
27d6ff93a58: Pull complete
7a09ec574a18: Pull complete
1336494f74e1: Pull complete
Digest: sha256:889099d7e0b0a9ee168b7cb261d2da8ff64bd7861c357e1caec59580d629ee9
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:

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

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

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

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

(c) Issue the following commands to compile Orbiter:

```
make
make install
```

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

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

Table 2.1: Orbiter session commands
The build number is the version number of the Orbiter software, as defined by the number of commits 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

\[
A = "I am a variable"
\]

and used anywhere later using the

\$(A)

syntax. Rules are defined using the following syntax

Label:
    Do something

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

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

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

The purpose of this command is to produce a file called

```
GF_16.tex
```

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

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

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

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

A sample makefile with all of the commands discussed in this user’s guide is distributed with Orbiter (in the examples directory). The file is reproduced in Section 19.1. It is advised to copy the example makefile from the orbiter tree to a location outside the orbiter distribution directory (otherwise, git update will cause error messages). It is also fine to create a new custom makefile, considering the remarks about **ORBITER_PATH** below.

One difficulty in installing Orbiter is the path of installation. In the sample makefile, there is a makefile variable called **ORBITER_PATH** which contains the path to the orbiter executable **orbiter.exe**. Depending on the local installation of orbiter, the makefile variable needs to be changed accordingly. The actual command to run the **F_16** example is as follows:
F_16:
   $(ORBITER_PATH)orbiter.out -v 3 -define F -finite_field -q 16 -end \ 
   -with F -do -finite_field_activity -cheat_sheet_GF -end

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

MY_PATH=../orbiter

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

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

to

ORBITER_PATH=

in the makefile.
2.4 Objects and Activities

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

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

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

The syntax to create an object is

```
 DEFINE LABEL KEYWORD PARAMETERS -end
```

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

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

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

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

creates the same finite field $F_2$ and uses it to construct PG(3, 2). The `projective_space` command requires additional options to set the dimension $n$ and the field $F_q$ in PG($n$, $q$). The `-n` command sets the dimension $n$. The `-field` command can be used to specify a praticular field. The `-q` command can be used to create a generic field $F_q$.

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

```
 WITH LABEL -do DESCRIPTION -end
```
<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-finite_field</td>
<td>A finite field $\mathbb{F}_q$. See Sections 3.2 and 3.3.</td>
</tr>
<tr>
<td>-polynomial_ring</td>
<td>A multivariate polynomial ring. See Section 8.2.</td>
</tr>
<tr>
<td>-linear_group</td>
<td>A linear group. See Section 5.2.</td>
</tr>
<tr>
<td>-permutation_group</td>
<td>A permutation group. See Section 5.1.</td>
</tr>
<tr>
<td>-projective_space</td>
<td>A projective space $\text{PG}(n, q)$. See Section 4.1.</td>
</tr>
<tr>
<td>-orthogonal_space</td>
<td>A non-degenerate orthogonal space $\mathcal{O}(n, q)$. See Section 4.7.</td>
</tr>
<tr>
<td>-BLT_set_classify</td>
<td>An object to classify BLT-sets. See Section 12.4.</td>
</tr>
<tr>
<td>-spread_classify</td>
<td>An object to classify spreads. See Section 12.1.</td>
</tr>
<tr>
<td>-formula</td>
<td>An algebraic / symbolic expression. See Section 2.8.</td>
</tr>
<tr>
<td>-cubic_surface</td>
<td>A cubic surface. See Section 7.1.</td>
</tr>
<tr>
<td>-quartic_curve</td>
<td>A quartic curve. See Section 7.2.</td>
</tr>
<tr>
<td>-classification_of_cubic_surfaces_with_double_sixes</td>
<td>An object to classify cubic surfaces using double sixes. See Section 7.3.</td>
</tr>
<tr>
<td>-collection</td>
<td>A collection of objects.</td>
</tr>
<tr>
<td>-geometric_object</td>
<td>A geometric object. See Section 4.10.</td>
</tr>
<tr>
<td>-graph</td>
<td>A graph. See Section 13.1.</td>
</tr>
<tr>
<td>-code</td>
<td>A code. See Section 10.2.</td>
</tr>
<tr>
<td>-spread</td>
<td>A spread. See Section 12.1.</td>
</tr>
</tbody>
</table>

Table 2.2: Orbiter Objects (Part I)
<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-translation_plane</td>
<td>A translation plane. See Section 12.2.</td>
</tr>
<tr>
<td>-spread_table</td>
<td>A table of spreads. See Section 12.3.</td>
</tr>
<tr>
<td>-packing_with_symmetry_assumption</td>
<td>A generator for packings with assumed symmetry. See Section 12.3.</td>
</tr>
<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>
<tr>
<td>-geometry_builder</td>
<td>An object to classify incidence geometries. See Section 11.4.</td>
</tr>
<tr>
<td>-vector_ge</td>
<td>A vector of group elements. See Section 5.3.</td>
</tr>
</tbody>
</table>

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

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

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

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

Table 2.4: Orbiter Activities
<table>
<thead>
<tr>
<th>Object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLT-sets</td>
<td>BLT sets of $Q(4, q)$ exist for all odd prime powers. The classification of BLT-sets of $Q(4, q)$ is known to Orbiter for all $q \leq 73$.</td>
</tr>
<tr>
<td>Cubic Surfaces</td>
<td>Cubic surfaces with 27 lines exist for all finite fields apart from $\mathbb{F}_2$, $\mathbb{F}_3$, $\mathbb{F}_5$. Orbiter knows the classification of cubic surfaces with 27 lines for all fields $\mathbb{F}_q$ of order $q \leq 128$.</td>
</tr>
<tr>
<td>Quartic curves</td>
<td>Orbiter knows the classification of smooth quartic curves with 28 bitangents in projective planes over all fields $\mathbb{F}_q$ for $q = 9, 13, 19, 25, 27, 29, 31$.</td>
</tr>
<tr>
<td>Spreads</td>
<td>A spread is a set of $q^k + 1$ pairwise non-intersecting $k$-dimensional subspaces of $\mathbb{F}_q^{2k}$. Spreads are related to translation planes of order $q^k$. Orbiter knows the classification of spreads for $(q, k) \in {(2, 2), (3, 2), (2, 4), (4, 2), (5, 2), (3, 3)}$.</td>
</tr>
<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 2.5: Mathematical Data Available in Orbiter

### 2.5 Mathematical Data

Orbiter serves as a repository for mathematical data. The knowledge base is concerned with classifications of geometric and combinatorial objects for small parameters. The types of objects for which a classification is available in Orbiter are listed in Table 2.5. The mathematical objects are stored in a catalogue, together with generators for their automorphism groups. The objects are indexed by a zero-based integer, called the *Orbiter Catalogue Number* (OCN). It is possible to access any object in the catalogue. Let us consider some examples:

The command

```
create_BLT_5_1:
  $\text{(ORBITER)} -v 2 \n  \text{define F -finite_field -q 5 -end} \n  \text{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 \texttt{catalogue_q5_iso1.tex} is written. For more details about BLT-sets, see Section 12.4.

The command

\begin{verbatim}
create_surface_4_0:
  $(ORBITER) -v 3 \
  -define F -finite_field -q 4 -end \
  -define P -projective_space -n 3 -field F -v 0 -end \
  -define S4_0 -cubic_surface -space P -catalogue 0 -end \
  -with S4_0 -do \
  -cubic_surface_activity \
  -report \ 
  -end
\end{verbatim}

recalls the cubic surface with Orbiter Catalogue Number 0 in $PG(3,4)$. A latex report \texttt{surface_catalogue_q4_iso0_report.tex} is written. For more details about cubic surfaces, see Section 7.1.
2.6 Set Builder

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

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

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

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

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

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

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

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

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

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

```
vector_example1:
▷ $(ORBITER) -v 2 \n▷ ▷ -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 \n▷ ▷ -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="\
1,0,0,0,0,1,1, \
0,1,0,0,1,0,1, \
0,0,1,0,1,1,0, \
0,0,0,1,1,1,1" \\
```

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="\
110111000100001010000\n11110111110100001011\n00000100000100010101\n11110101100010011110\n01010100000010011110\n00000100000100010101\n00100000000010010101\n00010001100000100111\n11101011110100111011\n00000000001001000101\n00000001000100010101\n01101111101100111111\n00000000000011001011\n00000001001101000101\n01101111101001010101\n01010101100000100111\n00000000000001000111\n00000001001111111101\n00000000000001000101\n00000000000001000101
```

29
matrix_example_co_1:
  » $(ORBITER) -v 2 \n  »   -define F -finite_field -q 2 -end \n  »   -define v -vector -field F -format 22 \n  »   -compact $(CONWAY_GEN1) -end \n  »   -print_symbols

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

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

creates a vector of length 20 and sets the 0-th and the 19-th coefficient to 1. Finally, the vector is displayed as a four-rowed matrix:

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

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

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

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

\[(0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2)\].

In order to create a constant vector, the -repeat command can be used as well. Simply use a repeat sequence consisting of a single number. For instance, the following example creates the all-one vector of length 11:
vector_example_all_one_11:
\[
\texttt{\textdollar\text{\$(ORBITER) -v 2 \textbackslash
\textbackslash$
\textdollar\textbackslash
\text{\textdollar\textend{quote}}
\textdollar\textbackslash
\text{\textdollar\textend{quote}}
\textdollar\textbackslash
\text{\textdollar\textend{quote}}
\textdollar\textbackslash
\text{\textdollar\textend{quote}}
\textdollar\textbackslash
\text{\textdollar\textend{quote}}}
\]

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

\[(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1).\]
Figure 2.4: The abstract syntax tree of the formula

2.8 Formula Builders

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

```
TEST_FORMULA="(-a+b*b)*x*x+a*b*x"
```

The command

```
$(ORBITER) -v 3 -define f -formula "test\_formula" "test\_formula" "" -formula "test\_formula" "test\_formula" "" \\
n $(TEST_FORMULA) -Tpng test\_formula.gv >test\_formula.png \\
open test\_formula.png
```

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

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

```
formula\_evaluate: \\
$(ORBITER) -v 3 \\
```

```
-define F -finite_field -q 5 -end \
-define f -formula \
  "test_formula" "test\_formula" "" \
  $(TEST\_FORMULA) \
  -with F -do -finite_field_activity \
  -evaluate f "a=2,b=3,x=4" -end
Chapter 3
Basic Algebra

3.1 Basic Number Theory

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

\texttt{PR29:}
\begin{verbatim}
 $\langle ORBITER \rangle \text{-v 1 -smallest\_primitive\_root 29}
\end{verbatim}

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

\texttt{PR\_915839:}
\begin{verbatim}
 $\langle ORBITER \rangle \text{-v 5 -primitive\_root 915839}
\end{verbatim}

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

\texttt{PR\_915839\_check:}
\begin{verbatim}
 $\langle ORBITER \rangle \text{-v 5 -power\_mod 43085 49842 915839}
\end{verbatim}

computes
\[ 43085^{49842} \equiv a \mod 915839 \]
which is 487320.

The command \texttt{-discrete\_log} can be used to compute the discrete logarithm of \( a \) modulo
\( p \) with respect to \( b \). This means, a number \( k \) is computed such that
\[ b^k \equiv a \mod p. \]

For instance, the discrete log of 487320 with respect to the base 43085 modulo 915839 is
49842, based on the previous example. We can compute the discrete logarithm using the
command
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-power_mod</td>
<td>a n p</td>
<td>Computes $a^n \mod p$.</td>
</tr>
<tr>
<td>-discrete_log</td>
<td>b a p</td>
<td>Computes $n$ such that $a^n \equiv b \mod p$.</td>
</tr>
<tr>
<td>-extended_gcd</td>
<td>a b</td>
<td>Computes integers $g, u,$ and $v$ such that $g = \gcd(a, b) = ua + vb$.</td>
</tr>
<tr>
<td>-square_root_mod</td>
<td>a p</td>
<td>Computes a square root of $a$ modulo $p$, i.e. an integer $b$ such that $b^2 \equiv a \mod p$.</td>
</tr>
<tr>
<td>-square_root</td>
<td>a</td>
<td>Computes $\lfloor \sqrt{a} \rfloor$ of an integer $a$.</td>
</tr>
<tr>
<td>-inverse_mod</td>
<td>a p</td>
<td>Computes the modular inverse of $a$ modulo $p$, i.e. an integer $b$ with $ab \equiv 1 \mod p$.</td>
</tr>
<tr>
<td>-draw_mod_n</td>
<td>descr</td>
<td>Draws the integers modulo $n$ on a circle.</td>
</tr>
<tr>
<td>-order_of_q_mod_n</td>
<td>q n_min n_max</td>
<td>Computes the order $\text{ord}(q, n)$ of $q$ modulo $n$ for all $n$ with $n_{\text{min}} \leq n \leq n_{\text{max}}$ for which $\gcd(n, q) = 1$. Also computes $\varphi(n)$ and $\varphi(n)/\text{ord}(q, n)$.</td>
</tr>
<tr>
<td>-Chinese_remainders</td>
<td>R M</td>
<td>Solves a system of congruences with remainders $R$ and moduli $M$. $R$ and $M$ must be vectors whose labels are given.</td>
</tr>
</tbody>
</table>

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

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

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

computes the inverse of 1865025205 modulo 2147483647 which is 579785381.

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

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

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 \( b = 1865025205 \).

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

The output is

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

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

In order to compute the modular power

\[ a^e \mod n, \]

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

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

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

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

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

The command \texttt{-square\_root\_mod} can be used to compute modular square roots using the algorithm of Tonelli and Shanks (cf. [19]). For instance,
sqrt_mod:
▷ $(\text{ORBITER}) -v 2 -\text{s}
\begin{array}{c}
\text{\path{\text{\textsuperscript{\texttt{square_root_mod 33 41}}}}}\end{array}$

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

$$19^2 \equiv 33 \pmod{41}.$$

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

\texttt{order_of_2_mod_n:}
▷ $(\text{ORBITER}) -v 3 -\text{order_of_q_mod_n 2 3 151}$
▷ $(\text{ORBITER}) -v 1 -\text{csv_file_latex 1 }$
▷ ▷ $(\text{\texttt{order_of_q_mod_n_q2_3_151.csv}})$
▷ $(\text{pdflatex order_of_q_mod_n_q2_3_151.tex})$
▷ $(\text{open order_of_q_mod_n_q2_3_151.pdf})$

produces the output shown in Table 3.2.

The command

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|}
\hline
N & ORD & PHI & COF & N & ORD & PHI & COF & N & ORD & PHI & COF \\
\hline
3 & 2 & 2 & 1 & 53 & 52 & 52 & 1 & 103 & 51 & 102 & 2 \\
5 & 4 & 4 & 1 & 55 & 20 & 40 & 2 & 105 & 12 & 48 & 4 \\
7 & 3 & 6 & 2 & 57 & 18 & 36 & 2 & 107 & 106 & 106 & 1 \\
9 & 6 & 6 & 1 & 59 & 58 & 58 & 1 & 109 & 36 & 108 & 3 \\
11 & 10 & 10 & 1 & 61 & 60 & 60 & 1 & 111 & 36 & 72 & 2 \\
13 & 12 & 12 & 1 & 63 & 6 & 36 & 6 & 113 & 28 & 112 & 4 \\
15 & 4 & 8 & 2 & 65 & 12 & 48 & 4 & 115 & 44 & 88 & 2 \\
17 & 8 & 16 & 2 & 67 & 66 & 66 & 1 & 117 & 12 & 72 & 6 \\
19 & 18 & 18 & 1 & 69 & 22 & 44 & 2 & 119 & 24 & 96 & 4 \\
21 & 6 & 12 & 2 & 71 & 35 & 70 & 2 & 121 & 110 & 110 & 1 \\
23 & 11 & 22 & 2 & 73 & 9 & 72 & 8 & 123 & 20 & 80 & 4 \\
25 & 20 & 20 & 1 & 75 & 20 & 40 & 2 & 125 & 100 & 100 & 1 \\
27 & 18 & 18 & 1 & 77 & 30 & 60 & 2 & 127 & 7 & 126 & 18 \\
29 & 28 & 28 & 1 & 79 & 39 & 78 & 2 & 129 & 14 & 84 & 6 \\
31 & 5 & 30 & 6 & 81 & 54 & 54 & 1 & 131 & 130 & 130 & 1 \\
33 & 10 & 20 & 2 & 83 & 82 & 82 & 1 & 133 & 18 & 108 & 6 \\
35 & 12 & 24 & 2 & 85 & 8 & 64 & 8 & 135 & 36 & 72 & 2 \\
37 & 36 & 36 & 1 & 87 & 28 & 56 & 2 & 137 & 68 & 136 & 2 \\
39 & 12 & 24 & 2 & 89 & 11 & 88 & 8 & 139 & 138 & 138 & 1 \\
41 & 20 & 40 & 2 & 91 & 12 & 72 & 6 & 141 & 46 & 92 & 2 \\
43 & 14 & 42 & 3 & 93 & 10 & 60 & 6 & 143 & 60 & 120 & 2 \\
45 & 12 & 24 & 2 & 95 & 36 & 72 & 2 & 145 & 28 & 112 & 4 \\
47 & 23 & 46 & 2 & 97 & 48 & 96 & 2 & 147 & 42 & 84 & 2 \\
49 & 21 & 42 & 2 & 99 & 30 & 60 & 2 & 149 & 148 & 148 & 1 \\
51 & 8 & 32 & 4 & 101 & 100 & 100 & 1 & 151 & 15 & 150 & 10 \\
\hline
\end{tabular}
\caption{The order of 2 modulo $n$}
\end{table}
Table 3.3: The values of the Euler function

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

Eulerfunction\_150:

- $(\text{ORBITER})$ -v 1 -eulerfunction\_interval 1 150
- $(\text{ORBITER})$ -v 1 -csv\_file\_latex 1 \
- $\text{table}\_\text{eulerfunction}\_1\_150.csv$
- pdflatex $\text{table}\_\text{eulerfunction}\_1\_150.tex$
- open $\text{table}\_\text{eulerfunction}\_1\_150.pdf$

computes Euler's totient function for all integers $n$ with $1 \leq n \leq 150$. The result is shown in Table 3.3.

A power map sends $a$ to $a^k$ for some fixed $k$. Orbiter can compute power maps modulo $p$.

For instance, the following command computes the function $a \mapsto a^k \mod 11$:

power\_function\_2\_mod\_11:

- $(\text{ORBITER})$ -v 5 -power\_function\_mod\_n 2 11
- $(\text{ORBITER})$ -v 1 -csv\_file\_latex 1 power\_function\_k2\_n11.csv
- pdflatex power\_function\_k2\_n11.tex
- open power\_function\_k2\_n11.pdf

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

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

Figure 3.1: Cycle of powers of 2 modulo 13

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

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

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

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

\[
\begin{align*}
x & \equiv 2 \mod 5 \\
x & \equiv 2 \mod 6 \\
x & \equiv 5 \mod 7
\end{align*}
\]

40
The following command creates one vector for the remainders and one for the moduli and then invokes the \texttt{-Chinese_remainders} command.

\textbf{Chinese\_remainders\_A:}

\begin{verbatim}
$\texttt{(ORBTER) -v 2 \n-define R -vector -dense "2,2,5" -end \n-define M -vector -dense "5,6,7" -end \n-Chinese\_remainders R M}
\end{verbatim}

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

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

\[
\begin{align*}
x & \equiv 2 \pmod{2147483647} \\
x & \equiv 3 \pmod{5915587277}
\end{align*}
\]

using the command

\textbf{Chinese\_remainders\_C2:}

\begin{verbatim}
$\texttt{(ORBTER) -v 2 \n-define R -vector -dense "2,3" -end \n-define M -vector -dense "2147483647,5915587277" -end \n-Chinese\_remainders R M}
\end{verbatim}

The answer is

\[
x \equiv 5684294357108828365 \pmod{12703626939758759219}.
\]

A quick check with Maple shows that

\[
\begin{align*}
5684294357108828365 & \pmod{2147483647} \equiv 2 \\
5684294357108828365 & \pmod{5915587277} \equiv 3
\end{align*}
\]

as required.
### 3.2 Finite Fields

Let $\mathbb{F}_q$ denote the finite field with $q$ elements. Up to isomorphism, there is only one field of order $q$. See Section 17.2 for a list of limitations of Orbiter. 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. For instance,

\begin{verbatim}
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
\end{verbatim}

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. The command

\begin{verbatim}
F.7:
  $ (ORBITER) -v 3 \n  $ define F -finite_field -q 7 -end \n  $ with F -do -finite_field_activity -cheat_sheet_GF -end
  pdflatex GF.7.tex
  open GF.7.pdf
\end{verbatim}

creates a cheat sheet for $\mathbb{F}_7$ as shown below. The element $\alpha$ is a primitive element.
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-cheat_sheet_GF</td>
<td></td>
<td>Produce a cheat sheet in latex which shows information about the field, including addition and multiplication tables.</td>
</tr>
<tr>
<td>-product_of v</td>
<td>v</td>
<td>Compute the product of all field elements in the vector v.</td>
</tr>
<tr>
<td>-sum_of v</td>
<td>v</td>
<td>Compute the sum of all field elements in the vector v.</td>
</tr>
<tr>
<td>-negate v</td>
<td>v</td>
<td>Negate each field element in the vector v.</td>
</tr>
<tr>
<td>-inverse v</td>
<td>v</td>
<td>Compute the multiplicative inverse of each field element in the vector v.</td>
</tr>
<tr>
<td>-power_map k v</td>
<td>k v</td>
<td>Compute the k-th power of each field element in the vector v.</td>
</tr>
</tbody>
</table>

Table 3.6: Finite Field Activities

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

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

\[ + \]

\[
\begin{array}{cccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 \\
0 & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\
1 & 1 & 2 & 3 & 4 & 5 & 6 & 0 \\
2 & 2 & 3 & 4 & 5 & 6 & 0 & 1 \\
3 & 3 & 4 & 5 & 6 & 0 & 1 & 2 \\
4 & 4 & 5 & 6 & 0 & 1 & 2 & 3 \\
5 & 5 & 6 & 0 & 1 & 2 & 3 & 4 \\
6 & 6 & 0 & 1 & 2 & 3 & 4 & 5 \\
\end{array}
\]
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 \texttt{product\_of} finite field activity to compute the product of these elements. The answer is 10 which is congruent to \(-1\) mod 11:

\[
\text{\texttt{F\_11\_product\_of\_all\_nonzero\_elements:}} \quad \texttt{\$ (ORBITER) -v 3 \}} \quad \texttt{\}$ (ORBITER) -v 3 \}} \quad \texttt{-define F -finite\_field -q 11 -end \}} \quad \texttt{-define S -vector -field F -loop 1 11 1 -end \}} \quad \texttt{-with F -do -finite\_field\_activity -product\_of S -end \}}
\]

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

\[
\text{\texttt{F\_7\_vandermonde:}} \quad \texttt{\$ (ORBITER) -v 3 \}} \quad \texttt{\}$ (ORBITER) -v 3 \}} \quad \texttt{-define F -finite\_field -q 7 -end \}} \quad \texttt{-with F -do -finite\_field\_activity \}} \quad \texttt{-Vandermonde\_matrix \}} \quad \texttt{-end \}}
\]

The output is shown below. The first matrix is \( V = (x_j^i) \). The second matrix is \( V^{-1} \):
There is a second ordering of the elements which is used occasionally. In this labeling, every non-zero element is written as a power of a fixed primitive element. So, if $\alpha$ is a primitive element, we arrange the elements of $\mathbb{F}_p$ as $0, 1, \alpha, \alpha^2, \ldots, \alpha^{q-2}$.

The cheat sheet contains this list of field elements at the very end. In Figure 3.2, the addition and multiplication tables of $\mathbb{F}_7$ are shown with respect to the cyclic ordering of elements as $0, 3^0, 3^1, 3^2, \ldots, 3^6 = 0, 1, 3, 2, 6, 4, 5, 1$.

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

For small field orders, the Orbiter employs precomputed tables for the arithmetic operations such as addition and multiplication and computing inverses. Precomputing these tables can be time-consuming. The option `-without_tables` can be given to avoid precomputing tables. Here is an example. We create the field $\mathbb{F}_{101}$ without precomputed tables:

```bash
F_101.wo:
$ (ORBITER) -v 3 \
```
\define \F \finite_field \q 101 \without_tables \end \\
\define \F \with \F \do \finite_field_activity \cheat_sheet_{\GF} \end \\
pdflatex GF_101.tex \\
open GF_101.pdf
3.3 Extension Fields

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

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

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

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

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

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

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

$$0,$$

$$1,$$

$$X,$$

$$X + 1.$$  

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

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

47
The addition of polynomials is as in $\mathbb{F}_2[X]$, so

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

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

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

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

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

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

Table 3.7 lists Orbiter activities for finite fields. This extends Table 3.6 in Section 3.3.

The isomorphism type of the resulting field only depends on the order $q$ of the field, and not on the choice of the polynomial. However, for practical computations, the choice of the polynomial matters. For instance, results can only be shared between different computer algebra
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-trace</td>
<td></td>
<td>Computes the partition of the field elements</td>
</tr>
<tr>
<td></td>
<td></td>
<td>according to the value of their absolute trace.</td>
</tr>
<tr>
<td>-norm</td>
<td></td>
<td>Computes the partition of the field elements</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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 $q \leq 1024$
The field $\mathbb{F}_4$:
\[
\text{\$\text{(ORBITER)} -v 3 \\backslash
\text{-define F -finite_field -q 4 -end \backslash
\text{-with F -do -finite_field_activity -cheat_sheet_GF -end}
\backslash
\text{pdflatex GF.4.tex
\backslash
\text{open GF.4.pdf}
\]
\]
creates a report for the field $\mathbb{F}_4$. The command

The field $\mathbb{F}_{16}$:
\[
\text{\$\text{(ORBITER)} -v 3 \\backslash
\text{-define F -finite_field -q 16 -end \backslash
\text{-with F -do -finite_field_activity -cheat_sheet_GF -end}
\backslash
\text{pdflatex GF.16.tex
\backslash
\text{open GF.16.pdf}
\]
\]
creates a cheat sheet for $\mathbb{F}_{16}$. This command produces Table 3.9.

Unlike other computer algebra systems (GAP [29] and Magma [14]), Orbiter does not use Conway polynomials to create field extensions. Instead, it provides the option to override
<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]

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

The polynomial used to create the finite field. For subfield relationships, the cheat sheet will indicate the irreducible polynomials of all subfields for a given field. For instance, Table 3.10 shows the subfields of $\mathbb{F}_{64}$ generated by the polynomial $X^6 + X^5 + 1$ whose numerical rank is 97.

The lexicographic ordering has an interesting side-effect for the ordering of elements in extension fields. The elements of the prime subfield are always listed before any other elements in the extension field. For this reason, the addition and multiplication tables of the extension field contain the respective table of the prime field in the upper left corner. For instance, in Figure 3.4, we find the tables for $\mathbb{F}_3$ in the upper left corners of the tables of $\mathbb{F}_9$, for instance. Recall that omit the zero element in the multiplication tables.

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

Table 3.11: Finite Field Activities for Linear Algebra

### 3.4 Linear Algebra Over Finite Fields

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

```
RREF:
▷ $(ORBITER) -v 2 \n▷▷ -define F -finite_field -q 2 -end \n▷▷ -define v -vector -field F -format 2 \n▷▷▷ -dense "1,1,1,0,1,1,0,0,1" \n▷▷ -end \n▷▷ -with F -do -finite_field_activity \n▷▷ -RREF v -normalize_from_the_right \n▷▷ -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{-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{-RREF} command:

\begin{verbatim}
V7_VANDERMONDE_EXTENDED="\n1,0,0,0,0,0,1,0,0,0,0,0,0,0,  
1,1,1,1,1,1,0,0,1,0,0,0,0,0,  
1,2,4,1,2,4,1,0,0,1,0,0,0,0,  
1,3,2,6,4,5,1,0,0,1,0,0,0,0,  
1,4,2,1,4,2,1,0,0,0,1,0,0,0,  
1,5,4,6,2,3,1,0,0,0,0,1,0,0,  
1,6,1,6,1,6,1,0,0,0,0,0,0,0,0,1"
\end{verbatim}

\begin{verbatim}
RREF_V7:  
  \$\text{ORBITER} -v 2 \  
  \$ \$\text{define F -finite_field -q 7 -end} \  
  \$ \$\text{define V -vector -format 7} \  
  \$ \$\text{dense $(V7_VANDERMONDE_EXTENDED)$} \  
  \$ \$\text{end} \  
  \$ \$\text{with F -do -finite_field_activity} \  
  \$ \$\text{-RREF V} \  
  \$ \$\text{end}
\end{verbatim}

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

\begin{verbatim}
A matrix over the field \( \mathbb{F}_7 \)
\end{verbatim}


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

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

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

After making pivot 1:

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

output truncated

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

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

56
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 \n  >   -define F2 -finite_field -q 2 -end \n  >   -define v -vector -field F2 -format 2 \n  >   -dense "1,1,1,1,0,1,1,0,0,1" \n  >   -end \n  >   -with F2 -do \n  >   -finite_field_activity \n  >   -nullspace v \n  >   -normalize_from_the_right \n  >   -end
```

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

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

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

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

computes all eigenvectors and eigenvalues of the matrix

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

over \( \mathbb{F}_5 \).

Orbiter can produce a list of all conjugacy classes of endomorphisms of \( \mathbb{F}_q^d \) by means of their rational normal forms. For instance
produces a list of all conjugacy classes of $\text{GL}(3, 2)$. 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 [41]. This command relies on the Orbiter catalogue of irreducible polynomials. For an introduction to the rational normal form of endomorphisms, see [47].

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

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

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

Class 0 / 6

3, 1, 0

centralizer order 7

class size 24

Class 1 / 6

2, 1, 0

centralizer order 7

class size 24
<table>
<thead>
<tr>
<th>Class</th>
<th>Order</th>
<th>Class Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 / 6</td>
<td>3</td>
<td>56</td>
</tr>
<tr>
<td>3 / 6</td>
<td>4</td>
<td>42</td>
</tr>
<tr>
<td>4 / 6</td>
<td>8</td>
<td>21</td>
</tr>
<tr>
<td>5 / 6</td>
<td>168</td>
<td>1</td>
</tr>
</tbody>
</table>
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

```plaintext
normal_basis_2,3:
▷ $\$(ORBITER) -v 2 \
▷ ▷ -define F -finite_field -q 2 -end \
▷ ▷ -with F -do -finite_field_activity \ 
▷ ▷ -normal_basis 3 -end
```

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

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

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

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

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

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

Thus,

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

under $\Phi$, as required.

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

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

The following code computes the field reduction from $\mathbb{F}_{64}$ to $\mathbb{F}_8$. Elements in the small field are represented as colors. The $(i, j)$-th block is the matrix of $a = i8 + j$ in the chosen basis.
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-write_code_for_division</code></td>
<td><code>fname A B</code></td>
<td>Write C++ source code for the polynomial division of $A$ by $B$. See Section 10.4.</td>
</tr>
<tr>
<td><code>-polynomial_division</code></td>
<td>$A B$</td>
<td>Divides polynomial $B$ by polynomial $A$.</td>
</tr>
<tr>
<td><code>-extended_gcd_for_polynomials</code></td>
<td>$A B$</td>
<td>Computes the extended gcd of polynomials $A$ and $B$.</td>
</tr>
<tr>
<td><code>-polynomial_mult_mod</code></td>
<td>$A B M$</td>
<td>Computes the product of polynomials $A$ and $B$ modulo the polynomial $M$.</td>
</tr>
<tr>
<td><code>-polynomial_power_mod</code></td>
<td>$A N M$</td>
<td>Computes the $n$-th power of the polynomial $A$ modulo the polynomial $M$.</td>
</tr>
<tr>
<td><code>-Berlekamp_matrix</code></td>
<td>$A$</td>
<td>Compute the Berlekamp matrix associated with the polynomial $A$.</td>
</tr>
<tr>
<td><code>-normal_basis</code></td>
<td>$d$</td>
<td>Computes a normal basis for $\mathbb{F}_{q^d}$ over $\mathbb{F}_q$.</td>
</tr>
<tr>
<td><code>-polynomial_find_roots</code></td>
<td>$A$</td>
<td>Computes the roots of the polynomial $A$.</td>
</tr>
<tr>
<td><code>-nullspace</code></td>
<td>$A$</td>
<td>Computes the right nullspace of the matrix $A$.</td>
</tr>
<tr>
<td><code>-RREF</code></td>
<td>$A$</td>
<td>Computes the RREF of the matrix $A$.</td>
</tr>
<tr>
<td><code>-weight Enumerator</code></td>
<td>$A$</td>
<td>Computes the weight enumerator of the code whose generator matrix is $A$.</td>
</tr>
<tr>
<td><code>-Walsh_Hadamard_transform</code></td>
<td><code>fname n</code></td>
<td>Computes the Walsh-Hadamard transform for the $n$-variable boolean function in the given file.</td>
</tr>
<tr>
<td><code>-algebraic_normal_form</code></td>
<td><code>fname n</code></td>
<td>Computes the algebraic normal form for the $n$-variable boolean function in the given file.</td>
</tr>
<tr>
<td><code>-apply_trace_function</code></td>
<td><code>fname</code></td>
<td>Applies the absolute trace function to the function in the given file.</td>
</tr>
<tr>
<td><code>-apply_power_function</code></td>
<td><code>fname d</code></td>
<td>Applies the raise-to-the-power-$d$ function to the function in the given file.</td>
</tr>
<tr>
<td><code>-identity_function</code></td>
<td><code>fname_csv</code></td>
<td>Creates the identity function and stores in the given csv file.</td>
</tr>
<tr>
<td><code>-Walsh_matrix</code></td>
<td>$n$</td>
<td>Creates the Walsh matrix of order $n$.</td>
</tr>
</tbody>
</table>

Table 3.12: Finite Field Activities (Part I)
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-Vandermonde_matrix</td>
<td>$n$</td>
<td>Creates the Vandermonde matrix of order $q \times q$. The entry $(i, j)$ is $x_j^i$ where $w_0, \ldots, x_{q-1}$ is the list of field elements in ordered according to the Orbiter ranks.</td>
</tr>
<tr>
<td>-transversal</td>
<td>$L_1$ $L_2$ $P$</td>
<td>Computes the unique transversal to the lines $L_1$ and $L_2$ through the point $P$ in $\text{PG}(3, q)$. The lines are given by a basis consisting of 8 field elements.</td>
</tr>
<tr>
<td>-intersection_of_two_lines</td>
<td>$L_1$ $L_2$</td>
<td>Computes the intersection of two lines in $\text{PG}(3, q)$. The lines are given by a basis consisting of 8 field elements.</td>
</tr>
<tr>
<td>-rank_point_in_PG</td>
<td>$P$</td>
<td>Computes the orbiter point rank of the point $P$ in $\text{PG}(n, q)$. $P$ is a label of a vector, which is the coefficient vector.</td>
</tr>
<tr>
<td>-unrank_point_in_PG</td>
<td>$r$</td>
<td>Computes the orbiter point in $\text{PG}(n, q)$ from the Orbiter rank value $r$.</td>
</tr>
<tr>
<td>-inverse_isomorphism_klein_quadric</td>
<td>$L36$</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)
The output is shown in Figure 3.6. Note that the dimension of the vector space is 2, so the block matrices are $2 \times 2$. Observe that $\mathbb{F}_{64}$ has many subfields. Figure 3.7 shows the field reduction from $\mathbb{F}_{64}$ to $\mathbb{F}_4$ (left) and from $\mathbb{F}_{64}$ to $\mathbb{F}_2$ (right). Here, the block matrices have size $3 \times 3$ and $6 \times 6$, respectively.

The minimum polynomials associated with the $n$-th roots over $\mathbb{F}_q$ can be computed using the `-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:
▷ $(ORBITER) -v 3 \$
▷ ▷ -define F -finite_field -q 8 -override_polynomial 11 -end \$
▷ ▷ -with F -do -coding_theoretic_activity \$
▷ ▷ ▷ -nth_roots 21 \$
▷ ▷ -end
▷ pdflatex Nth_roots_q8_n21.tex
▷ open Nth_roots_q8_n21.pdf
```

The output is:

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

$\beta^0 = 100000 = 1$
$\beta^1 = 000100 = \alpha^3$
$\beta^2 = 100001 = \alpha^5 + 1$
$\beta^3 = 111101 = \alpha^5 + \alpha^3 + \alpha^2 + \alpha + 1$
\[ \beta^4 = 011111 = \alpha^5 + \alpha^4 + \alpha^3 + \alpha^2 + \alpha \]
\[ \beta^5 = 101010 = \alpha^4 + \alpha^2 + 1 \]
\[ \beta^6 = 110100 = \alpha^3 + \alpha + 1 \]
\[ \beta^7 = 100111 = \alpha^5 + \alpha^4 + \alpha^3 + 1 \]
\[ \beta^8 = 101101 = \alpha^5 + \alpha^3 + \alpha^2 + 1 \]
\[ \beta^9 = 011101 = \alpha^5 + \alpha^3 + \alpha^2 + \alpha \]
\[ \beta^{10} = 011011 = \alpha^5 + \alpha^4 + \alpha^2 + \alpha \]
\[ \beta^{11} = 001011 = \alpha^5 + \alpha^3 + \alpha^2 + 1 \]
\[ \beta^{12} = 010100 = \alpha^3 + \alpha \]
\[ \beta^{13} = 100011 = \alpha^5 + \alpha^4 + \alpha^3 \]
\[ \beta^{14} = 000011 = \alpha^5 + \alpha^4 + \alpha^3 \]
\[ \beta^{15} = 011001 = \alpha^5 + \alpha^3 + \alpha^2 + \alpha \]
\[ \beta^{16} = 010100 = \alpha^3 + \alpha^2 + \alpha + 1 \]
\[ \beta^{17} = 100110 = \alpha^4 + \alpha^3 + 1 \]
\[ \beta^{18} = 100100 = \alpha^3 + \alpha \]
\[ \beta^{19} = 001001 = \alpha^5 + \alpha^4 + \alpha^2 \]
\[ \beta^{20} = 001100 = \alpha^3 + \alpha^2 \]

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:

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>r_i</td>
<td>Cyc(r_i)</td>
<td>m_{\beta_i}(X)</td>
<td>m_{\beta_i}(X)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>-----</td>
<td>----------</td>
<td>----------------</td>
<td>----------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>(0)</td>
<td>(100000)X^0 + (100000)X^1</td>
<td>X + 1</td>
<td></td>
<td></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>
<td></td>
<td></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>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>(3)</td>
<td>(111101)X^0 + (100000)X^1</td>
<td>X + 2</td>
<td></td>
<td></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>
<td></td>
<td></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>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>(6)</td>
<td>(110100)X^0 + (100000)X^1</td>
<td>X + 4</td>
<td></td>
<td></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>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>(9)</td>
<td>(011101)X^0 + (100000)X^1</td>
<td>X + 3</td>
<td></td>
<td></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>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>12</td>
<td>(12)</td>
<td>(001001)X^0 + (100000)X^1</td>
<td>X + 6</td>
<td></td>
<td></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>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>15</td>
<td>(15)</td>
<td>(101001)X^0 + (100000)X^1</td>
<td>X + 7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>18</td>
<td>(18)</td>
<td>(010100)X^0 + (100000)X^1</td>
<td>X + 5</td>
<td></td>
<td></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:

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

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

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 \
  -define F -finite_field -q 1024 -end \
  -with F -do -finite_field_activity \
  -Vandermonde_matrix \
  -end
```

rm Vandermonde_1024.csv
rm Vandermonde_inv_1024.csv

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

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

```
SRC=$(MY_PATH)/src
MY_CPP = g++
MY_CC = gcc
CPPFLAGS = -Wall -I../../../DEV.22/orbiter/src/lib -std=c++14
```
LIB = $(SRC)/lib/liborbiter.a -lpthread
LFLAGS = -lm -Wl,-rpath -Wl,/usr/local/gcc-8.2.0/lib64

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

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

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

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

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

Orbiter can deal with multivariate polynomial rings with coefficients over finite fields. Orbiter creates the 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 \
  >   -define F -finite_field -q 4 -end \
  >   -define R -polynomial_ring -field F \
  >   -number_of_variables 4 \
  >   -homogeneous_of_degree 3 \
  >   -variables "x0,x1,x2,x3" "x_0,x_1,x_2,x_3" \
  >   -end
\end{verbatim}

For more on rings, see Chapter 8.
Chapter 4

Geometry

4.1 Finite Projective Spaces

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

```
P3_2_easy:
▷ $(ORBITER) -v 2 \ \\
▷ ▷ -define F -finite_field -q 2 -end \ \\
▷ ▷ -define P -projective_space -n 3 -field F -end
```

This means that Orbiter offers indexing for the subspaces of $\text{PG}(n, q)$ of a fixed dimension. For instance, there are enumerators for points and lines. Besides these, there are enumerators for subspaces of any dimension. The incidence matrix between points and lines with respect to this ordering can be computed. The indexing is used to establish the permutation representations of the projective group, as will be described in Section 5.2. The indexing of points is not the lexicographic ordering. It emphasizes the role of frames in the geometry by assigning the smallest rank values to the members of the standard frame. After that, the other points are listed.

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

```
P2_4:
▷ $(ORBITER) -v 2 \ \\
▷ ▷ -define F -finite_field -q 4 -end \ \\
▷ ▷ -define P -projective_space -n 2 -field F -v 0 -end \ \\
```

71
Figure 4.1: The plane PG(2, 4)

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:
There are 7 elements in the Baer subgeometry.

Normalized from the left:
Here is a slightly larger example. The following command creates a cheat sheet for PG(3, 2).

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

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

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

\[
\begin{align*}
q &= 2 \\
p &= 2 \\
e &= 1 \\
n &= 3 \\
\text{Number of points} &= 15
\end{align*}
\]
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:

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

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

Lines sorted by Pluecker coordinates

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

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

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

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

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

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

\vdots

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

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

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

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

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

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

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

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

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

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

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

\[\vdots\]

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

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

<table>
<thead>
<tr>
<th>$h$</th>
<th>monomial</th>
<th>vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$X_0$</td>
<td>$(1, 0, 0, 0)$</td>
</tr>
<tr>
<td>1</td>
<td>$X_1$</td>
<td>$(0, 1, 0, 0)$</td>
</tr>
<tr>
<td>2</td>
<td>$X_2$</td>
<td>$(0, 0, 1, 0)$</td>
</tr>
<tr>
<td>3</td>
<td>$X_3$</td>
<td>$(0, 0, 0, 1)$</td>
</tr>
</tbody>
</table>
4.2 Indexing Points and Lines

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

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

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

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

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

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

This is the standard frame. We start the labeling of points with the standard frame. After these \(n + 2\) points, we list the remaining points in lexicographic ordering (utilizing right-normalized representative). Thus, for \(\text{PG}(2, 2)\) the ordering is

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

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

Consider the group \(\text{PGL}(3, 2)\) acting on \(\text{PG}(2, 2)\), for instance. The points of \(\text{PG}(2, 2)\) are listed in 4.1.

Let us look at an example. The following command computes the rank of

\[ P(3, 3, 1) = P(\omega + 1, \omega + 1, 1) \]

in \(\text{PG}(2, 4)\):

\begin{verbatim}
PG_2_4_rank_point:
▷ $ (ORBITER) -v 2 \n▷ ▷ -define v -vector -dense "3,3,1" -format 1 -end \n▷ ▷ -define F -finite_field -q 4 -end \n▷ ▷ -with F -do -finite_field_activity \n▷ ▷ ▷ -rank_point_in_PG v -end
\end{verbatim}
Algorithm 1 Rank

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

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

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

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

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

\[
\text{PG}_{2.4}\text{unrank}\_point:\n\]
\[
\text{define } v \text{ -vector -dense } "19,20" \text{ -end } \\n\text{define } F \text{ -finite_field -q } 4 \text{ -end } \\n\text{with } F \text{ -do -finite_field_activity } \\n\text{unrank}\_point \text{ in } PG 2 v \text{ -end }
\]

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

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

\[
\text{PG}_{2.8}\text{incidence}\_matrix:\n\]
\[
\text{(ORBITER) -v } 2 \\n\text{define } F \text{ -finite_field -q } 8 \text{ -end } \\n\text{define } P \text{ -projective_space -n } 2 \text{ -field } F \text{ -v } 0 \text{ -end } \\n\text{with } P \text{ -do -projective_space_activity } \\n\text{export}\_point\_line\_incidence\_matrix \\n\text{end } \\n\text{(ORBITER) -v } 2 \\n\text{define all_one -vector -repeat } 1 73 \text{ -end } \\n\text{draw_matrix } \\n\text{input_csv_file } PG_{n2,q8}\_incidence\_matrix.csv \\n\text{box_width } 20 \text{ -bit_depth } 8 \\n\text{partition } 3 \\n\text{all_one all_one } \\n\text{end } \\n\text{open } PG_{n2,q8}\_incidence\_matrix\_draw.bmp
\]

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

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

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

The command

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

produces the drawing of $\text{PG}(2, 16)$ shown in Figure 4.3. The $\text{-nodes_empty}$ command is used to suppress the drawing of the nodes. The $\text{-xin 20000}$ and $\text{-yin 20000}$ options double the input coordinate system (recall from Table 16.2 that the default values are 10000), which has the effect that the text appears smaller relative to the grid.

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

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

83
Figure 4.3: The plane \( \text{PG}(2, 16) \)

The output is shown below:

Considering the cyclic group generated by

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

The group is transitive on points and on lines.

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

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

Fixed points:
Fixed lines:
Row scheme:

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

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

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

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

The cyclic incidence matrix is shown in Figure 4.4.
If the number of points is not a prime, the group acts imprimitively. By considering various subgroups, tactical decompositions are created. For instance, for PG(2, 4), with 21 points, we can consider a subgroup the Singer cycle of order 3, which induces a partition with 7 classes of size 3 on both points and lines:

```
PGL_2_4_incma_singer_sub_3:
  ▶ $(ORBITER) -v 2 \
  ▶ ▶ -list_arguments \
  ▶ ▶ -define R -vector -repeat 3 7 -end \
  ▶ ▶ -define C -vector -repeat 3 7 -end \
  ▶ ▶ -draw_matrix \
  ▶ ▶ -input_csv_file PG_2_4_singer_incma_subgroup_index_3.csv \
  ▶ ▶ -box_width 40 -bit_depth 24 \
  ▶ ▶ -partition 3 R C \
  ▶ ▶ -end
  ▶ open PG_2_4_singer_incma_subgroup_index_3_draw.bmp
```

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

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

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

using the $q$-binomial coefficient.

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

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

```
GR_3_2_2:
▷ (ORBITER) -v 2 \
▷ ▷ -define F -finite_field -q 2 -end \
▷ ▷ -with F -do -finite_field_activity \
▷ ▷ ▷ -cheat_sheet_Gr 3 2 -end
▷ pdflatex Gr_3_2_2.tex
▷ open Gr_3_2_2.pdf
```

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

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

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

87
In the next example, we unrank six lines in PG(3, 5).

```
P_3_5_unrank_lines:
  $(ORBITER) -v 2 \n  -define v -vector \n  -dense "0,36,72,108,144,805" \n  -end \n  -define F -finite_field -q 5 -end \n  -define P -projective_space -n 3 -field F -v 0 -end \n  -with P -do \n  -projective_space_activity \n  -rank_lines_in_PG v \n  -end
```

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

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

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

Conversely, given a set of points \( V \) in \( \text{PG}(n,q) \), the ideal \( I(V) \) is the set of homogeneous polynomials in \( \mathbb{F}_q[X_0, \ldots, X_n] \) which vanish on all of \( V \). This set is an ideal in the polynomial ring. In \( \text{PG}(n,q) \), every set is algebraic of degree at most \((n+1)(q-1)\) [30]. The associated polynomial is unique and known as the algebraic normal form of the set.

In order to work with algebraic sets, multivariate polynomial rings are required. For details, see Section 8.2.

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

\[
X_0^3 + 3X_2^3 + 10X_1^2X_2 + X_0X_2^2 = 0.
\]

Using the indexing of monomials from Table 8.4, we record the coefficient vector of the equation as sequence

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

The Orbiter command

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

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

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

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

Based on the partition ordering of Figure 8.5, the equation is coded as coefficient vector

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

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

Hirschfeld_surface_equation="0,0,0,0,0,0,1,0,1,0,0,1,0,0,0,0,0,0,0"

Hirschfeld_surface_q4.txt:

```
$\text{ORBITER)} -v 2 \n  -define F -finite_field -q 4 -end \n  -define R -polynomial_ring -field F \n  -number_of_variables 4 \n  -homogeneous_of_degree 3 \n  -end \n  -define P -projective_space -n 3 -field F -v 0 -end \n  -define H4 -geometric_object P \n  -projective Variety R \n  "Hirschfeld\_surface\_q4" \n  "Hirschfeld\_surface\_q4"
```
A file called Hirschfeld_surface_q4.txt is created. The file contains the Orbiter ranks of the 45 points on the surface.
4.6 The Klein Quadric and the Plücker Map

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

```latex
\text{GR.4.2.2:}
\text{\triangleright \ $(\text{ORBITER}) -v \ 2 \ \
\text{\triangleright \ \triangleright \ -define \ F \ -finite_field \ -q \ 2 \ -end \ 
\text{\triangleright \ \triangleright \ -with \ F \ -do \ -finite_field_activity \ 
\text{\triangleright \ \triangleright \ \triangleright \ -cheat_sheet_Gr \ 4 \ 2 \ -end
\text{\triangleright \ pdflatex \ Gr.4.2.2.tex
\text{\triangleright \ 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 list is produced (output shortened):

There are 35 lines:

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

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

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

$L_3 = \begin{bmatrix} 0010 \\ 0001 \end{bmatrix} = \text{Pl}(0,1,0,0,0,0)$

The Plücker coordinates satisfy

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

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

The Orbiter labeling of points of the $Q^+(5,q)$ quadric (see Section 4.7) can then be used to enumerate the lines of PG(3,q) in a second, different way. In the example of PG(3,2), this yields the following list (output shortened):
0 = P_{1000} = L_0 = \begin{bmatrix} 1000 \\ 0100 \end{bmatrix} \quad 2 = P_{0100} = L_4 = \begin{bmatrix} 1000 \\ 0010 \end{bmatrix} \\
1 = P_{1010} = L_{34} = \begin{bmatrix} 0010 \\ 0001 \end{bmatrix} \quad \vdots \\
34 = P_{1011} = L_{26} = \begin{bmatrix} 1101 \\ 0011 \end{bmatrix}
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 4.2. Here, $p(X,Y) = c_1 X^2 + c_2 XY + c_3 Y^2 \in \mathbb{F}_q[X,Y]$ is irreducible over $\mathbb{F}_q$. To create an orthogonal space, the command 

```
-orthogonal_space \epsilon d q -end
```

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

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

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

For instance, the following command creates $Q(3, 2)$ together with its group $\text{PGO}^+(4, 2)$:
The next command creates $Q(4, 2)$ together with its group $PGO(5, 2)$. There are 15 points and 15 lines. The geometry is a configuration $15_3$ which is also known as the Cremona-Richmond configuration.

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 $PTO(n+1, q)$. When $q$ is prime, the group $PGO(n+1, q)$ is created instead (the groups are isomorphic in this case, and $PGO(n+1, q)$ is a bit more efficient). For large orthogonal spaces, creating the group is expensive in terms of time and memory. The a command $-without\_group$ can be used to prevent the group from being created. For instance

$-define 0 -orthogonal\_space 1 6 2 -end$

creates an object $O$ of type $Q^+(5, 2)$. In Table 4.4, Orbiter activities for orthogonal spaces are shown.
Figure 4.7: Incidence matrix of $Q(4, 2)$

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

Table 4.4: Activities related to orthogonal spaces
The command

```
$ (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
```

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></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>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</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>0</td>
<td>9</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>9</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>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</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>0</td>
<td>1</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>1</td>
</tr>
</tbody>
</table>

The number of points is 35 points:

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

The number of lines is 105

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

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

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

\texttt{Op\_6\_64\_line\_rank:}
\begin{verbatim}
▷ $(ORBITER) -v 4 \ 
▷ ▷ -define F -finite_field -q 64 -end \ 
\end{verbatim}
computes the Orbiter rank of the line through the points with rank \(15447347\) and \(15225451\), respectively. The rank of the line is \(16767254\). These ranks refer to the orthogonal geometry. They are different from the ranks of points and lines in projective spaces.

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

```
Op_6_64_report:
  $(ORBITER) -v 4 \\
  -define F -finite_field -q 64 -end \\
  -define O -orthogonal_space 1 6 F -without_group -end \\
  -with 0 -do -orthogonal_space_activity \\
  -unrank_line_through_two_points 15447347 15225451 \\
  -end
```

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

The group is not available.
The quadratic form is:

\[X_0X_1 + X_2X_3 + X_4X_5 = 0\]

<table>
<thead>
<tr>
<th>(\rightarrow)</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.

According to Table 4.2, Orbiter uses the equation

$$X_0X_1 + X_2X_3 + X_4X_5 = 0$$

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

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

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

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

The elliptic quadric has 26 points.
The coding of points and line in orthogonal spaces is different from the coding of points in projective spaces. We will create and print a set called BLT set (after [4]). This is a set of \( q + 1 \) points on the \( Q(4,q) \) quadric satisfying a special geometric property. According to Table 4.2, Orbiter uses the equation
\[
X_0^2 + X_1X_2 + X_3X_4 = 0
\]
to define the \( Q(4,q) \) quadric. The following example creates the BLT-set with Orbiter catalogue number #1 in \( Q(4,7) \):

BLT_database_7_1:
\[
\text{
$\text{\$ORBITER\text{\text{-v\text{2}}}}$
\text{
\text{-define F -finite_field -q 7 -end}}
\text{
\text{-define P -projective_space -n 4 -field F -v 0 -end}}
\text{
\text{-define S -geometric_object P}}$
\text{
\text{-BLT_database 1}}$
\text{
\text{-end}}$
\text{
\text{-with S -do -combinatorial_object_activity -save}}$
\text{
\text{-end}}$
\]

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

BLT_database_7_1_print:
\[
\text{
$\text{\$ORBITER\text{\text{-v\text{2}}}}$
\text{
\text{-define F -finite_field -q 7 -end}}
\text{
\text{-define O -orthogonal_space 0 5 F -without_group -end}}$
\text{
\text{-define S -set -file_orbiter_format BLT_7_1.txt -end}}$
\text{
\text{-with O -do -orthogonal_space_activity}}$
\text{
\text{-print_points S -end}}$
\text{
\text{pdflatex S_set_report.tex}}$
\text{
\text{open S_set_report.pdf}}$
\]

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

<table>
<thead>
<tr>
<th>A set of points of size 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Points:</td>
</tr>
<tr>
<td>0 : ( P_0 = (0, 1, 0, 0, 0) )</td>
</tr>
<tr>
<td>1 : ( P_1 = (0, 0, 1, 0, 0) )</td>
</tr>
<tr>
<td>2 : ( P_{40} = (0, 1, 2, 6, 2) )</td>
</tr>
<tr>
<td>3 : ( P_{41} = (0, 1, 4, 3, 1) )</td>
</tr>
<tr>
<td>4 : ( P_{225} = (1, 6, 6, 5, 1) )</td>
</tr>
<tr>
<td>5 : ( P_{270} = (1, 5, 5, 4, 4) )</td>
</tr>
<tr>
<td>6 : ( P_{241} = (1, 2, 2, 2, 1) )</td>
</tr>
</tbody>
</table>
More on BLT-sets can be found in Section 12.4.

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

Doily_W_2:

```
$ (ORBITER) -v 2 \\
  -define F -finite_field -q 2 -end \\
  -define O -orthogonal_space 0 5 F -without_group -end \\
  -define W2_points -set -loop 0 15 1 -end \\
  -define W2_lines -set -loop 0 15 1 -end \\
  -with O -do \\
  -orthogonal_space_activity \\
  -print_points W2.points \\
  -end \\
  -with O -do \\
  -orthogonal_space_activity \\
  -print_lines W2.lines \\
  -end
```

`pdflatex W2_points_set_report.tex`
`open W2_points_set_report.pdf`
`pdflatex W2_lines_set_of_lines_report.tex`
`open W2_lines_set_of_lines_report.pdf`
4.8 Hermitian Varieties

Orbiter has enumerators for points of the hermitian variety $H(k, Q)$. 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

\begin{verbatim}
H_2.4:
\$ (ORBITER) -v 2 \n\$ -define F -finite_field -q 4 -end \n\$ -with F -do -finite_field_activity \n\$ \$ -cheat_sheet_hermitian 2 -end \n\$ pdflatex H_2.4.tex \n\$ open H_2.4.pdf
\end{verbatim}

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

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

$$\begin{align*}
P_0 &= (1, 1, 0) = 4 & P_5 &= (3, 0, 1) = 9 \\
P_1 &= (2, 1, 0) = 5 & P_6 &= (0, 1, 1) = 10 \\
P_2 &= (3, 1, 0) = 6 & P_7 &= (0, 2, 1) = 13 \\
P_3 &= (1, 0, 1) = 7 & P_8 &= (0, 3, 1) = 17 \\
P_4 &= (2, 0, 1) = 8 &
\end{align*}$$

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

The command

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

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 4.5-4.8.

Table 4.9 lists Orbiter global commands related to projective geometries. These commands do not need an object of type projective space in order to be invoked.

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

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

which is a Baer collineation. It fixes a subgeometry PG(3, 2). The command

\texttt{fix\_structure\_2A:}
\begin{itemize}
  \item \texttt{$\langle$ORBITER$\rangle$ -v 2 \}
  \item \texttt{\hspace{1cm} -define F -finite\_field -q 4 -end \}
  \item \texttt{\hspace{1cm} -define P -projective\_space -n 3 -field F -v 0 -end \}
  \item \texttt{\hspace{1cm} -with P -do \}
  \item \texttt{\hspace{1cm} -projective\_space\_activity \}
  \item \texttt{\hspace{1cm} \hspace{1cm} -cheat\_sheet\_for\_decomposition\_by\_element\_PG 1 \}
  \item \texttt{\hspace{1cm} \hspace{1cm} \hspace{1cm} "1,0,0,0, 0,1,0,0, 0,0,1,0, 0,0,0,1, 1" \}
  \item \texttt{\hspace{1cm} \hspace{1cm} \hspace{1cm} \texttt{fix\_structure\_2A \}}
  \item \texttt{\hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \texttt{-end \}}
  \item \texttt{\hspace{1cm} pdflatex fix\_structure\_2A.tex}
  \item \texttt{\hspace{1cm} open fix\_structure\_2A.pdf}
\end{itemize}

can be used.

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

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

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

The command
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-export_point_line_incidence_matrix</td>
<td></td>
<td>Create a csv file of the point line incidence matrix.</td>
</tr>
<tr>
<td>-table_of_cubic_surfaces_compute_properties</td>
<td>fname $q_0$ col-offset</td>
<td>See Section 7.5.</td>
</tr>
<tr>
<td>-cubic_surface_properties_analyze</td>
<td>fname $q_0$</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 $\text{PG}(n, q)$.</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 $\text{PG}(n, q)$. The subgroup must be a linear group, and the description of $H$ must come from the commands from Section 5.2.</td>
</tr>
<tr>
<td>-table_of_quartic_curves</td>
<td></td>
<td>Export the classification of quartic curves to a csv file.</td>
</tr>
<tr>
<td>-table_of_cubic_surfaces</td>
<td></td>
<td>Export the classification of cubic surfaces to a csv file.</td>
</tr>
<tr>
<td>-classify-surfaces_with_double_sixes</td>
<td>label control</td>
<td>Classify cubic surfaces using the approach of double sixes. See Section 7.3.</td>
</tr>
</tbody>
</table>

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

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

Table 4.7: Projective Space Activities (Part 3)
Table 4.8: Projective Space Activities (Part 4)

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

trans:

\[$\$(ORBITER) -v 5 \$

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

\[\$ -define P -projective_space -n 3 -field F -v 0 -end \$

\[\$ -with P -do \$

\[\$ -projective_space_activity \$

\[\$ -move_two_lines_in_hyperplane_stabilizer_text \$

\[\$ "1,0,0,0, 0,0,0,1" "1,1,0,2, 0,0,1,0" \$

\[\$ "1,0,0,0, 0,0,0,1" "0,1,0,1, 0,0,1,0" \$

\[\$ -end \$

computes a projectivity (transvection) to do so:

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

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

It is possible to define algebraic varieties directly from an algebraic equation. We distinguish between managed variables and arbitrary variables. We require that the polynomial is ho-
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-create_points_on_quartic</code></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><code>-create_points_on_parabola</code></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><code>-smooth_curve</code></td>
<td>( \epsilon \ N \ b \ t_{\text{min}} ) ( t_{\text{max}} ) ( \text{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><code>-make_table_of_surfaces</code></td>
<td></td>
<td>Produces a latex table summarizing the surfaces in the Orbiter catalogue.</td>
</tr>
<tr>
<td><code>-create_surface_reports</code></td>
<td>field-orders</td>
<td>Produce reports for all surfaces in the Orbiter catalogue over the give field orders.</td>
</tr>
<tr>
<td><code>-create_surface_atlas</code></td>
<td>( q_{\text{max}} )</td>
<td>Produce reports for all surfaces in the Orbiter catalogue for field orders ( q \leq q_{\text{max}} ).</td>
</tr>
<tr>
<td><code>-create_dickson_atlas</code></td>
<td></td>
<td>Produce reports of Dickson surfaces.</td>
</tr>
</tbody>
</table>

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

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

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

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

for \( f_3 \) and

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

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

del_Pezzo_F13ab_report:

\[
\begin{align*}
&\text{\texttt{\$ORBITER}} -v 3 \ \backslash \\
&\quad \text{\texttt{-define F -finite_field -q 13 -end}} \ \\
&\quad \text{\texttt{-define P -projective_space -n 3 -field F -v 0 -end}} \ \\
&\quad \text{\texttt{-define f3 -formula}} \ \\
&\quad \quad \text{\texttt{"del_Pezzo_F13a" "del\_Pezzo\_F13a" "x,y,z" \}} \\
&\quad \quad \text{\texttt{"x*x*x*x+y*y*y*y+z*z*z*z+8*x*x*y*y+8*x*x*z*z+8*y*y*z*z" \}} \\
&\quad \quad \text{\texttt{-define f4 -formula}} \ \\
&\quad \quad \text{\texttt{"del_Pezzo_F13b" "del\_Pezzo\_F13b" "x,y,z" \}} \\
&\quad \quad \text{\texttt{"x*x*x*x+y*y*y*y+z*z*z*z-x*x*y*y" \}} \\
&\quad \quad \text{\texttt{-define del_Pezzo13 -collection "f3,f4" \}} \\
&\quad \quad \text{\texttt{-with P -do}} \ \\
&\quad \quad \text{\texttt{-projective_space_activity \}} \\
&\quad \quad \text{\texttt{-analyze_del_Pezzo_surface del_Pezzo13 "" \}} \\
&\quad \quad \text{\texttt{-end}} \\
&\quad \text{\texttt{pdflatex del_Pezzo_F13b_report.tex \}} \\
&\quad \text{\texttt{open del_Pezzo_F13b_report.pdf}} \\
\end{align*}
\]

The third argument after the \texttt{-formula} command specifies the managed variables, which are \( x, y, z \). The command \texttt{-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 \texttt{-analyze_del_Pezzo_surface} once, and it applies to both surfaces.
4.10 Geometric Objects

Orbiter can create objects in projective space. To do so, define an object of type `geometric_object`. The definition of a geometric object requires a projective geometry object. For this reason, the definition requires an extra argument, which is the label of a previously created projective geometry object. After that, one of the commands shown in Tables 4.10 and 4.11 can be issued. Modifier options as shown in Table 4.12 apply.

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

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

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

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

The Edge curve is given by the equation

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

where \( f \) is a primitive element of \( \mathbb{F}_q \). Let us pick \( q = 17 \). The next example creates the Edge curve in PG(2, 17) and saves it to file. The equation is encoded using the ordering of quartic monomials from Table 8.4.

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

EDGE_CURVE_Q17_AS_POINTS="4, 7, 16, 19, 20, 23, 32, 35, 89, 100, 244, 251"
```
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-hyperoval</td>
<td></td>
<td>To create a hyperoval</td>
</tr>
<tr>
<td>-subiaco_oval</td>
<td>f_short</td>
<td>Create the Subiaco oval</td>
</tr>
<tr>
<td>-subiaco_hyperoval</td>
<td></td>
<td>Create the Subiaco hyperoval</td>
</tr>
<tr>
<td>-adelaide_hyperoval</td>
<td></td>
<td>Create the Adalaide hyperoval</td>
</tr>
<tr>
<td>-translation</td>
<td>i</td>
<td>Create the translation hyperoval with exponent i</td>
</tr>
<tr>
<td>-Segre</td>
<td></td>
<td>Create the Segre hyperoval</td>
</tr>
<tr>
<td>-Payne</td>
<td></td>
<td>Create the Payne hyperoval</td>
</tr>
<tr>
<td>-Cherowitzo</td>
<td></td>
<td>Create the Cherowitzo hyperoval</td>
</tr>
<tr>
<td>-OKeefe_Penttila</td>
<td></td>
<td>Create the O’Keefe, Penttila hyperoval</td>
</tr>
<tr>
<td>-BLT_database</td>
<td>k</td>
<td>Create the kth BLT-set of order q from the database ($k = 0, 1, \ldots$)</td>
</tr>
<tr>
<td>-elliptic_quadric_ovoide</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 [7]</td>
</tr>
<tr>
<td>-ttp_construction_A_hyperoval</td>
<td></td>
<td>Create the twisted tensor product code of type A [7]</td>
</tr>
<tr>
<td>-ttp_construction_B</td>
<td></td>
<td>Create the twisted tensor product code of type B [7]</td>
</tr>
</tbody>
</table>

Table 4.10: Orbiter Objects (Part 1)
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-unital_XXq_YZq_ZYq</td>
<td></td>
<td>Create the unital with equation $XXq+YZq+ZYq = 0$</td>
</tr>
<tr>
<td>-desarguesian_line_spread_in_PG_3_q</td>
<td></td>
<td>Create the desarguesian line spread in $\text{PG}(3,q)$ as a set of 2-subspaces</td>
</tr>
<tr>
<td>-Buekenhout_Metz</td>
<td></td>
<td>Create the Buekenhout Metz unital</td>
</tr>
<tr>
<td>-Uab</td>
<td>$a$ $b$</td>
<td>Create the Buekenhout Metz unital in the form of Barwick and Ebert [5]</td>
</tr>
<tr>
<td>-whole_space</td>
<td></td>
<td>Create the whole space</td>
</tr>
<tr>
<td>-hyperplane</td>
<td>$\text{pt}$</td>
<td>Create the hyperplane given by dual coordinates associated with the given point</td>
</tr>
<tr>
<td>-segre Variety</td>
<td>$a$ $b$</td>
<td>Create the Segre variety</td>
</tr>
<tr>
<td>-Maruta_Hamada_arc</td>
<td></td>
<td>Create the Maruta Hamada arc</td>
</tr>
<tr>
<td>-projective Variety</td>
<td>lab_ascii lab_tex $d$ coeffs</td>
<td>Create a projective variety of degree $d$ from an equation. By default, the coefficients of the equation are listed in the partition ordering. A different ordering can be specified. A label for the variety in ascii and in tex is required. See Section 4.5.</td>
</tr>
<tr>
<td>-intersection_of_zariski_open_sets</td>
<td>$l$ $d$ $n$ $C_1$ ... $C_n$</td>
<td>Create the intersection of the Zariski open sets given by equations $C_1, \ldots C_n$ of degree $d$ with label $l$, see Section 4.5.</td>
</tr>
<tr>
<td>-projective_curve</td>
<td>$l$ $r$ $d$ $C$</td>
<td>Create the projective curve of degree $d$ with label $l$, with coefficient vector $C$ in $r$ variables</td>
</tr>
</tbody>
</table>

Table 4.11: Orbiter Objects (Part 2)

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

Table 4.12: Orbiter Objects: Modifiers
The following command computes the line type of the Edge curve:

```
Edge_curve.17_line_type:
> echo $(FILE_Q17) > edge_q17.csv
> $(ORBITER) -v 2 \
> -define F -finite_field -q 17 -end \
> -define R -polynomial_ring -field F \
> -number_of_variables 3 \
> -homogeneous_of_degree 4 \
> -end \
> -define P -projective_space -n 2 -field F -v 0 -end \
> -define C -geometric_object P \
> -projective_variety R \
> "Edge.q17" "Edge\_q17" \
> $(EDGE_CURVE_Q17_EQUATION) \
> -end \
> -with C -do -combinatorial_object_activity -save \
> -end
```

The line type is

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

Group Theory

5.1 Permutation Groups

Permutation groups can be represented on a computer using the technique of stabilizer chains, or Sims chains (cf. [36, 63]). The stabilizer chain is defined with respect to a sequence of points in the permutation domain called a base. A set of generators which allows to generate each group along the chain is called a strong generating set. Many algorithms for permutation groups rely on knowing a base and strong generating set. In Orbiter, permutation groups can be created from a base and strong generating set. Many types of groups come with their own built-in base and strong generating set. On the other hand, it is also possible to create groups from generating sets which are either not strong or for which a base is not known. For efficiency purposes, small basic orbits are desired.

In order to establish the permutation representation of a group, the technique of indexing is used. Indexing sets up a fixed bijection between the permutation domain (the set we act on) and the integer interval \([0, n-1]\) for some \(n\). The integer associated to an element in the permutation domain is called the rank. Conversely, given an integer in \([0, n-1]\), the element in the permutation domain associated with it is obtained by the unrank function. The process of converting integers to elements of the permutation domain and vice-versa is indexing. We have seen indexing for projective points in Section 4.1.

In Section 5.2, we will discuss matrix groups over finite fields. The enumerators for projective points from Section 4.1 are used to realize the permutation domain. This enumerator relies on an enumerator for finite fields, as discussed in Sections 3.2 and 3.3. For extension fields, the enumerator for finite fields in turn depends on the choice of the irreducible polynomial which is used to create the field. For affine groups, a different enumerator is used to describe the permutation domain. This enumerator uses the base-\(q\) representation of integers, which associates a vector over \(\mathbb{F}_q\) of length \(n\) with an integer in \([0, q^n-1]\).

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

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

\begin{verbatim}
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
\end{verbatim}

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

\begin{verbatim}
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 \\
\end{verbatim}
Next, let us consider the symmetric group Sym(n). The following command creates Sym(3):

Symmetric_3:
$$(\text{ORBITER}) \ -v \ 3 \ \$$
$$(\text{define} \ G \ -\text{permutation} \ \text{group} \ -\text{symmetric} \ \text{group} \ 3 \ -\text{end})$$
$$(\text{with} \ G \ -\text{do})$$
$$(\text{-group_theoretic_activity})$$
$$(\text{-report})$$
$$(\text{-end})$$
pdflatex Perm3_report.tex
open Perm3_report.pdf

The report is shown below:

<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>

**Stabilizer chain**

**Basic Orbit 0**

0

1

2
Figure 5.2: The group table of Sym(3)

Basic orbit 0 has size 3
0, 1, 2

Basic Orbit 1

1

2

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

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

Figure 5.3: The elements of Sym(3)
-define all_one_c -vector -repeat 1 3 -end \
-draw_matrix \
  -input_csv_file Sym3_elts_matrix.csv \
  -box_width 50 -bit_depth 8 \
  -partition 3 \
  all_one_r all_one_c \
-end 
open Sym3_elts_matrix_draw.bmp
5.2 Linear Groups

Orbiter provides support for matrix groups and their various permutation representationes. For background information about the classical groups of matrices over finite fields, see cf. [68]. Any group in Orbiter is associated with a permutation action. There can be multiple actions for the same group though. Using homomorphisms of permutation groups, new actions can be formed from old actions. Basic group actions are projective, affine, and general linear, as well as orthogonal, unitary and tensor product. Product actions can be defined also. In order to establish a permutation representation, the elements (aka points) of the permutation domain need to be made available. One way would be to make a table of all elements in the permutation domain. However, this would be time and memory intensive. For this reason, a different technique is used that creates points only when needed. The way this works is that the permutation domain is encoded implicitly, using a fixed bijection to a suitable integer interval (zero based), called the domain. Whenever we want the $i$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 $V \simeq \mathbb{F}_q^n$ be a finite dimensional vector space over $\mathbb{F}_q$. The set of subspaces of $V$ form the projective geometry $\text{PG}(n - 1, q)$.

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

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

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

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

125
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
\[ M_k \cdot N_l = \left( M \cdot N^{\phi^{-k}} \right)_{k+l}, \] (5.2)
\[ (M_k)^{-1} = \left( \left( M^{-1} \right)^{\phi^k} \right)_{-k}. \] (5.3)
The affine group \( AGL(n,q) \) is the semidirect product of \( GL(n,q) \) with \( \mathbb{F}_q^n \). The affine semi-linear group \( A\Gamma L(n,q) \) is the semidirect product of \( AGL(n,q) \) with \( \text{Aut}(\mathbb{F}_q) \). The elements of \( A\Gamma L(n,q) \) are triples
\[ M_{a,k} := (M, a, k) \in GL(n,q) \times \mathbb{F}_q^n \times \text{Aut}(\mathbb{F}_q), \]
which act on \( \mathbb{F}_q^n \):
\[ (x, (M, a, k)) \mapsto (x \cdot M + a)^{\phi^k}. \]
The multiplication in \( A\Gamma L(n,q) \) is
\[ M_{a,k} \cdot N_{b,l} = (MN)_{a+b^{\phi^{-k}}+b^{\phi^{-k}}, k+l}. \]
The inverse of an element is
\[ \left( M_{a,k} \right)^{-1} = \left( M^{-1} \right)_{a^{\phi^k}M^{-1}, -k}. \]
A correlation is a one-to-one mapping between the set of points and the set of hyperplanes which reverses incidence. So, if \( \rho \) is a correlation and \( P \) is a point and \( \ell \) is a hyperplane then \( P^\rho \) is a hyperplane and \( \ell^\rho \) is a point and
\[ \ell^\rho \in P^\rho \iff P \in \ell. \]
A correlation of order two is called polarity. The standard polarity is the map
\[ \rho : \mathcal{P} \leftrightarrow \mathcal{L}, \ P(x) \leftrightarrow [x]. \]
A group \( G \) can act on \( V \) in one of the types listed in Table 5.1. One can create a matrix group over a finite field \( \mathbb{F}_q \) is created as described in in two steps. In the first step, the field \( \mathbb{F}_q \) is created as described in Sections 3.2 and 3.3. The field is stored in the symbol table. Then, the group is created using the symbolic label for the field. The basic types of matrix groups in Orbiter are listed in Table 5.2.

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

Table 5.1: Basic actions

**PGL$_{4,2}$:**

```
$(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 $G r_1(\mathbb{F}_2^4)$. 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>$P\Gamma L(n, q)$</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>$P\Sigma L(n, q)$</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>$A\Gamma L(n, q)$</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>$A\Sigma L(n, q)$</td>
</tr>
<tr>
<td>-PGO</td>
<td>$n \ q$</td>
<td>$PGO(n, q)$</td>
</tr>
<tr>
<td>-PGOp</td>
<td>$n \ q$</td>
<td>$PGO^+(n, q)$</td>
</tr>
<tr>
<td>-PGOm</td>
<td>$n \ q$</td>
<td>$PGO^-(n, q)$</td>
</tr>
<tr>
<td>-PGGO</td>
<td>$n \ q$</td>
<td>$P\Gamma O(n, q)$</td>
</tr>
<tr>
<td>-PGGOp</td>
<td>$n \ q$</td>
<td>$P\Gamma O^+(n, q)$</td>
</tr>
<tr>
<td>-PGGOm</td>
<td>$n \ q$</td>
<td>$P\Gamma O^-(n, q)$</td>
</tr>
<tr>
<td>-GL$_{_d_q_wr_Sym_n}$</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:

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

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

The Action

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

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

The group is a matrix group.
The group acts on projective space PG(3, 2)
q = 2
p = 2
e = 1
n = 3
Number of points = 15
Number of lines = 35
The finite field $\mathbb{F}_2$

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

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

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

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

$1^0 \equiv 1$
$1^1 \equiv 1$

Base and Stabilizer Chain

Group order 20160
$tI=15, 14, 12, 8,$
Base: $(0, 1, 2, 3)$
Strong generators for a group of order 20160:

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

1,0,0,0,0,1,0,0,0,0,1,0,1,0,0,1,
1,0,0,0,0,1,0,0,0,0,1,0,0,1,0,1,
1,0,0,0,0,1,0,0,0,0,1,0,0,0,1,1,
Stabilizer chain

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

Basic Orbit 0

Basic orbit 0 has size 15
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14
Basic Orbit 1

Basic orbit 1 has size 14
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14

Basic Orbit 2

Basic orbit 2 has size 12
2, 3, 4, 6, 7, 8, 9, 10, 11, 12, 13, 14
Basic Orbit 3

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.

```bash
PGL.4.2.export:
  ▶ $(ORBITER) -v 2 \n  ▶ ▶ -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:

```bash
PGL.4.2.generated:
  ▶ $(ORBITER) -v 2 \n  ▶ ▶ -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:

```text
Row,C0,C1,C2,C3,C4,C5,C6,C7,C8,C9,C10,C11,C12,C13,C14
134
```
The command

L_5_3:
\[
\texttt{\$(ORBITER) -v 2 \ \ |}
\]
\[
\texttt{define F -finite_field -q 3 -end \ |}
\]
\[
\texttt{define G -linear_group -PSL 5 F -end \ |}
\]
\[
\texttt{with G -do \ |}
\]
\[
\texttt{-group_theoretic_activity \ |}
\]
\[
\texttt{-report \ |}
\]
\[
\texttt{-end \ |}
\]
\[
\texttt{pdflatex PSL_5_3_report.tex}
\]
\[
\texttt{open PSL_5_3_report.pdf}
\]

creates PSL(5, 3) of order 237783237120.

The command

PSP_4_4:
\[
\texttt{\$(ORBITER) -v 2 \ |}
\]
\[
\texttt{-define F -finite_field -q 4 -end \ |}
\]
\[
\texttt{-define G -linear_group -PGL 4 F \ |}
\]
\[
\texttt{symplectic_group \ |}
\]
\[
\texttt{-end \ |}
\]
\[
\texttt{-with G -do \ |}
\]
\[
\texttt{-group_theoretic_activity \ |}
\]
\[
\texttt{-report \ |}
\]
\[
\texttt{-end}
\]
\[
\texttt{pdflatex PGL_4_4_Sp_4_4_report.tex}
\]
\[
\texttt{open PGL_4_4_Sp_4_4_report.pdf}
\]

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

The command

PGO_5_2:
\[
\texttt{\$(ORBITER) -v 2 \ |}
\]
creates the group \( \text{PGO}(5, 2) \) acting on the 15 points of the \( Q(4, 2) \) quadric. The following latex report is produced:

### The Group \( \text{PGO}(5, 2) \)

The order of the group \( \text{PGO}(5, 2) \) is 720

The group acts on a set of size 15

Strong generators for a group of order 720:

<table>
<thead>
<tr>
<th>[ \begin{bmatrix} 1 &amp; 0 &amp; 0 &amp; 0 &amp; 0 \ 0 &amp; 1 &amp; 0 &amp; 0 &amp; 0 \ 0 &amp; 0 &amp; 1 &amp; 0 &amp; 0 \ 0 &amp; 0 &amp; 0 &amp; 1 &amp; 0 \ 1 &amp; 0 &amp; 0 &amp; 1 &amp; 1 \end{bmatrix} ]</th>
<th>[ \begin{bmatrix} 1 &amp; 0 &amp; 0 &amp; 0 &amp; 0 \ 0 &amp; 1 &amp; 0 &amp; 0 &amp; 0 \ 0 &amp; 0 &amp; 1 &amp; 0 &amp; 0 \ 1 &amp; 0 &amp; 0 &amp; 1 &amp; 1 \ 0 &amp; 0 &amp; 0 &amp; 0 &amp; 1 \end{bmatrix} ]</th>
<th>[ \begin{bmatrix} 1 &amp; 0 &amp; 0 &amp; 0 &amp; 0 \ 0 &amp; 1 &amp; 0 &amp; 0 &amp; 0 \ 1 &amp; 1 &amp; 1 &amp; 0 &amp; 1 \ 1 &amp; 0 &amp; 0 &amp; 1 &amp; 1 \ 0 &amp; 0 &amp; 0 &amp; 0 &amp; 1 \end{bmatrix} ]</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ \begin{bmatrix} 1 &amp; 0 &amp; 0 &amp; 0 &amp; 0 \ 0 &amp; 1 &amp; 0 &amp; 0 &amp; 0 \ 0 &amp; 0 &amp; 1 &amp; 0 &amp; 1 \ 1 &amp; 1 &amp; 0 &amp; 1 &amp; 1 \ 0 &amp; 1 &amp; 0 &amp; 1 &amp; 0 \end{bmatrix} ]</td>
<td>[ \begin{bmatrix} 1 &amp; 0 &amp; 0 &amp; 0 &amp; 0 \ 1 &amp; 1 &amp; 1 &amp; 0 &amp; 0 \ 0 &amp; 0 &amp; 1 &amp; 0 &amp; 0 \ 0 &amp; 0 &amp; 0 &amp; 1 &amp; 0 \ 0 &amp; 0 &amp; 0 &amp; 0 &amp; 1 \end{bmatrix} ]</td>
<td>[ \begin{bmatrix} 1 &amp; 0 &amp; 0 &amp; 0 &amp; 0 \ 0 &amp; 1 &amp; 0 &amp; 0 &amp; 1 \ 0 &amp; 0 &amp; 1 &amp; 0 &amp; 0 \ 0 &amp; 0 &amp; 1 &amp; 1 &amp; 0 \ 0 &amp; 0 &amp; 0 &amp; 0 &amp; 1 \end{bmatrix} ]</td>
</tr>
</tbody>
</table>

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

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

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

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

The finite field \( \mathbb{F}_2 \)

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

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

\[
+ \quad 0 \\
0 & 0 & 1 \\
1 & 1 & 0
\]
\[
\cdot \quad 1 \\
1 & 1
\]
\[1^0 \equiv 1\]
\[1^1 \equiv 1\]

**Base and Stabilizer Chain**

Group order 720

tl=15, 8, 3, 1, 1, 2,

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

Strong generators for a group of order 720:

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

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:
\[ G := \text{Group}([ (6, 13)(7, 14)(8, 15)(9, 12), \\
(3, 13)(4, 14)(5, 15)(9, 11), \\
(2, 12)(3, 14)(4, 13)(8, 10), \\
(2, 8, 9, 10, 12, 15)(3, 14, 7)(4, 13, 6)(5, 11), \\
(1, 10)(4, 11)(7, 12)(9, 14), \\
(1, 7)(3, 5)(4, 9)(10, 12)(11, 14)(13, 15)]); \]

Magma export:

Compact form:

Generators in compact permutation form are:
\[
\begin{array}{cccccccccccccccc}
6 & 15 \\
0 & 1 & 2 & 3 & 4 & 12 & 13 & 14 & 11 & 9 & 10 & 8 & 5 & 6 & 7 \\
0 & 1 & 12 & 13 & 14 & 5 & 6 & 7 & 10 & 9 & 8 & 11 & 2 & 3 & 4 \\
0 & 11 & 13 & 12 & 4 & 5 & 6 & 9 & 8 & 7 & 10 & 1 & 3 & 2 & 14 \\
0 & 7 & 13 & 12 & 10 & 3 & 2 & 8 & 9 & 11 & 4 & 14 & 5 & 6 & 1 \\
9 & 1 & 2 & 10 & 4 & 5 & 11 & 7 & 13 & 0 & 3 & 6 & 12 & 8 & 14 \\
6 & 1 & 4 & 8 & 2 & 5 & 0 & 7 & 3 & 11 & 13 & 9 & 14 & 10 & 12 \\
-1
\end{array}
\]

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

\begin{verbatim}
$\textsc{ORBITER}$ -v 2 -define F -finite_field -q 2 -end -define G -linear_group -PGO 7 F -end -with G -do -group_theoretic_activity -report -end
\end{verbatim}

\texttt{pdflatex PGO7_2_report.tex}
\texttt{open PGO7_2_report.pdf}
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-Janko1</td>
<td></td>
<td>first Janko group, needs PGL(7, 11)</td>
</tr>
<tr>
<td>-monomial</td>
<td></td>
<td>subgroup of monomial matrices</td>
</tr>
<tr>
<td>-diagonal</td>
<td></td>
<td>subgroup of diagonal matrices</td>
</tr>
<tr>
<td>-null_polarity_group</td>
<td></td>
<td>null polarity group</td>
</tr>
<tr>
<td>-symplectic_group</td>
<td></td>
<td>symplectic group</td>
</tr>
<tr>
<td>-singer</td>
<td>$k$</td>
<td>subgroup of index $k$ in the Singer cycle</td>
</tr>
<tr>
<td>-singer_and_frobenius</td>
<td>$k$</td>
<td>subgroup of index $k$ in the Singer cycle, extended by the Frobenius automorphism of $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$</td>
</tr>
<tr>
<td>-borel_upper</td>
<td></td>
<td>Borel subgroup of upper triangular matrices</td>
</tr>
<tr>
<td>-borel_lower</td>
<td></td>
<td>Borel subgroup of lower triangular matrices</td>
</tr>
<tr>
<td>-identity_group</td>
<td></td>
<td>identity subgroup</td>
</tr>
<tr>
<td>-subgroup_from_file</td>
<td>$fl$</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>$lon s_1 \ldots s_n$</td>
<td>Generate a subgroup from generators. The label “l” is used to denote the subgroup; $o$ is the order of the subgroup; $n$ is the number of generators and $s_1, \ldots, s_n$ are the generators for the subgroup in vector form.</td>
</tr>
</tbody>
</table>

Table 5.3: Commands for creating subgroups

### 5.3 Subgroups

There are many ways to create subgroups of a group. Table 5.3 lists some commands to do so.

We start with an example of an explicit permutation group using a known base and strong generating set, using the `bsgs` command. Here is the cyclic group of order 13 acting on the permutation domain $[0, 12]$. The base is (0). When creating a group, we supply a label in ascii text and in tex. Then we specify the degree of the action, and the group order. After that, we specify the number of generators and the generators themselves. The labels will be used in reports about the group, for instance.

```
GEN_C13="1,2,3,4,5,6,7,8,9,10,11,12,0"
# (0,1,2,3,4,5,6,7,8,9,10,11,12)
```

C13:
The makefile variable `GEN_C13` is used to define the generator of the group, which is the cycle

\[(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12)\].

The generator is given in list notation, which is the second row in the array

\[
\begin{bmatrix}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 0
\end{bmatrix}
\].

The command creates the group from the known base 0. After that, several activities are invoked. Specifically, these are group theoretic activities. They will be discussed in more detail in Section 5.6.

Let us take a closer look at the three activities performed in this example. The -export_orbiter command exports the group in Orbiter makefile format. The file `C13.makefile` is generated, which can be used to recreate the permutation group in an Orbiter makefile. Here is the content of the file:

`C13.generated`:

\[
\begin{verbatim}
$(ORBITER) -v 2 \
-define gens -vector -dense $(GEN_C13) -end \
-define G -permutation_group \
-bsgs C13 C_{13} 13 13 0 1 \
gen \
-define -end \
-with G -do \
group_theoretic_activity \
-export_orbiter \
-end \
-with G -do \
group_theoretic_activity \
-export_group_table \
-end \
-with G -do \
group_theoretic_activity \
-report \
-end \
-with G -do \
group_theoretic_activity \
-save_elements_csv "C13_elts.csv" \
-end
\end{verbatim}
\]
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"
```

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

### Stabilizer chain

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

### Basic Orbit 0

Basic orbit 0 has size 13
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12
It is possible to create a permutation group as a subgroup of the symmetric group, using the known base for the symmetric group. Because the base of the symmetric group is large, this way of creating the group is less efficient than creating the group with a known (small) base. Here is an example. We create $C_{13}$ as a subgroup of $\text{Sym}(13)$.

```
C13_as_subgroup:
  ➤ $(ORBITER) -v 2 \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```
creates the first Janko group as a subgroup of PGL(7, 11).

The command

```
PGL_3.11_singer:
▷ $(ORBITER) -v 2 \n▷ ▷ -define G -linear_group -PGL 3 11 -singer 19 -end \n▷ ▷ -with G -do \n▷ ▷ -group_theoretic_activity \n▷ ▷ ▷ -report \n▷ ▷ -end
▷ pdflatex PGL_3.11_Singer_3.11.19_report.tex
▷ open PGL_3.11_Singer_3.11.19_report.pdf
```

creates a subgroup of the Singer cycle of order 7. The Singer cycle in 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:
▷ $(ORBITER) -v 2 \n▷ ▷ -define G -linear_group -PGL 3 11 -singer_and_frobenius 19 -end \n▷ ▷ -with G -do \n▷ ▷ -group_theoretic_activity \n▷ ▷ ▷ -report \n▷ ▷ -end
▷ pdflatex PGL_3.11_Singer_and_Frob3.11.19_report.tex
▷ open PGL_3.11_Singer_and_Frob3.11.19_report.pdf
```

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

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

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

It is isomorphic to a subgroup of SL(2, 3). The Orbiter command
quaternion:
▷ $(\text{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 \ 
▷ ▷ ▷ ▷ -group_table \ 
▷ ▷ ▷ ▷ -report \ 
▷ ▷ ▷ -end \ 
▷ pdflatex GL_2_3_Subgroup_quaternion_8_elements.tex 
▷ open GL_2_3_Subgroup_quaternion_8_elements.pdf 
▷ pdflatex GL_2_3_Subgroup_quaternion_8_report.tex 
▷ open GL_2_3_Subgroup_quaternion_8_report.pdf

creates the group. The command produces the list of group elements shown below.

Element 0 / 8 of order 1:

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

(0)(1)(2)(3)(4)(5)(6)(7)(8)

Element 1 / 8 of order 4:

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

(0)(1,5,2,7)(3,4,6,8)

Element 2 / 8 of order 2:

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

(0)(1,2)(3,6)(4,8)(5,7)
Element 3 / 8 of order 4:
\[
\begin{bmatrix}
1 & 2 \\
2 & 2
\end{bmatrix}
\]
(0)(1,7,2,5)(3,8,6,4)

Element 4 / 8 of order 4:
\[
\begin{bmatrix}
1 & 1 \\
1 & 2
\end{bmatrix}
\]
(0)(1,4,2,8)(3,7,6,5)

Element 5 / 8 of order 4:
\[
\begin{bmatrix}
0 & 1 \\
2 & 0
\end{bmatrix}
\]
(0)(1,3,2,6)(4,5,8,7)

Element 6 / 8 of order 4:
\[
\begin{bmatrix}
2 & 2 \\
2 & 1
\end{bmatrix}
\]
(0)(1,8,2,4)(3,5,6,7)

Element 7 / 8 of order 4:
\[
\begin{bmatrix}
0 & 2 \\
1 & 0
\end{bmatrix}
\]
(0)(1,6,2,3)(4,7,8,5)

The group table is created as csv file:

```
Row, C0, C1, C2, C3, C4, C5, C6, C7
0, 0, 1, 2, 3, 4, 5, 6, 7
1, 1, 2, 3, 0, 5, 6, 7, 4
2, 2, 3, 0, 1, 6, 7, 4, 5
3, 3, 0, 1, 2, 7, 4, 5, 6
4, 4, 7, 6, 5, 2, 1, 0, 3
5, 5, 4, 7, 6, 3, 2, 1, 0
6, 6, 5, 4, 7, 0, 3, 2, 1
```

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

cube_group:

```
$\texttt{ORBITER} -v 2 \backslash
  \texttt{-define gens -vector -dense} \backslash
  \texttt{"0,1,0,2,0,0,0,0,1,"} \backslash
  \texttt{\texttt{-end}} \backslash
  \texttt{-define G -linear_group -GL 3 3} \backslash
  \texttt{-subgroup_by_generators "cube" "48" 3} \backslash
  \texttt{-with G -do} \backslash
  \texttt{-group_theoretic_activity} \backslash
  \texttt{-print_elements.tex} \backslash
  \texttt{-report} \backslash
  \texttt{-end} \backslash
```

```
pdflatex GL_3_3_Subgroup_cube_48_report.tex
open GL_3_3_Subgroup_cube_48_report.pdf
pdflatex GL_3_3_Subgroup_cube_48_elements.tex
open GL_3_3_Subgroup_cube_48_elements.pdf
```

The tetrahedral subgroup can be created as well:

tetra_group:

```
$\texttt{ORBITER} -v 3 \backslash
  \texttt{-define G -linear_group -GL 3 3} \backslash
  \texttt{-subgroup_by_generators "tetra" "12" 2} \backslash
  \texttt{-end} \backslash
  \texttt{-with G -do} \backslash
  \texttt{-group_theoretic_activity} \backslash
  \texttt{-print_elements.tex} \backslash
  \texttt{-report} \backslash
  \texttt{-end} \backslash
```

```
pdflatex GL_3_3_Subgroup_tetra_12_report.tex
open GL_3_3_Subgroup_tetra_12_report.pdf
pdflatex GL_3_3_Subgroup_tetra_12_elements.tex
open GL_3_3_Subgroup_tetra_12_elements.pdf
```
The Hesse group of order 216 extended by the automorphism group of the field can be created in $PG(3, 4)$

```
GENERATORS_HESSE_GROUP="\n3000300030 \
2000201230 \
1000100111 \
1000220200 \
1002312010 \
0331003211 \
2200011331"
```

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

The group has order 432.

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

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

151
Weyl\_E8:

```
$\text{(ORBITER)} -v 3 \\
  -\text{define\ gens\ -vector\ -dense} \\
  -\text{end} \\
  -\text{define\ G\ -linear\_group\ -GL\ 8\ 3} \\
  -\text{subgroup\_by\_generators} \\
  -\text{"Weyl\_E8"\ "696729600"\ 2\ gens} \\
  -\text{end} \\
  -\text{with\ G\ -do} \\
  -\text{group\_theoretic\_activity} \\
  -\text{report} \\
  -\text{end}
```

```
pdflatex GL_8_3_Subgroup_Weyl_E8_696729600_report.tex
open GL_8_3_Subgroup_Weyl_E8_696729600_report.pdf
```

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

```
```

We can test if a group is a subgroup of another. In the following example, we test whether PGO$^+(6,2)$ is a subgroup of PSp(6,2). The fact that it is depends on the choice of forms associated with the groups and on the fact that the characteristic is two.

```
test\_subgroup:
  $\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{symplectic\_group} \\
  -\text{end} \\
  -\text{with\ G1\ -and\ G2\ -do} \\
  -\text{group\_theoretic\_activity} \\
  -\text{is\_subgroup\_of} \\
  -\text{end}
```

152
Since the subgroup index is small (36), we create a set of coset representatives using the following command:

```bash
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:

```
coset 0 / 36:
\[
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
= \begin{bmatrix}
100000 \\
010000 \\
001000 \\
000100 \\
000010 \\
000001
\end{bmatrix}
\]

::

coset 35 / 36:
\[
\begin{bmatrix}
1 & 0 & 1 & 1 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 1 \\
0 & 1 & 1 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 & 1 & 0 \\
1 & 1 & 0 & 1 & 0 & 0
\end{bmatrix}
= \begin{bmatrix}
101110 \\
101000 \\
011101 \\
011111 \\
100010 \\
110100
\end{bmatrix}
\]
```

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

```bash
coset_reps_read:
```

153
> $(ORBITER) -v 2 \\
>  ▶ -define F -finite_field -q 2 -end \\
>  ▶ -define G1 -linear_group -PGOp 6 F -end \\
>  ▶ -define G2 -linear_group -PGL 6 F \\
>  ▶ ▶ -symplectic_group \\
>  ▶ ▶ -end \\
>  ▶ -define CR -vector_ge -action G2 \\
>  ▶ ▶ -read_csv \\
>  ▶ ▶ PGOp_6_2_coset_reps.csv Element \\
>  ▶ -end
It is sometimes necessary to control the finite field that is used in the construction of a matrix group. For prime fields, this is not an issue. For extension fields, the choice of polynomial does matter, as the generators depend on specific choices made for the finite field. Magma and GAP use Conway polynomials, which are difficult to compute. Orbiter has a built-in table of primitive polynomials. As explained in Section 3.3, Orbiter allows to specify the polynomial that should be used to create the finite field. The next example shows an instance where choosing the polynomial is important. We are recreating a group from the electronic Atlas on finite simple groups [71].

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

```magma
F<w>:=GF(9);
x:=CambridgeMatrix(1,F,3,[
  "164",
  "506",
  "851"])]
;y:=CambridgeMatrix(1,F,3,[
  "621",
  "784",
  "066"])
;G<x,y>:=MatrixGroup<3,F|x,y>;
```

The generators are given using the Magma command `CambridgeMatrix`, which allows for more efficient coding of field elements. The field elements are coded as base-3 integers (like in Orbiter) with respect to the Magma version of $\mathbb{F}_9$. The polynomial for $\mathbb{F}_9$ can be determined using the following Magma command, which can be typed into Magma (or the free Magma online calculator at [66]):

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

It results in

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

```
G<w>:=GF(9);
print DefiningPolynomial(G);
```
U_3.3:
\begin{verbatim}
$ (ORBITER) -v 3 \\
$ -define F -finite_field -q 9 -override_polynomial "17" -end \\
$ -define G -linear_group -PGL 3 F \\
$ -subgroup_by_generators "U_3.3" "6048" 2 \\
$ "1,6,4, 5,0,6, 8,5,1, \\
$ 6,2,1, 7,8,4, 0,6,6" \\
$ -end \\
$ -with G -do \\
$ -group_theoretic_activity \\
$ -report \\
$ -end \\
pdflatex PGL_3.9_Subgroup_U_3.3_6048_report.tex \\
open PGL_3.9_Subgroup_U_3.3_6048_report.pdf
\end{verbatim}

Group theoretic activities will be discussed in Section 5.6.

As an example of a large group, consider the Conway group Co3. Following [65], the group can be generated using two matrices of dimension 22 over \( \mathbb{F}_2 \). We use the makefile variables to give each generator in compact form. Then we define vectors for each of the generators. We concatenate the two generators to form one long vector, which is passed to the \texttt{-subgroup_by_generators} command. Finally, we create a report for the group.

\begin{verbatim}
CONWAY_GEN1="\n1101110001000001010000\n11110111110100001011\n0000010000001000101\n11111011010001001110\n01010100000001001111\n000001000001000101\n0010000000100010101\n00010000110100001111\n11101001001101001111\n00000000001001001011\n00000000000000001111\n00000000000011001010\n01101111111010011111\n00000000000001000101\n00000000000010000111\n00000000000010000101"
\end{verbatim}
CONWAY_GEN2="\
0101000010111010111111
01100100011110111000
0011010000111111101111
000110111001011010011
1010010000100010111110
110100000001001000011
1100101010001111010101
1000110100110101010101
0100110001000000000111
1100001010010110010001
0101110110011100000101
0101111101001111100101
1000101011010101010101
0001000001110001011111
0011001011011010101011
0100110010110001110001
0101011101111011000111
1101010110011101111001
0100110001001000100001
1100101100001001110001
0101110110010100000001
0000001101111000101110
1101101010101110000101""

Co3:
▷ $(ORBITER) -v 2 \n▷ ▷ -define F -finite_field -q 2 -end \n▷ ▷ -define g1 -vector -field F -format 22 -compact $(CONWAY_GEN1) -end \n▷ ▷ -define g2 -vector -field F -format 22 -compact $(CONWAY_GEN2) -end \n▷ ▷ -define gens -vector -concatenate g1 -concatenate g2 -end \n▷ ▷ -define G -linear_group -PGL 22 2 \n▷ ▷ ▷ -subgroup_by_generators "Co3" "495766656000" 2 gens \n▷ ▷ ▷ -end \n▷ ▷ -with G -do \n▷ ▷ -group_theoretic_activity \n▷ ▷ ▷ -report \n▷ ▷ -end
▷ pdflatex PGL_22_2_Subgroup.Co3.495766656000_report.tex
▷ open PGL_22_2_Subgroup.Co3.495766656000_report.pdf
The next example creates the Ree group in 7 dimensions over the field $\mathbb{F}_{2^7}$. Again, we use makefile variables to specify the two generators as $7 \times 7$ matrices over $\mathbb{F}_{2^7}$ and concatenate them, before passing them to the `-subgroup_by_generators` command.

```
Ree_gen1="21,5,1,6,17,1,1, 3,13,5,21,6,6,18, 21,3,21,22,6,14, \n14,18,1,5,13,6,7, 3,3,2,1,24,16,3, 17,3,22,10,16,24,26, \n21,21,6,18,20,25"
```

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

```
Ree_27:
  ▶ $(ORBITER) -v 2 \n  ▶ ▶ -define F -finite_field -q 27 -override_polynomial "34" -end \n  ▶ ▶ -define g1 -vector -field F -format 7 -dense $(Ree_gen1) -end \n  ▶ ▶ -define g2 -vector -field F -format 7 -dense $(Ree_gen2) -end \n  ▶ ▶ -define gens -vector -concatenate g1 -concatenate g2 -end \n  ▶ ▶ -define G -linear_group -PGL 7 F \n  ▶ ▶ ▶ -subgroup_by_generators "Ree_27" "10073444472" 2 gens \n  ▶ ▶ ▶ -end \n  ▶ ▶ -with G -do \n  ▶ ▶ -group_theoretic_activity \n  ▶ ▶ ▶ -report \n  ▶ ▶ -end
```
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-wedge</td>
<td></td>
<td>action on the exterior square</td>
</tr>
<tr>
<td>-wedge_detached</td>
<td></td>
<td>action on the exterior square. Unlike -wedge, this command does not establish the homomorphism to the original group. Instead, the group is created as subgroup of the larger general linear group.</td>
</tr>
<tr>
<td>-PGL2OnConic</td>
<td></td>
<td>induced action of PGL(2, q) on the conic in the plane PG(2, q)</td>
</tr>
<tr>
<td>-subfield_structure_action</td>
<td>s</td>
<td>action by field reduction to the subfield of index s</td>
</tr>
<tr>
<td>-on_k_subspaces</td>
<td>k</td>
<td>induced action on k dimensional subspaces</td>
</tr>
<tr>
<td>-on_tensors</td>
<td></td>
<td>induced action of GL(d, q) ⋊ Sym(n) on the tensor space</td>
</tr>
<tr>
<td>-on_rank_one_tensors</td>
<td></td>
<td>induced action of GL(d, q) ⋊ Sym(n) on the tensor space</td>
</tr>
<tr>
<td>-restricted_action</td>
<td>s</td>
<td>restricted action on the set s</td>
</tr>
</tbody>
</table>

Table 5.4: Commands for creating new actions from old

5.5 Induced Actions

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

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

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 $\text{GL}(2, 2) \ltimes \text{Sym}(3)$

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

The Action

Group action $\text{GL}(2, 2) \ltimes \text{Sym}(3)_{\text{res}255}$ of degree 255

Base and Stabilizer Chain

Group order 1296
$\text{tl}=3, 2, 1, 3, 2, 3, 2, 3, 2,$

Strong generators for a group of order 1296.

\[
\begin{align*}
&\left( \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} ; \text{id} \right), \\
&\left( \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} ; \text{id} \right), \\
&\left( \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} ; \text{id} \right), \\
&\left( \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} ; (1,2) \right), \\
&\left( \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} ; (0,1) \right), \\
&0,1,2,1,0,0,1,1,0,0,1,1,0,1,1,1, \\
&0,1,2,1,0,0,1,1,0,1,0,1,1,1,0,1, \\
&0,1,2,1,0,0,1,1,0,1,1,1,0,0,1, \\
&0,1,2,1,0,0,1,1,0,1,0,0,1,1,1, \\
&0,1,2,1,0,1,1,1,0,0,1,1,0,0,1, \\
&0,1,2,0,1,1,0,1,0,0,1,1,0,0,1, \\
&0,2,1,1,0,0,1,1,0,1,0,0,1,1,0,1, \\
&1,0,2,1,0,0,1,1,0,0,1,1,0,0,1, \\
&1,0,2,1,0,0,1,1,0,0,1,0,0,0,1, \\
&1,0,2,1,0,0,1,1,0,0,1,1,0,0,1, \\
&... \\
&160
\end{align*}
\]
Stabilizer chain

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

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

T3r1:

```
$ (ORBITER) -v 4 \
  -define G \n  -linear_group -GL_d_q_wr_Sym_n 2 2 3 \n  -on_rank_one_tensors -end \n  -with G -do \n  -group_theoretic_activity \n  -report \n  -end
```

`pdflatex GL_2_2_wreath_Sym3_report.tex`

`open GL_2_2_wreath_Sym3_report.pdf`

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

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

0,1,2,1,0,0,1,0,1,0,0,1,1,0,1,1,
0,1,2,1,0,0,1,0,1,0,1,1,0,1,1,
0,1,2,1,0,0,1,1,0,1,1,1,0,0,1,
0,1,2,1,0,0,1,1,0,1,1,0,0,1,
0,1,2,1,0,0,1,1,0,0,1,1,0,0,1,
0,2,1,1,0,0,1,0,1,0,1,0,1,0,1,
1,0,2,1,0,0,1,1,0,1,1,0,0,1,
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 \texttt{PGL(2,8)} of PG$(1,8)$ and act on 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 \ }
  pdflatex PGGL_2.8_OnConic_2.8_report.tex
  open PGGL_2.8_OnConic_2.8_report.pdf
\end{verbatim}

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

\begin{center}
\textbf{The Group} \texttt{PGL(2,8)OnConic(2,8)}
\end{center}

The order of the group \texttt{PGL(2,8)OnConic(2,8)} is 1512

The group acts on a set of size 73

Strong generators for a group of order 1512:

\begin{align*}
  &\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, &
\end{align*}
The Action

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

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

The group is a matrix group.
The base action is on projective space $\text{PG}(1, 8)$
$q = 8$
$p = 2$
$e = 3$
$n = 1$
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>DNE</td>
<td>DNE</td>
<td>1</td>
<td>DNE</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1 = 1</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>$\alpha = \gamma$</td>
<td>2</td>
<td>6</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>$\alpha + 1 = \gamma^5$</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>
\[
\begin{array}{c|cccccccc}
+ & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
0 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
1 & 1 & 0 & 3 & 2 & 5 & 4 & 7 & 6 \\
2 & 2 & 3 & 0 & 1 & 6 & 7 & 4 & 5 \\
3 & 3 & 2 & 1 & 0 & 7 & 6 & 5 & 4 \\
4 & 4 & 5 & 6 & 7 & 0 & 1 & 2 & 3 \\
5 & 5 & 4 & 7 & 6 & 1 & 0 & 3 & 2 \\
6 & 6 & 7 & 4 & 5 & 2 & 3 & 0 & 1 \\
7 & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\
\end{array}
\]

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

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

Base and Stabilizer Chain

Group order 1512

\( tl=9, 8, 7, 3, \)

Stabilizer chain

<table>
<thead>
<tr>
<th>Level</th>
<th>Base pt</th>
<th>Orbit length</th>
<th>Subgroup order</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>9</td>
<td>1512</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>8</td>
<td>168</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>7</td>
<td>21</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>
Basic Orbit 0

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

Basic Orbit 1

Basic orbit 1 has size 8
1, 2, 3, 4, 5, 6, 7, 8
In the following example, we will demonstrate two types of induced actions. One is the action induced on $k$-dimensional subspaces. The second is the restricted action on an invariant subset. The example we show is related to cubic surfaces. At first, we create the Eckardt surface in $PG(3,13)$ from the arc

$$\{0,1,2,3,43,113\}.$$
Then we export the set of 45 tritangent planes to file and we produce a report about the surface and its automorphism group. The next command creates the stabilizer of the surface from the generators given in the report, creates the induced action on planes, and restricts the action to the 45 tritangent planes stored in the file. Here is the fill command sequence, including a makefile variable for the generators of the stabilizer of the surface:

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

**surface_q13_stab_on_tritangents_orbits:**

```
$($ORBITER) -v 3 \\
  -define F -finite_field -q 13 -end \\
  -define P -projective_space -n 3 -field F -v 0 -end \\
  -define S -cubic_surface -space P -arc_lifting "0,1,2,3,43,113" -end \\
  -with S -do \\
  -cubic_surface_activity \\
  -report_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 \\
  -define Orb -orbits -group Gr \\
```

168
▷▷▷ -on_points \\
▷▷ -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 -group_theoretic_activity option. Tables 5.5 and 5.6 list the possible commands that can come after it.

The command

PGL_3_2_elements:
  ▶ $(ORBITER) -v 5 \n  ▶ ▶ -define G -linear_group -PGL 3 2 -end \n  ▶ ▶ -with G -do \n  ▶ ▶ -group_theoretic_activity \n  ▶ ▶ ▶ -save_elements_csv "PGL_3_2_elements.csv" \n  ▶ ▶ -end

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

The command

Sym_3_elements:
  ▶ $(ORBITER) -v 3 \n  ▶ ▶ -define G -permutation_group -symmetric_group 3 -end \n  ▶ ▶ -with G -do \n  ▶ ▶ -group_theoretic_activity \n  ▶ ▶ ▶ -print_elements_tex \n  ▶ ▶ ▶ -end \n  ▶ $(ORBITER) -v 2 \n  ▶ ▶ -draw_options \n  ▶ ▶ ▶ -nodes \n  ▶ ▶ ▶ ▶ -embedded -radius 250 \n  ▶ ▶ ▶ ▶ -xin 10000 -yin 10000 \n  ▶ ▶ ▶ ▶ -xout 1000000 -yout 600000 \n  ▶ ▶ ▶ ▶ -scale 0.3 -line_width 1.0 \n  ▶ ▶ ▶ -end \n  ▶ ▶ -tree_draw -file Perm3_elements_tree.txt -end \n  ▶ pdflatex Perm3_elements_tree_draw.tex \n  ▶ open Perm3_elements_tree_draw.pdf

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

It is possible to compute all powers of a fixed element, as in the following command:
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-apply</td>
<td>$a \ s$</td>
<td>Applies the group element given by the coded vector $s$ to the element $a$.</td>
</tr>
<tr>
<td>-multiply</td>
<td>$s_1 \ s_2$</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 $s$, which is given in coded form. Produces a latex report.</td>
</tr>
<tr>
<td>-export_orbiter</td>
<td></td>
<td>Exports the group to Orbiter.</td>
</tr>
<tr>
<td>-export_gap</td>
<td></td>
<td>Exports the group to GAP [29].</td>
</tr>
<tr>
<td>-export_magma</td>
<td></td>
<td>Exports the group to Magma [14].</td>
</tr>
<tr>
<td>-search_element_of_order</td>
<td>$i$</td>
<td>Finds all elements of order $i$ in the group ($i \in \mathbb{N}$).</td>
</tr>
<tr>
<td>-element_rank</td>
<td>$s$</td>
<td>Determines the rank of the group element $s$ in the given group. $s$ is given in coded form.</td>
</tr>
<tr>
<td>-element_unrank</td>
<td>$r$</td>
<td>Produces the group element whose rank is $r$.</td>
</tr>
<tr>
<td>-find_singer_cycle</td>
<td></td>
<td>Finds all Singer cycles whose matrix is a companion matrix.</td>
</tr>
<tr>
<td>-poset_classification_control</td>
<td>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 $g_ig_j$, $1 \leq i, j \leq n$.</td>
</tr>
<tr>
<td>Command</td>
<td>Arguments</td>
<td>Purpose</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>-classify_arcs</td>
<td>description</td>
<td>Classify arcs in geometries. See Section 6.6.</td>
</tr>
<tr>
<td>-linear_codes</td>
<td>$d \ n_{\text{max}}$</td>
<td>Classify linear codes with prescribed minimum distance $d$. Assumes that the group is PGL($r, q$) or PTL($r, q$). For each $n \leq n_{\text{max}}$, the $[n, k, \geq d]$ codes are classified with $n - k = r$. See Section 10.</td>
</tr>
<tr>
<td>-tensor_classify</td>
<td>$d$</td>
<td>Classifies tensors of tensor-rank at most $d$.</td>
</tr>
<tr>
<td>-tensor_permutations</td>
<td></td>
<td>Computes the permutation representation of generators of wreath product.</td>
</tr>
<tr>
<td>-reverse_iso</td>
<td></td>
<td>Given a set of generators of a subgroup of PGO$^+(6, q)$ as $6 \times 6$ matrixes, compute the inverse image of the generators in PGL($4, q$) (if possible).</td>
</tr>
<tr>
<td>-classify_cubic_curves</td>
<td>descr</td>
<td>Classifies cubic curves. Expects an arc description options as in Table 6.4.</td>
</tr>
</tbody>
</table>

Table 5.6: Group theoretic activities (Part 2)

![Figure 5.4: The elements of Sym(3) in lex-order](image_url)
Cycle_12_power:

```
$\$(ORBITER) -v 5 \n$\$ -define G -permutation_group -symmetric_group 12 -end \n$\$ -with G -do \n$\$ -group_theoretic_activity \n$\$ -consecutive_powers \n$\$ "1,2,3,4,5,6,7,8,9,10,11,0" 12 \n$\$ -end
```

We create the 12 powers of the cycle 

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

The output is

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

The command

```
PGL_3.4_singer:
```

```
$\$(ORBITER) -v 5 \n```

173
finds all Singer cycles in $\mathrm{PGL}(3, 4)$ 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 $\mathrm{GL}(2, 8)$. We multiply the elements coded by 0,1,2,3 and 4,5,6,7:

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

The output is

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

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:
  \>$\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
  \> 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:
  \>$\text{(ORBITER)} -v 5 \$
  \>  \> -define G -linear_group -GL 2 7 -end \$
  \>  \> -with G -do \$
  \>  \> -group_theoretic_activity \$
  \>  \>  \> -inverse "0,1,2,3" \$
  \>  \> -end
  \> pdflatex GL_2.7_inv.tex
  \> open GL_2.7_inv.pdf

The output is

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

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

GL\_2.7\_power:

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

```
pdflatex GL\_2.7\_power.tex
open GL\_2.7\_power.pdf
```

The output is

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

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

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

on\_planes:

```
$ (ORBITER) -v 2 \\
  -define F -finite_field -q 8 -end \\
  -define P -projective_space -n 3 -field F -v 0 -end \\
  -define G -linear_group -PGL 4 F -end \\
  -define G\_on\_planes -modified\_group -from G \\
  -on_k\_subspaces 3 \\
  -with G\_on\_planes -do \\
  -group\_theoretic\_activity \\
  -apply "0,8,1,6,4,3,7,2,5" \\
  "1,0,0,0, 0,1,0,0, 0,0,0,2, 0,0,1,1" \\
  -end
```

```
pdflatex PGL\_4.8\_Gr\_4.3\_apply.tex
open PGL\_4.8\_Gr\_4.3\_apply.pdf
```

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

1,0,0,0,1,0,0,0,0,2,0,0,1,1, maps:
0 ↦→ 8
8 ↦→ 1
1 ↦→ 3
6 ↦→ 5
4 ↦→ 7
3 ↦→ 4
7 ↦→ 6
2 ↦→ 0
5 ↦→ 2
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-classes</td>
<td></td>
<td>Compute a report of the conjugacy classes of elements.</td>
</tr>
<tr>
<td>-centralizer_of_element</td>
<td>label coding</td>
<td>Compute the centralizer of the coded group element, using label to create file names.</td>
</tr>
<tr>
<td>-normalizer_of_cyclic_subgroup</td>
<td>label s</td>
<td>Compute the normalizer of the cyclic subgroup generated by the element s.</td>
</tr>
<tr>
<td>-normalizer</td>
<td></td>
<td>Compute the normalizer of a subgroup in the larger group.</td>
</tr>
</tbody>
</table>

Table 5.7: Group theoretic activities based on Magma

5.7 Group Theoretic Activities Based on Magma

Through its interface to Magma [14], Orbiter can perform group theoretic computations. Table 5.7 list the group theoretic commands that rely on Magma. The communication to and from magma happens through files. This is a three step process: An Orbiter session receives a command to compute the conjugacy classes of a group. The Orbiter session writes a magma file. This file is read and executed by Magma. Magma writes a second file containing the conjugacy classes in coded form. Another Orbiter session reads the magma output file, decodes the information and produces the desired list of conjugacy classes. A latex report is written containing the classes, as well as related information regarding centralizers and normalizers.

For instance, the three-step command sequence

```
PGGL_2_4_classes:
  > $(ORBITER) -v 3 \n  >   -define G \n  >   -linear_group -PGGL 2 4 \n  >   -end \n  >   -with G -do \n  >   -group_theoretic_activity \n  >     -classes \n  >   -end
  > $(MAGMA_PATH)magma PGGL_2_4_classes.magma
  > $(ORBITER) -v 3 \n  >   -define G \n  >   -linear_group -PGGL 2 4 \n  >   -end \n  >   -with G -do \n  >   -group_theoretic_activity \n  >     -classes \n  >   -end
```
computes the classes of elements in $\text{PGL}(2, 4)$ using Orbiter-Magma-Orbiter. The first Orbiter command produces the file $\text{PGGL}_2_4\_\text{classes}.\text{magma}$. The magma command reads this file and produces the file $\text{PGGL}_2_4\_\text{classes}\_\text{out}.\text{txt}$. The second Orbiter command reads the file $\text{PGGL}_2_4\_\text{classes}\_\text{out}.\text{txt}$ and produces the latex report $\text{PGGL}_2_4\_\text{classes}\_\text{out}.\text{tex}$.

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

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

Representing element is $c_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ of order 2 and with 3 fixed points. 0, 1, 1, 0, 1,

The normalizer is generated by:

Strong generators for a group of order 12:

$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}_1, \begin{bmatrix} \omega^2 & 0 \\ 0 & 1 \end{bmatrix}_1, \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}_1$

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

The command sequence

```
PGGL_2_4\_\text{cent}\_2A:
  $\text{(ORBITER)} -v 3 \$
  $\text{-define G }$
  $\text{-linear\_group -PGGL 2 4 -end }$
  $\text{-with G -do }$
  $\text{-group\_theoretic\_activity }$
```
computes the centralizer of the Baer involution
\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\].

The centralizer is a group of order 40320, isomorphic to $\text{PGL}(4,2).\mathbb{Z}_2$. Orbiter produces a list of strong generators, shown below:

Strong generators for a group of order 40320:

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\], \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
1 & 1 & 1 & 1 \\
\end{bmatrix}
\], \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
1 & 1 & 0 & 1 \\
\end{bmatrix}
\], \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 \\
1 & 0 & 0 & 1 \\
\end{bmatrix}
\], \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\], \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\], \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 \\
1 & 1 & 1 & 1 \\
\end{bmatrix}
\], \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\].
The end of the report has a list of generators in coded form. This list can be used to create the centralizer in Orbiter.

Orbiter can compute the normalizer of a subgroup. The group must be constructed as a subgroup $H$ of a larger group $G$ containing $H$. Typically, the group $G$ is the group in which $H$ is generated as a subgroup (either the full linear group of the full symmetric group). Then, the normalizer of $H$ in $G$ is computed. Here is an example in a symmetric group. We first create a subgroup of order 5, using a makefile variable. The generator is

$$(0, 1, 2, 3, 4)(5, 6, 7, 8, 9)(10)(11)(12).$$

We store it in a makefile variable:

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

The command

```
Normalizer_of_H5:
  ▷ $(ORBITER) -v 2 \n  ▷ ▷ -define G -permutation_group -symmetric_group 13 \n  ▷ ▷ ▷ -subgroup_by_generators H5 5 1 \n  ▷ ▷ ▷ ▷ $(GENERATORS_H5) -end \n  ▷ ▷ -with G -do \n  ▷ ▷ -group_theoretic_activity \n  ▷ ▷ ▷ -normalizer \n  ▷ ▷ -end
  ▷ pdflatex Perm13_Subgroup_H5_5_normalizer.tex
  ▷ open Perm13_Subgroup_H5_5_normalizer.pdf
```

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

```
quit;
```
The Orbiter command has to be run one more time after that. The following report is produced:

<table>
<thead>
<tr>
<th>The group Perm13SubgroupH5order5 of order 5 is:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strong generators for a group of order 5:</td>
</tr>
<tr>
<td>(0, 1, 2, 3, 4)(5, 6, 7, 8, 9)</td>
</tr>
</tbody>
</table>

1, 2, 3, 4, 0, 6, 7, 8, 9, 5, 10, 11, 12,
Inside the group of order 6227020800, the normalizer has order 1200:

<table>
<thead>
<tr>
<th>Strong generators for a group of order 1200:</th>
</tr>
</thead>
<tbody>
<tr>
<td>(11, 12),</td>
</tr>
<tr>
<td>(10, 11),</td>
</tr>
<tr>
<td>(5, 9, 8, 7, 6),</td>
</tr>
<tr>
<td>(1, 2, 4, 3)(6, 7, 9, 8),</td>
</tr>
<tr>
<td>(0, 5)(1, 9)(2, 8)(3, 7)(4, 6)</td>
</tr>
</tbody>
</table>

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

Consider this example of a subgroup which is not cyclic: The group

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

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

```
Normalizer_of_Z22_in_PGL_2_9:
  \$\$(\text{ORBITER}) -v 2 \$
  \$\$ -define G -linear_group -PGL 2 9 \$
  \$\$ -subgroup_by_generators Z22 4 2 \$
  \$\$ -with G -do \$
  \$\$ -group_theoretic_activity \$
  \$\$ -normalizer \$
```
It produces a report showing that the normalizer is a group of order 24 (it is isomorphic to Sym(4), though the report does not tell us this fact directly):

The group PGL(2, 9)SubgroupZ22order4 of order 4 is:
Strong generators for a group of order 4:
\[
\begin{bmatrix}
\alpha^4 & 0 \\
0 & 1 \\
\end{bmatrix},
\begin{bmatrix}
0 & 1 \\
1 & 0 \\
\end{bmatrix}
\]

1,0,0,2,
0,1,1,0,
Inside the group of order 720, the normalizer has order 24:

Strong generators for a group of order 24:
\[
\begin{bmatrix}
\alpha^4 & 0 \\
0 & 1 \\
\end{bmatrix},
\begin{bmatrix}
\alpha^2 & 0 \\
0 & 1 \\
\end{bmatrix},
\begin{bmatrix}
\alpha^4 & \alpha^4 \\
\alpha^4 & 1 \\
\end{bmatrix},
\begin{bmatrix}
\alpha^4 & \alpha^6 \\
\alpha^2 & 1 \\
\end{bmatrix}
\]

1,0,0,2,
1,0,0,5,
1,1,1,2,
1,7,5,2,
Chapter 6

Orbit Algorithms

6.1 Schreier Trees

Orbiter provides several different orbit algorithms. The most basic orbit algorithm uses Schreier trees. It is explained in [17, 36, 63]. This algorithm has memory and time complexity proportional to the size of the orbit. It therefore is limited to small problems. More elaborate algorithms exist, provided the set on which we act has additional structure. These algorithms offer sublinear complexity. Orbiter offers posets based algorithms, which exploit an underlying poset structure. They will be discussed in Section 6.2. Orderly generation using canonical forms is discussed in Section 15.2.

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

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

```
orbits PGL_4_2_on_points:
  $(ORBITER) -v 4 \n  ▶ ▶ -define G -linear_group -PGL 4 2 -end \n  ▶ ▶ -define Orb -orbits -group G \n  ▶ ▶ ▶ -on_points \n  ▶ ▶ -end
  $(ORBITER) -v 3 \n  ▶ ▶ -draw_layered_graph \n  ▶ ▶ ▶ PGL_4_2_0.layered_graph \n  ▶ ▶ -radius 500 -spanning_tree -embedded \n  ▶ ▶ ▶ -line_width 1.1 -x_stretch 1.4 -scale 0.25 \n  ▶ ▶ -end
  pdflatex PGL_4_2_0_draw.tex
  open PGL_4_2_0_draw.pdf
  pdflatex PGL_4_2_orbits_report.tex
  open PGL_4_2_orbits_report.pdf
```
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-orbits_on_subsets</code></td>
<td>$k$</td>
<td>Compute orbits on $k$-subsets.</td>
</tr>
<tr>
<td><code>-orbits_on_points</code></td>
<td></td>
<td>Compute orbits in the action that was created.</td>
</tr>
<tr>
<td><code>-orbits_of</code></td>
<td>$i$</td>
<td>Compute orbit of point $i$ in the given action.</td>
</tr>
<tr>
<td><code>-stabilizer</code></td>
<td></td>
<td>Compute the stabilizer of the orbit representative (needs <code>-orbits_on_points</code>).</td>
</tr>
<tr>
<td><code>-orbits_on_set_system_from_file</code></td>
<td>$f\ldots l$</td>
<td>Reads the csv file “fname” and extract sets from columns $[f,\ldots, f + l - 1]$.</td>
</tr>
<tr>
<td><code>-orbit_of_set_from_file</code></td>
<td>$f$</td>
<td>Reads a set from the text file “fname” and computes orbits on the elements of the set.</td>
</tr>
<tr>
<td><code>-orbits_on_polynomials</code></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><code>-conjugacy_class_of</code></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><code>-orbits_on_group_elements_under_conjugation</code></td>
<td>$f$name-C $f$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 $f$name-C and $f$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
The Schreier tree for the action of $\text{PGL}(4, 2)$ on points of $\text{PG}(3, 2)$ is shown in Figure 6.1.

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

```
T3r1_orbits:
  > $(ORBITER) -v 4 \\
  >   -define G \\
  >   -linear_group -GL_d_q_wr_Sym_n 2 2 3 \\
  >   -on_rank_one_tensors -end \\
  >   -define Orb -orbits -group G \\
  >   -on_points \\
  >   -end \\
  >   -with Orb -do -orbits_activity \\
  >   -report \\
  >   -end \\
  >   -with Orb -do -orbits_activity \\
  >   -draw_tree 0 \\
  >   -end
```

In the next example, we compute the orbits of the linear group $\text{PGL}(4, 2)$ on homogeneous polynomials of degree 3 in 4 variables:

```
pdflatex GL_2_2_wreath_Sym3_orbit_0_tree.tex
open GL_2_2_wreath_Sym3_orbit_0_tree.pdf
```
Figure 6.2: The Schreier tree for the action on rank-one tensors

orbits_cubic_curves_q2:

```
$ (ORBITER) -v 4 \
  -define G -linear_group -PGL 3 2 -end \\
  -define Orb -orbits -group G \\
  -on_polynomials 3 \\
  -end
```

#pdflatex poly_orbits_d3_n3_q2.tex

#open poly_orbits_d3_n3_q2.pdf

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

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

PGGL_2_8_on_conic_orbits:

```
$ (ORBITER) -v 4 \
  -linear_group -PGGL 2 8 -PGL2OnConic -end \\
  -define Orb -orbits -group G \\
  -on_points \\
```

188
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 PGL(2, 8)OnConic:
Strong generators for a group of order 1512:

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

0,1,0,0,1,1,
0,0,0,6,0,
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 : orbit length : number of orbits
0 : 1 : 1
1 : 9 : 1
2 : 63 : 1

Orbits classified:
Set 0 has size 1 : \{1\}
Set 1 has size 1 : \{0\}
Set 2 has size 1 : \{2\}

Orbits of length 1:
Orbit 1: (1)

0 : 1 = (0, 1, 0)
Orbits of length 9:
Orbit 0: ( 0, 2, 3, 29, 48, 38, 55, 60, 67 )

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

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

0 : 4 = ( 1, 1, 0 )
1 : 5 = ( 2, 1, 0 )
2 : 18 = ( 0, 1, 1 )
3 : 7 = ( 4, 1, 0 )
4 : 16 = ( 6, 0, 1 )

Orbits of length 1:
Orbit 1: ( 1 )
Stabilizer of orbit representative 1:
Strong generators for a group of order 1512:

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

1,0,0,1,1,
1,0,0,6,0,
1,0,2,1,0,
0,1,1,0,0,
Generator 0 / 4 is:

\[
\begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix}
\]
 Generator 1 / 4 is:
\[
\begin{bmatrix}
\gamma & 0 \\
0 & 1
\end{bmatrix}_0
\]

 Generator 2 / 4 is:
\[
\begin{bmatrix}
1 & 0 \\
\gamma & 1
\end{bmatrix}_0
\]

 Generator 3 / 4 is:
\[
\begin{bmatrix}
0 & 1 \\
1 & 0
\end{bmatrix}_0
\]

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:
\[
\begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix}_1, \begin{bmatrix}
\gamma^6 & 0 \\
0 & 1
\end{bmatrix}_0, \begin{bmatrix}
\gamma^4 & 0 \\
\gamma^2 & 1
\end{bmatrix}_0
\]
1,0,0,1,1,
1,0,0,2,0,
1,0,3,5,0,
Generator 0 / 3 is:
\[
\begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix}_1
\]
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, 26, 21, 72, 54, 39, 13, 20, 43, 70, 50, 61, 17, 22, 44, 35, 23, 46, 40, 51, 28, ...12, 31, 16 )
Stabilizer of orbit representative 4:
Strong generators for a group of order 24:
\[
\begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix}
\begin{bmatrix}
1 & 0 \\
\gamma^5 & 1
\end{bmatrix}
\begin{bmatrix}
1 & 0 \\
\gamma & 1
\end{bmatrix}
\]
1,0,0,1,1,
1,0,3,1,2,
1,0,5,1,0,
1,0,2,1,0,
Generator 0 / 4 is:
\[
\begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix}
\]
Generator 1 / 4 is:

\[
\begin{bmatrix}
1 & 0 \\
\gamma^5 & 1
\end{bmatrix}_2
\]

Generator 2 / 4 is:

\[
\begin{bmatrix}
1 & 0 \\
\gamma^3 & 1
\end{bmatrix}_0
\]

Generator 3 / 4 is:

\[
\begin{bmatrix}
1 & 0 \\
\gamma & 1
\end{bmatrix}_0
\]
6.2 Poset Classification

A partially ordered set (poset) is a set together with a partial order. For instance, the set of subsets of a fixed set form an order structure with respect to set-inclusion. The Hasse diagram is a diagram whose nodes represent the element. Nodes are arranged from top to bottom, and relations are indicated by lines. Transitivity is implied. For instance, Figure 6.3 shows the power set lattice of a four-element subset.

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

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

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

Orbiter uses the algorithm of Schmalz [62] to perform poset classification. Two versions are available: one for subset-type posets and one for subspace-type posets. Figure 6.4 shows the subspace lattice of $V(3, 2) = \mathbb{F}_2^3$. The basis elements are listed, using the enumerator for elements on the projective geometry 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

\[
\texttt{-preferred\_choice \( n \ a \ b \)}
\]

can be used to force Orbiter to choose \( b \) instead of \( a \) at node \( n \).
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-problem_label</td>
<td>str</td>
<td>Use str as a prefix for files that are created.</td>
</tr>
<tr>
<td>-path</td>
<td>p</td>
<td>Use path p for files that are created.</td>
</tr>
<tr>
<td>-depth</td>
<td>d</td>
<td>Set search depth to d.</td>
</tr>
<tr>
<td>-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 A_{sup} and A_{inf}.</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>Command</th>
<th>Arguments</th>
<th>Purpose</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_with_flag_orbits</td>
<td></td>
<td>Produce a bitmap drawing of the neighboring relations in the poset with 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_is_group_order</td>
<td></td>
<td>When drawing the poset of orbits, display the group order in the orbit nodes.</td>
</tr>
<tr>
<td>-node_label_is_element</td>
<td></td>
<td>When drawing the poset of orbits, display the element rank in the orbit 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 transporter that maps the given object to the canonical form. Here, $L$ must be a list of integers (comma separated and enclosed in double quotes) encoding an object. This option can be repeated.</td>
</tr>
<tr>
<td>-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 representative.</td>
</tr>
</tbody>
</table>

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

The lattice of subsets of a set $X$ is $\mathcal{P}(X)$, the set of all subsets of $X$, ordered with respect to inclusion. Assume that a group $G$ acts on $X$, and hence on the lattice by means of the induced action on subsets. The orbits of $G$ on subsets form a new poset, the poset of orbits. Poset classification is the process of computing the poset of orbits. Orbiter has an algorithm to perform poset classification. In many cases, we are not interested in the full lattice of subsets $\mathcal{P}(X)$ but rather in a subposet of it. We require that the subposet is closed under the group action and that the following property holds:

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

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

```
PGL_3_2_singer:
  $(ORBITER) -v 3 \n  -orbiter_path $(ORBITER_PATH) \n  -define Control -poset_classification_control \n  -problem_label PGL_3_2_singer -W -depth 7 \n  -draw_poset \n  -report -end \n  -end \n  -define G -linear_group -PGL 3 2 -singer 1 -end \n  -define Orb -orbits -group G \n  -on_subsets 7 Control \n  -end
  pdflatex PGL_3_2_singer_poset.tex
  open PGL_3_2_singer_poset.pdf
```

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

```
PG_3_2_subsets:
  $(ORBITER) -v 3 \n  -orbiter_path $(ORBITER_PATH) \n  -define Control -poset_classification_control \n  -problem_label PGL_4_2 \n  -depth 15 \n  -draw_options \n  -radius 200 -embedded \n  -report -end
```

198
A drawing of the poset of orbits as in Figure 6.5 is produced.

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

$\text{PGL}_3.2_{\text{on_lines}}$:  
\> $(\text{ORBITER}) -v 3 \backslash$
The following example computes the orbits of $\text{PGO}(5, 2)$ on the power set lattice of points of $\mathbb{Q}(4, 2)$:

$\text{PGO}_5 \text{ on subsets:}$

```
$($\text{ORBITER}) \ -v \ 3$
$-\text{orbiter_path} \ ($(\text{ORBITER\_PATH})$
$-\text{define Control} \ -\text{poset\_classification\_control}$
$-\text{problem\_label} \ \text{PGO\_5\_2} \ -\text{depth} \ 15$
$-\text{w} \ -\text{end}$
$-\text{define F} \ -\text{finite\_field} \ -q \ 2 \ -\text{end}$
$-\text{define G} \ -\text{linear\_group} \ -\text{PGO} \ 5 \ F \ -\text{end}$
$-\text{define Orb} \ -\text{orbits} \ -\text{group} \ \text{G}$
$-\text{on\_subsets} \ 15 \ \text{Control}$
$-\text{end}$
```

```
pdflatex PGO_5_2_poset.tex
open PGO_5_2_poset.pdf
```

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

Orbiter can compute the orbits of a group on the lattice of subspaces of a finite vector space. The orthogonal group is the stabilizer of a non-degenerate quadric. Suppose we want to classify the subspaces in PG(3, 2) under the action of the orthogonal group. In PG(3, 2) there are two distinct nondegenerate quadrics, $Q^+(3, 2)$ and $Q^-(3, 2)$. The $Q^+(3, 2)$ quadric is a finite version of the quadric given by the equation

$$x_0x_1 + x_2x_3 = 0,$$

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

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

The $Q^+(3, 2)$ quadric given by the equation above consists of the nine points $P_0, P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8$. The quadric is stabilized by the group $PGO^+(4, 2)$ of order 72. The command

\[
\text{subspaces.Op.4.2:}
\]

\[
\begin{align*}
\&\texttt{\$\{ORBITER\} -v 5} \& \\
\&\texttt{\$\{ORBITER\} -orbiter_path $\{ORBITER\}_\text{PATH}\} \& \\
\&\texttt{\$\{ORBITER\} -define Control -poset_classification_control} \& \\
\&\texttt{\$\{ORBITER\} -node_label_is_element} \&
\end{align*}
\]
Figure 6.8: Hasse-diagram for the orbits of the orthogonal group \( \text{PGO}^+(4, 2) \) on subspaces of \( \text{PG}(3, 2) \)

```
▷ ▷ ▷ -draw_poset -draw_options -radius 200 -end \ 
▷ ▷ ▷ -problem_label Op_4_2 -W -depth 4 \ 
▷ ▷ ▷ -report -end \ 
▷ ▷ -end \ 
▷ ▷ -define G -linear_group -PGL 4 2 -orthogonal 1 -end \ 
▷ ▷ -define Orb -orbits -group G \ 
▷ ▷ ▷ -on_subspaces 4 Control \ 
▷ ▷ -end
```

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

Orbiter can compute the orbits of a group on set-partitions. The set-partition needs to have three parts of equal size.

The command

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

computes the orbits of the cyclic group $C_6$ on set-partitions of type 2 + 2 + 2. There are 15 set-partitions, and they fall into 5 orbits, with stabilizer orders $3, 1, 2, 2, 6$.

The orbit count gives

$$6 \left( \frac{1}{3} + \frac{1}{1} + \frac{1}{2} + \frac{1}{2} + \frac{1}{6} \right) = 15.$$
computes the orbits of the group $\text{PGL}(2, 17)$ on set-partitions of type $6 + 6 + 6$. The number of set-partitions is

$$\frac{\binom{18}{6} \cdot \binom{12}{6}}{3!} = 2858856$$

There are 720 orbits. The orbit stabilizer statistic is

$$(1^{480}, 2^{184}, 3^{11}, 4^{20}, 6^{15}, 8, 12^{6}, 18, 24, 36).$$

The orbit-stabilizer count confirms that

$$4896 \left( \frac{480}{1} + \frac{184}{2} + \frac{11}{3} + \frac{20}{4} + \frac{15}{6} + \frac{1}{8} + \frac{6}{12} + \frac{1}{18} + \frac{1}{24} + \frac{1}{36} + \right) = 2858856.$$
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-q</td>
<td>$q$</td>
<td>Specify the size of the field $\mathbb{F}_q$.</td>
</tr>
<tr>
<td>-d</td>
<td>$d$</td>
<td>Require that no more than $d$ points lie on a line.</td>
</tr>
<tr>
<td>-n</td>
<td>$n$</td>
<td>The size of the matrix group.</td>
</tr>
<tr>
<td>-target_size</td>
<td>$t$</td>
<td>Specify the size of the arc to be $t$.</td>
</tr>
<tr>
<td>-conic_test</td>
<td></td>
<td>Require that no 6 points of the arc lie on a conic.</td>
</tr>
<tr>
<td>-affine</td>
<td></td>
<td>Classify arcs in the affine geometry, assuming that $x_0 = 0$ is the hyperplane at infinity. The condition that no more that $d$ point lie on a line applies to affine lines only.</td>
</tr>
<tr>
<td>-no_arc_testing</td>
<td></td>
<td>Do not test the at most $d$ points per line condition.</td>
</tr>
<tr>
<td>-forbidden_point_set</td>
<td>set</td>
<td>The arc must not contain any of the given points.</td>
</tr>
</tbody>
</table>

**Table 6.4: Commands for Classifying Arcs**

### 6.6 Arcs and Caps in Projective Spaces

In $\text{PG}(n,q)$, an arc is a set of points, no $n + 1$ in a hyperplane. A cap is set of points, no three collinear. Here, we restrict our attention to arcs in $\text{PG}(2,q)$. Arcs in higher dimensional projective spaces are equivalent to MDS codes and will be treated in Section 10. Our main examples will be the construction of the Lunelli-Sce hyperoval in $\text{PG}(2,16)$ (cf. [48]) and the Pellegrino cap in $\text{AG}(4,3)$. The uniqueness of this cap was proven by Hill [31].

A $(k,d)$-arc in a projective plane $\pi$ is a set $S$ of $k$ points such that very line intersects $S$ in at most $d$ points. Arcs are related to linear codes and other structures. Two arcs $S_1$ and $S_2$ are equivalent if there is a projectivity $\Phi$ such that $\Phi(A) = B$. The problem of classifying arcs is the problem of determining the orbits of the projectivity group on arcs. At times, we consider the larger group of collineations. In that case, the problem of classifying arcs is the problem of determining the orbits of the collineation group on arcs. Orbiter can solve such classification problems, at least for small parameter cases. Table 6.4 list the commands available to classify arcs. Here is an example. A hyperoval in a plane $\text{PG}(2,2^e)$ is a $(2^e + 2,2)$-arc. It is interesting to classify the hyperovals up to collineation equivalence under the group $\text{PGL}(3,2^e)$. The command

```
subspaces.Op.4.2:
  \>$\$(\text{ORBITER}) -v 5 \$
  \>$\$ -orbiter_path $\$(\text{ORBITER_PATH}) \$
  \>$\$ -define Control -poset_classification_control \$
  \>$\$ -node_label_is_element \$
```
performs the classification of hyperovals in PG(2, 16). There are exactly two hyperovals in this plane. Orbiter also finds the stabilizers of these arcs. They have orders 16320 and 144, respectively. The two hyperovals are the regular hyperoval and the Lunelli-Sce hyperoval. Here is the relevant output from the Orbiter report (in the output, the Lunelli-Sce hyperoval is orbit 0, and the regular hyperoval is orbit 1):

**Orbits at Level 18**

There are 2 orbits at level 18.

**Orbit 0 / 2 at Level 18**

Node number: 4212

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

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

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

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}

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 : 156 = ( 11, 8, 1 )
15 : 229 = ( 4, 13, 1 )
16 : 256 = ( 15, 14, 1 )
17 : 264 = ( 7, 15, 1 )

Strong generators for a group of order 16320:

\[
\begin{bmatrix}
\delta^6 & 0 & 0 \\
0 & \delta^3 & 0 \\
0 & 0 & 1
\end{bmatrix}, \quad \begin{bmatrix}
\delta^9 & 0 & 0 \\
0 & \delta^7 & 0 \\
0 & 0 & 1
\end{bmatrix}, \quad \begin{bmatrix}
\delta^2 & 0 & 0 \\
0 & \delta^{11} & 0 \\
\delta^4 & \delta^7 & 1
\end{bmatrix},
\]

\[
\begin{bmatrix}
\delta^{10} & 0 & 0 \\
0 & \delta^3 & 0 \\
\delta & \delta^{11} & 1
\end{bmatrix}, \quad \begin{bmatrix}
\delta & 0 & 0 \\
\delta^{12} & \delta^2 & \delta^5 \\
\delta^{14} & \delta^{10} & 1
\end{bmatrix}, \quad \begin{bmatrix}
\delta^5 & 0 & 0 \\
\delta^5 & \delta^3 & \delta^6 \\
\delta^5 & \delta^{14} & 1
\end{bmatrix}, \quad \begin{bmatrix}
\delta^{12} & \delta^2 \\
\delta^4 & \delta^3 & \delta^7 \\
\delta^6 & \delta^3 & 1
\end{bmatrix}, \quad \begin{bmatrix}
\delta^{11} & \delta^6 & \delta^{10} \\
\delta^5 & \delta^3 & \delta^6 \\
\delta^{10} & \delta^6 & 1
\end{bmatrix}
\]
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 -n 2 -field F -v 0 -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
```

Cubic surfaces are associated with arcs of size 6 (in a many-to-one relationship when consid-
ering 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 as-
sociated 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:
```

209
\texttt{$(ORBITER) \ -v \ 4$ \ \\
$\texttt{-orbiter\_path $(ORBITER\_PATH)$}$ \\
$\texttt{-define \textit{F -finite\_field} \ -q \ 32 \ -end}$ \\
$\texttt{-define \textit{P -projective\_space -n} \ 2 \ \textit{-field F -v} \ 0 \ -end}$ \\
$\texttt{-with \textit{P -do}}$ \\
$\texttt{-projective\_space\_activity}$ \\
$\texttt{-classify \textit{arcs}}$ \\
$\texttt{-poset\_classification\_control}$ \\
$\texttt{-problem\_label nc\_arcs\_q32\_d2}$ \\
$\texttt{-W \ -depth \ 6}$ \\
$\texttt{-draw\_poset \ -draw\_options \ -end}$ \\
$\texttt{-report \ -end}$ \\
$\texttt{-end}$ \\
$\texttt{-target\_size \ 6}$ \\
$\texttt{-test\_nb\_Eckardt\_points \ 13}$ \\
$\texttt{-d \ 2}$ \\
$\texttt{-conic\_test}$ \\
$\texttt{-end}$ \\
$\texttt{-end}$ \\
$\texttt{pdflatex nc\_arcs\_q32\_d2\_poset.tex}$ \\
$\texttt{open nc\_arcs\_q32\_d2\_poset.pdf}$
6.7 Cubic Curves

Orbiter can classify cubic curves in $\text{PG}(2,q)$. To this end, the $(9,3)$-arcs in $\text{PG}(2,q)$ are classified first. From this classification, the classification of curves is computed. This classification only works for arcs which contain a $(9,3)$ arc. For very small fields, this is not always the case.

Here is an example. The command

cubic_curves_PG_2.8:

```bash
$\textit{(ORBITER)} -v 3 -define G \\
  -define F -finite_field -q 8 -end \\
  -define P -projective_space -n 2 -field F -v 0 -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 \\
  -recognize "0,1,2,3,35,28" \\
  -draw_options -radius 200 -embedded -end \\
  -recognize "1,2,3,51,28,61,46,71,40" \\
  -draw_poset \\
  -Kramer_Mesner_matrix 6 9 \\
  -end \\
$\textit{(ORBITER)} -v 2 -draw_matrix \\
  -input_csv_file cc_8_KM_6_9.csv \\
  -box_width 50 -bit_depth 8 -end
```

draws the diagram.

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

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

Cubic Surfaces

7.1 Creation

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

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

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

Table 7.4 lists activities for a cubic surface object.

Let us look at some examples. The next command creates the unique surface with 27 lines over the field $\mathbb{F}_4$, 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 -n 3 -field F -v 0 -end 
▷ ▷ -define S -cubic_surface -space P -catalogue 0 -end 
▷ ▷ -with S -do 
▷ ▷ -cubic_surface_activity 
▷ ▷ ▷ -report 
▷ ▷ ▷ -report_with_group 
▷ ▷ ▷ -end 
▷ ▷ -with S -do 
▷ ▷ -cubic_surface_activity 
▷ ▷ ▷ -export something "points" 
```
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-space</td>
<td>P</td>
<td>Specify the 3-dimensional projective space $P = \mathbb{P}(3, q)$.</td>
</tr>
<tr>
<td>-label_txt</td>
<td>label</td>
<td>Override the ascii label of the curve.</td>
</tr>
<tr>
<td>-label_tex</td>
<td>label</td>
<td>Override the latex label of the curve.</td>
</tr>
<tr>
<td>-label_for_summary</td>
<td>label</td>
<td>Override the ascii label of the curve, to be used in summary commands.</td>
</tr>
<tr>
<td>-catalogue</td>
<td>$i$</td>
<td>Create the $i$-th surface in the Orbiter catalogue. Here, $i$ is an index variable used to index all surfaces in $\mathbb{P}(3, q)$. The index $i$ is zero-based. The automorphism group is created as well.</td>
</tr>
<tr>
<td>-by_coefficients</td>
<td>list-of-coeff-pairs</td>
<td>Create a surface from a list of coefficient-monomial pairs. The automorphism group is not created.</td>
</tr>
<tr>
<td>-by_rank</td>
<td>rank $q_0$</td>
<td>Create a surface from the numerical rank of the equation over the subfield $\mathbb{F}_{q_0}$. Here, we think of the equation as a point in $\mathbb{P}(19, q_0)$ and use the Orbiter point rank.</td>
</tr>
<tr>
<td>-family_Eckardt</td>
<td>$a$ $b$</td>
<td>Create the Eckardt surface with parameters $(a, b)$ as in see [12] (where it is called the Hilbert, Cohn-Vossen surface). The equation is $X_3^3 - b^2(X_0^2 + X_1^2 + X_2^2)X_3 + \frac{b^3}{a}(a^2 + 1)X_0X_1X_2 = 0$. The automorphism group is created as well.</td>
</tr>
</tbody>
</table>

Table 7.1: Commands to create a cubic surface (Part 1)
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-family_G13</td>
<td>a</td>
<td>Create a member of the $G_{13}$ family with parameter $a$. The surface has 13 Eckardt points.</td>
</tr>
<tr>
<td>-family_F13</td>
<td>a</td>
<td>Create a member of the $F_{13}$ family with parameter $a$. The surface has 13 Eckardt points.</td>
</tr>
<tr>
<td>-family_bes</td>
<td>a c</td>
<td>Create a member of the “bes”-family with parameter $a$. The surface has 5 Eckardt points.</td>
</tr>
<tr>
<td>-family_general_abcd</td>
<td>a b c d</td>
<td>Create a member of the general family with parameters $a, b, c, d$.</td>
</tr>
<tr>
<td>-arc_lifting</td>
<td>A</td>
<td>Create the surface associated with the arc $A = a_1, \ldots, a_6$ in 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>
<tr>
<td>-arc_lifting_with_two_lines</td>
<td>A L</td>
<td>Create the surface associated with the arc $A = a_1, \ldots, a_6$ in PG(2, q) by means of the Clebsch map, defined by the lines $\ell_1$ and $\ell_2$ whose ranks are given in $L$. If both of the lines are given as 0, the program will pick a suitable set of lines automatically.</td>
</tr>
</tbody>
</table>

Table 7.2: Commands to create a cubic surface (Part 2)
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-Cayley_form</td>
<td>k l m n</td>
<td>Create the surface from Cayley’s normal form, using the parameters k, l, m, n.</td>
</tr>
<tr>
<td>-by_equation</td>
<td></td>
<td>Create the surface from an equation.</td>
</tr>
<tr>
<td>-by_double_six</td>
<td>D</td>
<td>Create the surface from a double six D.</td>
</tr>
<tr>
<td>-by_skew_hexagon</td>
<td>label label</td>
<td>Create the surface from a skew hexagon.</td>
</tr>
<tr>
<td>-select_double_six</td>
<td>L</td>
<td>Relabel the lines by choosing the 12 lines in L as new double six. The entries in L are line indices with respect to the old double six. They are integers in the interval [0, 26]. This command can be repeated. In each application, the double six refers to the previous labeling.</td>
</tr>
<tr>
<td>-override_group</td>
<td>descr</td>
<td>Override the automorphism group of the surface by the given group.</td>
</tr>
<tr>
<td>-transform</td>
<td>elt</td>
<td>Apply the transformation given by the group element.</td>
</tr>
<tr>
<td>-transform_inverse</td>
<td>elt</td>
<td>Apply the inverse transformation given by the group element.</td>
</tr>
</tbody>
</table>

Table 7.3: Commands to create a cubic surface (Part 3)
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-report</code></td>
<td></td>
<td>Produce a latex report about the cubic surface.</td>
</tr>
<tr>
<td><code>-report_with_group</code></td>
<td></td>
<td>Produce a latex report about the cubic surface. The report include group theoretic information about the automorphism group and the action on the surface.</td>
</tr>
<tr>
<td><code>-export_points</code></td>
<td></td>
<td>Writes the set of points on the surface to a file.</td>
</tr>
<tr>
<td><code>-all_quartic_curves</code></td>
<td></td>
<td>Creates all quartic curves with 28 bitangents from the surface by projecting along the tangent cone of a point not on any line. Creates latex files.</td>
</tr>
<tr>
<td><code>-export_all_quartic_curves</code></td>
<td></td>
<td>Creates all quartic curves with 28 bitangents from the surface by projecting along the tangent cone of a point not on any line. Writes a csv file with the curves that have been created.</td>
</tr>
<tr>
<td><code>-export_tritangent_planes</code></td>
<td></td>
<td>Writes a file containing all tritangent planes of the surface.</td>
</tr>
</tbody>
</table>

Table 7.4: Activities related to cubic surfaces
Two reports are created, one with information about the group and the other without it.

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

eckardt_13_4_12:

```
$ (ORBITER) -v 6 \
  -define F -finite_field -q 13 -end \
  -define P -projective_space -n 3 -field F -v 0 -end \
  -define Eckardt_4_12 -cubic_surface -space P -family Eckardt 4 12 -end \
  -with Eckardt_4_12 -do  \
    -cubic_surface_activity  \
    -report  \
    -report_with_group  \
  -end
```

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

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

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

218
\[(1+1) \ast a \ast b \ast c \ast d - a \ast a \ast b \ast d \ast d - (1+1) \ast a \ast a \ast c \ast d \ast d \\
- (1+1) \ast a \ast b \ast b \ast c \ast c + a \ast b \ast b \ast c \ast d + (1+1) \ast a \ast b \ast c \ast c \ast d + a \ast b \ast c \ast d \ast d \\
- b \ast b \ast c \ast c \ast d - a \ast a \ast b \ast c + a \ast a \ast c \ast d + a \ast a \ast d \ast d + a \ast b \ast b \ast c + a \ast b \ast c \ast c \\
- (1+1+1+1) \ast a \ast b \ast c \ast d - a \ast c \ast c \ast d + a \ast c \ast d \ast d + b \ast b \ast c \ast c \ast X1 \ast X2 \ast X3 \\
+ c \ast a \ast (a \ast d - b \ast c - a + b + c - d) \ast (b - d) \ast X1 \ast X3 \ast X3\]

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_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 -n 3 -field F -v 0 -end \\
  \> -define F_2345 -cubic_surface -space P \\
  \> \> -by_equation "F.alpha_beta_gamma_delta" \\
  \> \> "\DF{\alpha,\beta,\gamma,\delta}\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" \\
  \> \> -override_group 6 2 \\
  \> \> "1,5,0,0,3,6,0,0,0,1,1,3,0,5,5,0,3, \\
  \> \> 1,0,2,5,0,1,6,1,0,0,3,5,0,0,4,4" \\
  \> \> -end \\
  \> -with F_2345 -do \\
  \> -cubic_surface_activity \\
  \> \> -report \\
  \> \> -report_with_group \\
  \> \> -end
```

\[219\]
pdflatex surface_equation_F_alpha_beta_gamma_delta_q7_with_group.tex
open surface_equation_F_alpha_beta_gamma_delta_q7_with_group.pdf
Table 7.5: Options to create a quartic curve

### 7.2 Quartic Curves

Cubic surfaces with 27 lines are associated with quartic curves with 28 bitangents (see [32]), which in turn are associated with del Pezzo surfaces. Orbiter can classify quartic curves based on a known classification of cubic surfaces. Orbiter also has a catalogue of quartic curves for small field sizes.

Table 7.5 lists options to create a quartic curve object.

Table 7.6 lists activities for a quartic curve object.

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

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

```plaintext
quartic_curve_9_0_report:
  ▷ $(ORBITER) -v 3 \n  ▷ ▷ -define F -finite_field -q 9 -end \n  ▷ ▷ -define P -projective_space -n 2 -field F -v 0 -end \n  ▷ ▷ -define C -quartic_curve -space P -catalogue 0 -end \n```
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-report</td>
<td></td>
<td>Produce a latex report about the curve.</td>
</tr>
<tr>
<td>-report_with_group</td>
<td></td>
<td>Produce a latex report about the curve. The report include group theoretic information about the automorphism group and the action on the curve.</td>
</tr>
<tr>
<td>-create_surface</td>
<td></td>
<td>Create a cubic surface from the curve.</td>
</tr>
<tr>
<td>-extract_orbit_on_bitangents_by_length</td>
<td>l</td>
<td>Extract the bitangents in the unique orbit of length $l$. If there is no orbit of length $l$, or if there are multiple orbits of length $l$, an error is raised.</td>
</tr>
</tbody>
</table>

Table 7.6: Activities related to quartic curves

```plaintext
▷▷ -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:

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

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

**The gradient**

The gradient of the quartic curve is:

$$\alpha^7 X_1^3 + \alpha^2 X_2^3$$

$$(0, 4, 7, 0, 0, 0, 0, 0, 0)$$

$$\alpha^3 X_0^3 + X_2^3$$

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

222
\[ \alpha^4 X_0^3 + \alpha^6 X_1^3 \]
\[(2, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)\]

**General information**

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

**All points on the curve**

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

0 : \(P_0 = (1, 0, 0)\)  
1 : \(P_1 = (0, 1, 0)\)  
2 : \(P_2 = (0, 0, 1)\)  
3 : \(P_3 = (1, 1, 1)\)  
4 : \(P_4 = (1, 1, 0)\)  
5 : \(P_5 = (2, 1, 0)\)  
6 : \(P_6 = (3, 0, 1)\)  
7 : \(P_7 = (6, 0, 1)\)  
8 : \(P_8 = (5, 1, 1)\)  
9 : \(P_9 = (6, 1, 1)\)  
10 : \(P_{10} = (2, 2, 1)\)  
20 : \(P_{20} = (3, 5, 1)\)  
11 : \(P_{11} = (4, 2, 1)\)  
21 : \(P_{21} = (7, 5, 1)\)  
12 : \(P_{12} = (6, 2, 1)\)  
22 : \(P_{22} = (3, 7, 1)\)  
13 : \(P_{13} = (1, 3, 1)\)  
23 : \(P_{23} = (4, 7, 1)\)  
14 : \(P_{14} = (4, 3, 1)\)  
24 : \(P_{24} = (5, 7, 1)\)  
15 : \(P_{15} = (7, 3, 1)\)  
25 : \(P_{25} = (0, 8, 1)\)  
16 : \(P_{16} = (0, 4, 1)\)  
26 : \(P_{26} = (1, 8, 1)\)  
17 : \(P_{17} = (5, 4, 1)\)  
27 : \(P_{27} = (2, 8, 1)\)  
18 : \(P_{18} = (7, 4, 1)\)  
19 : \(P_{19} = (2, 5, 1)\)  

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

The Kovalevski points are:

0 : \(P_7 = (4, 1, 0) = c_{13} \cap c_{14} \cap c_{36} \cap c_{46}\)  
1 : \(P_8 = (5, 1, 0) = a_2 \cap a_4 \cap c_{25} \cap c_{45}\)  
2 : \(P_9 = (6, 1, 0) = b_1 \cap b_6 \cap c_{12} \cap c_{26}\)  
3 : \(P_{10} = (7, 1, 0) = a_3 \cap b_5 \cap c_{35} \cap d\)  
4 : \(P_{11} = (8, 1, 0) = b_2 \cap b_3 \cap c_{24} \cap c_{34}\)  
5 : \(P_{12} = (1, 0, 1) = a_3 \cap a_4 \cap c_{23} \cap c_{24}\)  
6 : \(P_{13} = (2, 0, 1) = c_{34} \cap c_{36} \cap c_{45} \cap c_{56}\)
7: $P_{15} = (4, 0, 1) = b_3 \cap b_6 \cap c_{13} \cap c_{16}$
8: $P_{16} = (5, 0, 1) = a_5 \cap a_6 \cap c_{25} \cap c_{26}$
9: $P_{18} = (7, 0, 1) = a_2 \cap b_1 \cap c_{35} \cap c_{46}$
10: $P_{19} = (8, 0, 1) = b_4 \cap b_5 \cap c_{14} \cap c_{15}$
11: $P_{20} = (0, 1, 1) = a_2 \cap b_3 \cap c_{14} \cap c_{56}$
12: $P_{21} = (2, 1, 1) = b_2 \cap b_4 \cap c_{26} \cap c_{46}$
13: $P_{22} = (3, 1, 1) = a_4 \cap b_5 \cap c_{12} \cap c_{36}$
14: $P_{23} = (4, 1, 1) = a_6 \cap b_1 \cap c_{23} \cap c_{45}$
15: $P_{26} = (7, 1, 1) = c_{16} \cap c_{25} \cap c_{34} \cap d$
16: $P_{27} = (8, 1, 1) = a_3 \cap a_5 \cap c_{13} \cap c_{15}$
17: $P_{28} = (0, 2, 1) = c_{12} \cap c_{13} \cap c_{25} \cap c_{35}$
18: $P_{29} = (1, 2, 1) = b_1 \cap b_5 \cap c_{16} \cap c_{56}$
19: $P_{31} = (3, 2, 1) = a_3 \cap a_6 \cap c_{34} \cap c_{46}$
20: $P_{33} = (5, 2, 1) = a_2 \cap b_4 \cap c_{24} \cap d$
21: $P_{35} = (7, 2, 1) = b_2 \cap b_6 \cap c_{23} \cap c_{36}$
22: $P_{36} = (8, 2, 1) = a_4 \cap b_3 \cap c_{15} \cap c_{26}$
23: $P_{37} = (0, 3, 1) = a_5 \cap b_1 \cap c_{24} \cap c_{36}$
24: $P_{39} = (2, 3, 1) = a_2 \cap a_6 \cap c_{12} \cap c_{16}$
25: $P_{40} = (3, 3, 1) = b_3 \cap b_4 \cap c_{35} \cap c_{45}$
26: $P_{42} = (5, 3, 1) = a_4 \cap b_2 \cap c_{13} \cap c_{56}$
27: $P_{43} = (6, 3, 1) = a_3 \cap b_6 \cap c_{14} \cap c_{25}$
28: $P_{45} = (8, 3, 1) = c_{15} \cap c_{23} \cap c_{46} \cap d$
29: $P_{47} = (1, 4, 1) = a_6 \cap b_2 \cap c_{14} \cap c_{35}$
30: $P_{48} = (2, 4, 1) = b_3 \cap b_5 \cap c_{23} \cap c_{25}$
31: $P_{49} = (3, 4, 1) = a_5 \cap b_6 \cap c_{56} \cap d$
32: $P_{50} = (4, 4, 1) = a_2 \cap a_3 \cap c_{26} \cap c_{36}$
33: $P_{52} = (6, 4, 1) = b_1 \cap b_4 \cap c_{13} \cap c_{34}$
34: $P_{54} = (8, 4, 1) = c_{12} \cap c_{15} \cap c_{24} \cap c_{45}$
35: $P_{55} = (0, 5, 1) = a_4 \cap a_6 \cap b_3 \cap b_6$
36: $P_{56} = (1, 5, 1) = c_{13} \cap c_{26} \cap c_{45} \cap d$
37: $P_{59} = (4, 5, 1) = c_{24} \cap c_{25} \cap c_{46} \cap c_{56}$
38: $P_{60} = (5, 5, 1) = c_{12} \cap c_{14} \cap c_{23} \cap c_{34}$
39: $P_{61} = (6, 5, 1) = a_2 \cap a_5 \cap b_2 \cap b_5$
40: $P_{63} = (8, 5, 1) = c_{15} \cap c_{16} \cap c_{35} \cap c_{36}$
41: $P_{64} = (0, 6, 1) = a_1 \cap b_5 \cap c_{26} \cap c_{34}$
42: $P_{65} = (1, 6, 1) = a_1 \cap b_4 \cap c_{25} \cap c_{36}$
43: $P_{66} = (2, 6, 1) = a_1 \cap b_6 \cap c_{24} \cap c_{35}$
44: $P_{67} = (3, 6, 1) = a_1 \cap a_2 \cap c_{13} \cap c_{23}$
45: $P_{68} = (4, 6, 1) = a_1 \cap b_2 \cap c_{12} \cap d$
46: $P_{69} = (5, 6, 1) = a_1 \cap a_3 \cap b_1 \cap b_3$
47: $P_{70} = (6, 6, 1) = a_1 \cap a_4 \cap c_{16} \cap c_{46}$
48: $P_{71} = (7, 6, 1) = a_1 \cap a_5 \cap c_{14} \cap c_{45}$
49: $P_{72} = (8, 6, 1) = a_1 \cap a_6 \cap c_{15} \cap c_{56}$
50: $P_{73} = (0, 7, 1) = a_3 \cap b_2 \cap c_{16} \cap c_{45}$
The points off the curve are:

$$P_{37} = (5, 8, 1) = b_5 \cap b_6 \cap c_{45} \cap c_{46}$$

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

The points off the curve are:

$$P_6 = (3, 1, 0)$$

$$P_7 = (4, 1, 0)$$

$$P_8 = (5, 1, 0)$$

$$P_9 = (6, 1, 0)$$

$$P_{10} = (7, 1, 0)$$

$$P_{11} = (8, 1, 0)$$

$$P_{12} = (1, 0, 1)$$

$$P_{13} = (2, 0, 1)$$

$$P_{15} = (4, 0, 1)$$

$$P_{16} = (5, 0, 1)$$

$$P_{18} = (7, 0, 1)$$

$$P_{19} = (8, 0, 1)$$

$$P_{20} = (0, 1, 1)$$

$$P_{21} = (2, 1, 1)$$

$$P_{22} = (3, 1, 1)$$

$$P_{23} = (4, 1, 1)$$

$$P_{26} = (7, 1, 1)$$

$$P_{27} = (8, 1, 1)$$

$$P_{28} = (0, 2, 1)$$

$$P_{29} = (1, 2, 1)$$

$$P_{31} = (3, 2, 1)$$

$$P_{33} = (5, 2, 1)$$
The lines and their points of contact are:

\[
\begin{align*}
\mathbf{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) 4 \times \\
\mathbf{a}_2 &= \begin{bmatrix} 1 & 0 & \alpha^6 \\ 0 & 1 & 1 \end{bmatrix}^{51} = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 1 \end{bmatrix}^{51} \quad P_{83} = P(1, 8, 1) 4 \times \\
\mathbf{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) 4 \times \\
\mathbf{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) 4 \times \\
\mathbf{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) 4 \times \\
\mathbf{a}_6 &= \begin{bmatrix} 1 & 0 & \alpha^2 \\ 0 & 1 & \alpha^2 \end{bmatrix}^{77} = \begin{bmatrix} 1 & 0 & 7 \\ 0 & 1 & 7 \end{bmatrix}^{77} \quad P_{5} = P(2, 1, 0) 4 \times \\
\mathbf{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) 4 \times \\
\mathbf{b}_2 &= \begin{bmatrix} 1 & 0 & \alpha^7 \\ 0 & 1 & \alpha^6 \end{bmatrix}^{45} = \begin{bmatrix} 1 & 0 & 4 \\ 0 & 1 & 5 \end{bmatrix}^{45} \quad P_{14} = P(3, 0, 1) 4 \times \\
\mathbf{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) 4 \times \\
\mathbf{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) 4 \times \\
\mathbf{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) 4 \times \\
\mathbf{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) 4 \times \\
\mathbf{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) 4 \times \\
\mathbf{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) 4 \times \\
\mathbf{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) 4 \times \\
\mathbf{c}_{15} &= \begin{bmatrix} 1 & 0 & \alpha^5 \\ 0 & 1 & 0 \end{bmatrix}^{60} = \begin{bmatrix} 1 & 0 & 6 \\ 0 & 1 & 0 \end{bmatrix}^{60} \quad P_{1} = P(0, 1, 0) 4 \times
\end{align*}
\]
\[
c_{16} = \begin{bmatrix} 1 & 0 & \alpha^6 \\ 0 & 1 & \alpha \\ \end{bmatrix}^{35} = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 5 \\ \end{bmatrix}^{35} = P_{51} = \mathbf{P}(5,4,1) 4\times
\]
\[
c_{23} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & \alpha^5 \\ \end{bmatrix}^{16} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 6 \\ \end{bmatrix}^{16} = P_{82} = \mathbf{P}(0,8,1) 4\times
\]
\[
c_{24} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & \alpha^7 \\ \end{bmatrix}^{14} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 4 \\ \end{bmatrix}^{14} = P_{25} = \mathbf{P}(6,1,1) 4\times
\]
\[
c_{25} = \begin{bmatrix} 1 & 0 & \alpha^2 \\ 0 & 1 & \alpha^4 \\ \end{bmatrix}^{72} = \begin{bmatrix} 1 & 0 & 7 \\ 0 & 1 & 2 \\ \end{bmatrix}^{72} = P_{76} = \mathbf{P}(3,7,1) 4\times
\]
\[
c_{26} = \begin{bmatrix} 1 & 0 & \alpha^2 \\ 0 & 1 & \alpha^3 \\ \end{bmatrix}^{78} = \begin{bmatrix} 1 & 0 & 7 \\ 0 & 1 & 8 \\ \end{bmatrix}^{78} = P_{44} = \mathbf{P}(7,3,1) 4\times
\]
\[
c_{34} = \begin{bmatrix} 1 & 0 & \alpha^4 \\ 0 & 1 & \alpha^3 \\ \end{bmatrix}^{28} = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 8 \\ \end{bmatrix}^{28} = P_{38} = \mathbf{P}(1,3,1) 4\times
\]
\[
c_{35} = \begin{bmatrix} 1 & 0 & \alpha^6 \\ 0 & 1 & \alpha^4 \\ \end{bmatrix}^{52} = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 2 \\ \end{bmatrix}^{52} = P_{24} = \mathbf{P}(5,1,1) 4\times
\]
\[
c_{36} = \begin{bmatrix} 1 & 0 & \alpha^4 \\ 0 & 1 & \alpha^7 \\ \end{bmatrix}^{24} = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 4 \\ \end{bmatrix}^{24} = P_{78} = \mathbf{P}(5,7,1) 4\times
\]
\[
c_{45} = \begin{bmatrix} 1 & 0 & \alpha^4 \\ 0 & 1 & \alpha^6 \\ \end{bmatrix}^{25} = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 5 \\ \end{bmatrix}^{25} = P_{34} = \mathbf{P}(6,2,1) 4\times
\]
\[
c_{46} = \begin{bmatrix} 1 & 0 & \alpha^6 \\ 0 & 1 & \alpha \\ \end{bmatrix}^{53} = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 3 \\ \end{bmatrix}^{53} = P_{46} = \mathbf{P}(0,4,1) 4\times
\]
\[
c_{56} = \begin{bmatrix} 1 & 0 & \alpha^4 \\ 0 & 1 & 1 \\ \end{bmatrix}^{21} = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 1 \\ \end{bmatrix}^{21} = P_{4} = \mathbf{P}(1,1,0) 4\times
\]
\[
d = \begin{bmatrix} 1 & \alpha^6 & 0 \\ 0 & 0 & 1 \\ \end{bmatrix}^{59} = \begin{bmatrix} 1 & 5 & 0 \\ 0 & 0 & 1 \\ \end{bmatrix}^{59} = P_{2} = \mathbf{P}(0,0,1) 4\times
\]

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

Line type: $1^{28}$

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

227
Lines on points off the curve:

Off point 0 = $P_0 = (3, 1, 0)$ 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 }

point types for points off the curve: 4$^{63}$
Regarding the problem of classification, we first fix the field order $q$ for which we want to classify the quartic curves. Next, we observe that quartic curves with 28 bitangents are related to cubic surfaces with 27 lines over the same field. This means we can exploit the previously classified list of cubic surfaces towards the goal of classifying quartic curves. The Orbiter knowledge base contains the classification of cubic surfaces with 27 lines over $\mathbb{F}_q$ for small values of $q$. Because of this dependency, there is a restriction on the size of $q$ for which
this algorithm can be applied. Next, we consider the list of cubic surfaces with 27 lines over the field $\mathbb{F}_q$. For each surface, and for each orbit on points not on lines, we perform a projection operation to create one quartic curve. This guarantees that every isomorphism type of quartic curve with 28 bitangents will be created.

Let us look at some examples of this algorithm. We try $q = 7$ and $q = 13$. In each case, we need a makefile variable to set the number of (isomorphism types of) cubic surfaces with 27 lines over $\mathbb{F}_q$. For $q = 7$, there is exactly one isomorphism type, so we put

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

The next command creates a list of quartic curves using a projection construction. For each isomorphism type of cubic surface, and for each point not on any line, we consider the intersection of the tangent cone with the surface and project onto a plane not containing the point. Because of symmetry, it suffices to perform this projection only for a set of representatives of the orbits on points not on lines. Here is the Orbiter command for $q = 7$:

```
quartic_curves_q7:
  $(ORBITER\_PATH)orbiter.out -v 3 \n  -list\_arguments \n  -draw\_options -end \n  -define F -finite\_field -q 7 -end \n  -define P -projective\_space -n 3 -field F -end \n  -loop L 0 $(NB\_CUBIC\_SURFACES\_Q7) 1 \n  -define S\_%L -cubic\_surface -space P -catalogue %L -end \n  -end\_loop \n  -print\_symbols \n  -loop L 0 $(NB\_CUBIC\_SURFACES\_Q7) 1 \n  -define S\_%L -cubic\_surface -do \n  -cubic\_surface\_activity \n  -export\_all\_quartic\_curves \n  -end \n  -end\_loop \n  -print\_symbols
```

The resulting curves are written to file. Unfortunately, in this example, there is no point which does not lie on any line of the surface. This means that no quartic curve with 28 lines exists over $\mathbb{F}_7$.

We move on to the next example, which is $q = 13$. Again, we use a makefile variable to set the number of isomorphism types of cubic surfaces with 27 lines over $\mathbb{F}_{13}$. There are exactly 4 types:

\[ \text{NB\_CUBIC\_SURFACES\_Q13}=4 \]
Just like before, we create all quartic curves arising from projections:

```
quartic_curves_q13:
  > $(ORBITER_PATH)orbiter.out -v 3 \
  >   -list_arguments \ 
  >   -draw_options -end \ 
  >   -define F -finite_field -q 13 -end \ 
  >   -define P -projective_space -n 3 -field F -end \ 
  >   -loop L 0 $(NB_CUBIC_SURFACES_Q13) 1 \ 
  >     -define S_%L -cubic_surface -space P -catalogue %L -end \ 
  >   -end_loop \ 
  >   -print_symbols \ 
  >   -loop L 0 $(NB_CUBIC_SURFACES_Q13) 1 \ 
  >     -with S_%L -do \ 
  >     -cubic_surface_activity \ 
  >     -export_all_quartic_curves \ 
  >     -end \ 
  >   -end_loop \ 
  >   -print_symbols
```

We combine the output files into one:

```
quartic_curves_q13_combine:
  > $(ORBITER_PATH)orbiter.out -v 3 \
  >   -csv_file_concatenate_from_mask $(NB_CUBIC_SURFACES_Q13) \ 
  >   -surface_catalogue_q13_iso%ld_quartics.csv \ 
  >   quartics_q13.csv
```

The next command processes the curves that have been created and performs a classification up to isomorphism. The result is the classification of quartic curves with 28 bitangents over the field $\mathbb{F}_{13}$:

```
quartic_curves_q13_classify:
  > $(ORBITER) -v 3 \
  >   -list_arguments \ 
  >   -define F -finite_field -q 13 -end \ 
  >   -define P -projective_space -n 2 -field F -v 0 -end \ 
  >   -with P -do \ 
  >   -projective_space_activity \ 
  >   -classify_quartic_curves_with_substructure \ 
  >   quartics_q13.csv \ 
  >   1 4 4 quartic_curves_q13 \ 
  >   -end \ 
  >   -print_symbols
```
We find exactly two isomorphism types. The data is exported to C++ source code. The file `quartic_curves_q13.cpp` is created. This file is now part of Orbiter’s knowledge base of geometric objects.
7.3 Classification

There are several different approaches to classify cubic surfaces with 27 lines over finite fields $F_q$ in Orbiter. Classification means to determine the non-equivalent surfaces under the action of the collineation group $P\Gamma L(4,q)$ of $PG(3,q)$. The approach described in [12] relies on Schlaeffi’s notion of a double six as a substructure [61]. The approach described in [37] utilizes the relation to non-conical six-arcs in a plane. A third approach is described in [38]. All three approaches are available in Orbiter.

In PG(3,4), there is only one type of cubic surfaces with 27 lines. It is a member of the Hirschfeld family, described in [33]. The following Orbiter command can be used to construct this surface and to prove its uniqueness for $F_4$. The following command utilizes the algorithm of [12] to do so:

```
surface classify_q4:
  $ (ORBITER) -v 5 \ 
  -define F -finite_field -q 4 -end \ 
  -define P -projective_space -n 3 -field F -v 0 -end \ 
  -with P -do \ 
  -projective_space_activity \ 
  -classify_surfaces_with_double_sixes Surf27 -W -end \ 
  -end \ 
  -with Surf27 -do \ 
  -classification_of_cubic_surfaces_with_double_sixes_activity \ 
  -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 $P\Gamma L(4,4)$ of degree 85
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 orthogonal group**

**The Action**

Group action PGL(4,4)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 PGL(4,4)OnWedges100 of degree 100
\[ \vdots \]
Strong generators for a group of order 5529600:

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

Poset classification up to depth 5
The Orbits

Number of Orbits By Level

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

Summary of Orbit Representatives

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

Table 7.7: 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>4</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>1</td>
<td>{ 0, 3, 56, 76, 97 }</td>
<td>(96, 57600)</td>
<td>4</td>
<td>5</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>4</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>3</td>
<td>{0, 3, 56, 80, 93}</td>
<td>(120, 46080)</td>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Poset of Orbits in Detail

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

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

Of these, 1 impose 19 conditions.
Of these, 1 are associated with double sixes. They are:
0/1 is orbit 3/4 $\{0, 3, 56, 80, 93\}_{120}$ orbit length 46080
The overall number of five plus one configurations associated with double sixes in $\text{PG}(3,4)$ is: 46080

Flag orbits for double sixes

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

(1) Flag orbit 0 / 1 down=$\{3, 0\}$ up=$\{0, -1\}$ is $\{0, 3, 56, 80, 93, 16, 340, 38, 61, 156, 0, 16, 340, 38, 61, 156, 165, 72, 54, 25, 356, 0\}$ with a stabilizer of order 120

Strong generators for a group of order 120:

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

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

The orbits are:

(1) 0/1 \{16, 340, 38, 61, 156, 165, 155, 72, 54, 25, 356, 0\}_{1440} orbit length 1370880

Strong generators for a group of order 1440:

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

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

nb received = 0
The overall number of objects is: 1370880

Flag orbits for surfaces

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

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

Strong generators for a group of order 1440:

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

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:

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

The overall number of objects is: 38080

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

The order of the group is 1974067200

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

The order of the group is 1974067200

The group has 1 orbits:

The orbits are:
(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} \text{ orbit length 38080}

Strong generators for a group of order 51840:

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

The overall number of objects is: 38080

**Surface 4#0**

**The equation**

The equation of the surface is:

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

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

Number of points on the surface 45

The automorphism group of the surface has order 51840
The automorphism group is the following group
Strong generators for a group of order 51840:

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

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

General information
Points on lines: 5^{27}
Lines on points: 3^{45}

The 27 Lines
\[
\ell_0 = a_1 = \begin{bmatrix} 1 & 0 & \omega^2 & 0 \\ 0 & 1 & 1 & \omega \end{bmatrix}_{72} = \begin{bmatrix} 1 & 0 & 3 & 0 \\ 0 & 1 & 1 & 2 \end{bmatrix}_{72} = \text{Pl}(3,2,3,0,3,1)_{308}
\]
\[
\ell_1 = a_2 = \begin{bmatrix} 1 & 0 & \omega & 0 \\ 0 & 1 & 0 & \omega^2 \end{bmatrix}_{54} = \begin{bmatrix} 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 3 \end{bmatrix}_{54} = \text{Pl}(2,3,0,0,2,1)_{238}
\]
\ell_2 = a_3 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}_{25} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}_{25} = \text{Pl}(1, 1, 0, 0, 1, 1)_{177}

\ell_3 = a_4 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}_{356} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}_{356} = \text{Pl}(0, 1, 0, 0, 0, 0)_{1}

\ell_4 = a_5 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}_0 = \begin{bmatrix} 1 & 0 & 1 & 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}
\[ \ell_{16} = c_{16} = \begin{bmatrix} 1 & 0 & \omega^2 & 0 \\ 0 & 1 & 0 & \omega \end{bmatrix}_{71} = \begin{bmatrix} 1 & 0 & 3 & 0 \\ 0 & 1 & 0 & 2 \end{bmatrix}_{71} = \text{Pl}(3, 2, 0, 0, 3, 1)_{299} \]

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

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

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

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

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

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

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

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

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

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

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

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

All Points on surface

The surface has 45 points
Eckardt Points

The surface has 45 Eckardt points:

0 : $E_{56} = a_5 \cap b_6 \cap c_{56} = P_0 = P_0 = P(1,0,0,0) = P(1,0,0,0), \ T = 0$
1 : $E_{51} = a_5 \cap b_1 \cap c_{15} = P_1 = P_1 = P(0,1,0,0) = P(0,1,0,0), \ T = 4$
2 : $E_{46} = a_4 \cap b_6 \cap c_{46} = P_2 = P_2 = P(0,0,1,0) = P(0,0,1,0), \ T = 20$
3 : $E_{41} = a_4 \cap b_1 \cap c_{14} = P_3 = P_3 = P(0,0,0,1) = P(0,0,0,1), \ T = 84$
4 : $E_{32} = a_3 \cap b_2 \cap c_{23} = P_4 = P_4 = P(1,1,1,1) = P(1,1,1,1), \ T = 27$
5 : $E_{35} = a_5 \cap b_2 \cap c_{25} = P_5 = P_5 = P(1,1,0,0) = P(1,1,0,0), \ T = 1$
6 : $E_{54} = a_5 \cap b_4 \cap c_{45} = P_6 = P_6 = P(\omega,1,0,0) = P(2,1,0,0), \ T = 2$
7 : $E_{53} = a_5 \cap b_3 \cap c_{35} = P_7 = P_7 = P(\omega^2,1,0,0) = P(3,1,0,0), \ T = 3$
8 : $E_{36} = a_3 \cap b_6 \cap c_{36} = P_8 = P_8 = P(0,1,0,0) = P(0,1,0,0), \ T = 5$
9 : $E_{16} = a_1 \cap b_6 \cap c_{16} = P_9 = P_9 = P(\omega,0,1,0) = P(2,0,1,0), \ T = 10$
10 : $E_{26} = a_2 \cap b_6 \cap c_{26} = P_{10} = P_{10} = P(\omega^2,0,1,0) = P(3,0,1,0), \ T = 15$
11 : $E_{14,23,56} = c_{14} \cap c_{23} \cap c_{56} = P_{11} = P_{11} = P(0,1,1,0) = P(0,1,1,0), \ T = 9$
12 : $E_{13,24,56} = c_{13} \cap c_{24} \cap c_{56} = P_{12} = P_{12} = P(1,1,1,0) = P(1,1,1,0), \ T = 6$
13 : $E_{65} = a_6 \cap b_5 \cap c_{56} = P_{13} = P_{13} = P(\omega,1,1,0) = P(2,1,1,0), \ T = 12$
14 : $E_{12,34,56} = c_{12} \cap c_{34} \cap c_{56} = P_{14} = P_{14} = P(\omega^2,1,1,0) = P(3,1,1,0), \ T = 18$
15 : $E_{15,23,46} = c_{15} \cap c_{23} \cap c_{46} = P_{15} = P_{15} = P(\omega^2,0,1,0) = P(1,0,0,1), \ T = 21$
16 : $E_{43} = a_3 \cap b_5 \cap c_{13} = P_{16} = P_{26} = P(0,1,0,1) = P(0,1,0,1), \ T = 25$
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 : $E_{15,16,23,34} = a_6 \cap c_{21} \cap c_{16} = P_{19} = P_{31} = P(1,\omega,0,1) = P(1,2,0,1), \ T = 24$
20 : $E_{61} = a_6 \cap b_1 \cap c_{16} = P_{20} = P_{34} = P(0,\omega^2,0,1) = P(0,3,0,1), \ T = 67$
21 : $E_{15} = a_1 \cap b_5 \cap c_{15} = P_{21} = P_{35} = P(1,\omega,0,1) = P(1,3,0,1), \ T = 23$
22 : $E_{42} = a_4 \cap b_5 \cap c_{24} = P_{22} = P_{38} = P(0,0,1,1) = P(0,0,1,1), \ T = 41$
23 : $E_{13,25,46} = c_{13} \cap c_{25} \cap c_{46} = P_{23} = P_{39} = P(1,0,1,1) = P(1,0,1,1), \ T = 26$
24 : $E_{14,25,36} = c_{14} \cap c_{25} \cap c_{36} = P_{24} = P_{42} = P(0,1,1,1) = P(0,1,1,1), \ T = 30$
25 : $E_{62} = a_6 \cap b_2 \cap c_{26} = P_{25} = P_{47} = P(\omega,\omega,1,1) = P(2,2,1,1), \ T = 53$
26 : $E_{25} = a_2 \cap b_5 \cap c_{25} = P_{26} = P_{48} = P(\omega^2,\omega,1,1) = P(3,2,1,1), \ T = 80$
27 : $E_{16,25,34} = c_{16} \cap c_{25} \cap c_{34} = P_{27} = P_{51} = P(\omega,\omega^2,1,1) = P(2,3,1,1), \ T = 55$
28 : $E_{12} = a_1 \cap b_2 \cap c_{12} = P_{28} = P_{32} = P(\omega^2,\omega^2,1,1) = P(3,3,1,1), \ T = 79$
29 : $E_{43} = a_4 \cap b_3 \cap c_{34} = P_{29} = P_{53} = P(0,0,0,1) = P(0,0,2,1), \ T = 62$
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 : $E_{35} = a_3 \cap b_5 \cap c_{35} = P_{31} = P_{39} = P(\omega,\omega,1,1) = P(2,1,2,1), \ T = 49$
32 : $E_{63} = a_6 \cap b_3 \cap c_{36} = P_{32} = P_{60} = P(\omega^2,\omega,1,1) = P(3,1,2,1), \ T = 76$
33 : $E_{14,26,35} = c_{14} \cap c_{26} \cap c_{35} = P_{33} = P_{61} = P(0,\omega,\omega,1) = P(0,2,2,1), \ T = 51$
34 : $E_{23} = a_2 \cap b_3 \cap c_{23} = P_{34} = P_{62} = P(1,\omega,\omega,1) = P(1,2,2,1), \ T = 39$
35 : $E_{13} = a_1 \cap b_3 \cap c_{13} = P_{35} = P_{67} = P(\omega,\omega,\omega,1) = P(2,3,2,1), \ T = 50$
36 : $E_{16,24,35} = c_{16} \cap c_{24} \cap c_{35} = P_{36} = P_{68} = P(\omega^2,\omega^2,\omega,1) = P(3,3,2,1), \ T = 74$
37 : $E_{45} = a_4 \cap b_5 \cap c_{45} = P_{37} = P_{69} = P(0,0,\omega^2,1) = P(0,0,3,1), \ T = 83$
38 : $E_{64} = a_6 \cap b_4 \cap c_{46} = P_{38} = P_{70} = P(1,0,\omega^2,1) = P(1,0,3,1), \ T = 31$
39 : $E_{12,36,45} = c_{12} \cap c_{36} \cap c_{45} = P_{39} = P_{75} = P(\omega,\omega,\omega^2,1) = P(2,1,3,1), \ T = 59$
40 : $E_{34} = a_3 \cap b_4 \cap c_{34} = P_{40} = P_{76} = P(\omega^2,\omega,\omega^2,1) = P(3,1,3,1), \ T = 71$
41 : \( E_{24} = a_2 \cap b_4 \cap c_{24} = P_{41} = P_{79} = P(\omega, \omega, \omega^2, 1) = P(2, 2, 3, 1), T = 58 \)
42 : \( E_{13,26,45} = c_3 \cap c_{26} \cap c_{45} = P_{42} = P_{80} = P(\omega^2, \omega, \omega^2, 1) = P(3, 2, 3, 1), T = 70 \)
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: \( (0, 4, 20, 24, 27, 1, 2, 3, 5, 10, 15, 9, 6, 12, 18, 21, 25, 22, 46, 24, 67, 23, 41, 26, 30, 53, 80, 55, 79, 62, 36, 49, 76, 51, 39, 50, 74, 83, 31, 59, 71, 58, 70, 72, 33 ) \)
Line type of Eckardt points: \( 5^{27}, 3^{240}, 1^{90} \)
Plane type of Eckardt points: \( 1^{45}, 9^{40} \)

**Hesse planes**

Number of Hesse planes: 40
Set of Hesse planes: \( (7, 8, 11, 13, 14, 16, 17, 19, 28, 29, 32, 34, 35, 37, 38, 40, 42, 43, 44, 45, 47, 48, 52, 54, 56, 57, 60, 61, 63, 64, 65, 66, 68, 69, 73, 75, 77, 78, 81, 82 ) \)
subspace 0 / 40 is 7:

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

subspace 39 / 40 is 82:

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

0 : 7 : \( E_{56}, E_{31}, E_{15,24,36}, E_{16,25,34}, E_{12}, E_{14,26,35}, E_{23}, E_{45}, E_{64} \)

39 : 82 : \( E_{41}, E_{52}, E_{16}, E_{12,34,56}, E_{15,24,36}, E_{35}, E_{23}, E_{64}, E_{13,26,45} \)

**Axes**

Number of axes: 240
Axes:
0 : 0 = 0,0 = \( E_{23}, E_{31}, E_{12} \)
239 : 239 = 119,1 = \( E_{12,36,45}, E_{14,26,35}, E_{13,25,46} \)

246
Tritangent planes

The 45 tritangent planes are:

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

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

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

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

Karaoglu [37] describes a different algorithm, based on non-conical six-arcs and Steiner 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 -n 3 -field F -v 0 -end \
  -with P -do \n  -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. In PG(3, 4), the only surface with 27 lines is the Hirschfeld surface.
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-identify_Eckardt</td>
<td></td>
<td>Identifies the isomorphism type of the Eckardt surface with parameter $a$. All values of $a$ are considered.</td>
</tr>
<tr>
<td>-identify_F13</td>
<td></td>
<td>Identifies the isomorphism type of the $F_{13}$ surface with parameter $a$. All values of $a$ are considered.</td>
</tr>
<tr>
<td>-identify_Bes</td>
<td></td>
<td>Identifies the isomorphism type of the Bes surface with parameters $a$ and $c$. All values of $a, c$ are considered.</td>
</tr>
<tr>
<td>-identify_general_abcd</td>
<td></td>
<td>Identifies the isomorphism type of the general surface with parameters $a, b, c, d$. All values of $a, b, c, d$ are considered.</td>
</tr>
<tr>
<td>-isomorphism_testing</td>
<td>S1 S2</td>
<td>Computes an isomorphism from surface S1 to surface S2 or concludes that none exists.</td>
</tr>
<tr>
<td>-recognize</td>
<td>S</td>
<td>Identifies the isomorphism type of the given surface $S$.</td>
</tr>
<tr>
<td>-create_source_code</td>
<td></td>
<td>Creates source code for the classification of cubic surfaces with 27 lines over the given field.</td>
</tr>
<tr>
<td>-sweep_Cayley</td>
<td></td>
<td>Identifies all surfaces given by the Cayley normal form over the given field.</td>
</tr>
</tbody>
</table>

Table 7.8: Activities related to the classification of cubic surfaces

### 7.4 Isomorphism Testing and Recognition

Besides classification, Orbiter provides recognition and isomorphism testing of cubic surfaces. Table 7.8 lists the relevant Orbiter commands. These commands are activities of type “classification of cubic surfaces with double sixes.”

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

```
surface_recognize_q7_abcd_2_3_3_4:
  $(ORBITER) -v 3 \n  ▶ -define F -finite_field -q 7 -end \n  ▶ ▶ -define P -projective_space -n 3 -field F -v 0 -end \n  ▶ ▶ ▶ -with P -do \n  ▶ ▶ ▶ ▶ -projective_space_activity \n  ▶ ▶ ▶ ▶ ▶ -classify_surfaces_with_double_sixes Surf -W -end \n  ▶ ▶ ▶ ▶ ▶ ▶ -end \n```
identifies the surface (cf. Table 8.5)

\[ X_0^2X_3 + X_1^2X_2 + X_1X_2^2 + X_0X_3^2 + X_1X_2X_3 = 0 \]  \hspace{1cm} (7.1)

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

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

\hspace{1cm} (7.2)

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

\[
X_0^2X_2 + X_2^2X_2 + X_1X_2^3 + X_0X_2^2 + X_1X_2^2 + X_2X_3 + \delta^{13}X_1X_3^2 + \delta^{13}X_2X_3^2 + \delta^{11}X_0X_3^2 + X_1X_2X_3 = 0
\]

and

\[
\delta^{11}X_0^2X_3 + \delta^{12}X_1X_2 + \delta^{12}X_1X_2^2 + \delta^{11}X_0X_3^2 + X_1X_2X_3 = 0,
\]

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

```
surface_isomorph_16:
   $\text{ORBITER}) -v 3 \\
   -define F -finite_field -q 16 -end \\
   -define P -projective_space -n 3 -field F -v 0 -end \\
   -with P -do \\
   -projective_space_activity \\
   -classify_surfaces_with_double_sixes Surf27 -W -end \\
```
computes an isomorphism from the first surface to the second, given by the matrix:

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

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

$$X_0^2X_3 + X_1^2X_2 + X_1X_2^2 + X_0X_3^2 + X_1X_2X_3 = 0.$$
finds that the surface is isomorphic to the surface with OCN equal to 0. An isomorphism will be computed as well.

The command

```
surface_sweep_Cayley_13:
> $(ORBITER) -v 3 \
> > -define F -finite_field -q 13 -end \
> > -define P -projective_space -n 3 -field F -v 0 -end \
> > -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> > > -sweep_Cayley \n> > -end \n> > -print_symbols
```

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

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

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

```
orbits_cubic_curves_q2:
  $\text{(ORBITER)} \ -v\ 4 \$
  $\triangleright \ -define\ G\ -linear\_group\ -PGL\ 3\ 2\ -end\ \$
  $\triangleright \ -define\ Orb\ -orbits\ -group\ G\ \$
  $\triangleright \ -\_on\_polynomials\ 3\ \$
  $\triangleright \ -end$
  $\#pdflatex\ poly\_orbits\_d3\_n3\_q2.tex$
  $\#open\ poly\_orbits\_d3\_n3\_q2.pdf$
```

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

```
poly\_orbits\_d3\_n3\_q2\_F2.csv: poly\_orbits\_d3\_n3\_q2.csv
  $\text{(ORBITER)} \ -v\ 4 \$
  $\triangleright \ -define\ F\ -finite\_field\ -q\ 2\ -end\ \$
  $\triangleright \ -define\ P\ -projective\_space\ -n\ 3\ -field\ F\ -v\ 0\ -end\ \$
  $\triangleright \ -with\ P\ -do\ \$
  $\triangleright \ -projective\_space\_activity\ \$
  $\triangleright \ -table\_of\ cubic\_surfaces\_compute\_properties\ \$
  $\triangleright \ -end$
```

and

```
Dickson\_q2\_analyze: poly\_orbits\_d3\_n3\_q2\_F2.csv
  $\text{(ORBITER)} \ -v\ 4 \$
  $\triangleright \ -define\ F\ -finite\_field\ -q\ 2\ -end\ \$
  $\triangleright \ -define\ P\ -projective\_space\ -n\ 3\ -field\ F\ -v\ 0\ -end\ \$
  $\triangleright \ -with\ P\ -do\ \$
  $\triangleright \ -projective\_space\_activity\ \$
  $\triangleright \ -cubic\_surface\_properties\_analyze\ \$
  $\triangleright \ -end$
  $\#pdflatex\ poly\_orbits\_d3\_n3\_q2\_F2\_report.tex$
  $\#open\ poly\_orbits\_d3\_n3\_q2\_F2\_report.pdf$
```

252
To investigate the properties of these surfaces over the extension field $\mathbb{F}_4$, the following two commands can be used:

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

and

```
Dickson_q4_analyze: poly_orbits_d3_n3_q2_F4.csv
  $ (ORBITER) -v 4 \n  -define F -finite_field -q 4 -end \n  -define P -projective_space -n 3 -field F -v 0 -end \n  -with P -do \n  -projective_space_activity \n  -cubic_surface_properties_analyze \n  poly_orbits_d3_n3_q2_F4.csv 2 \n  -end
  pdflatex poly_orbits_d3_n3_q2_F4_report.tex
  open poly_orbits_d3_n3_q2_F4_report.pdf
```
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 $q = 17$:

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

cubic_surfaces_tables_17:
  $(ORBITER) -v 3 \ 
  $(MAKE_TABLE_OF_CUBIC_SURFACES)

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

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

produces a latex table from the csv file.
8.1 Polynomials Over Finite Fields

For \( p \) prime, the finite field \( \mathbb{F}_p \) of order \( p \) can be constructed as factoring of the integers modulo \( p \). In this section, we will consider polynomials over \( \mathbb{F}_p \). The ring of polynomials in one variable with coefficients in \( \mathbb{F}_p \) is denoted as \( \mathbb{F}_p[X] \). Table 8.1 lists Orbiter activities for polynomials over finite fields. The activities are finite field activities. For instance, the command

\[
\text{poly\_division:} \quad \text{\$ORBITER) -v 2 \ \ \ \ \} \\
\text{\texttt{\textgreater\textgreater\textgreater\textgreater -define F -finite\_field -q 2 -end \ \textless\textless\textless\textless}} \\
\text{\texttt{\textgreater\textgreater\textgreater\textgreater -with F -do \ \textless\textless\textless\textless}} \\
\text{\texttt{\textgreater\textgreater\textgreater\textgreater -finite\_field\_activity \ \textless\textless\textless\textless}} \\
\text{\texttt{\textgreater\textgreater\textgreater\textgreater -polynomial\_division "1,0,0,0,0,0,0,0,0,0,1" "1,0,1,1" -end}}
\]

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

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

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

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

with

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

The coefficient lists in the arguments are from the lowest term up.

It is perhaps more convenient to create the polynomials as vectors, as in Section 2.7. The following example uses vectors named \( A \) and \( B \). After that, the division command is called.

\[
\text{poly\_division2:} \quad \text{\$ORBITER) -v 2 \ \ \ \ \} \\
\text{\texttt{\textgreater\textgreater\textgreater\textgreater -define A \ (-} \\
\text{\texttt{\textgreater\textgreater\textgreater\textgreater -poly\_division \ "1,0,0,0,0,0,0,0,0,0,1" \ "1,0,1,1" \ -end}}
\]
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-polynomial_division</td>
<td>$A(X) B(X)$</td>
<td>Polynomial division of $A(X)$ by $B(X)$ over $\mathbb{F}_q$. $A(X)$ and $B(X)$ are given as coefficient list, starting from the lowest coefficient.</td>
</tr>
<tr>
<td>-extended_gcd_for_polynomials</td>
<td>$A(X) B(X)$</td>
<td>Extended gcd for polynomials $A(X)$ and $B(X)$ over $\mathbb{F}_q$. $A(X)$ and $B(X)$ are given as coefficient list, starting from the lowest coefficient.</td>
</tr>
<tr>
<td>-polynomial_mult_mod</td>
<td>$A(X) B(X) M(X)$</td>
<td>Multiply the polynomials $A(X)$ and $B(X)$ modulo $M(X)$ in $\mathbb{F}_q[X]$.</td>
</tr>
<tr>
<td>-Berlekamp_matrix</td>
<td>$A(X)$</td>
<td>Computes the rank of the Berlekamp matrix associated to the polynomial $A(X)$ over $\mathbb{F}_q$. The polynomial $A(X)$ is irreducible over $\mathbb{F}_q$ if the Berlekamp matrix has rank $d - 1$ where $d$ is the degree of $A(X)$. The Berlekamp matrix is $F - I$ where $F$ is the Frobenius matrix and $I$ is the identity matrix. The Frobenius matrix is the matrix of the Frobenius endomorphism with respect to the standard basis of the polynomial ring: $1, X, X^2, \ldots, X^{d-1}$.</td>
</tr>
<tr>
<td>-polynomial_find_roots</td>
<td>$A(X)$</td>
<td>Find the roots of $A(X)$ over $\mathbb{F}_q$.</td>
</tr>
<tr>
<td>-make_table_of_irreducible_polynomials</td>
<td>$d$</td>
<td>Produces a list of all irreducible polynomials of degree $d$ over $\mathbb{F}_q$.</td>
</tr>
</tbody>
</table>

Table 8.1: Finite Field Activities Related to Polynomials
The command `-extended_gcd_for_polynomials` takes two polynomials \( A(X) \) and \( B(X) \) and computes polynomials \( U(X) \) and \( V(X) \) and \( G(X) \) such that \( G(X) \) is the greatest common divisor of \( A(X) \) and \( B(X) \) and
\[
G(X) = U(X) \cdot A(X) + V(X) \cdot B(X).
\]

For instance,

```bash
poly_gcd:
```
```
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 \
-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
\[
(3X^2 + 2X + 1) \cdot (5X^2 + 4X + 3) \mod (X^3 + 7) \mod 7.
\]

```bash
poly_mult_mod1:
```
```
define F -finite_field -q 7 -end \
with F -do \
finite_field_activity \
-polynomial_mult_mod "1,2,3" "3,4,5" "6,0,0,1" -end
```
```
which has a result of
\[
X^2 + 4X + 4.
\]

The coefficients are given from the lowest to the highest term. For the opposite order, the following command computes
\[
(2X^2 + X + 3) \cdot (4X^2 + 3X + 5) \mod (X^3 + 7) \mod 7.
\]
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:

```bash
poly_mult_mod_F4:
  $(ORBITER) -v 2 \n  -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 \n  -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 \n  -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
```

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 \( \mathbb{F}_2 \):

```bash
mult_polynomials_2_5_7:
  $(ORBITER) -v 2 \n  -define F -finite_field -q 2 -end \n  -with F -do \n  -finite_field_activity -mult_polynomials 5 7 -end
  pdflatex polynomial_mult_5_7.tex
  open polynomial_mult_5_7.pdf
```

The next command performs polynomial long division based on numerical polynomials:
Here is a somewhat larger example for numerical arguments. We wish to multiply 999 by 997 modulo 1033. The first command performs multiplication:

```
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
```

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 \
```

259
In this last command, the formula $a \cdot b$ is used and evaluated over $\mathbb{F}_{1024}$, using $a = 999$ and $b = 997$.

Orbiter allows polynomial arithmetic modulo a factor polynomial. The coefficient vector of the polynomial can be created as a vector. Here is an example which performs arithmetic modulo the CRC32 polynomial. The goal is to compute the multiplicative inverse of $X$. In order to do so, we use the fact that the CRC32 polynomial is irreducible, and hence the factor ring is a finite field of order $2^{32}$. The inverse of a polynomial can be computed by raising to the power of $2^{32} - 2$:

```
\$ (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"
```

```
\$ (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 \ndefine B -vector -field F -sparse 33 $(\text{INVERSE\_SPARSE}) -end \nwith F -do \n  -finite\_field\_activity \n  -polynomial\_mult\_mod A B M \n  -end

The product is indeed 1.

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

```
Berlekamp\_matrix\_2,3:
\$ (ORBITER) -v 2 \n  -define F -finite\_field -q 2 -end \n  -define v -vector -field F -dense "1,1,0,1" -end \n  -with F -do \n  -finite\_field\_activity \n  -Berlekamp\_matrix v -end
```

computes the Berlekamp matrix associated with the polynomial $X^3 + X + 1$ over $\mathbb{F}_2$. The matrix is

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

Since the matrix has rank 2, the polynomial is irreducible.

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

For instance, the command

```
search\_primitive\_poly_2:
\$ (ORBITER) -v 3 \n  -search\_for\_primitive\_polynomial\_in\_range 2 2 2 10 #| grep //
```

searches for primitive polynomials over $\mathbb{F}_2$ of degree 2 to 10. The unix command `grep` is used to filter the output for lines containing the given pattern “//”. This yields the list
Primitive polynomials over the base field $\mathbb{F}_s$ are converted into integers, using the base-s representation of integers. For instance, the polynomial $X^2 + X + 1$ is read as binary string 111, which in turn translates to the integer 7 (we use $s = 2$).

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

```
irred_3_4:
  $ (ORBITER) -v 6 \\
  -define F -finite_field -q 4 -end \\
  -with F -do \\
  -finite_field_activity \\
  -make_table_of_irreducible_polynomials 3 -end \\
  pdflatex Irred_q4_d3.tex \\
  open Irred_q4_d3.pdf
```

The output is:

There are 20 irreducible polynomials of degree 3 over the field $\mathbb{F}_4$:
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 polynomial rings. Table 8.2 lists the commands for creating a multivariate polynomial ring.

Table 8.3 lists the activities for a multivariate polynomial ring.

There are two orderings of the monomials which can be chosen:

1. The partition ordering is grouping terms according to the partition that results from the degrees of the variables first, and then applies the lexicographic ordering as a tie breaker.

2. The lexicographic ordering orders the monomials lexicographically.

By default, the partition ordering is used. Table 8.4 shows the monomials in the partition ordering for degrees 1, 2, 3 and 4 in a plane.
Table 8.4: The partition ordering of monomials of degree 1, 2, 3 and 4 in a plane.

Table 8.5 shows the partition ordering monomials of degree at most 3 in PG(3, q).

The following example shows how a Cremona map can be defined. At first, we define 4 polynomials as makefile variables. After that, we invoke Orbiter to create a polynomial ring and to evaluate the map.

```
CREMONA_MAP_Y0="3*y0*y0*y0*y0*y0*y0*y2+4*y0*y0*y0*y1*y1*y2+
+2*y0*y0*y0*y1*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*y0+y0*y1*y1+y1*y1*
+12*y0*y0*y1*y2+y2+3*y0*y1*y1*y1*y1*
+5*y0*y1*y1*y2+y2+0*y1*y2*y2*y2*y2" 

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

CREMONA_MAP_Y3="0"
```
Table 8.5: The partition ordering of monomials of degree 1, 2 and 3 in PG(3, q)
Cremona map:

```perl
$(ORBITER) -v 3 \\
  -define F -finite_field -q 13 -end \\
  -define P -projective_space -n 2 -field F -v 0 -end \\
  -define R -polynomial_ring \\
    -field F \\
    -number_of_variables 3 \\
    -homogeneous_of_degree 6 \\
    -monomial_ordering.lex \\
    -variables "y0,y1,y2" "y_0,y_1,y_2" \\
    -end \\
  -define Y0 -formula \\
  -define Y1 -formula \\
  -define Y2 -formula \\
  -define Cremona -collection "Y0,Y1,Y2" \\
  -with P -do \\
    -projective_space_activity \\
    -map R Cremona "" \\
  -end
```

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 PG(2, 11):

```perl
arcs_5_2_q11:
$(ORBITER) -v 4 \\
  -define F -finite_field -q 11 -end \\
  -define P -projective_space -n 2 -field F -v 0 -end \\
  -with P -do \\
    -projective_space_activity \\
    -classify_arcs \\
    -poset_classification_control \\
    -problem_label arcs_5_2_q11 \\
    -W -depth 5 \\
    -report -end \\
    -end \\
  -d 2 \\
  -end
```

267
It finds exactly two isomorphism types of arcs. The representative sets are 
\[ \{0, 1, 2, 3, 37\}, \{0, 1, 2, 3, 49\}. \]

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

\[ \text{arcs}_5_2.q11.ideal: \]
\[ $(\text{ORBITER})$ -v 2 \]
\[ $>$ define F -finite_field -q 11 -end \]
\[ $>$ define R -polynomial_ring \]
\[ $>$ define C -combinatorial_objects \]
\[ $>$ file_of_points arcs_5_2_q11_lvl_5 \]
\[ $>$ end \]
\[ $>$ with C -do \]
\[ $>$ combinatorial_object_activity \]
\[ $>$ ideal R \]
\[ $>$ end \]

The ideals are generated by
\[ 7x0*x1 + 5x0*x2 + 10*x1*x2 \]
and
\[ 4x0*x1 + 8x0*x2 + 10*x1*x2, \]
respectively.

Let us consider a smooth cubic surface with 9 lines and 4 Eckardt points. Suppose we have the set of points and we wish to determine the equation of the object. To do so, we first define the object from the given set of points.

\[ \text{PTS_OF_SURFACE_ORBIT211_Q3_L9_E4} = "\]
\[ 0,1,2,5,7,8,10,14,9,12, \]
\[ 15,3,16,37,31,34,20,19,17,32,36,33" \]
Then, we create a ring and compute the ideal:

```
surface_9lines_4E_ideal:
 ▶ $(ORBITER) -v 2 \
 ▶ ▶ -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 "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:

```
SURFACE_F_9="x0*x0*x1 - x0*x1*x1 -x0*x1*x3 -x2*x2*x3 - x2*x3*x3"
```

```
F_9_q7:
 ▶ $(ORBITER) -v 3 \
 ▶ ▶ -define F -finite_field -q 7 -end \n ▶ ▶ -define P -projective_space -n 3 -field F -v 0 -end \n ▶ ▶ -define F_9 -cubic_surface -space P \n ▶ ▶ ▶ -by_equation "F_9" \n ▶ ▶ ▶ \"\DF_9\D\" "x0,x1,x2,x3" \n ▶ ▶ ▶ $(SURFACE_F_9) \n ▶ ▶ ▶ "\Dno parameters\D" \n ▶ ▶ ▶ -end \n ▶ ▶ -with F_9 -do \n ▶ ▶ -cubic_surface_activity \n ▶ ▶ ▶ -report \n ▶ ▶ ▶ -end
```

pdflatex surface_equation_F_9_q7_report.tex
open surface_equation_F_9_q7_report.pdf
In the next example, we wish to explore the relationship between conics and \((5, 2)\)-arcs. We consider the plane \(\text{PG}(2, 11)\). 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:

```plaintext
random_k_subsets_PG_2_11:
> $(ORBITER) -v 4 \
> ▶ -create_random_k_subsets 133 5 20
```

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

```plaintext
line_type_in_PG_2_11:
> $(ORBITER) -v 3 \
> ▶ -orbiter_path $(ORBITER_PATH) \
> ▶ -define F -finite_field -q 11 -end \
> ▶ -define P -projective_space -n 2 -field F -v 0 -end \
> ▶ -define C -combinatorial_objects \
> ▶ ▶ -file_of_points random_k_subsets_n133_k5_nb20.csv \
> ▶ ▶ -end \
> ▶ ▶ -with C -do \
> ▶ ▶ -combinatorial_object_activity \
> ▶ ▶ ▶ -line_type P random_sets \
```

It turns out that the second set is an arc. It is the set \(\{3, 33, 40, 83, 102\}\). We create the conic through these 5 points:

```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
```

270
The ideal is generated by
\[ 10x0x0 + 3x0x1 + 8x0x2 + 2x1x1 + 10x2x2. \]

The conic contains the following 12 points:
\[ \{3, 15, 19, 33, 40, 42, 46, 50, 83, 88, 102, 108\}. \]

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

```
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"
```

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

```
ocite{Endrass_F7.txt}
```

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

```
ocitic_prepare:
```

271
There are 165 monomials. They are listed in the file `octic_monomials_sorted.txt`.
Chapter 9

Applications

9.1 Number Theory

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

\[ \text{inverse}\ mod\ a: \]
\[ \triangleright \ (\text{ORBITER}) \ -v \ 2 \ -\text{inverse}\ mod \ 18059241 \ 58014043 \]

computes the inverse of 18059241 modulo 58014043.

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

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

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

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

where

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

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

\[ \text{jacobi\ a: } \]
\[ \triangleright \ (\text{ORBITER}) \ -v \ 5 \ -\text{jacobi} \ 2221 \ 7817 \]

273
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-jacobi</td>
<td>a p</td>
<td>Computes the Jacobi symbol ( \left( \frac{a}{p} \right) ).</td>
</tr>
<tr>
<td>-sift_smooth</td>
<td>a n primes</td>
<td>Computes all smooth numbers in the interval ([a, a + n - 1]). Smooth means that they factor completely over the list of primes given.</td>
</tr>
<tr>
<td>-random</td>
<td>n fname</td>
<td>Creates ( n ) random numbers and writes them to the csv file \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:

\[
\begin{align*}
\left( \frac{2221}{7817} \right) &= \left( \frac{7817}{2221} \right) \cdot (-1)^{\frac{2221-1}{2} \cdot \frac{7817-1}{2}} \\
&= \left( \frac{7817}{2221} \right) \\
&= \left( \frac{2221}{2221} \right) \cdot \left( \frac{577}{2221} \right) \\
&= (-1)^{\frac{2221^2-1}{2} \cdot \frac{577}{2221}} \\
&= (-1) \cdot \left( \frac{577}{2221} \right) \\
&= (-1) \cdot \left( \frac{2221}{577} \right) \cdot (-1)^{\frac{577-1}{2} \cdot \frac{2221-1}{2}} \\
&= (-1) \cdot \left( \frac{577}{577} \right) \\
&= (-1) \cdot \left( \frac{2221}{577} \right) \\
&= (-1) \cdot \left( \frac{490}{577} \right)
\end{align*}
\]
\[
= (-1) \cdot \left( \frac{2}{577} \right) \cdot \left( \frac{245}{577} \right) \\
= (-1) \cdot (-1)^{\frac{577^2 - 1}{8}} \cdot \left( \frac{245}{577} \right) \\
= (-1) \cdot \frac{245}{577} \\
= (-1) \cdot \frac{577}{245} \cdot (-1)^{\frac{245 - 1}{2} \cdot \frac{577 - 1}{2}} \\
= (-1) \cdot \frac{577}{245} \\
= (-1) \cdot \frac{87}{245} \\
= (-1) \cdot \left( \frac{245}{87} \right) \cdot (-1)^{\frac{87 - 1}{2} \cdot \frac{245 - 1}{2}} \\
= (-1) \cdot \frac{245}{87} \\
= (-1) \cdot \frac{71}{87} \\
= (-1) \cdot \frac{87}{71} \cdot (-1)^{\frac{71 - 1}{2} \cdot \frac{87 - 1}{2}} \\
= \left( \frac{87}{71} \right) \\
= \left( \frac{16}{71} \right) \\
= \left( \frac{2}{71} \right)^4 \cdot \left( \frac{1}{71} \right) \\
= \left( (-1)^{\frac{71^2 - 1}{8}} \right)^4 \cdot \left( \frac{1}{71} \right) \\
= \left( \frac{1}{71} \right) \\
= 1
\]

The answer 1 tells us that 2221 is a square modulo 7817. Because 7817 is prime, the Jacobi symbol and the Legendre symbol agree on this input pair. We can use the `square_root_mod` command from Section 3.1 to compute a square root of 2221 modulo 7817 and verify this fact. The command

```
$ $(ORBITER) -v 2 -square_root_mod 2221 7817$
```

yields that 7634 is a square root. Indeed,

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

<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-orbits_on_polynomials</td>
<td>$d$</td>
<td>Computes the representation of the group $G$ on homogeneous polynomials of degree $d$. This is a group theoretic activity as described in Section 5.6. The group $G$ must be constructed first.</td>
</tr>
</tbody>
</table>

### 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

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

creates $G = \text{PGL}(4,3)$ and computes the representation on polynomials of degree 3 in 4 variables. The representation has degree 20. The second command produces bitmap drawings for the representing matrices associated with a generating set of the group. Figure 9.1 shows the representing matrices for a generating set of size 9.
Figure 9.1: Representation of $\text{PGL}(4, 3)$ on cubic polynomials
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-solovay_strassen</td>
<td>a n</td>
<td>Performs n Solovay / Strassen tests on the number a</td>
</tr>
<tr>
<td>-miller_rabin</td>
<td>a n</td>
<td>Performs n Miller / Rabin tests on the number a</td>
</tr>
<tr>
<td>-fermat</td>
<td>a n</td>
<td>Performs n Fermat tests on the number a</td>
</tr>
<tr>
<td>-find_pseudoprime</td>
<td>a 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 global cryptographic commands are shown. Some cryptographic commands require a finite field and appear as a finite field activity, see Table 9.4. For instance,

**EC_add:**

```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
```

add...
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-EC_add</td>
<td>$a \ b \ i_1 \ i_2$</td>
<td>On the elliptic curve $y^2 \equiv x^3 + ax + b$ in $\mathbb{F}_q$, add the points with indices $i_1$ and $i_2$, each given as a pair $x, y$</td>
</tr>
<tr>
<td>-EC_points</td>
<td>$a \ b$</td>
<td>Computes all points of the elliptic curve $y^2 \equiv x^3 + ax + b$ over $\mathbb{F}_q$</td>
</tr>
<tr>
<td>-EC_multiple_of</td>
<td>$a \ b \ pt \ n$</td>
<td>Computes the $n$ fold multiple of the given point $pt$ on the elliptic curve $y^2 \equiv x^3 + ax + b$ over $\mathbb{F}_q$</td>
</tr>
<tr>
<td>-EC_cyclic_subgroup</td>
<td>$a \ b \ pt$</td>
<td>Computes the cyclic subgroup generated by the given point $pt$ on the elliptic curve $y^2 \equiv x^3 + ax + b$ over $\mathbb{F}_q$</td>
</tr>
<tr>
<td>-EC_Koblitz_encoding</td>
<td>$a \ b \ s \ pt \ plain$</td>
<td>Computes the Koblitz encoding of “plain” (all caps) on the elliptic curve $y^2 \equiv x^3 + ax + b$ over $\mathbb{F}_q$ using the base point $pt$ and the secret exponent $s$</td>
</tr>
<tr>
<td>-EC_bsgs</td>
<td>$a \ b \ pt \ n \ cipher$</td>
<td>Prepare the baby-step giant-step tables for the ciphertext “cipher” on the elliptic curve $y^2 \equiv x^3 + ax + b$ over $\mathbb{F}_q$ using the base point $pt$ of order $n$</td>
</tr>
<tr>
<td>-EC_bsgs_decode</td>
<td>$a \ b \ pt \ n \ cipher \ round-keys$</td>
<td>Decodes the ciphertext “cipher” on the elliptic curve $y^2 \equiv x^3 + ax + b$ over $\mathbb{F}_q$ using the base point $pt$ of order $n$ and the round keys “keys”</td>
</tr>
<tr>
<td>-EC_discrete_log</td>
<td>$a \ b \ pt \ base-pt$</td>
<td>Computes the elliptic curve discrete log analogue of $pt$ with respect to $base-pt$ on the elliptic curve $y^2 \equiv x^3 + ax + b$ over $\mathbb{F}_q$</td>
</tr>
<tr>
<td>-NTRU_encrypt</td>
<td>$N \ p \ H \ R \ M$</td>
<td>NTRU encryption for the message $M(X)$ using the public key $H(X)$ and one-time-key $R(X)$</td>
</tr>
<tr>
<td>-polynomial_center_lift</td>
<td>$A(X)$</td>
<td>Compute the center lift mod $q$ for the coefficients of $A$</td>
</tr>
<tr>
<td>-polynomial_reduce_mod_p</td>
<td>$p \ A(X)$</td>
<td>Reduce the coefficients of the polynomial $A$ modulo $p$</td>
</tr>
</tbody>
</table>

Table 9.4: Finite Field Activities related to Cryptography
The elliptic curve $y^2 = x^3 + 5x + 7 \mod 199$ computes the cyclic subgroup generated by the point $(1, 4)$ on the curve $y^2 = x^3 + x + 3 \mod 11$. The command

```
EC_points_199:
  $(ORBITER) -v 2 \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

```
EC_bsgs:
  > $(ORBITER) -v 2 \n  > > -define F -finite_field -q 199 -end \n  > > -with F -do \n  > > -finite_field_activity \n  > > -EC_bsgs 5 7 "147,164" 212 \n  > > "172,158,45,195,50,22,10,103,55,33,50,22,145,105,31,74,73,155,67,60,25,6" \n  > > -end
```

performs a baby-step-giant-step brute force attack on the ciphertext sequence

\[
R_i = (172, 158), (45, 195), (50, 22), (10, 103), (55, 33), (50, 22), (145, 105), (31, 74), (73, 155), (67, 60), (25, 6),
\]

using the base point \((147, 164)\) on the curve \( y^2 = x^3 + 5x + 7 \) mod 199, assuming a group order of 212. The command

```
EC_bsgs_decode:
  > $(ORBITER) -v 2 \n  > > -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  > > "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
```
decodes the ciphertext sequence
\[ T_i = (127, 188), (51, 141), (85, 29), (106, 90), (41, 105), (179, 71), (171, 2), (16, 197), (183, 72), (27, 129), (37, 10), \]
assuming round keys
\[ k_i = 50, 179, 169, 13, 153, 169, 115, 116, 188, 110, 176, \]
using the base point \((147, 164)\) on the curve \(y^2 = x^3 + 5x + 7 \mod 199\), and assuming a group order of 212.

The next sequence of examples discusses the NTRU cryptosystem (cf. Example 7.53 in [35]). In the example, we choose the parameters of the cryptosystem to be \((N, p, q, d) = (7, 41, 3, 2)\). Orbiter uses the following convention for polynomials over a finite field \(\mathbb{F}_q\): The coefficients of \(A(X) = a_0 + a_1X + \cdots + a_dX^d\) are listed as a sequence, starting with the constant term and ending with the leading coefficient. The cryptosystem requires coefficients \(a_i\) in the range \(-\frac{p}{2} \leq a_i \leq \frac{p}{2}\). So, in an extension to the conventions for field elements in \(\mathbb{F}_q\), Orbiter allows negative coefficients as well. The assumption is that \(q\) is prime and negative coefficients are considered modulo \(q\).

In the example, Alice picks the private polynomials
\[ f(x) = x^6 - x^4 + x^3 + x^2 - 1 \text{ (with } d+1 \text{ coefficients equal to plus one and } d \text{ coefficients equal to minus one)} \]
and
\[ g(x) = x^6 + x^4 - x^2 - x \text{ with } d \text{ coefficients plus one and } d \text{ coefficients minus one}. \]
We also need the polynomial \(x^N - 1\). The makefile commands

\begin{verbatim}
NTRU_N=7
NTRU_P=3
NTRU_Q=41
NTRU_D=2
NTRU_XN1="-1,0,0,0,0,0,0,1,"
ALICE_PRIVATE_F="-1,0,1,1,-1,0,1"
ALICE_PRIVATE_G="0,-1,-1,0,1,0,1"
\end{verbatim}

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

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

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

To this end, we can use the \texttt{extended gcd for polynomials} command from Table 9.1. The following two makefile commands do the job:

\begin{verbatim}
NTRU_Alice1:
  \>$\$(ORBITER) -v 2 \ 
  \>$\define F -finite_field -q $\$(NTRU_Q) -end \ 
  \>$\with F -do \ 
  \>$-finite_field_activity \ 
  \>$-extended_gcd_for_polynomials \ 
  \>$\ $(NTRU_XN1) $(ALICE_PRIVATE_F) \ 
  \>$-end

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

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

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

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

There is a chance that the polynomial $f(x)$ does not have an inverse in either $\mathbb{F}_p[x]$ or in $\mathbb{F}_q[x]$. In that case, Alice simply chooses a different polynomial $f(x)$ and tries again. Alice can now compute her public key:

\begin{verbatim}
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
\end{verbatim}
The public key is assigned to the makefile variable ALICE_PUBLIC_KEY. Now, Bob chooses his message to Alice and his one-time-key. The message must be the center lift of a polynomial in $F_p[x]$. The round-key must have exactly $d$ coefficients one and $d$ coefficients $-1$ (rest zeroes).

BOB_MESSAGE="1,-1,1,1,0,-1"

BOB_ONE_TIME_KEY="-1,1,0,0,0,-1,1"

The encryption proceeds using the NTRU_encrypt command, and the result is stored in the makefile variable BOB_ENCRYPT:

NTRU_encrypt:
▷ $(ORBITER) -v 2 \\
▷ ▷ -define F -finite_field -q $(NTRU_Q) -end \\
▷ ▷ -with F -do \\
▷ ▷ -finite_field_activity \\
▷ ▷ -NTRU_encrypt $(NTRU_N) $(NTRU_P) $(ALICE_PUBLIC_KEY) \\
▷ ▷ $(BOB_ONE_TIME_KEY) $(BOB_MESSAGE) -end

BOB_ENCRYPT= "25,3,40,2,4,19,31"

Decryption is done in five steps.

NTRU_decrypt1:
▷ $(ORBITER) -v 2 \\
▷ ▷ -define F -finite_field -q $(NTRU_Q) -end \\
▷ ▷ -with F -do \\
▷ ▷ -finite_field_activity \\
▷ ▷ -polynomial_mult_mod $(ALICE_PRIVATE_F) \\
▷ ▷ ▷ $(BOB_ENCRYPT) $(NTRUE_XN1) \\
▷ ▷ -end

ALICE_C1="40,1,40,40,33,10,1"

NTRU_decrypt2:
▷ $(ORBITER) -v 2 \\
▷ ▷ -define F -finite_field -q $(NTRU_Q) -end \\

284
Decryption produces Bob’s message to Alice.

ToDo:

• RSA

• $\sqrt{\text{mod}}$
• quadratic sieve
• pseudoprimes
Chapter 10
Coding Theory

10.1 Introduction

Orbiter supports research in coding theory. Global Orbiter commands for coding theory are summarized in Table 10.1. Additional commands, associated with objects of type code will be discussed below and in later sections.

The command

```
Allen_Gates_noise_1_percent:
  $(ORBITER) -v 3 \n  -random_noise_in_bitmap_file \n  allen_Gates.bmp \n  allen_Gates_1.bmp \n  1 100 \n  open allen_Gates_1.bmp
```

simulates random noise at the 1 percent level applied to the file `allen_Gates.bmp`, see Figure 10.1. The original is on the left. The effect of noise can be seen on the right. The picture shows Paul Allen and Bill Gates in the early 1970s.

The command

```
Hamming_space_4_2_distance_matrix:
  $(ORBITER) -Hamming_space_distance_matrix 4 2
```

creates the distance matrix of the Hamming graph $H(n,q)$. The data is written to the file `Hamming_n4_q2.csv`. The command

```
Hamming_space_4_2_distance_matrix_draw:
  $(ORBITER) -v 2 -draw_matrix \n  -input_csv_file Hamming_n4_q2.csv \n  -box_width 20 -bit_depth 24
```
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-make_macwilliams_system</td>
<td>(q\ n\ k)</td>
<td>Create the MacWilliams equations for the weight enumerator of the dual code.</td>
</tr>
<tr>
<td>-table_of_bounds</td>
<td>(n_{\text{max}}\ q)</td>
<td>Make a table of bounds for (q)-ary linear code for all (k \leq n \leq n_{\text{max}})</td>
</tr>
<tr>
<td>-make_bounds_for_d_given_n_and_k_and_q</td>
<td>(n\ k\ q)</td>
<td>Make bounds for the minimum distance of a ([n, k]_q) code</td>
</tr>
<tr>
<td>-Hamming_space_distance_matrix</td>
<td>(n\ q)</td>
<td>Make the distance matrix of the Hamming graph (H(n, q)).</td>
</tr>
<tr>
<td>-random_noise_in_bitmap_file</td>
<td>(f1\ f2\ n\ d)</td>
<td>Apply random noise a the (d/n) level to the bitmap file (f1) and write to (f2).</td>
</tr>
<tr>
<td>-introduce_errors</td>
<td>CRC-options</td>
<td>Introduce errors to a file. See Table 10.6.</td>
</tr>
<tr>
<td>-check_errors</td>
<td>CRC-options</td>
<td>Find errors in a CRC coded file. See Table 10.6.</td>
</tr>
<tr>
<td>-extract_block</td>
<td>CRC-options</td>
<td>Extract a block from a CRC coded file. See Table 10.6.</td>
</tr>
</tbody>
</table>

Table 10.1: Global Coding Theoretic Commands

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

```
▷▷▷ -partition 4 16 16 \
▷▷ -end
▷ open Hamming_n4_q2_draw.bmp
```

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

The command

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

creates the coefficient matrix of the MacWilliams system for the $[7,4,2]$ Hamming code:
For examples concerning the bounds, see Section 10.8.

Tables 10.2 and 10.3 list coding theoretic activities in Orbiter. Depending on the activity, an object of type code or an object of type finite field is required.

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

```
CODE 5 2 3

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

code_5_2_3_diagram:

```
$(ORBITER) -v 2 \n  define F -finite_field -q 2 -end \n  with F -do -coding_theoretic_activity \n  code_diagram "code_5_2_3" \n  metric_balls 1 \n  end \n$(ORBITER) -v 2 \n  draw_matrix \n  input_csv_file code_5_2_3_diagram.01.5.4.csv \n  box_width 25 -bit_depth 24 \n  partition 4 8 4 \n  end
```

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

```
Hamming_5_2_graph:

$(ORBITER) -v 2 \n  define G -graph -Hamming 5 2 -end \n  with G -do \n  graph_theoretic_activity -export_csv -end
```

290
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-BCH</td>
<td>$n q t$</td>
<td>Compute a table of BCH codes of length $n$ over $\mathbb{F}_q$.</td>
</tr>
<tr>
<td>-BCH_dual</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-general_code_binary</td>
<td>$n$ text</td>
<td></td>
</tr>
<tr>
<td>-code_diagram</td>
<td>label codewords n</td>
<td></td>
</tr>
<tr>
<td>-code_diagram_from_file</td>
<td>fname</td>
<td></td>
</tr>
<tr>
<td>-enhance</td>
<td>radius</td>
<td></td>
</tr>
<tr>
<td>-metric_balls</td>
<td>radius</td>
<td></td>
</tr>
<tr>
<td>-long_code</td>
<td>$n$ generators</td>
<td></td>
</tr>
<tr>
<td>-encode_text_5bits</td>
<td>input fname</td>
<td></td>
</tr>
<tr>
<td>-field_induction</td>
<td>fname-in fname-out nb-bits</td>
<td></td>
</tr>
<tr>
<td>-crc32</td>
<td>text</td>
<td></td>
</tr>
<tr>
<td>-crc32_hexdata</td>
<td>hexdata</td>
<td></td>
</tr>
<tr>
<td>-crc32_test</td>
<td>block-length</td>
<td></td>
</tr>
<tr>
<td>-crc256_test</td>
<td>message-length $R$ $k$</td>
<td></td>
</tr>
<tr>
<td>-crc32_remainders</td>
<td>msg-length</td>
<td></td>
</tr>
<tr>
<td>-crc32_file_based</td>
<td>fname-in fname-out block-length</td>
<td></td>
</tr>
<tr>
<td>-crc_new_file_based</td>
<td>fname</td>
<td></td>
</tr>
<tr>
<td>-weight_enumerator</td>
<td>matrix</td>
<td>Compute the complete weight enumerator of the linear code generated by the $m \times n$ matrix $L$</td>
</tr>
</tbody>
</table>

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

Table 10.3: Coding Theoretic Activities (Part II)
Using the unix dot program, this command sequence creates the drawing of $H(5, 2)$ shown in Figure 10.3.

```bash
$ (ORBITER) -v 2 -draw_matrix
$ -input_csv_file Hamming_5_2.csv
$ -box_width 8 -bit_depth 24 -partition 4 32 32
$ dot -Tpng Hamming_5_2.gv >Hamming_5_2.png
```
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-field</td>
<td>$F$</td>
<td>Specify the field of definition.</td>
</tr>
<tr>
<td>-linear_code_through_generator_matrix</td>
<td>$M$</td>
<td>Create a code defined by a generator matrix.</td>
</tr>
<tr>
<td>-linear_code_from_projective_set</td>
<td>$nmk S$</td>
<td>Create a code defined by a projective set in the dual.</td>
</tr>
<tr>
<td>-linear_code_by_columns_of_parity_check</td>
<td>$nmk M$</td>
<td>Create a code defined by a affine set in the dual.</td>
</tr>
<tr>
<td>-first_order_Reed_Muller</td>
<td>$m$</td>
<td>Create a first order Reed-Muller code of degree $m$.</td>
</tr>
<tr>
<td>-BCH</td>
<td>$n d$</td>
<td>BCH code of length $n$ with prescribed minimum distance $d$.</td>
</tr>
<tr>
<td>-Reed_Solomon</td>
<td>$n d$</td>
<td>Not yet implemented.</td>
</tr>
<tr>
<td>-Gilbert_Varshamov</td>
<td>$n k d$</td>
<td>Create a Gilbert-Varshamov code of length $n$ with dimension $k$ and minimum distance at least $d$.</td>
</tr>
</tbody>
</table>

Table 10.4: Commands to Create Codes

<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-dual</td>
<td></td>
<td>Compute the dual code.</td>
</tr>
</tbody>
</table>

Table 10.5: Code modifications

10.2 Linear Codes

In this section, we will see how linear codes can be created and studied in Orbiter. A code object is used to represent a specific code. Table 10.4 list the commands to create a code object. Table 10.5 list code modifications. These are commands used to create a new code from an old one.

The following command creates the first order Reed-Muller code in three variables:

```
RM_3_1:
  ▷ $(ORBITER) -v 2 \
  ▷ ▷ -define F -finite_field -q 2 -end \
  ▷ ▷ -define C -code -field F \
  ▷ ▷ ▷ -first_order_Reed_Muller 3 \
  ▷ ▷ -end \n  ▷ -with C -and F -do -coding_theoretic_activity \n```

294
Let us create the Hamming code. The dual of the Hamming code is the simplex code, so we create the simplex code first. The following makefile variable is defined to hold the generator matrix of the simplex code:

```
SIMPLEX_CODE_GENERATOR="
1,0,1,0,1,0,1, \\
0,1,1,0,0,1,1, \\
0,0,0,1,1,1,1"
```

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

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

The following latex output is produced:

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

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

Basis for Perp:
\[
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 1
\end{bmatrix}
\]

295
It is possible to create the Hamming code by taking the dual of the simplex code. The following command does so:

```
Hamming code:
$ $(ORBITER) -v 2 \
  -define F -finite_field -q 2 -end \
  -define v -vector -field F -format 3 \
  -dense $(SIMPLEX_CODE_GENERATOR) \
  -end \
  -define C -code -field F \
  -linear_code_through_generator_matrix v \
  -dual \
  -end \
  -with C -do -coding_theoretic_activity \
  -export_magma Hamming.magma \
  -end
```

The command also exports the code to magma by means of the magma file `Hamming.magma`, shown below:

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

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

```
RM_3_1_and_codewords:
$ $(ORBITER) -v 2 \
  -define F -finite_field -q 2 -end \
  -define C -code -field F -first_order_Reed_Muller 3 -end \
  -with C -and F -do -coding_theoretic_activity \
  -export_magma RM_3_1.magma \
  -end \
  -with C -and F -do -coding_theoretic_activity \
  -export_codewords RM_3_1_codewords.csv \
  -end \
  -with C -and F -do -coding_theoretic_activity \
  -export_genma RM_3_1_genma.csv \
  -end
```

296
Alternatively, we can store the generator matrix in a makefile variable:

```plaintext
CODE_RM_3_1_GENMA="\n1111111\n0101010\n0011001\n0001111"
```

The following command creates the Hamming code from its generator matrix directly:

```plaintext
RM_3_1_from_generator_matrix:
▷ $(ORBITER) -v 2 \\
▷ ▷ -define F -finite_field -q 2 -end \\
▷ ▷ -define genma -vector -format 8 -field F \\
▷ ▷ ▷ -compact $(CODE_RM_3_1_GENMA) \\
▷ ▷ -end \\
▷ ▷ -define C -code -field F \\
▷ ▷ ▷ -linear_code_through_generator_matrix genma \\
▷ ▷ -end \\
▷ #pdflatex code_n8_k4_q2.tex \\
▷ #open code_n8_k4_q2.pdf
```

The following command creates the Hamming code and produces a list of codewords.

```plaintext
RM_3_1_and_codewords:
▷ $(ORBITER) -v 2 \\
▷ ▷ -define F -finite_field -q 2 -end \\
▷ ▷ -define C -code -field F -first_order_Reed_Muller 3 -end \\
▷ ▷ -with C -and F -do -coding_theoretic_activity \\
▷ ▷ ▷ -export_magma RM_3_1.magma \\
▷ ▷ -end \\
▷ ▷ -with C -and F -do -coding_theoretic_activity \\
▷ ▷ ▷ -export_codewords RM_3_1_codewords.csv \\
▷ ▷ -end \\
▷ ▷ -with C -and F -do -coding_theoretic_activity \\
▷ ▷ ▷ -export_genma RM_3_1_genma.csv \\
▷ ▷ -end
```

The Hamming code is cyclic. To see this, we need to consider the action of the Singer cycle on the set of points of PG(2, 2). The following command creates the Singer cycle:
Figure 10.4: The Hamming code

Hamming.singer:

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

This produces the following output:

Strong generators for a group of order 7:
Basic Orbit 0

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

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

```
SIMPLEX_CODE_GENMA_CYCLIC="\n1,0,0,1,1,1,0, \n0,1,0,0,1,1,1, \n0,0,1,1,1,0,1"
```

```
Hamming_cyclic_generator:
▷ $(ORBITER) -v 2 \n▷ ▷ -define F -finite_field -q 2 -end \n▷ ▷ -define v -vector -format 3 -field F \n▷ ▷ ▷ -dense $(SIMPLEX_CODE_GENMA_CYCLIC) \n▷ ▷ -end \n▷ ▷ -with F -do -finite_field_activity \n▷ ▷ -nullspace v \n▷ ▷ -end
▷ pdflatex nullspace_3_7.tex
▷ open nullspace_3_7.pdf
```

This produces the following output:
Orbiter can compute the weight enumerator and the minimum distance of codes. Let us consider the Hamming code, for example. We use a makefile variable for the generator matrix:

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

The next command computes the weight enumerator:

```
Hamming_weight Enumerator:
> $(ORBITER) -v 2 \
>   -define F -finite_field -q 2 -end \
>   -define v -vector -field F -format 4 \n>   -dense $(HAMMING_CODE_GENERATOR) \n>   -end \n>   -define C -code -field F \n>   -linear_code_through_generator_matrix v \n>   -end \n>   -with C -do \n>   -coding_theoretic_activity \n>   -weight Enumerator \n>   -end
```
We find that the weight enumerator is

\[(1, 0, 0, 7, 7, 0, 0, 1).\]

The next command computes the minimum distance of the code:

```
Hamming_minimum_distance:
  $(ORBITER) -v 2 \n  -define F -finite_field -q 2 -end \n  -define v -vector -field F -format 4 \n  -dense $(HAMMING_CODEGENERATOR) \n  -end \n  -with F -do \n  -coding_theoretic_activity \n  -minimum_distance v \n  -end
```

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

```
Golay23_minimum_distance:
  $(ORBITER) -v 2 \n  -define F -finite_field -q 2 -end \n  -define v -vector -field F -format 12 \n  -dense $(GOLAY23_CODEGENERATOR) \n  -end \n  -with F -do \n  -coding_theoretic_activity \n  -minimum_distance v \n  -end
```

301
10.3 Golay Codes

The Golay code of length 23 is a perfect code of dimension 12 and minimum distance 7. The metric balls of radius three centered around codewords cover the whole Hamming space. We can create the code by listing the columns of a generator matrix in Orbiter ranks of points in PG(11, 2). The following makefile variable does that:

```
GOLAY_23_COLUMN_RANKS_PROJECTIVELY="0, 1, 2, 3, 4, 5, 6, 7, \
8, 9, 10, 11, 132, 913, 1460, 1750, 1898, 2518, 2787, 2874, \
3320, 3357, 3662"
```

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

```
Golay23_code_words:  
  $(ORBITER) -v 2 \  
  -define v -vector -dense $(GOLAY_23_COLUMN_RANKS_PROJECTIVELY) -end \  
  -define F -finite_field -q 2 -end \  
  -define C -code -field F \  
  -linear_code_from_from_projective_set 12 v -end \  
  -with C -and F -do -coding_theoretic_activity \  
  -export_magma Golay23.magma \  
  -end \  
  -with C -and F -do -coding_theoretic_activity \  
  -export_codewords Golay23_codewords.csv \  
  -end \  
  -with C -and F -do -coding_theoretic_activity \  
  -export_genma Golay23_genma.csv \  
  -end \  
  #pdflatex code_n23_k12_q2.tex \  
  #open code_n23_k12_q2.pdf
```
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-input</td>
<td>fname</td>
<td>Input file name.</td>
</tr>
<tr>
<td>-output</td>
<td>fname</td>
<td>Output file name.</td>
</tr>
<tr>
<td>-block_length</td>
<td>$L$</td>
<td>Set block length to $L$ field elements.</td>
</tr>
<tr>
<td>-block_based_error_generator</td>
<td></td>
<td>Apply block-based error generator.</td>
</tr>
<tr>
<td>-file_based_error_generator</td>
<td>threshold</td>
<td>Apply file-based error generator.</td>
</tr>
<tr>
<td>-nb_repeats</td>
<td>$N$</td>
<td>Set the number of repeats to $N$.</td>
</tr>
<tr>
<td>-threshold</td>
<td>$t$</td>
<td>Set probability of error per experiment to $t/1000000$.</td>
</tr>
<tr>
<td>-error_log</td>
<td>fname</td>
<td>Set file name for error logging.</td>
</tr>
<tr>
<td>-selected_block</td>
<td>$i$</td>
<td>Set block number.</td>
</tr>
</tbody>
</table>

Table 10.6: CRC options

## 10.4 CRC Codes

A CRC code can be used to detect communication errors. It is a cyclic code, and hence generated by a polynomial over a finite field. The message is encoded as a string, which is then thought of as a polynomial, called the information polynomial. Assume that the check polynomial has degree $d$. The information polynomial is then divided by the check polynomial. The remainder is added to the information polynomial multiplied by $X^d$. This is the codeword, which is sent.

Table 10.6 summarizes options associated with commands for CRC-codes.

Here is an example. We consider a short string of English text and encode it with 5 bits per character. This is done using the `-encode_text_5bits` command. The encoded text is stored in a csv file, which we decide to call `text.csv`.

```
encode_text_5bits:
  $(ORBITER) -encode_text_5bits \n  "Hithere" "text.csv"
  $(ORBITER) -v 2 \n  -define F -finite_field -q 2 -end \n  -with F -do \n  -coding_theoretic_activity \n  -polynomial_division_from_file \n  text.csv 13 -end
  pdflatex polynomial_division_file_13.tex
  open polynomial_division_file_13.pdf
```

303
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  $ -coding_theoretic_activity \n  $ -polynomial_division_from_file \n  $ -text_with_error.csv 13 \n  $ -end
```

This creates the following output:

```
text.csv / 13 =
1010110100110101010111000010111100 / 1101 =
1101101110000110111111010000101
==================================
1101 | 1010110100110101010111000010111100
  1101
  ====
  111110101101101010111000010111100
  1101
  ====
  1010100110101010111000010111100
  1101
  ====
  111100110101010111000010111100
  1101
  ====
  1000110101010111000010111100
  1101
  ====
  101110101010111000010111100
  1101
  ====
  110110101010111000010111100
  1101
  ====
  101010111000010111100
```
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:

```bash
encode_text_5bits_1error:
  $(ORBITER) -encode_text_5bits 'Hithere' 'text.csv'
  $(ORBITER) -v 2
  -define F -finite_field -q 2 -end
  -with F -do
```

/13 = 1841528453 Remainder 5
The following output is created:

<table>
<thead>
<tr>
<th>Number</th>
<th>Binary Code</th>
<th>Polynomial</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0101011101101101011011110000110111101</td>
<td>X^2{-2}</td>
</tr>
<tr>
<td>1</td>
<td>0101011101101101011011110000110111101</td>
<td>X^2 + X + 1</td>
</tr>
<tr>
<td>2</td>
<td>0101011101101101011011110000110111101</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0101011101101101011011110000110111101</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0101011101101100111011110000110111101</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0101011101101101011011110000110111101</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>0101011101101101011011110000110111101</td>
<td>X</td>
</tr>
<tr>
<td>7</td>
<td>0101011101101100111011110000110111101</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>0101011101101101011011110000110111101</td>
<td>X^2{-2}</td>
</tr>
<tr>
<td>9</td>
<td>0101011101101101011011110000110111101</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>0101011101101101011011110000110111101</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0101011101101100111011110000110111101</td>
<td>2</td>
</tr>
<tr>
<td>12</td>
<td>0101011101101101011011110000110111101</td>
<td>X</td>
</tr>
<tr>
<td>13</td>
<td>0101011101101101011011110000110111101</td>
<td>X + 1</td>
</tr>
<tr>
<td>14</td>
<td>0101011101101100111011110000110111101</td>
<td>4</td>
</tr>
<tr>
<td>15</td>
<td>0101011101101100111011110000110111101</td>
<td>X^2{-2}</td>
</tr>
<tr>
<td>16</td>
<td>0101011101101100111011110000110111101</td>
<td>1</td>
</tr>
<tr>
<td>17</td>
<td>0101011101101101011011110000110111101</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>0101011101101101011011110000110111101</td>
<td>2</td>
</tr>
<tr>
<td>19</td>
<td>0101011101101101011011110000110111101</td>
<td>X^2{-2}</td>
</tr>
<tr>
<td>20</td>
<td>0101011101101101011011110000110111101</td>
<td>3</td>
</tr>
<tr>
<td>21</td>
<td>0101011101101100111011110000110111101</td>
<td>4</td>
</tr>
<tr>
<td>22</td>
<td>0101011101101100111011110000110111101</td>
<td>X^2{-2}</td>
</tr>
<tr>
<td>23</td>
<td>0101011101101101011011110000110111101</td>
<td>1</td>
</tr>
<tr>
<td>24</td>
<td>0101011101101101011011110000110111101</td>
<td>0</td>
</tr>
<tr>
<td>25</td>
<td>0101011101101101011011110000110111101</td>
<td>2</td>
</tr>
<tr>
<td>26</td>
<td>0101011101101101011011110000110111101</td>
<td>X^2{-2}</td>
</tr>
<tr>
<td>27</td>
<td>0101011101101101011011110000110111101</td>
<td>3</td>
</tr>
<tr>
<td>28</td>
<td>0101011101101101011011110000110111101</td>
<td>X</td>
</tr>
<tr>
<td>29</td>
<td>0101011101101100111011110000110111101</td>
<td>4</td>
</tr>
<tr>
<td>30</td>
<td>0101011101101100111011110000110111101</td>
<td>X^2{-2}</td>
</tr>
<tr>
<td>31</td>
<td>0101011101101101011011110000110111101</td>
<td>1</td>
</tr>
<tr>
<td>32</td>
<td>0101011101101101011011110000110111101</td>
<td>0</td>
</tr>
<tr>
<td>33</td>
<td>0101011101101101011011110000110111101</td>
<td>X</td>
</tr>
</tbody>
</table>

306
It shows that 5 single bit errors are undetected.

The following command performs an exhaustive search over all binary CRC polynomials of degree \( k = 10 \) which can detect every error pattern of Hamming weight at most \( t = 3 \) in messages of length \( n = 128 \).

\[
\text{CRC}_{3,128,10}:
\begin{align*}
\text{\texttt{\$(ORBITER) -v 1 \ \textbf{-define F -finite_field -q 2 -end \ \textbf{-with F -do -coding_theoretic_activity \ \textbf{-find_CRC_polynomials 3 128 10 \ \textbf{-end}}}}} \end{align*}
\]

The program finds 244 polynomials in about 1 minute.

Here is a collection of CRC polynomials from various sources:

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

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

**CRC8_ATM**="1,8,1,2,1,1,1,0"

**CRC16_CCITT**="1,16,1,12,1,5,1,0"

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

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

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

**CRC64_ROCKSOFT**="1,64,1,63,1,61,1,59,1,58,1,56,1,55,1,52,1,49,1,48,1,47,1,46,1,44,1,41,1,37,1,36,1,34,1,32,1,31,1,28,1,26,1,23,1,22,1,19,1,16,1,13,1,12,1,10,1,9,1,6,1,4,1,3,1,0"
We test whether the polynomial crc32 is irreducible:

crc32_Berlekamp_matrix:

```
$ (ORBITER) -v 2 \
  -define F -finite_field -q 2 -end \n  -define v -vector -field F -sparse 33 $(CRC32_ETHHERNET) -end \n  -with F -do \n  -finite_field_activity \n  -Berlekamp_matrix v \n  -end
```

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

CRC_F256_roots_771:

```
$ (ORBITER) -v 3 \n  -define F -finite_field -q 256 -end \n  -with F -do -coding_theoretic_activity \n  -nth_roots 771 \n  -end
```

We create a BCH code of length 771 over $F_{256}$ with designed distance 2:

CRC_F256_BCH_code_d2:

```
$ (ORBITER) -v 2 \n  -define F -finite_field -q 256 -end \n  -define C -code -field F \n  -BCH 771 2 \n  -end \n  -with C -and F -do -coding_theoretic_activity \n  -export_magma BCH_lq8_n771_d2.magma \n  -end
```

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

\[ \text{POLY}_{Q256}^{\text{DEG}30}^{\text{DENSE}} = "1,26,210,24,138,148,\
160,58,108,199,95,56,9,205,194,193,3,248,110,\
150,24,169,192,212,112,144,97,109,174,28,253,1" \]

The polynomial in dense coding is:

\[ \text{POLY}_{Q256}^{\text{DEG}30}^{\text{DENSE}} = "1,26,210,24,138,148,\
160,58,108,199,95,56,9,205,194,193,3,248,110,\
150,24,169,192,212,112,144,97,109,174,28,253,1" \]

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

\[ \text{F256_BCH_write_code_for_division_d16:} \]
\[ $(\text{ORBITER}) -v 2 \]
\[ -\text{define F -finite_field -q 256 -end} \]
\[ -\text{define A -vector -field F -sparse 772 "1,771,1,0" -end} \]
\[ -\text{define B -vector -field F -dense $(\text{POLY}_{Q256}^{\text{DEG}30}^{\text{DENSE}})$ -end} \]
\[ -\text{with F -do} \]
\[ -\text{write_code_for_division} \]
\[ -\text{alfa A B} \]
\[ -\text{end} \]
\[ g++ crc.alfa.cpp -o crc.alfa.out \]
\[ ./crc.alfa.out \]
We confirm that the polynomial divides $X^{771} - 1$ as it should:

F256_BCH_code_d16_division:

```bash
$ (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 \n  -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:

```bash
$ (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 \n  -polynomial_division A B -end
```
10.5 Reed-Muller Codes

The following command creates the Reed Muller code RM$_{3,1}$.

```
RM_3_1_Hamming_space_diagram:
  > $(ORBITER) -v 2 \
  >   -define F -finite_field -q 2 -end \n  >   -with F -do \n  >   -coding_theoretic_activity \n  >   -code_diagram "RM_3_1" \n  >   $(REED_MULLER_3_1_CODEWORDS) 8 \n  >   -metric_balls 1 \n  >   -end
```

The following command produces a diagram of the characteristic function of the code in the Hamming space $H(8, 2)$, shown in Figure 10.5. The different codewords are given different colors.

```
RM_3_1_draw:
  > $(ORBITER) -v 2 \
  >   -draw_matrix \n  >   -input_csv_file RM_3_1_holes_8_16.csv \n  >   -box_width 25 -bit_depth 8 \n  >   -partition 4 16 16 \n  >   -end
  > $(ORBITER) -v 2 \
  >   -draw_matrix \n  >   -input_csv_file RM_3_1_diagram_01_8_16.csv \n  >   -box_width 25 -bit_depth 8 \n  >   -partition 4 16 16 \n  >   -end
  > $(ORBITER) -v 2 \
  >   -draw_matrix \n  >   -input_csv_file RM_3_1_diagram_8_16.csv \n  >   -box_width 25 -bit_depth 8 \n  >   -partition 4 16 16 \n  >   -end
  > open RM_3_1_diagram_8_16_draw.bmp
```
Figure 10.5: Boolean function representation of RM_{3,1} in $H(8, 2)$
10.6 BCH Codes

Let $\beta$ be an $n$-th root of unity over $\mathbb{F}_q$. The minimum polynomial of $\beta$ over $\mathbb{F}_q$ is denoted as $m_{\beta, \mathbb{F}_q}$. The BCH code of length $n$ and designed distance $d$ is the cyclic code with generator polynomial

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

To create the polynomial $m_{\beta^a, \mathbb{F}_q}$, we consider the $q$-cyclotomic set of $a$ modulo $n$, which is

$$\{aq^i \mod n | i \in \mathbb{Z}\}.$$

Suppose we want to make a BCH-code of length 21 over $\mathbb{F}_8$. In Section 3.3, we considered the $q$-cyclotomic sets modulo 21 for $q = 8$. Let us produce a pictorial representation. Omitting the singletons, a transversal is given by the sets containing 1, 2, 4, 5, 7, 10, 13. For this reason, we issue the command

draw_cyclotomic_mod_21_q8:

```bash
$\text{(ORBITER)} -v 2 \\
  \hspace{1cm} -\text{draw.options} \\
  \hspace{2cm} -\text{radius 100} \\
  \hspace{3cm} -\text{line_width 1.0 } -\text{embedded} \\
  \hspace{4cm} -\text{end} \\
  \hspace{5cm} -\text{draw_mod_n } -\text{n 21 } -\text{file mod_21_cyclotomic} \\
  \hspace{6cm} -\text{cyclotomic_sets 8 } "1,2,4,5,7,10,13" \ -\text{end} \\
  \hspace{7cm} \text{pdflatex mod_21_cyclotomic_draw.tex} \\
  \hspace{8cm} \text{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:

F_8.BCH_code_d3:

```bash
$\text{(ORBITER)} -v 3 \\
  \hspace{1cm} -\text{define F } -\text{finite_field } -q 8 -\text{override_polynomial 11 } -\text{end} \\
  \hspace{2cm} -\text{with F } -\text{do} \\
  \hspace{3cm} -\text{coding_theoretic_activity} \\
  \hspace{4cm} -\text{make_BCH_code 21 3 } \\
  \hspace{5cm} -\text{end} \\
  \hspace{6cm} \text{pdflatex BCH_codes_q8_n21_d3.tex} \\
  \hspace{7cm} \text{open BCH_codes_q8_n21_d3.pdf}
```

The code is described in a latex output file:
Figure 10.6: The 8-cyclotomic sets modulo 21

BCH-code:
\( n = 21, \ k = 17, \ d_0 = 3, \ q = 8, \)
\( g(x) = m_1m_2 = X^4 + 4X^3 + 4X^2 + 3X + 4 \)
Chosen cyclotomic sets:
\{ 1, 8 \}
\{ 2, 16 \}
The generator polynomial has degree 4

- dense "4,3,4,4,1"
- sparse "4,0,3,1,4,2,4,3,1,4"
The generator matrix is:

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

And now for $d = 5$:

F_8_BCH_code_d5:

$\textbf{\textless}$ (ORBITER) -v 3 \textbf{\texttt{\textbackslash}}
$\textbf{\textless}$ $\textbf{\textgreater}$ -define F -finite_field -q 8 -override_polynomial 11 -end \textbf{\texttt{\textbackslash}}
$\textbf{\textless}$ $\textbf{\textgreater}$ -with F -do \textbf{\texttt{\textbackslash}}
$\textbf{\textless}$ $\textbf{\textgreater}$ $\textbf{\textgreater}$ -coding_theoretic_activity \textbf{\texttt{\textbackslash}}
$\textbf{\textless}$ $\textbf{\textgreater}$ $\textbf{\textgreater}$ $\textbf{\textgreater}$ -make_BCH_code 21 5 \textbf{\texttt{\textbackslash}}
$\textbf{\textless}$ $\textbf{\textgreater}$ -end

pdflatex BCH_codes_q8_n21_d5.tex
open BCH_codes_q8_n21_d5.pdf

The output file is:

BCH-code:

$n = 21, k = 14, d_0 = 5, q = 8,$
$g(x) = m_1m_2m_3m_4 = X^7 + 3X^6 + 3X^5 + 2X^4 + X^3 + 2X^2 + X + 2$

Chosen cyclotomic sets:

\{ 1, 8 \}
\{ 2, 16 \}
The generator polynomial has degree 7

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

The generator matrix is:

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

We compute the minimum distance:

```
F_8_BCH_code_d5_minimum_distance:
> $(ORBITER) -v 2 \ 
> -define F -finite_field -q 8 -override_polynomial 11 -end \ 
> -define v -vector -format 14 -field F \ 
> -compact $(CODE_BCH_F8_N21_D5_GENMA_OVERRIDE_POLYNOMIAL11) \ 
> -end \ 
> -with F -do \ 
> -coding_theoretic_activity \ 
> -minimum_distance v \ 
> -end
```

# important: use the same polynomial as when creating the code.
#
# d=5
The minimum distance turns out to be \( d = 5 \).

Finally, we create the BCH code with minimum distance \( d = 7 \):

\[
\text{F}_8\text{BCH\_code\_d7:} \\
\quad \textsc{\$ (ORBITER) -v 3 \ }
\quad \textsc{\$ -define F -finite\_field -q 8 -override\_polynomial 11 -end \ }
\quad \textsc{\$ -with F -do \ }
\quad \textsc{\$ -coding\_theoretic\_activity \ }
\quad \textsc{\$ -make\_BCH\_code 21 7 \ }
\quad \textsc{\$ -end}
\]

The output file is:

\[
\text{BCH-code:} \\
n = 21, k = 11, d_0 = 7, q = 8, \\
g(x) = m_1m_2m_3m_4m_5m_6 = X^{10} + X^9 + 2X^8 + 5X^7 + 2X^6 + 4X^4 + 6X^3 + 5X^2 + 6X + 6 \\
\text{Chosen cyclotomic sets:} \\
\quad \{ 1, 8 \} \\
\quad \{ 2, 16 \} \\
\quad \{ 3 \} \\
\quad \{ 4, 11 \} \\
\quad \{ 5, 19 \} \\
\quad \{ 6 \} \\
\text{The generator polynomial has degree 10} \\
\quad \text{-dense "6,6,5,6,4,0,2,5,2,1,1"} \\
\quad \text{-sparse "6,0,6,1,5,2,6,3,4,4,2,6,5,7,2,8,1,9,1,10"}
\]
The generator matrix is:

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

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:
\[
(\text{ORBITER}) -v \ 2 \\
\text{-draw\_options -nodes\_empty -radius 10}
\]
\[
\text{-line\_width 0.4 -embedded -end}
\]
\[
\text{-draw\_mod\_n -n 255 -file mod_255_cyclotomic_1_and_3}
\]
\[
\text{-cyclotomic\_sets 2 "1,3" -end}
\]
\[
\text{pdflatex mod_255_cyclotomic_1_and_3\_draw.tex}
\]
\[
\text{open mod_255_cyclotomic_1_and_3\_draw.pdf}
\]

The drawing is shown in Figure 10.7.

Suppose we want to make a BCH-code over $\mathbb{F}_{256}$. In order to keep the degree of the generator polynomial low, we try a quadratic field extension. This way, each cyclotomic set has size either 1 or 2. Since

$$256^2 - 1 = (256 + 1)(256 - 1) = 257 \cdot 3 \cdot 5 \cdot 17,$$

we can consider a code of length $n = 771 = 257 \cdot 3$. The following command computes the 256-cyclotomic cosets modulo 771:

BCH_F256_roots_771:
\[
(\text{ORBITER}) \ -v \ 3 \\
\text{-define F -finite\_field -q 256 -end}
\]
\[
\text{-with F -do -coding\_theoretic\_activity}
\]

318
The next command creates a BCH-code of length 771 over $\mathbb{F}_{256}$ with minimum distance at least 16:

```
$\text{ORBITER} -v 3 -define F -finite_field -q 256 -end -with F -do -coding_theoretic_activity -make BCH_code 771 16 -end pdflatex BCH_codes_q256_n771_d16.tex open BCH_codes_q256_n771_d16.pdf
```
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

```
code_rs_6_4_7="\ 
621000 \ 
062100 \ 
006210 \ 
000621"
```

computes the weight enumerator, which turns out to be

$$(1, 0, 0, 120, 360, 972, 948).$$

In polynomial form, this is

$$1y^6 + 120x^3y^3 + 360x^4y^2 + 972x^5y + 948x^6.$$
This confirms that the minimum distance is three.

Let us consider an example of a Reed-Solomon code in characteristic two: The Reed Solomon code of designed distance 3 over $\mathbb{F}_8$ is the cyclic code generated by

$$(X - \alpha)(X - \alpha^2) = X^2 + 6X + 5.$$ 

The associated cyclic generator matrix is

$$\begin{bmatrix}
5 & 6 & 1 & 0 & 0 & 0 & 0 \\
0 & 5 & 6 & 1 & 0 & 0 & 0 \\
0 & 0 & 5 & 6 & 1 & 0 & 0 \\
0 & 0 & 0 & 5 & 6 & 1 & 0 \\
0 & 0 & 0 & 0 & 5 & 6 & 1
\end{bmatrix}.$$ 

We use the makefile variable `CODE_RS_8` to hold this generator matrix. The following command computes the weight enumerator

```
CODE_RS_8="\n5610000 \n0561000 \n0056100 \n0005610 \n0000561"
```

```
RREF_RS_8_weight Enumerator:
```
```
\$ (ORBITER) -v 2 \n\$ define F -finite_field -q 8 -end \n\$ define v -vector -format 5 -field F \n\$ define C -code -field F \n\$ -compact $(CODE_RS_8) \n\$ -end \n\$ -linear_code_through_generator_matrix v \n\$ -with C -do \n\$ -coding_theoretic_activity \n\$ -weight Enumerator \n\$ -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.

321
The reduced matrix is shown in Figure 10.8. Let us compute the weight enumerator of the reduced code. The command

```bash
RS_8_reduced="\n0100011000000000000000
0011100100000000000000
1100110010000000000000
0000100110000000000000
0000011100100000000000"\n```

322
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 \$
2\$
8\$
reduced) \$
▷ ▷ -end \$
▷ ▷ -define C -code -field F \$
▷ ▷ -linear_code_through_generator_matrix v \$
▷ ▷ -end \$
▷ ▷ -with C -do \$
▷ ▷ -coding_theoretic_activity \$
▷ ▷ ▷ -weight Enumerator \$
▷ ▷ -end

computes the weight enumerator of the binary code. It is

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

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="\n1110010000000000000000 \n1101000100000000000000 \n1011000010000000000000 \n0111000001000000000000 \n1100100000100000000000 \n1010100000100000000000 \n0110100000010000000000 \n0000000000000000000000"
100110000000010000000 \  
010110000000010000000 \  
001110000000001000000 \  
111110000000000100000 \  
110001000000000010000 \  
101001000000000001000 \  
011001000000000001001 \  
100101000000000000001"  

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  

We compute the weight enumerator

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) \  
▷ ▷ ▷ -end \  
▷ ▷ -define C -code -field F \  
▷ ▷ ▷ -linear_code_through_generator_matrix v \  
▷ ▷ ▷ -end \  
▷ ▷ -with C -do \  
▷ ▷ -coding_theoretic_activity \  
▷ ▷ ▷ -weight Enumerator \  
▷ ▷ -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

```
bounds_for_d_given_n15_k6_q2:
  $ (ORBITER) -v 2 -make_bounds_for_d_given_n_and_k_and_q 15 6 2
```

gives upper and lower bounds on the optimal minimum distance $d_{\text{max}}$ 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

```
coding_theory_bounds_q2:
  $ (ORBITER) -v 2 -table_of_bounds 20 2
```

produces a table of bounds for binary codes with $n, k \leq 20$. A file

```
table_of_bounds_n20_q2.csv
```

is computed. The command

```
GV_n15_k6_d5:
  $ (ORBITER) -v 2 -define F -finite_field -q 2 -end -with F -do -coding_theoretic_activity -make_gilbert_varshamov_code 15 6 5 -end
```

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

325
To compute the minimum distance of the code, we do:

```
CODE_GV_N15_K6="\n111111111100000\n111110000010000\n111001100001000\n1101010000100\n10101010000010\n10110100100001"
```

The weight enumerator is

\[ \begin{align*}
1 & y^{15} + 27 x^6 y^9 + 24 x^8 y^7 + 9 x^{10} y^5 + 3 x^{12} y^3.
\end{align*} \]

From this, we see that the code has minimum distance 6, which is better than predicted.
10.9 Classification of Optimal Linear Codes

The classification problem of optimal codes in coding theory is the problem of determining the equivalence classes of codes for a given set of values of \( n \) and \( k \) and \( q \) with a lower bound on \( d \). Orbiter can be used to classify linear codes with given redundancy and bounded minimum distance. The redundancy of a linear \([n, k]\) code is the parameter \( r = n - k \). Codes with redundancy \( r \) can be identified with subsets of \( \text{PG}(r-1, q) \). Under this correspondence, a code with minimum distance at least \( d \) corresponds to a subset such that any \( d-1 \) elements are independent. We use the notation \( \Lambda_{r-1,s}(q) \) to denote the poset of subsets of \( \text{PG}(r-1, q) \) for which any \( d-1 \)-subset (if any) is independent. Under the correspondence, the action of \( \text{PGL}(r, q) \) on \( \Lambda_{r-1,s}(q) \) corresponds to the orbits of equivalent linear codes. For this reason, we are interested in determining the orbits of \( \text{PGL}(r, q) \) on \( \Lambda_{r-1,s}(q) \). An orbit of size \( n \) represents an isometry class of \([n, n-r, d; q]\) codes with \( d \geq s + 1 \). The projective stabilizer of the subset is the automorphism group of the code. The Orbiter command

```plaintext
codes_8_4_4:
  >> $(ORBITER) -v 6 \
  >>   -orbiter_path $(ORBITER_PATH) \ 
  >>   -define G \ 
  >>   -linear_group -PGL 4 2 -end \ 
  >>   -with G -do \ 
  >>   -group_theoretic_activity \ 
  >>   -poset_classification_control \ 
  >>     -problem_label codes_8_4_4 \ 
  >>     -draw_poset \ 
  >>     -draw_options -embedded -radius 250 \ 
  >>     -line_width 1.0 -spanning_tree -end \ 
  >>     -report -end \ 
  >>   -end \ 
  >>   -linear_codes 3 8 \ 
  >>   -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 \( \Lambda_3(2) \). 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.

The command

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

\[
\text{ORBITER} -v 2 -\text{conj\_classes}\_\text{Sym}_n 10
\]

\[
\text{open classes}\_\text{Sym}_\text{10}\text{.csv}
\]

produces a list of the conjugacy classes of Sym(10). The list is written to a csv file. A pie chart of the class size distribution is shown in Fig. 11.1.

The next command computes the character table of the symmetric group Sym(4):

\[
\text{Char}\_\text{Sym}_4:
\]

\[
\text{ORBITER} -v 2 -\text{character\_table\_symmetric\_group} 4
\]

The command produces the following output:

The character table of Sym(4) is the matrix

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

The following command creates the character table of Sym(4).
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-random_permutation</td>
<td>$n$ fname</td>
<td>Creates a random permutation in $\text{Sym}(n)$ and stores it in the given file.</td>
</tr>
<tr>
<td>-create_random_k_subsets</td>
<td>$n$ $k$ $N$</td>
<td>Creates $N$ random $k$-subsets of an $n$-set.</td>
</tr>
<tr>
<td>-read_poset_file</td>
<td>fname</td>
<td>Reads a poset from the given file.</td>
</tr>
<tr>
<td>-read_poset_file_with_grouping</td>
<td>fname x-stretch</td>
<td>Reads a poset from the given file and sets stretch factor for orbit grouping.</td>
</tr>
<tr>
<td>-list_parameters_of_SRG</td>
<td>$v_{\text{max}}$</td>
<td>Performs a sift for putative parameter sets of SRGs.</td>
</tr>
<tr>
<td>-conjugacy_classes_Sym_n</td>
<td>$n$</td>
<td>Compute a list of conjugacy classes of $\text{Sym}(n)$.</td>
</tr>
<tr>
<td>-tree_of_all_k_subsets</td>
<td>$n$ $k$</td>
<td>Creates a tree-file for all $k$-subsets of an $n$-set.</td>
</tr>
<tr>
<td>-Delandtsheer_Doyen</td>
<td></td>
<td>See Section 11.7.</td>
</tr>
<tr>
<td>-tdo_refinement</td>
<td></td>
<td>See Section 11.8.</td>
</tr>
<tr>
<td>-tdo_print</td>
<td></td>
<td>See Section 11.8.</td>
</tr>
<tr>
<td>-convert_stack_to_tdo</td>
<td></td>
<td>See Section 11.8.</td>
</tr>
<tr>
<td>-maximal_arc_parameters</td>
<td></td>
<td>See Section 11.8.</td>
</tr>
<tr>
<td>-arc_parameters</td>
<td></td>
<td>See Section 11.8.</td>
</tr>
<tr>
<td>-pentomino_puzzle</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 11.1: Commands related to Combinatorics (Part 1)
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-draw_layered_graph</td>
<td>options</td>
<td>Draws a graph.</td>
</tr>
<tr>
<td>-make_elementary_symmetric_functions</td>
<td>$n$ $k_{\text{max}}$</td>
<td>Computes the elementary symmetric functions in $n$ variables of degree $1,\ldots,k_{\text{max}}$</td>
</tr>
<tr>
<td>-Dedekind_numbers</td>
<td>$n_{\text{min}}$ $n_{\text{max}}$ $q_{\text{min}}$ $q_{\text{max}}$</td>
<td>Computes the Dedekind numbers $D_{n,q}$ for $n_{\text{min}} \leq n \leq n_{\text{max}}$ and $q_{\text{min}} \leq q \leq q_{\text{max}}$</td>
</tr>
<tr>
<td>-rank_k_subset</td>
<td>$n$ $k$ $L$</td>
<td>Computes the ranks of $k$-subsets chosen from an $n$-set. $L$ is a list of $k$-sets taken from an $n$-set.</td>
</tr>
<tr>
<td>-geometry_builder</td>
<td></td>
<td>See Section 11.4.</td>
</tr>
<tr>
<td>-character_table_symmetric_group</td>
<td>$n$</td>
<td>Computes the character table of $\text{Sym}(n)$ using the algorithm of Burnside.</td>
</tr>
<tr>
<td>-domino_portrait</td>
<td>$D$ $s$ $\text{fname}$</td>
<td>Computes a domino portrait for a graphics file in r/g/b format using double $D$ domino sets.</td>
</tr>
</tbody>
</table>

Table 11.2: Commands related to Combinatorics (Part 2)
Figure 11.1: The conjugacy classes of Sym(10) arranged by size

Char_Sym_4:
▷ $(\text{ORBITER}) -v 2 \ -\text{character}\_\text{table}\_\text{symmetric}\_\text{group} \ 4$

The following command illustrates how to create random $k$-subsets of a set of size $n$. In the example, we create 20 5-subsets of a 10-element set:

\textit{random\_k\_subsets:}
▷ $(\text{ORBITER}) -v 4 \$
▷ ▷ $-\text{create}\_\text{random}\_k\_\text{subsets} \ 10 \ 5 \ 20$

Using the lexicographic order, the $k$-subsets of an $n$-element set are ranked. The following command computes the ranks of a number of 3-subsets of a 10-element set:

\textit{rank\_k\_subsets\_test:}
▷ $(\text{ORBITER}) -v 2 \$
▷ ▷ $-\text{rank}\_k\_\text{subset} \ 10 \ 3 \ 0,1,2,0,3,4,1,3,5,2,4,5,3,6,7,1,6,8,0,6,9$

Orbiter can create the Sylvester type Hadamard matrix of size $2^n$ (also called the Walsh matrix). The following command creates the matrix of size $2^4 \times 2^4$ and produces a graphical representation:
Walsh\_matrix\_4:
\[
\text{\texttt{\$(ORBITER) -v 3 \backslash}}\\
\text{\texttt{ -define F -finite\_field -q 2 -end \backslash}}\\
\text{\texttt{-with F -do -finite\_field\_activity \backslash}}\\
\text{\texttt{-Walsh\_matrix 4 -end}}\\
\text{\texttt{\$(ORBITER) -v 2 -draw\_matrix \backslash}}\\
\text{\texttt{-input\_csv\_file Walsh\_01\_4.csv \backslash}}\\
\text{\texttt{-box\_width 10 -bit\_depth 24 -partition 3 16 16 -end}}\\
\text{\texttt{#pdflatex GF\_2.tex}}\\
\text{\texttt{#open GF\_2.pdf}}
\]

The following command creates the matrix of Dedekind numbers of order at most 10:

\textbf{Dedekind\_10\_10:}
\[
\text{\texttt{\$(ORBITER) -v 3 -Dedekind\_numbers 2 10 2 10}}
\]

The following command creates the elementary symmetric functions in 4 variables.

\textbf{elementary\_symmetric\_functions\_4:}
\[
\text{\texttt{\$(ORBITER) -make\_elementary\_symmetric\_functions 4 4}}
\]

The output is:

\[
\begin{align*}
\text{k=1 :} & \quad x_0 + x_1 + x_2 + x_3 \\
\text{k=2 :} & \quad x_0x_1 + x_0x_2 + x_0x_3 + x_1x_2 + x_1x_3 + x_2x_3 \\
\text{k=3 :} & \quad x_0x_1x_2 + x_0x_1x_3 + x_0x_2x_3 + x_1x_2x_3 \\
\text{k=4 :} & \quad x_0x_1x_2x_3
\end{align*}
\]

Orbiter can compute domino portraits. To do so, we need an input file in r/g/b format of size \((D + 1)s \times Ds\), where \(D = 7\) for double-six dominos.

\textbf{domino\_portrait:}
\[
\text{\texttt{\$(ORBITER) -v 3 -domino\_portrait 7 4 anton\_28x32 -end}}
\]

The portrait is shown in Figure 11.2. It is possible to compare the domino portrait with a grayscale version of the input image. The following command creates a grayscale image of the input file that was written during the previous command.
Figure 11.2: Domino Portrait
Figure 11.3: Domino portrait input file in grayscale

```bash
domino_portrait_input:
  $ (ORBITER) -v 2 \
  -define all_one_r -vector -repeat 1 28 -end \
  -define all_one_c -vector -repeat 1 32 -end \
  -draw_matrix \ 
  -grayscale \ 
  -invert_colors \ 
  -input_csv_file anton_28x32.m.csv \ 
  -box_width 20 -bit_depth 8 \ 
  -partition 3 \ 
  all_one_c all_one_r \ 
  -end \
  open anton_28x32.m_draw.bmp
```

The grayscale version of the input file is shown in Figure 11.3.
11.2 Diophantine Systems

Diophantine systems of equations and inequalities arise frequently in Combinatorics. In Table 11.3, Orbiter commands for creating and solving diophantine systems are shown. In Table 11.4, Orbiter activities for diophantine systems are shown.

Consider the matrix

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

Suppose we want to find all column vectors \( \mathbf{x} \) with entries in \( 0, 1 \) such that

\[ A\mathbf{x} = \mathbf{1} \]

where \( \mathbf{1} \) is the all-one column vector. Orbiter offers two algorithms to do this. One is McKay’s possolve, the other is Knuth’s dancing links (DLX). In order to get started, we need to create a diophant object. In the following example, we use the makefile variable \( \text{TEST\_SYSTEM} \) for the coefficient matrix and \( \text{TEST\_RHS} \) for the right hand side.

\[
\text{TEST\_SYSTEM}="\\
0,1,0,1,0,0, \\
0,0,1,0,1,0, \\
1,0,1,0,0,0, \\
0,1,0,1,0,1, \\
1,0,0,0,0,1, \\
1,0,1,0,0,0, \\
0,1,0,0,1,1
"
\]

\[
\text{TEST\_RHS}="1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1"
\]

\[
solve\_test\_system:
\]
\[>(\text{ORBITER}) -v 4 \]
\[>\text{-define A -vector -format 7 -dense $(\text{TEST\_SYSTEM}) -end} \]
\[>\text{-define D -diophant} \]
\[>\text{-label test\_system} \]
\[>\text{-coefficient\_matrix A} \]
\[>\text{-RHS $(\text{TEST\_RHS})} \]
\[>\text{-x\_min\_global 0 -x\_max\_global 1} \]
\[>\text{-end} \]
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-label</td>
<td>label</td>
<td>Use the given name as file name.</td>
</tr>
<tr>
<td>-coefficient_matrix</td>
<td>A</td>
<td>Set the coefficient matrix to the previously created vector with format information.</td>
</tr>
<tr>
<td>-coefficient_matrix_csv</td>
<td>fname</td>
<td>Read the coefficient matrix from the given csv-file.</td>
</tr>
<tr>
<td>-RHS</td>
<td>list-of-integers</td>
<td>3n values: (RHS-low, RHS-high, RHS-type) for each row of the system.</td>
</tr>
<tr>
<td>-RHS_csv</td>
<td>fname</td>
<td>Read the RHS information from the given csv file.</td>
</tr>
<tr>
<td>-RHS_constant</td>
<td>low,high,type</td>
<td>Set the RHS according to low,high,type.</td>
</tr>
<tr>
<td>-x_max_global</td>
<td>a</td>
<td>Set the upper bound for all variables to a</td>
</tr>
<tr>
<td>-x_min_global</td>
<td>a</td>
<td>Set the lower bound for all variables to a</td>
</tr>
<tr>
<td>-x_bounds</td>
<td>list-of-values</td>
<td>Set the lower and upper bounds for all variables.</td>
</tr>
<tr>
<td>-x_bounds_csv</td>
<td>fname</td>
<td>Read the lower and upper bounds for all variables from the given file.</td>
</tr>
<tr>
<td>-has_sum</td>
<td>s</td>
<td>For the sum of the variables to be s.</td>
</tr>
<tr>
<td>-maximal_arc</td>
<td>s d secants subset</td>
<td>Create system for a maximal arc of size s and degree d in PG(2,q). Use the given set of two pencil lines. The subset picks the lines from the given pencils which are external.</td>
</tr>
<tr>
<td>-q</td>
<td>q</td>
<td>Use PG(2,q) for maximal arcs.</td>
</tr>
<tr>
<td>-override_polynomial</td>
<td>a</td>
<td>Use polynomial numerically coded as a for creating $\mathbb{F}_q$.</td>
</tr>
</tbody>
</table>

Table 11.3: Orbiter Commands to create Diophantine systems
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-print</td>
<td></td>
<td>Print the system.</td>
</tr>
<tr>
<td>-solve_mckay</td>
<td></td>
<td>Solve the system using McKay’s pos-solve.</td>
</tr>
<tr>
<td>-solve_DLX</td>
<td></td>
<td>Solve the system using Knuth’s dancing links.</td>
</tr>
<tr>
<td>-solve_standard</td>
<td></td>
<td>Solve the system using the standard solver.</td>
</tr>
<tr>
<td>-draw</td>
<td></td>
<td>Produce a drawing of the coefficient matrix of the system.</td>
</tr>
<tr>
<td>-draw_as_bitmap</td>
<td>$w \ b$</td>
<td>Produce a bitmap drawing of the coefficient matrix of the system, using boxes of $w$ pixels with. Set the color bit-depth to $b$ ($b = 8$ or $b = 24$). The output is a bmp-file.</td>
</tr>
<tr>
<td>-perform_column_reductions</td>
<td></td>
<td>Eliminate variables which must be zero and write a reduced system.</td>
</tr>
<tr>
<td>-test_single_equation</td>
<td></td>
<td>For each row of the system, compute the number of solutions of the system restricted to the nonzero coefficients.</td>
</tr>
<tr>
<td>-project_to_single_equation_and_solve</td>
<td>$i \ j$</td>
<td>Solve the system assuming the $j$th solution to the restricted system consisting of the $i$th row.</td>
</tr>
<tr>
<td>-project_to_two_equations_and_solve</td>
<td>$i \ j \ r \ m$</td>
<td>Solve the system assuming any solution to the restricted system consisting of the $i$th and the $j$-th row whose number is congruent to $r$ mod $m$.</td>
</tr>
</tbody>
</table>

Table 11.4: Orbiter activities for Diophantine systems
There are two commands to solve a diophantine system: -solve_mckay and -solve_DLX. The latter is more restrictive, as it allows only 0,1 variables. Here is the McKay solver:

```
 McKay_test:
  $ (ORBITER) -v 4 \
  -define A -vector -format 7 -dense $(TEST_SYSTEM) -end \
  -define D -diophant \
  -label test_system \
  -coefficient_matrix A \
  -RHS $(TEST_RHS) \
  -x_min_global 0 -x_max_global 1 \
  -end \
  -with D -do \
  -diophant_activity -solve_mckay \
  -end
```

The solutions are written to the file DLX_test.sol. And now the dancing links solver:

```
 DLX_test:
  $ (ORBITER) -v 4 \
  -define A -vector -format 7 -dense $(TEST_SYSTEM) -end \
  -define D \
  -diophant -label test_system \
  -coefficient_matrix A \
  -RHS $(TEST_RHS) \
  -x_min_global 0 -x_max_global 1 \
  -end \
  -with D -do \
  -diophant_activity -solve_DLX \
  -end
```
11.3 Combinatorial Linear Spaces

A linear space is a pair \((S, \mathcal{L})\) where \(S\) is a set and \(\mathcal{L}\) is a set of subsets of \(S\) such that each set \(L \in \mathcal{L}\) satisfies \(|L| \geq 2\) and moreover for any two \(a, b \in S\) there is exactly one element \(L \in \mathcal{L}\) such that both \(a\) and \(b\) belong to \(L\). The usual notions of isomorphism and automorphism apply. For finite linear spaces, a first combinatorial property is the number \(a_i\) which counts the number of sets \(L \in \mathcal{L}\) of size \(i\). The vector \((a_2, \ldots, a_n)\) is the line type of \((S, \mathcal{L})\). The equation

\[
\binom{n}{2} = \sum_{j=2}^{n} a_j \binom{j}{2}
\]

(11.1)

is satisfied. The equation can be used to generate all possible line types of a putative linear space. Here is an example. For \(|S| = 6\), (11.1) becomes

\[
x_0 \binom{6}{2} + x_1 \binom{5}{2} + x_2 \binom{4}{2} + x_3 \binom{3}{2} + x_4 \binom{2}{2} = \binom{6}{2}.
\]

The extended coefficient matrix of the system is

\[
\begin{bmatrix}
15 & 10 & 6 & 3 & 1 \\
\end{bmatrix}
\]

The Orbiter command

\texttt{linsp6:}

\begin{verbatim}
$\text{(ORBITER)} \ -v \ 4 \ \\
\ -define \ A \ -vector \ -format \ 1 \ -dense \ "15,10,6,3,1" \ -end \ \\
\ -define \ D \ -diophant \ -label \ linsp6 \ \\
\ -coefficient \ matrix \ A \ \\
\ -RHS \ "15,15,1" \ \\
\ -x_min_global \ 0 \ \\
\ -x_max_global \ 15 \ \\
\ -end \ \\
\ -with \ D \ -do \ \\
\ -diophant_activity \ -solve_mckay \ \\
\ -end
\end{verbatim}

# Found 15 solutions with 22 backtrack steps

solves the system using McKay's program possolve [50]. The program finds 15 solutions, written to the file \texttt{linsp6.sol}.

Let us consider a problem from [11]. Suppose we are interested in linear spaces on 30 points. with line type \((7,5^{27},4^{24})\). This notation means that we assume one 7-line, 27 5-lines and 24 4-lines. The type of a point \(P\) is the set of integers

\[p_j = \#j\text{-lines though } P.\]
We are trying to precompute the matrix of point types

\((p_{ij})\)

where \(j = 7, 5, 4\) and \(i\) belongs to an index set of all possible point types. Fixing a point \(P\), counting points \(Q \neq P\) collinear with \(P\) yields

\[
6p_7 + 4p_5 + 3p_4 = 29, \quad p_7 \leq 1, \quad p_5 \leq 27, \quad p_4 \leq 24.
\]

Using the Orbiter commands

\begin{verbatim}
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
\end{verbatim}

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

343
linsp30 pt_distribution:
▷ $(ORBITER) -v 4 \n▷ ▷ -define A -vector -format 6 -dense \n▷ ▷ ▷ "1,1,1,1,1,0,0,5,2,5,2,1,5,3,7,10,1,10,1,0,10,3,21" \n▷ ▷ ▷ -end \n▷ ▷ ▷ -define D -diophant \n▷ ▷ ▷ ▷ -label linsp30 pt_distribution \n▷ ▷ ▷ ▷ -coefficient_matrix A \n▷ ▷ ▷ ▷ -RHS "30,30,1,7,7,1,135,135,1,96,96,1,0,351,2,0,276,2" \n▷ ▷ ▷ ▷ -x_min_global 0 -x_max_global 30 \n▷ ▷ ▷ ▷ -end \n▷ ▷ ▷ -with D -do \n▷ ▷ ▷ ▷ ▷ -diophant_activity -solve_mckay \n▷ ▷ ▷ ▷ ▷ -end \n▷ ▷ ▷ ▷ -with D -do \n▷ ▷ ▷ ▷ ▷ -diophant_activity -draw_as_bitmap 20 8 \n▷ ▷ ▷ -end

we determine the possibilities

$$ (b_0, b_1, b_2, b_3) = \begin{pmatrix} 2 & 5 & 23 & 0 \\ 3 & 4 & 22 & 1 \\ 4 & 3 & 21 & 2 \\ 5 & 2 & 20 & 3 \\ 6 & 1 & 19 & 4 \\ 7 & 0 & 18 & 5 \end{pmatrix} $$
11.4 Classification of Configurations and Geometries

A partial linear space is a set system on a fixed set $V$. We write $\mathcal{L} = (V, \mathcal{B})$, where $\mathcal{B}$ is a set of distinct subsets of $V$, called lines. The members of $V \cup \mathcal{B}$ are called elements. For two elements $x, y$, we say that $x$ is incident with $y$, written $xIy$, if either $x \in y$ or $y \in x$. We require that any line has at least two points and any two points are contained in at most one line. A decomposition of a linear space is a partition $\Pi = (C_1, \ldots, C_n)$ of $V \cup \mathcal{B}$ such that each $C_i$ either is a subset of $V$ or a subset of $\mathcal{B}$. A decomposition is called tactical if for all $i$, the incidence number

$$\iota(C_i, C_j) = |\{ y \in C_j, xIy \}|$$

does not depend on the choice of $x \in C_i$. Any linear space has a tactical decomposition, as the discrete partition (every element is in its own class) is tactical. Let $\text{Aut}(\mathcal{L})$ be the automorphism group of the linear space, which is the subgroup of $\text{Sym}(V)$ which preserves incidence. For $\alpha \in \text{Aut}(\mathcal{L})$ we say that the decomposition $\Pi$ preserves $\alpha$ if $\alpha$ fixes every class of $\Pi$. For $A \leq \text{Aut}(\mathcal{L})$, we say that $\Pi$ preserves $A$ is $\Pi$ preserves every element $\alpha \in A$. Mostly, we are interested in those decompositions $\Pi$ which preserve $\text{Aut}(\mathcal{L})$. In light of this, the discrete decomposition is not that interesting.

Any linear space has a coarsest tactical decomposition that preserves its automorphism group: The orbit partition of the automorphism group acting on $V \cup \mathcal{B}$ will do. Up to ordering of the classes, the coarsest tactical refinement is unique. Computing the orbit decomposition is challenging as it involves computing the automorphism group. Computationally, there are easier ways to get to admissible decompositions. One is by means of successive refinements. If a class $C_i$ does not have the property that $\iota(C_i, C_j)$ is well-defined for all $x \in C_i$, then a refinement of $C_i$ will do. The coarsest refinement of $C_i$ has the property that if $C_i$ preserves some group $A$ then the refinement will do, too. This shows that there is an algorithm to compute a tactical decomposition of any given linear space $\mathcal{P}$. Simply start with the decomposition of two classes, one the set of points and one the set of blocks, and refine. The output may or may not be equal to the decomposition arising from the orbit partition of $\text{Aut}(\mathcal{L})$.

Let us consider the opposite question. Given a tactical decomposition, does there exist a linear space whose coarsest tactical decomposition is the given one? If so, how many nonisomorphic partial linear spaces are there for a given tactical decomposition? in other words, we would like to classify the linear spaces which admit a given tactical decomposition. The -geometry_builder option can answer these kinds of questions. Table 11.5 shows the options for the geometry builder.

The command

```
geo_10_3:
  $\verb|(ORBITER) -v 2 \|
  $\verb|> -define Test_lines -set -loop 4 11 1 -end \|
  $\verb|> -define Geo -geometry_builder \|
  $\verb|> > -V 10 -B 10 -TDO 3 -fuse 1 \|
  $\verb|> > -fname_GEO 10_3 \|
```

345
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-V</td>
<td>part</td>
<td>The initial partition of points (rows).</td>
</tr>
<tr>
<td>-B</td>
<td>part</td>
<td>The initial partition of blocks (columns).</td>
</tr>
<tr>
<td>-TDO</td>
<td>tdo</td>
<td>The initial row-tactical decomposition scheme.</td>
</tr>
<tr>
<td>-fuse</td>
<td>fuse</td>
<td>The partition of row classes.</td>
</tr>
<tr>
<td>-girth_test</td>
<td>g</td>
<td>Require the girth of the collinearity graph to be at least $g$.</td>
</tr>
<tr>
<td>-lambda</td>
<td>$\lambda$</td>
<td>Set $\lambda$ for two-design test. Every pair of points lies in $\lambda$ blocks.</td>
</tr>
<tr>
<td>-find_square</td>
<td></td>
<td>Construct linear spaces.</td>
</tr>
<tr>
<td>-simple</td>
<td></td>
<td>Construct simple designs (needs -lambda)</td>
</tr>
<tr>
<td>-search_tree</td>
<td></td>
<td>Write a file containing the search tree (at the level of rows of the partial geometry).</td>
</tr>
<tr>
<td>-search_tree_flags</td>
<td></td>
<td>Write a file containing the search tree (at the level of flags of the partial geometry). A flag is an incident point-block pair.</td>
</tr>
<tr>
<td>-orderly</td>
<td></td>
<td>User orderly generation.</td>
</tr>
<tr>
<td>-special_test_orderly</td>
<td></td>
<td>Use a special test. This option only applies to orderly generation.</td>
</tr>
<tr>
<td>-split</td>
<td>$l \ r \ m$</td>
<td>Split the search tree. After $l$ lines, continue only cases congruent to $r$ modulo $m$.</td>
</tr>
<tr>
<td>-fname_GEO</td>
<td>fname</td>
<td>Set the output file name base (no extension).</td>
</tr>
<tr>
<td>-output_to_inc_file</td>
<td></td>
<td>Set output to inc file.</td>
</tr>
<tr>
<td>-output_to_sage_file</td>
<td></td>
<td>Set output to sage file.</td>
</tr>
<tr>
<td>-output_to_blocks_file</td>
<td></td>
<td>Set output to a file containing the blocks in coded form.</td>
</tr>
<tr>
<td>-output_to_blocks_latex_file</td>
<td></td>
<td>Set output to a file containing the blocks in latex.</td>
</tr>
</tbody>
</table>

Table 11.5: Orbiter commands to build geometries
classifies the configurations 10_3. It uses isomorphism tests after 4, 5, 6, 7, 8, 9 and 10 points. The positions of the tests is defined using a set called Test_lines. The set of test lines is defined using a loop command. The command shows that there are exactly 10 configurations of this kind. One of them is the Desargues configuration. Four different output files can be written. Each contains all geometries, but the file format is different.

1. The option -output_to_inc_file writes 10_3.inc. The file contains the incidences in increasing order. The position in the incidence matrix is given. One linear space is given per row, except for the first row and the last. The first row contains the number of points, the number of lines, and the number of incidences. The incidences are given in numeric form. The last row start with -1. Each incidence is the numerical position of the point/block pair in the incidence matrix. The position is the numbering of the matrix entries in the incidence matrix in row-major ordering, starting with zero for the top left entry. The index of the incidence in row i (zero-based) and column j is b·i+j, where b is the number of blocks in the geometry. In this case, with b = 10, zero represents the incidence between point 0 and block zero. The number 99 represents the incidence between point 9 and block 9. Here is the file 10_3.inc:

```
10 10 30
0 1 2 10 13 14 20 25 26 31 33 35 41 44 47 52 53 58 62 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 87 89 96 97 98
0 1 2 10 13 14 20 25 26 31 33 35 41 44 47 52 56 58 62 67 69 73 78 79 84 86 89 95 97 98
0 1 2 10 13 14 20 25 26 31 33 35 41 47 48 52 54 57 62 66 68 73 77 79 84 86 89 95 98 99
0 1 2 10 13 14 20 25 26 31 33 35 41 47 48 52 54 57 62 66 68 73 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 78 79 84 86 89 95 97 99
120, 24, 12, 10, 6, 4^2, 3^2, 2
```

2. The option -output_to_sage_file writes 10_3.sage. This file is meant to be read by Sage [64].

3. The option -output_to_blocks_file writes 10_3.blocks. Here is the content of the file 10_3.blocks:

```
10 10 3
0 15 26 44 51 68 81 109 114 116
0 15 26 46 49 68 81 109 114 116
0 15 26 46 49 68 83 106 115 116
0 15 26 46 52 69 77 106 114 116
0 15 26 46 56 69 80 101 106 119
0 15 26 46 56 69 80 103 104 119
```
contains the blocks. Each number represents the rank of a 3-subset corresponding to a block in the lexicographic ordering of all 3-subsets.

4. The option -output_to_blocks_latex_file writes 10_3.blocks_long. The file 10_3.blocks_long contains a list of all blocks written out in a format ready for use in latex.

It is possible to create graphical representations of the search tree. The command below does so for the example that we just did. Note the additional option -search_tree. This option causes Orbiter to create a file containing the search tree. The name of the file is derived from the file name given with the fname_GEO option. Here, the fname_GEO option sets the output file to 10_3. The -search_tree option then creates the file 10_3_tree.txt. In a second invocation of Orbiter, the -tree_draw command is used to draw a tree from the file 10_3_tree.txt that was just created. The vertex color represents the outcome of the isomorphism test. A green node is accepted. A red node is rejected. The search will continue for the green nodes only. A green node at the bottom of the tree corresponds to an isomorphism type of a geometry satisfying all the requirements. Here, the 10 green nodes at the very bottom of the diagram represent the 10 isomorphism types of configurations 10_3.

geo_10_3_tree:
  $(ORBITER) -v 20 \
  $-define Test_lines -set -loop 0 11 1 -end \
  $-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 40 \
  $-paperheight 220 \
  $-paperwidth 330 \
  $-xin 10000 -yin 10000 \
  $-xout 1000000 -yout 500000 \
  $-scale 2 -line_width 0.3 \
  $-nodes_empty \
  $-end \
  $-tree_draw \

348
The size of the tree can be determined by counting the lines of the file \texttt{10\_3\_tree.txt} and subtracting one:

\texttt{wc 10\_3\_tree.txt}

The word count command yields the following output:

\begin{verbatim}
1471 13543 37633 10\_3\_tree.txt
\end{verbatim}

This means that there are 1471 lines in the file. Hence the search tree has 1470 nodes. The resulting tree is shown in Figure 11.4.

Any incidence structure defines a graph on its underlying set of points. The vertices are the points of the incidence structure. Two vertices are adjacent if and only if the incidence structure contains a block which contains the associated points. In a geometric context, the graph is known as the collinearity graph of the geometry. The distance between two points is the distance of the associated vertices in the collinearity graph. The girth is the length of the shortest cycle. It is often desired to classify incidence structures with a given girth. This means that we are given an integer \( g \) (the girth), and that we are looking for incidence structures whose collinearity graph has no cycles of length less than \( g \). For instance, the following example classifies all cubic graphs on 10 vertices with girth at least 5:
geo_petersen:

- $(ORBITER) -v 8 \\
- -define Test_lines -set -loop 3 11 1 -end \\
- -define Geo -geometry_builder \\
- -V 10 -B 15 -TDO 3 -fuse 1 \\
- -fname_GEO petersen -girth 5 \\
- -search_tree \\
- -test Test_lines \\
- -end

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:

- $(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

This command takes about 8 minutes of time to complete.

The next command classifies the configurations $15_3$ with girth 4. Only one configuration arises, the Cremona Richmond, with automorphism group $\text{Sym}(6)$ of order 720.

geo_15_3_g4:

- $(ORBITER) -v 2 \\
- -define Test_lines -set -loop 4 16 1 -end \\
- -define Geo -geometry_builder \\
- -V 15 -B 15 -TDO 3 \\
- -fuse 1 -fname_GEO 15_3_g4 \\
- -girth 4 \\
- -search_tree \\
- -test Test_lines \\
- -end

- $(ORBITER) -v 2 \\
- -draw_options -embedded -radius 50 \\
- -nodes_empty
The next command classifies the configurations 40\textsuperscript{4} with girth 4. Exactly two configuration arise, both with a group of order 51840. Note the extra option \texttt{-special\_test\_not\_orderly} to speed up the search.

\texttt{40\_4\_g4.inc:}
\begin{verbatim}
$(ORBITER) \ -v 5 \ 
\ -define Test\_lines -set -loop 0 41 1 -end \ 
\ -define Geo -geometry\_builder \ 
\ -V 40 -B 40 -TDO 4 \ 
\ -fuse 1 \ 
\ -fname\_GEO 40\_4\_g4 \ 
\ -girth 4 \ 
\ -search\_tree \ 
\ -special\_test\_not\_orderly \ 
\ -test Test\_lines \ 
\ -output\_to\_sage\_file \ 
\ -output\_to\_inc\_file \ 
\ -end
\end{verbatim}

\texttt{geo\textunderscore 63\textunderscore 3\textunderscore g6:}
\begin{verbatim}
$(ORBITER) \ -v 2 \ 
\ -define Test\_lines -set -loop 4 64 1 -end \ 
\ -define Geo -geometry\_builder \ 
\ -V 63 -B 63 -TDO 3 \ 
\end{verbatim}

The search tree is shown in Figure 11.5.
Figure 11.5: The search tree for the configurations $40_4$ with girth 4
The search tree is much larger than for the previous problem.
11.5 Design Theory

A design is an incidence structure of points and blocks. The incidence matrix of a design has rows corresponding to the points and columns corresponding to the blocks. An entry in a certain row and column is one if and only if the point associated with the row is contained in the block associated with the column, zero otherwise. A decomposition of the design is a partition of the points and blocks such that each class consists either exclusively of points or exclusively of blocks.

A decomposition is point-tactical if for all points, the number of incident lines in the \( j \)th block class depends only on the class of the point. If the point belongs to class \( i \), this number is denoted as \( a_{ij} \). A decomposition is block-tactical if for all blocks, the number of incident points in the \( i \)th point class depends only on the class of the block. If the block belongs to class \( j \), this number is denoted as \( b_{ij} \).

A projective plane of order \( n \) is a design with \( n^2 + n + 1 \) points and equally many blocks (also called lines), each of size \( n + 1 \) such that any two points lie in exactly one block and any two blocks have exactly one point in common. Projective planes are known to exist for all \( n = q \) which are a power of a prime. This follows from a construction which utilizes the projective geometry \( \text{PG}(2,q) \). Points are the one-dimensional subspaces of \( \mathbb{F}_q^3 \), blocks are the two-dimensional subspaces of \( \mathbb{F}_q^3 \) and incidence is natural (inclusion of subspaces). The automorphism group of this design is the collineation group of the projective space. Projective planes other than these exist, though none are known when \( n \) is not a prime power. The number of lines through a point equals the number of points on a line. The fact that these numbers exist imply that there is a tactical decomposition. Namely, the trivial decomposition with two classes, one containing all points and one containing all lines. The structure constants of the decomposition are the numbers just described.

The command

```
design PG_2_3:
  > $(ORBITER) -v 8 \
  >   -define F -finite_field -q 3 -end \
  >   -define D -design -field F -family PG_2_q -end \
  >   -with D -do \ 
  >     -design_activity \ 
  >     -export_inc \ 
  >     -end
```

creates the design \( \text{PG}(2,3) \).

We have created the following design:

\[
\{19, 79, 126, 219, 256, 284, 371, 392, 465, 541, 619, 627, 653\}
\]

354
The stabilizer is generated by:
Strong generators for a group of order 5616:

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

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 \(k = 4\)).

In Section 15.4, we will show how to compute further properties of the design.

The command

\[
\text{wreath\_product\_designs\_n4\_k2\_inc.txt: }
\]

\[
\text{\$ (ORBITER) -v 8 \}
\]

\[
\text{\textasciitilde \textasciitilde -define D -design -wreath\_product\_designs 4 2 -end \}
\]

\[
\text{\textasciitilde \textasciitilde -with D -do \}
\]

\[
\text{\textasciitilde \textasciitilde \textasciitilde -design\_activity \}
\]

\[
\text{\textasciitilde \textasciitilde \textasciitilde \textasciitilde -export\_inc \}
\]

\[
\text{\textasciitilde \textasciitilde -end}
\]

creates a design on 8 points invariant under the wreath product \(\text{Sym}(4) \rtimes \text{Sym}(2)\). The design has 12 blocks of size 4. The command

\[
\text{wreath\_product\_designs\_n8\_k6\_inc.txt: }
\]

\[
\text{\$ (ORBITER) -v 8 \}
\]

355
creates a design on 16 points invariant under the wreath product Sym(8)≀Sym(2). The design has 3920 blocks of size 6. We will compute the automorphism groups of these two designs in Section 15.3.

One way to construct designs is by assuming a suitable group of symmetries. Let us consider an example. It is possible to construct \( t-(v,k,\lambda) \) designs invariant under a permutation group \( G \) acting on a set \( V \) with \( |V| = v \) as follows: Classify the orbits of \( G \) on subsets of size \( k \) and less. Construct a matrix which describes the relationship between the orbits on \( t \)-sets and the orbits on \( k \)-sets. This matrix is often referred to as the Kramer-Mesner matrix (cf. [40]). For each pair of \( t \)-orbit and \( k \)-orbit, for instance with representatives \( T \) and \( K \), say, we count the number of elements in the orbit of \( K \) which contain \( T \). The rows of the matrix are in correspondence to the \( t \)-orbits, while the columns are in correspondence to the \( k \)-orbits. The matrix entry \( a_{ij} \) is the number just defined where \( T \) is the representative of the \( i \)-th orbit on \( t \)-sets, and where \( K \) is the representative of the \( j \)-th orbit on \( k \)-sets. Let \( M_{t,k}(G) \) be the Kramer-Mesner matrix for the group \( G \leq \text{Sym}(V) \) defined in this way. The \( t-(v,k,\lambda) \) designs invariant under \( G \) are in one-to-one correspondence to the solutions of

\[
M_{t,k}(G) \cdot x = \lambda 1,
\]

where \( x \) is a column vector of zeros and ones and \( 1 \) is the column vector of all ones. The length of \( x \) is the number of \( k \)-orbits of \( G \) on \( V \), while the length of \( 1 \) is the number of \( t \)-orbits of \( G \) on \( V \). Any vector \( x \) satisfying the matrix equation corresponds to a design invariant under \( G \). Simply take the blocks of the design to be the union of those orbits of \( G \) on \( k \)-subsets whose associated entry in \( x \) is one. We assume the group \( \text{PGL}(2,32) \) in the action on points of the projective line \( \text{PG}(1,32) \) over the field \( \mathbb{F}_{32} \). 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

\[
\text{KM\_PGGL\_2\_32\_KM\_7\_8.csv}.
\]

The second command produces the graphical representation of the matrix shown in Figure 11.6 (different colors represent different values of entries in the matrix). The third Orbiter command creates the diophantine system associated with the Kramer-Mesner matrix.
Figure 11.6: Kramer-Mesner matrix $M_{7,8}(\text{PG}(2,32))$

```plaintext
KM_PGGL_2_32:
  ▶ $(\text{ORBITER}) -v 3 \$
  ▶ ▶ -define Control -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 \$
  ▶ ▶ -define G -linear_group -PGGL 2 32 -end \$
  ▶ ▶ -define Orb -orbits -group G \$
  ▶ ▶ ▶ -on_subsets 8 Control \$
  ▶ ▶ -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 \$
  ▶ pdfflatex 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 \$
```

357
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:
   $(ORBITER) -v 2 \n   <$> $\define Geo -geometry_builder \n   <$> <$> -V 9 -B 12 \n   <$> <$> -TDO 4 -fuse 1 \n   <$> <$> -fname_GEO AG_2_3 \n   <$> <$> -test 3,4,5,6,7,8,9 \n   <$> -end
```

The command produces the file `AG_2_3.inc`, which contains the following lines:

```
9 12 36
0 1 2 3 12 16 18 24 31 32 33 37 40 43 46 49 53 56 59 62 64 69 71 74 78 80 82 87 89 93 94 99 102 103 107
-1 1
432
```

This shows that the design is unique, and has an automorphism group of order 432. For the following commands, we will treat blocks of the design as sets of ranks of $k$-subsets. We can now create a table of all designs $AG(2, 3)$, as orbit under the group $\text{Sym}(9)$. The following command does that:

```
AG_2_3_BLOCKS="0,13,22,27,35,41,47,53,55,59,71,76"
```

```
LS_AG_2_3_design_table_create:
   $(ORBITER) -v 5 \n   <$> <$> -define B -vector -dense $(AG_2_3_BLOCKS) -end \n   <$> <$> -define D -design -list_of_blocks 9 3 B -end \n   <$> <$> -define Sym9 -permutation_group -symmetric_group 9 -end \n   <$> <$> -define T -design_table D "AG_2_3" Sym9 -end
```

The number of designs is $|\text{Sym}(9)|/432 = 362880/432 = 840$. To find all large sets, we establish the block-disjointness graph on this set of designs. After that, we find all cliques of size 7:
LS_AG_2_3_disjoint_sets_graph_and_cliques:

```
$(ORBITER) -v 2 \ 
  > define Gamma -graph \ 
  > > -disjoint_sets_graph \ 
  > > AG_2_3_design_table.csv \ 
  > > -end \ 
  > -with Gamma -do \ 
  > -graph_theoretic_activity \ 
  > > -save \ 
  > -end \ 
  > -with Gamma -do \ 
  > -graph_theoretic_activity \ 
  > > -find_cliques -target_size 7 -end \ 
  > -end \ 
  > -print_symbols
```

The files AG_2_3_design_table_disjoint_sets_sol.txt and AG_2_3_design_table_disjoint_sets_sol.csv are created, each containing the cliques of size 7. There are exactly 15360 cliques of size 7. It remains to classify the resulting 15360 large sets up to isomorphism. To do that, we first need to create the actual large sets from the cliques. The following command does that:

```
LS_AG_2_3_export_solutions:
```

```
 $(ORBITER) -v 20 \ 
  > define B -vector -dense $(AG_2_3_BLOCKS) -end \ 
  > define D -design -list_of_blocks 9 3 B -end \ 
  > define Sym9 -permutation_group -symmetric_group 9 -end \ 
  > define T -design_table D "AG_2_3" Sym9 -end \ 
  > with D -do \ 
  > -design_activity \ 
  > > -extract_solutions_by_index "AG_2_3" Sym9 \ 
  > > AG_2_3_design_table_disjoint_sets_sol.csv \ 
  > > solutions.csv \ 
  > > "" \ 
  > -end
```

The final step to classify the large sets up to isomorphism will be discussed in Section 15.4.
Delandtsheer and Doyen in [23] study line-transitive and point-imprimitive designs and show that they are rare in a certain sense. Orbiter can be used to construct such designs assuming that there is a grid structure on the set of points and assuming that the design is invariant under a chosen group $G$. The group $G$ is assumed to be a subgroup of the group $\text{AGL}(d_1, q_1) \times \text{AGL}(d_2, q_2)$ acting on a grid of size $q_1^{d_1} \times q_2^{d_2}$ in product action.

Finite projective planes often arise in this context. However, not all examples are projective planes. Orter can help to classify small examples. Let us consider an example. Suppose we want to classify all designs on 21 points with blocks of size $k = 5$ invariant under a cyclic group of order 21 preserving a grid of type $3 \times 7$. To this end, we consider the group $\text{AGL}(1, 3) \times \text{AGL}(1, 7)$. The subgroup is generated by the map

$$(\tau_1, \tau_2), \mathbb{Z}_3 \times \mathbb{Z}_7 \to \mathbb{Z}_3 \times \mathbb{Z}_7,$$

where

$$\tau_1: \mathbb{Z}_3 \to \mathbb{Z}_3, \ x \mapsto x + 1 \mod 3, \quad \tau_2: \mathbb{Z}_7 \to \mathbb{Z}_7, \ y \mapsto y + 1 \mod 7.$$

With blocks of size 5, we cover 10 pairs each. The group of order 21 allows to cover each of the $210 = \binom{21}{2}$ pairs exactly once using a single orbit of a block. The question remains to construct all blocks and to classify the resulting designs. The Desarguesian plane $\text{PG}(2, 4)$ provides a solution. The question is to decide whether there are any other, nonisomorphic designs. The following Orbiter commands can be used:

```
PP4= -d1 1 -q1 3 -d2 1 -q2 7 -K 5 -search_control -W -end -problem_label PP4

PP4_GROUP1=-subgroup "1,1,1,1, " "21" -group_label "cyclic21"

PP4_MASK1=\n  > -nb_orbits_on_blocks 1 \n  > -depth 5 \n  > -mask_label "no_mask"
```

The command $\text{DD\_PP4}$ sets up the orbits of the group on pairs and writes the file $\text{PP4\_pair\_covering.csv}$.

```
DD_PP4:
  > $(ORBITER) -v 6 \n  > > -Delandtsheer_Doyen $(PP4) $(PP4_GROUP1) $(PP4_MASK1) \n  > > > -end \n```

The command $\text{DD\_PP4\_system}$ creates a diophantine system of Steiner type and solves it.
DD_PP4_system:

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

-define D -diophant -label PP4 \$
-problem_of_Steiner_type 10 PP4.pair_covering.csv \$
-has_sum 1 \$
-end \$
-with D -do \$
-diophant_activity -solve_mckay \$
-end

It finds exactly one solution. This must be the PG(2, 4) design. Since there are no more designs, isomorphism testing is not needed.
11.8 Tactical Decompositions

Table 11.6 lists the Orbiter commands for decomposition refinement.

Suppose we want to study projective planes of order 16. It is a linear space with \(16^2 + 16 + 1 = 273\) points and equally many lines. Each point lies on 17 lines and each line contains 17 points. Any two points lie on exactly one line and any two lines intersect in exactly one point.

We decide to study maximal arcs of degree 4 in this plane (the degree has to divide the order of the plane). A maximal arc of degree \(d\) is a set of points so that each line intersects in either \(d\) or zero points. A line which intersects in \(d\) points is called a secant. A line which intersects in no point is called an external line. The command

\[
\text{max}_\text{arc}_\text{16}_\text{4}_\text{start}: \quad > \quad \$(\text{ORBITER}) -v 4 \text{-maximal}_\text{arc}_\text{parameters} 16 4
\]

creates a decomposition stack for the parameters of the arc and writes the file \text{max}_\text{arc}_\text{q16}_\text{r4}.stack

\[
\begin{array}{cccc}
\text{HTDO type=pt ptanz=2 btanz=2 fuse=simple} \\
221 & 52 \\
52 & 17 & 0 \\
221 & 13 & 4 \\
1 & 1 \\
\end{array}
\]

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

\[
\text{max}_\text{arc}_\text{16}_\text{4}_\text{convert}_\text{stack}_\text{tdo}: \quad > \quad \$(\text{ORBITER}) -v 4 \text{-convert}_\text{stack}_\text{to}_\text{tdo} \text{ max}_\text{arc}_\text{q16}_\text{r4}.stack
\]

does that. It creates the file \text{max}_\text{arc}_\text{q16}_\text{r4}.tdo. It also prints the decomposition stack:

\[
\begin{array}{l}
\text{lambda}_\text{scheme at level 2 : } \\
is 1 \times 1 \\
| 273_{\{ 1\}} \\
\text{-------------------} \\
273_{\{ 0\}} |
\end{array}
\]

363
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-lambda3</td>
<td>$\lambda_3$ s</td>
<td>Refine as 3-design with $\lambda_3$ and with block-size $s$</td>
</tr>
<tr>
<td>-solution</td>
<td>$i$ fname</td>
<td>Use solutions to system $i$ from file fname.</td>
</tr>
<tr>
<td>-range</td>
<td>$f$ $l$</td>
<td>Refine cases $i$ with $f \leq i &lt; f+l$ only.</td>
</tr>
<tr>
<td>-select</td>
<td>label</td>
<td>Select the case for refinement by label.</td>
</tr>
<tr>
<td>-o1</td>
<td>$s$</td>
<td>Omit $s$ variables from the first refinement system.</td>
</tr>
<tr>
<td>-o2</td>
<td>$s$</td>
<td>Omit $s$ variables from the second refinement system.</td>
</tr>
<tr>
<td>-D1_upper_bound_x0</td>
<td>$b$</td>
<td>Add the bound $x_0 \leq b$ in the first refinement.</td>
</tr>
<tr>
<td>-reverse</td>
<td></td>
<td>Sort the distributions in reverse order.</td>
</tr>
<tr>
<td>-reverse_inverse</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-nopacking</td>
<td></td>
<td>Do not use packing inequalities.</td>
</tr>
<tr>
<td>-dual_is_linear_space</td>
<td></td>
<td>Assume that the dual incidence structure is a linear space also. This is valid for projective planes, for instance.</td>
</tr>
<tr>
<td>-geometric_test</td>
<td></td>
<td>Subject the distributions to the geometric test.</td>
</tr>
<tr>
<td>-once</td>
<td></td>
<td>Find at most one refinement in each case. This can be used to test which cases can be refined.</td>
</tr>
<tr>
<td>-mckay</td>
<td></td>
<td>Use McKay’s solver instead (by default, a lexicographic solver is used).</td>
</tr>
<tr>
<td>-input_file</td>
<td>fname</td>
<td>Specify the input TDO-file for refinement.</td>
</tr>
</tbody>
</table>

Table 11.6: TDO refinement options
Next, we can compute all coarsest column-tactical refinements of the decomposition. To this end, the command

```
max_arc_16_4_refine:
▷ $(ORBITER) -v 4 -tdo_refinement \
▷ ▷ -input_file max_arc_q16_r4.tdo -dual_is_linear_space -end
▷
max_arc_16_4r_print:
▷ $(ORBITER) -v 4 -tdo_print max_arc_q16_r4r.tdo
```

is used. Because the incidence structure is a projective plane, the dual is a linear space also. Hence the option `-dual_is_linear_space` can be used, which is helpful to reduce possibilities. As it turns out, there is exactly one refinement, and it is tactical. The file `max_arc_q16_r4r.tdo` is produced. Note the added letter `r` at the end of the file name (r for refinement). We can use the following command to display the decomposition stack in the file:

```
max_arc_16_4r_print:
▷ $(ORBITER) -v 4 -tdo_print max_arc_q16_r4r.tdo
```

This produces the following output:

```
decomposition 0.1:
lambda_scheme at level 2 :
is 1 x 1
   | 273_{ 1}|
-------------------------
273_{ 0} | 17 |

row_scheme at level 4 :
is 2 x 2
   | 221_{ 1} 52_{ 2}|
-------------------------
52_{ 0} | 17 0|
221_{ 3} | 13 4|
```

```
| 52_{0} | 17 | 0 |
| 221_{3} | 13 | 4 |

**col_scheme at level 4:**
is 2 x 2

| 221_{1} | 52_{2} |

--------------------

| 52_{0} | 4 | 0 |
| 221_{3} | 13 | 17 |

**extra_col_scheme at level 3:**
is 1 x 2

| 221_{1} | 52_{2} |

--------------------

| 273_{0} | 17 | 17 |
Chapter 12

Finite Geometry

12.1 Spreads

A $t$-spread of $\text{PG}(n,q)$ is a set of disjoint $\text{PG}(t,q)$ that cover all of $\text{PG}(n,q)$ pointwise. $t$-spreads in $\text{PG}(n,q)$ exist if and only if $t+1$ divides $n+1$. In order to create a spread, Orbiter offers several commands, as summarized in Table 12.1. The following two commands create the two spreads of order 9, relying on the Orbiter knowledge base.

**create** _spread_ 9a:

```
$ (ORBITER) -v 3 \
-define F -finite_field -q 3 -end \
-define G -linear_group -PGL 4 F -end \
-define S -spread -kernel_field F \
-group G -k 2 -catalogue 0 \
-end
```

**create** _spread_ 9b:

```
$ (ORBITER) -v 3 \
-define F -finite_field -q 3 -end \
-define G -linear_group -PGL 4 F -end \
-define S -spread -kernel_field F \
-group G -k 2 -catalogue 1 \
-end
```

The first spread is the Desarguesian spread, with automorphism group of order 5760. The second spread is the Hall spread with automorphism group of order 1920.

Spreads can be defined using spread sets. A spread set is a set of $q^k$ matrices of size $k \times k$ over $\mathbb{F}_q$ such that $A_i - A_j$ is nonsingular for all $i \neq j$. Let us look at an example. The spread due to Rao and Rao [54] can be defined using the following makefile variable.
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-kernel_field</td>
<td>$F$</td>
<td>Define the kernel of the spread. $F$ must be an object of type finite field.</td>
</tr>
<tr>
<td>-group</td>
<td>$G$</td>
<td>Define the group acting on the spread. Should be $\text{PGL}(2k; F)$.</td>
</tr>
<tr>
<td>-k</td>
<td>$k$</td>
<td>Set the dimension of the spread.</td>
</tr>
<tr>
<td>-catalogue</td>
<td>$i$</td>
<td>Pull spread number $i$ from the catalogue of spreads associated with the given field and the given dimension.</td>
</tr>
<tr>
<td>-family</td>
<td>$L$</td>
<td>Define a spread from a named family $L$. So far, no family has been provided.</td>
</tr>
<tr>
<td>-spread_set</td>
<td>$S$</td>
<td>Define a spread from the named spread-set $S$. The spreadset $S$ must be a vector object. It must contain $q^3 k^2$ entries over $F$.</td>
</tr>
</tbody>
</table>

Table 12.1: Orbiter commands to define a spread

```
SPREAD_SET_27_RAO_RAO="\n 0,0,0,0,0,0,0,0,0, \n 1,1,0,2,1,1,0,0,2, \n 1,0,1,2,2,0,1,0, \n 1,2,2,1,2,0,2,2,2, \n 0,0,2,2,2,0,1,2,0, \n 1,1,2,0,2,1,2,1,0, \n 0,1,0,1,0,1,0,2,1, \n 2,0,2,0,2,1,1,0, \n 2,2,2,0,1,1,0,1,2, \n 2,0,0,1,0,2,1,2,1, \n 2,2,2,2,2,2,0,2, \n 2,1,2,0,2,0,2,0,1, \n 0,1,2,2,0,1,0,1,1, \n 1,0,0,0,1,0,0,0,1, \n 2,1,0,1,2,1,0,2,0, \n 0,2,0,2,2,1,1,2, \n 0,0,1,0,1,2,2,2,1, \n 2,0,1,2,2,1,1,0,1, \n 0,1,1,1,0,1,2,2, \n 2,2,0,2,0,0,2,2, \n 2,1,1,1,1,2,2,1,2, \n 2,2,1,2,1,0,2,0,0, \n 1,2,0,2,0,2,1,0,0, \n 368
```
Each line represents one matrix of the spread set, with matrix entries being listed consecutively. The following command can be used to define the spread:

```bash
create_spread_Rao_Rao_27:
  $\text{ORBITER} -v 3 \\
  -define F -finite_field -q 3 -end \\
  -define SS -vector -dense $(SPREAD_SET_27_RAO_RAO) -end \\
  -define G -linear_group -PGL 6 F -end \\
  -define S -spread -kernel_field F \\
  -group G -k 3 -spread_set SS \\
  -end
```

The following command creates the Desarguesian line-spread in $\text{PG}(3,2)$:

```bash
desarguesian_spread_in_PG_3_2:
  $\text{ORBITER} -v 3 \\
  -define FQ -finite_field -q 4 -end \\
  -define Fq -finite_field -q 2 -end \\
  -with FQ -and Fq -do -finite_field_activity \\
  -group G -k 3 -spread_set SS \\
  -end
```

The cheat sheet contains the following spread:

| Spread element 0 is (1, 0) = | \[ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \end{bmatrix} \] |
| Spread element 1 is (0, 1) = | \[ \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \end{bmatrix} \] |
| Spread element 2 is (1, 1) = | \[ \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ \end{bmatrix} \] |
| Spread element 3 is (2, 1) = | \[ \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ \end{bmatrix} \] |
Spread element 4 is \((3, 1) = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}\)^{22}

Spread elements by rank: \((0, 34, 9, 17, 22)\).

The following command creates the Desarguesian plane-spread in \(\text{PG}(5, 2)\):

\[
\text{desarguesian\_spread\_in\_PG\_5\_2:}
\]
\[
\text{	exttt{\$\{ORBITER\} -v 3 \}}
\]
\[
\text{	exttt{\$\{ORBITER\} -v 3 \}}
\]
\[
\text{	exttt{\$\{ORBITER\} -define FQ -finite\_field -q 8 -end \}}
\]
\[
\text{	exttt{\$\{ORBITER\} -define Fq -finite\_field -q 2 -end \}}
\]
\[
\text{	exttt{\$\{ORBITER\} -with FQ -and Fq -do -finite\_field\_activity \}}
\]
\[
\text{	exttt{\$\{ORBITER\} -cheat\_sheet\_desarguesian\_spread 2 -end \}}
\]
\[
\text{	exttt{pdflatex Desarguesian\_Spread\_5\_2.tex \}}
\]
\[
\text{	exttt{open Desarguesian\_Spread\_5\_2.pdf \}}
\]

Spread element 0 is \((1, 0) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}\) \((0)\)

Spread element 1 is \((0, 1) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}\) \((1394)\)

Spread element 2 is \((1, 1) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}\) \((189)\)

Spread element 3 is \((2, 1) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}\) \((671)\)

Spread element 4 is \((3, 1) = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}\) \((562)\)

Spread element 5 is \((4, 1) = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}\) \((1040)\)
Two $t$-spreads are isomorphic if there is a collineation which maps one to the other. The classification problem for $t$-spreads is the problem of determining a complete set of pairwise non-isomorphic $t$-spreads. The problem is computationally difficult. Orbiter can be used to classify spreads for small parameters. For greater classification power, the method of classification by substructure is used. Let us look at some examples.

At first, we look at an example which is sufficiently small and can be solved using the standard method. Here, the standard method is poset classification algorithm for partial spreads. Suppose we want to classify the line spreads in $\text{PG}(3, 4)$ under the action of $\text{PGL}(4, 4)$. Under the André, Bruck-Bose construction [3, 16], these spreads correspond to translation planes of order 16 with kernel $\mathbb{F}_4$. In order to classify the spreads of $\text{PG}(3, 4)$, we use the command

```
classify_spreads_16_4:
  $ \text{ORBITER} -v 4 \$
  $-\text{define F -finite_field -q 4 -end} \$
  $-\text{define P -projective_space -n 3 -field F -v 0 -end} \$
  $-\text{define C -spread_classifier} \$
  $-\text{projective_space P} \$
  $-k 2 \$
  $-\text{starter_size 17} \$
  $-\text{poset_classification_control} \$
  $-\text{draw_options} \$
  $-\text{radius 20} \$
  $-\text{nodes_empty} \$
  $-\text{line_width 0.2} \$
  $-\text{embedded} \$
  $-\text{end} \$
  $-\text{draw_poset} \$
  $-\text{problem_label spreads_16_4} \$
```
The command uses poset classification to classify the spreads. To this end, it computes the poset of orbits for the group $G = \text{PGL}(4,4)$ acting on the poset of partial spreads in $\text{PG}(3,4)$, shown in Figure 12.1. Up to isomorphism, there are exactly three line-spreads in $\text{PG}(3,4)$ (corresponding to the three nodes at the bottom of the poset of orbits in the figure). These three spreads are the dearguesian spread, the Hall spread, and the semifield spread, respectively. Here is the relevant output taken from the latex report:

There are 3 orbits at level 17.

**Orbit 0 / 3 at Level 17**

Node number: 1126

\[\{0, 25, 50, 75, 90, 107, 122, 140, 144, 157, 179, 204, 213, 238, 268, 334, 345\}_{1200}\]

Strong generators for a group of order 1200:

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

1,0,0,0,0,0,1,0,0,2,3,0,2,1,1,3,2,0, 
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
Figure 12.1: The poset of orbits of partial spreads in $PG(3,4)$
Orbit 1 / 3 at Level 17

Node number: 1127

\{0, 25, 50, 75, 90, 107, 140, 157, 179, 204, 213, 238, 265, 282, 299, 316, 356\}_{81600}

Strong generators for a group of order 81600:

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

1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,3,0,0,0,0,0,3,0,0,0,0,1,1,0,0,0,0,0,2,3,0,0,1,1,0,1,0,0,0,0,2,3,0,0,1,1,0,1,0,0,0,0,2,3,2,2,0,1,0,0,3,1,0,0,0,0,0,1,0,0,3,2,1,1,0,0,0,0,1,0,0,2,0,0,0,0,1,2,1,0,1,1,0,2,0,1,1,0,0,2,1,0,1,0,0,2,0,0,0,2,0,0,0,1,2,3,2,1,1,1,1,0,0,0,1,0,0,2,1,0,1,2,3,2,1,1,1,1,0
There are 0 extensions
Number of generators 7

Orbit 2 / 3 at Level 17

Node number: 1128

\{0, 25, 50, 75, 90, 108, 122, 140, 158, 183, 199, 217, 233, 250, 268, 312, 345\}_{576}
Let us now look at a more difficult problem. We wish to classify the spreads in $\text{PG}(3,5)$. To this end, we will use the method of classification by substructure. We pick a size $s$ of a partial spread, and classify all partial spreads of size $s$. These are the substructures. Next, we perform the lifting, which means we construct all spreads of $\text{PG}(3,5)$ containing one of the orbit representatives of the substructures. In a final step, we perform an isomorph classification on the set of liftings. This will furnish the desired classification of spreads of $\text{PG}(3,5)$. From a computational point of view, the lifting process is the bottleneck in this procedure. Because of this, we use specialized algorithms from graph theory, which enhance the performance of the lifting. Specifically, we perform a search for rainbow cliques. We will go over some examples to illustrate the technique. To begin with, we choose the parameter $s = 5$.

Table 12.2: Spreads in $\text{PG}(3,4)$ in the Orbiter Catalogue

| OCN | $|\text{Aut}|$ | Name            |
|-----|--------------|----------------|
| 0   | 1200        | Hall spread    |
| 1   | 81600       | Desarguesian spread |
| 2   | 576         | Semifield spread |

Strong generators for a group of order 576:

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

There are 0 extensions

Number of generators 6

The three spreads in $\text{PG}(3,4)$ can be distinguished by their stabilizer orders. Table 12.2 lists the line spreads in $\text{PG}(3,4)$ according to their orbiter catalogue number (OCN).
The command

```bash
classify_spreads_25_starter_lift_case_0:
  ▶ $(ORBITER) -v 3 \\
  ▶ ▶ -define F -finite_field -q 5 -end \\
  ▶ ▶ -define P -projective_space -n 3 -field F -v 0 -end \\
  ▶ ▶ -define C -spread_classifier \\
  ▶ ▶ ▶ -projective_space P \\
  ▶ ▶ ▶ -k 2 \\
  ▶ ▶ ▶ -starter_size 5 \\
  ▶ ▶ ▶ -recoordinatize \\
  ▶ ▶ ▶ -poset_classification_control \\
  ▶ ▶ ▶ ▶ -draw_options \\
  ▶ ▶ ▶ ▶ ▶ -radius 20 \\
  ▶ ▶ ▶ ▶ ▶ -nodes_empty \\
  ▶ ▶ ▶ ▶ ▶ -line_width 0.2 \\
  ▶ ▶ ▶ ▶ ▶ -embedded \\
  ▶ ▶ ▶ ▶ ▶ -end \\
  ▶ ▶ ▶ ▶ -W \\
  ▶ ▶ ▶ ▶ -draw_poset \\
  ▶ ▶ ▶ ▶ ▶ -problem_label_spreads_25 \\
  ▶ ▶ ▶ ▶ -end \\
  ▶ ▶ ▶ -output_prefix "" \\
  ▶ ▶ -end \\
  ▶ -with C -do -spread_classify_activity \\
  ▶ -compute_starter \\
  ▶ ▶ -problem_label_spreads_25 \\
  ▶ ▶ -W -depth 5 \\
  ▶ ▶ -report -end \\
  ▶ ▶ -end \\
  ▶ -end \\
  ▶ -with C -do -spread_classify_activity \\
  ▶ ▶ -prepare_lifting_single_case 0 \\
  ▶ ▶ -end
```

classifies the partial spreads of size $s = 5$ and prepares for the lifting of the first case only. In order to prepare for the lifting, a graph is constructed which describes the lines that can be added to the first partial spread. The vertices of the graph are the lines disjoint from the initial set of 5 lines in the partial spread. Two vertices are joined by an edge of the associated lines are disjoint. The vertices of the graph are colored according to the very first basis vector in the generator matrix of the subspace in reduced row echelon form. In order to find the rainbow clique in the graph, the command

```bash
spreads_25_starter_0_cliques:
  ▶ $(ORBITER) -v 2 \\
```

376
can be used.

The command

```bash
classify_spreads_25_starter_lift_all_cases:
```

```bash
$ (ORBITER) -v 3 \
  -define F -finite_field -q 5 -end \
  -define P -projective_space -n 3 -field F -v 0 -end \
  -define C -spread_classifier \
  -projective_space P \
  -k 2 \
  -starter_size 5 \
  -recoordinatize \
  -poset_classification_control \
  -draw_options \
  -radius 20 \
  -nodes_empty \
  -line_width 0.2 \
  -embedded \
  -end \
  -W \
  -draw_poset \
  -problem_label spreads_25 \
  -end \
  -output_prefix "" \
  -end \
  -with C -do -spread_classify_activity \
  -compute_starter \
  -problem_label spreads_25 \
  -W -depth 5 \
  -report -end \
  -end \
  -end \
  -with C -do -spread_classify_activity \
  -prepare_lifting_all_cases \
  -end
```

recomputes the partial spreads of size $s = 5$ and prepares for the lifting of all orbit representatives (there are 28). This leads to 28 graphs, each of which is written to a file. The next
command performs the rainbow clique finding in each of the 28 graphs:

```
spreads_25_starter_cliques:
  ▶ $(ORBITER) -v 2 \n  ▶ ▶ -loop L 0 29 1 \n  ▶ ▶ ▶ -define G -graph -load spreads_25_graph_%L.bin -end \n  ▶ ▶ ▶ -with G -do \n  ▶ ▶ ▶ ▶ -graph_theoretic_activity \n  ▶ ▶ ▶ ▶ ▶ -find_cliques -rainbow -target_size 21 -end \n  ▶ ▶ ▶ ▶ -end \n  ▶ ▶ ▶ -end_loop
```

The resulting cliques are again stored in files. The command

```
classify_spreads_25_isomorph:
  ▶ $(ORBITER) -v 3 \n  ▶ ▶ -define F -finite_field -q 5 -end \n  ▶ ▶ -define P -projective_space -n 3 -field F -v 0 -end \n  ▶ ▶ -define C -spread_classifier \n  ▶ ▶ ▶ -projective_space P \n  ▶ ▶ ▶ -k 2 \n  ▶ ▶ ▶ -starter_size 5 \n  ▶ ▶ ▶ -recoordinatize \n  ▶ ▶ ▶ ▶ -poset_classification_control \n  ▶ ▶ ▶ ▶ ▶ -draw_options \n  ▶ ▶ ▶ ▶ ▶ ▶ -radius 20 \n  ▶ ▶ ▶ ▶ ▶ ▶ -nodes_empty \n  ▶ ▶ ▶ ▶ ▶ ▶ -line_width 0.2 \n  ▶ ▶ ▶ ▶ ▶ ▶ -embedded \n  ▶ ▶ ▶ ▶ ▶ -end \n  ▶ ▶ ▶ ▶ -W \n  ▶ ▶ ▶ ▶ -draw_poset \n  ▶ ▶ ▶ ▶ ▶ -problem_label spreads_25 \n  ▶ ▶ ▶ ▶ ▶ -end \n  ▶ ▶ ▶ ▶ -output_prefix "" \n  ▶ ▶ ▶ -end \n  ▶ ▶ ▶ -with C -do -spread_classify_activity \n  ▶ ▶ ▶ ▶ -compute_starter \n  ▶ ▶ ▶ ▶ ▶ -problem_label spreads_25 \n  ▶ ▶ ▶ ▶ ▶ -W -depth 5 \n  ▶ ▶ ▶ ▶ ▶ -report -end \n  ▶ ▶ ▶ ▶ -end \n  ▶ ▶ ▶ -end \n  ▶ ▶ ▶ -with C -do -spread_classify_activity \n  ▶ ▶ ▶ ▶ -isomorph \n```

378
performs the final isomorph rejection on the spreads arising from the rainbow cliques in all cases. It results in a transversal of the isomorphism classes of spreads of $\text{PG}(3, 5)$. In total, 21 spreads are found. Of course, this agrees with the results in the literature, see [22].

Table 12.3 lists the solid spreads in $\text{PG}(7, 2)$ according to their Orbiter catalogue number (OCN).

| OCN | $|\text{Aut}|$ | Name |
|-----|-------------|------|
| 0   | 1008        |      |
| 1   | 1008        |      |
| 2   | 1728        |      |
| 3   | 216         |      |
| 4   | 360         |      |
| 5   | 288         |      |
| 6   | 3600        |      |
| 7   | 244800      |      |

Table 12.3: Spreads in $\text{PG}(7, 2)$ in the Orbiter Catalogue
12.2 Translation Planes

Orbiter can create translation planes from spreads. The construction of translation planes from spreads is due to André and Bruck, Bose (cf. [3, 16]). In order to perform the construction, we need a field $F = \mathbb{F}_q$ which is the kernel of the plane, a spread of $k$-subspaces, and the groups $\text{PGL}(2k; F)$ and $\text{PGL}(2k + 1, F)$. For instance, the command

create_translation_plane_9b:

```
$\text{ORBITER} -v 3 \n\text{define } F \text{-finite_field -q 3 -end} \n\text{define } G \text{-linear_group -PGL 4 F -end} \n\text{define } G_1 \text{-linear_group -PGL 5 F -end} \n\text{define } S \text{-spread -kernel_field F} \n\text{group } G \text{-k 2 -catalogue 1} \n\text{end} \n\text{define } T \text{-translation_plane S G G_1 -end} \n\text{with } T \text{-do -translation_plane_activity} \n\text{-export incma} \n\text{end} \n\text{with } T \text{-do -translation_plane_activity} \n\text{-report} \n\text{end} \n\text{define } A \text{-linear_group -import_group_of_plane T -end} \n\text{define } Orb \text{-orbits -group A} \n\text{end} \n\text{with } Orb \text{-do -orbits_activity} \n\text{-report} \n\text{end} \n\text{with } Orb \text{-do -orbits_activity} \n\text{-stabilizer 92} \n\text{end} \n\text{with } Orb \text{-do -orbits_activity} \n\text{-export_trees} \n\text{end} $\text{ORBITER} -v 2 \n\text{-draw_matrix} \n\text{-input_csv_file plane_catalogue_q3_k2_1_incma.csv} \n\text{-box_width 6 -bit_depth 8} \n\text{-partition 2 91 91} \n\text{-end} $\text{ORBITER} -v 3 \n\text{-draw_layered_graph} \n\text{orbit_PGL_5.3_on_andre_3.layered_graph} \n\text{-radius 250 -spanning_tree -embedded -nodes_empty} \n```
creates the (projective) Hall plane of order 9 from the Hall spread. In this example, we use the fact that $\text{PGL}(n,q) = \text{PGL}(n,q)$ if $q$ is prime. The example also creates a bitmap drawing of the incidence matrix of the plane, shown in Figure 12.2.

In the next example, we create a translation plane of order 16 with kernel of order 4:

```bash
create_translation_plane_16_4_0:
$ (ORBITER) -v 3 \
define F -finite_field -q 4 -end \
define G -linear_group -PGGL 4 F -end \
```
This plane is the Hall plane, and the spread is the Hall spread. The spread has a stabilizer of order 1200.

In the next example, we create a translation plane of order 16 with kernel of order 2:

```plaintext
create_translation_plane_16_2_0:
  $(ORBITER) -v 3 \
  -define F -finite_field -q 2 -end \
  -define G -linear_group -PGL 8 F -end \
  -define G1 -linear_group -PGL 9 F -end \
  -define S -spread -kernel_field F \
  -group G -k 4 -catalogue 0 \
  -end \
  -define T -translation_plane S G G1 -end \
  $(ORBITER) -v 2 \
  -draw_matrix \
  -input_csv_file plane_catalogue_q2_k4_0_incma.csv \
  -box_width 6 -bit_depth 8 \
  -partition 4 273 273 \
  -end \
  open plane_catalogue_q2_k4_0_incma_draw.bmp
```

The spread has a stabilizer of order 1008, which means that the associated translation plane has a stabilizer of order $1008 \times 256 = 258045$. According to [52], there are two planes whose associated spreads have this automorphism group order. They can be distinguished by the 2-rank of their incidence matrices. The Johnson-Walker plane has a 2-rank of 100. The Lorimer-Rahilly plane has a 2-rank of 106. Using Orbiter, we compute the 2-rank of the translation plane that we have created:

```plaintext
RREF_plane_16_2_0_rank_of_incma:
  $(ORBITER) -v 2 \
```

383
It turns out that the 2-rank of our plane is 106, so the plane is Lorimer-Rahilly.

Let us investigate the Rao / Rao plane from Section 12.1, which we know is isomorphic to the spread in the Orbiter catalogue with number 0 and projective stabilizer of order 84. The command

\[
\text{create_translation_plane_27_Rao_Rao:}
\]

\[
\text{\$\langle ORBITER \rangle \ -v 3 \ \} \\
\text{\ -define F -finite_field -q 3 -end \} \\
\text{\ -define SS -vector -dense \$\langle SPREAD\_SET\_27\_RAO\_RAO \rangle -end \} \\
\text{\ -define G -linear_group -PGL 6 F -end \} \\
\text{\ -define G1 -linear_group -PGL 7 F -end \} \\
\text{\ -define S -spread -kernel_field F \} \\
\text{\ -group G -k 3 -spread_set SS \} \\
\text{\ -end \} \\
\text{\ -define T -translation_plane S G G1 -end \} \\
\text{\ -with T -do -translation_plane_activity \} \\
\text{\ -export incma \} \\
\text{\ -end}
\]

creates the translation plane from the spread (there is an error message which we can ignore; this is because we did not create the stabilizer of the spread). To compute the 3-rank of the incidence matrix, we issue the following comand:

\[
\text{RREF_Rao_Rao\_plane\_incma\_rank:}
\]

\[
\text{\$\langle ORBITER \rangle \ -v 2 \ \} \\
\text{\ -define F -finite_field -q 3 -end \} \\
\text{\ -define v -vector -field F \} \\
\text{\ -file plane\_incma.csv \} \\
\text{\ -end \} \\
\text{\ -with F -do -finite_field_activity \} \\
\text{\ -RREF v -normalize_from_the_right \} \\
\text{\ -end}
\]

The 3-rank turns out to be 271. According to the Moorhouse tables [53], the plane is Moorhouse IV.
### 12.3 Packings

A packing of $\text{PG}(3, q)$ is a set of pairwise line-disjoint spreads of $\text{PG}(3, q)$ of size $q^2 + q + 1$. Each spread contains $q^2 + 1$ lines. A simple counting argument shows that every line is contained in exactly one spread of the packing. The classification problem for packings is the problem of determining a complete set of pairwise non-isomorphic packings. Orbiter can be used to classify packings for small parameters. It is sometimes useful to make a symmetry assumption. This means that only those packings will be found that satisfy the symmetry assumption. The reason for making such an assumption is that the problem becomes easier and hence more tractable. Often, an assumption is made that the packings are invariant under a (nontrivial) group $H$. This section describes various ways in which Orbiter can help find and classify packings, with or without symmetry assumption.

Table 12.4 list Orbiter commands related to the construction of packings with assumed symmetry.

Table 12.5 list Orbiter commands related to the construction of packings with assumed symmetry by picking long orbits.

The following command creates a table of all labeled spreads in $\text{PG}(3, 4)$. There are three isomorphism types of spreads in $\text{PG}(3, 4)$. The command computes the orbits of each. In total, this gives 5096448 labeled spreads.

```plaintext
spread_table_PG_3_4:
  - mkdir SPREAD_TABLES_4
  $(ORBITER) -v 6 \
  -define F -finite_field -q 4 -end \
  -define P -projective_space -n 3 -field F -v 0 -end \
  -define T -spread_table P 2 "0,1,2" "SPREAD_TABLES_4/"
```

There are 21 isomorphism types of spreads in $\text{PG}(3, 5)$. The regular spread has Orbiter catalogue number equal to 12. The following command creates a table of all labeled regular spreads:

```plaintext
spread_table_PG_3_4:
  - mkdir SPREAD_TABLES_4
  $(ORBITER) -v 6 \
  -define F -finite_field -q 4 -end \
  -define P -projective_space -n 3 -field F -v 0 -end \
  -define T -spread_table P 2 "0,1,2" "SPREAD_TABLES_4/"
```

There are 155000 regular spreads.
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-H</td>
<td>description</td>
<td>Specify the assumed group $H$ of symmetries. The orbits of $H$ on the set of spreads are considered. The packings will be constructed as union of orbits.</td>
</tr>
<tr>
<td>-N</td>
<td>description</td>
<td>Specify the normalizer of $H$.</td>
</tr>
<tr>
<td>-cliques_on_fixpoint_graph</td>
<td>$s$</td>
<td>Using poset classification, classify the orbits of $N$ on cliques of size $\leq s$ in the graph on fixed points.</td>
</tr>
<tr>
<td>-cliques_on_fixpoint_graph_control</td>
<td>descr</td>
<td>Specify poset classification options related to the classification of cliques on the fixed point graph as in Tables 6.2-6.3.</td>
</tr>
<tr>
<td>-fixp_clique_types_save_individually</td>
<td></td>
<td>Sort the cliques on fixed points by the type of their spreads and write one csv file for each possible type containing the index of the cliques of the given type.</td>
</tr>
<tr>
<td>-process_long_orbits</td>
<td>descr</td>
<td>Proceed on to long orbits using Table 12.5.</td>
</tr>
<tr>
<td>-spread_tables_prefix</td>
<td>$P$</td>
<td>Use prefix $P$ to access spread tables.</td>
</tr>
<tr>
<td>-report</td>
<td></td>
<td>Create a report of the classification process.</td>
</tr>
<tr>
<td>-regular_packing</td>
<td></td>
<td>Initialize Klein correspondence and identify (regular) spreads with external lines to the Klein quadric using the polarity of the Klein quadric.</td>
</tr>
</tbody>
</table>

Table 12.4: Orbiter commands related to the construction of packings with assumed symmetry
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-list_of_cases_from_file</td>
<td>fname</td>
<td>Define a subset of cases of fixed point cliques to be worked on. Only the cases listed the given file are considered.</td>
</tr>
<tr>
<td>-split</td>
<td>r m</td>
<td>Define a subset of cases of fixed point cliques to be worked on. Only those cases whose number is congruent to ( r ) modulo ( m ) are considered.</td>
</tr>
<tr>
<td>-orbit_length</td>
<td>l</td>
<td>Use orbits of length ( l ).</td>
</tr>
<tr>
<td>-clique_size</td>
<td>s</td>
<td>Use exactly ( s ) orbits of length ( l ).</td>
</tr>
<tr>
<td>-solution_path</td>
<td>P</td>
<td>Use ( P ) as a prefix for all solution files.</td>
</tr>
<tr>
<td>-create_graphs</td>
<td></td>
<td>For each case, create the graph that describes whether two orbits of length ( l ) are compatible.</td>
</tr>
<tr>
<td>-solve</td>
<td></td>
<td>Perform clique finding and write solutions to file.</td>
</tr>
<tr>
<td>-read_solutions</td>
<td></td>
<td>Read solutions from file.</td>
</tr>
</tbody>
</table>

Table 12.5: Orbiter commands related to the construction of packings with assumed symmetry related to picking long orbits
### 12.4 BLT-Sets

A BLT-set of $Q(4, q)$ is a set of $q + 1$ point on the quadric such that no point on the quadric is collinear to more than two points of the set. BLT sets are related to spreads of $PG(3, q)$, to flocks of the quadratic cone in $PG(3, q)$, and to many other objects in combinatorics and finite geometry. They exist whenever $q$ is odd. BLT-sets have been defined in [4]. It is an interesting problem to classify BLT-sets of $Q(4, q)$ under the orthogonal group. Some references are Law [42], Penttila-Royle [56], Penttila-Law [43, 44], Betten [8], AlAzemi-Betten-Chowdhury [1].

Orbiter can be used to create members of known families of BLT-sets and sets from a catalogue of BLT-sets over small fields. Besides that, Orbiter can be used to classify all BLT-sets for a given value of $q$. We will see how we create known examples of BLT-sets either from the catalogue or from known families. Afterwards, we will consider the problem of classification.

Table 12.6 shows options to create known BLT-sets. Table 12.7 shows options for known families or sporadic sets. For instance, the command

```plaintext
BLT_11_0:
  \$ (ORBITER) -v 2 \
  -define F -finite_field -q 11 -end \
  -define O -orthogonal_space 0 5 F -end \
  -with O -do -orthogonal_space_activity \
  -create_BLT_set -catalogue 0 -end \
  -end
```

creates the BLT-set #0 in $Q(4, 11)$. The command

```plaintext
BLT_11_Mondello:
  \$ (ORBITER) -v 2 \
  -define F -finite_field -q 11 -end \
  -define O -orthogonal_space 0 5 F -end \
  -with O -do -orthogonal_space_activity \
```

<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-catalogue</td>
<td>$i$</td>
<td>Create BLT-set number $i$ from the Orbiter catalogue ($i$ is zero-based).</td>
</tr>
<tr>
<td>-family</td>
<td>$F$</td>
<td>Create a BLT-set from family $F$. See Table 12.7 for possibilities for $F$.</td>
</tr>
</tbody>
</table>

Table 12.6: Commands for creating BLT-sets
<table>
<thead>
<tr>
<th>Command</th>
<th>Condition</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td></td>
<td>Linear BLT-set.</td>
</tr>
<tr>
<td>Fisher</td>
<td>$q \equiv \pm 1 \mod 10$</td>
<td>Mondello BLT-set due to Penttila [55].</td>
</tr>
<tr>
<td>Mondello</td>
<td></td>
<td>Fisher BLT-set [27].</td>
</tr>
<tr>
<td>FTWKB</td>
<td>$q \equiv \pm 2 \mod 3$</td>
<td>Fisher, Thas, Walker [69], Kantor, Betten [13] BLT-set.</td>
</tr>
<tr>
<td>Kantor1</td>
<td>$q = p^e, \ e &gt; 1$</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 [44].</td>
</tr>
<tr>
<td>LP_37_41a</td>
<td>$q = 37$</td>
<td>First BLT-set for $q = 37$ with ago=4, due to Law and Penttila [44].</td>
</tr>
<tr>
<td>LP_37_41b</td>
<td>$q = 37$</td>
<td>Second BLT-set for $q = 37$ with ago=4, due to Law and Penttila [44].</td>
</tr>
<tr>
<td>LP_71</td>
<td>$q = 71$</td>
<td>BLT-set for $q = 71$ due to Law and Penttila [44].</td>
</tr>
</tbody>
</table>

Table 12.7: Families of BLT-sets

```
▷▷▷ -create_BLT_set -family "Mondello" -end 
▷▷ -end
▷ pdflatex BLT_Mondello_q11.tex
▷ open BLT_Mondello_q11.pdf
```

creates the Mondello BLT-set in $Q(4,11)$. Orbiter creates the following report:

The quadratic form is:

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

The BLT-set is:
<table>
<thead>
<tr>
<th>i</th>
<th>Rank</th>
<th>Point</th>
<th>(a, b, c)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>846</td>
<td>(1, 6, 4, 10, 3)</td>
<td>(22, 11, 1)</td>
</tr>
<tr>
<td>1</td>
<td>851</td>
<td>(1, 5, 7, 10, 3)</td>
<td>(22, 110, 1)</td>
</tr>
<tr>
<td>2</td>
<td>1234</td>
<td>(1, 5, 1, 7, 7)</td>
<td>(37, 11, 1)</td>
</tr>
<tr>
<td>3</td>
<td>613</td>
<td>(1, 6, 10, 5, 1)</td>
<td>(73, 110, 1)</td>
</tr>
<tr>
<td>4</td>
<td>1307</td>
<td>(1, 1, 3, 8, 5)</td>
<td>(59, 36, 1)</td>
</tr>
<tr>
<td>5</td>
<td>1418</td>
<td>(1, 3, 9, 6, 10)</td>
<td>(95, 36, 1)</td>
</tr>
<tr>
<td>6</td>
<td>1022</td>
<td>(1, 9, 5, 10, 2)</td>
<td>(99, 96, 1)</td>
</tr>
<tr>
<td>7</td>
<td>835</td>
<td>(1, 2, 6, 3, 3)</td>
<td>(99, 36, 1)</td>
</tr>
<tr>
<td>8</td>
<td>950</td>
<td>(1, 10, 8, 2, 9)</td>
<td>(95, 96, 1)</td>
</tr>
<tr>
<td>9</td>
<td>789</td>
<td>(1, 8, 2, 4, 4)</td>
<td>(59, 96, 1)</td>
</tr>
<tr>
<td>10</td>
<td>611</td>
<td>(1, 7, 7, 5, 1)</td>
<td>(73, 11, 1)</td>
</tr>
<tr>
<td>11</td>
<td>1236</td>
<td>(1, 4, 4, 7, 7)</td>
<td>(37, 110, 1)</td>
</tr>
</tbody>
</table>

Plane intersection type is $4^{18} \ 3^{148}$
Plane invariant is too big (18 planes)

\[
\begin{array}{c|c|c}
\rightarrow & 18_1 & \downarrow 18_1 \\
\hline
12_0 & 6 & 12_0 & 4 \\
\end{array}
\]

\[C_0 = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}_{12}\]
\[C_1 = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17\}_{18}\]

\[
\begin{array}{c|c|c}
\rightarrow & 18_1 & \downarrow 18_1 \\
\hline
12_0 & 6 & 12_0 & 4 \\
\end{array}
\]

\[C_0 = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}_{12}\]
\[C_1 = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17\}_{18}\]

The classification of BLT-sets is a difficult problem. For recent contributions, see [1, 8, 42].
One approach is by means of the poset of partial BLT-sets. The following command classifies the poset of partial BLT-sets in $Q(4, 13)$:

```
BLT_13_deep_search:
  $\text{ORBITER} -v 2 \$
  $\text{-define F -finite_field -q 13 -end} \$
  $\text{-define O -orthogonal_space 0 5 F -end} \$
  $\text{-define C -BLT_set_classifier 0 -starter_size 14 -end} \$
```
The poset of partial BLT-sets is too big, and there are too many orbits. The technique of classification via substructure can help. Here is an example. We consider the same problem of BLT-sets of order 13. In the beginning, we classify all partial BLT-sets of size 5, and then create colored graphs for each of them:

```
BLT_13.classify_starter:
  $(ORBITER) -v 2 \
  -define F -finite_field -q 13 -end \
  -define O -orthogonal_space 0 5 F -end \
  -define C -BLT_set_classifier 0 -starter_size 5 -end \
  -with C -do -BLT_set.classify_activity \
  -compute_starter \
  -problem_label BLT_q13 \
  -W -depth 5 \
  -end \
  -end \
  -create_graphs \
  -end
```

In the next step, we compute all rainbow cliques in each of the graphs:

```
BLT_13.clique:
  $(ORBITER) -v 2 \
  -loop L 0 38 1 \
  -define G -graph -load BLT_q13_graph_5_%L.bin -end \
  -with G -do \
  -graph.theoretic.activity \
  -find_cliques -rainbow -target_size 9 -end \
  -end \
  -end_loop
```

Next, we create a data structure for isomorphism testing. The first step is to create a database of all partial BLT-sets of order at most 5:
The next step is to read the rainbow cliques from the clique finding process:

BLT_13.isomorph_read_DB:

```bash
$ (ORBITER) -v 2 \
  -define F -finite_field -q 13 -end \n  -define O -orthogonal_space 0 5 F -end \n  -define C -BLT_set_classifier 0 -starter_size 5 -end \n  -with C -do -BLT_set_classify_activity \n  -compute_starter \n  -problem_label BLT_q13 \n  -W -depth 5 \n  -end \n  -end \n  -with C -do -BLT_set_classify_activity \n  -isomorph \n  -prefix_iso "/BLT_q13" \n  -use_database_for_starter \n  -build_db \n  -solution_prefix "" \n  -base_fname "" \n  -end \n  -end
```

BLT_13.isomorph_read_solutions:

```bash
$ (ORBITER) -v 2 \
  -define F -finite_field -q 13 -end \n  -define O -orthogonal_space 0 5 F -end \n  -define C -BLT_set_classifier 0 -starter_size 5 -end \n  -with C -do -BLT_set_classify_activity \n  -compute_starter \n  -problem_label BLT_q13 \n  -W -depth 5 \n  -end \n  -end \n  -with C -do -BLT_set_classify_activity \n  -isomorph \n  -prefix_iso "/BLT_q13" \n  -use_database_for_starter \n  -read_solutions \n  -list_of_cases BLT_q13_list_of_cases_5_0_1.csv \n  -solution_prefix "" \n  -base_fname "BLT_q13_graph" \n  -end \n  -end
```

392
Then, we compute the stabilizer orbits, which are also known as the flag orbits:

```bash
BLT_13_isomorph_stabilizer_orbits:
  ▶ $(ORBITER) -v 2 \
  ▶ ▶ -define F -finite_field -q 13 -end \n  ▶ ▶ -define O -orthogonal_space 0 5 F -end \n  ▶ ▶ -define C -BLT_set_classifier 0 -starter_size 5 -end \n  ▶ ▶ -with C -do -BLT_set_classify_activity \n  ▶ ▶ ▶ -compute_starter \n  ▶ ▶ ▶ ▶ -problem_label BLT_q13 \n  ▶ ▶ ▶ ▶ -W -depth 5 \n  ▶ ▶ ▶ ▶ -end \n  ▶ ▶ ▶ -end \n  ▶ ▶ ▶ -with C -do -BLT_set_classify_activity \n  ▶ ▶ ▶ -isomorph \n  ▶ ▶ ▶ ▶ -prefix_iso "/BLT_q13" \n  ▶ ▶ ▶ ▶ -use_database_for_starter \n  ▶ ▶ ▶ ▶ -compute_orbits \n  ▶ ▶ ▶ ▶ -list_of_cases BLT_q13_list_of_cases_5_0_1.csv \n  ▶ ▶ ▶ ▶ -solution_prefix "" \n  ▶ ▶ ▶ ▶ -base_fname "BLT_q13_graph" \n  ▶ ▶ ▶ ▶ -end \n  ▶ ▶ ▶ -end

Finally, we perform isomorphism testing:

```bash
BLT_13_isomorph_testing:
  ▶ $(ORBITER) -v 4 \
  ▶ ▶ -define F -finite_field -q 13 -end \n  ▶ ▶ -define O -orthogonal_space 0 5 F -end \n  ▶ ▶ -define C -BLT_set_classifier 0 -starter_size 5 -end \n  ▶ ▶ -with C -do -BLT_set_classify_activity \n  ▶ ▶ ▶ -compute_starter \n  ▶ ▶ ▶ ▶ -problem_label BLT_q13 \n  ▶ ▶ ▶ ▶ -W -depth 5 \n  ▶ ▶ ▶ ▶ -report -end \n  ▶ ▶ ▶ ▶ -end \n  ▶ ▶ ▶ ▶ -with C -do -BLT_set_classify_activity \n  ▶ ▶ ▶ ▶ -isomorph \n  ▶ ▶ ▶ ▶ ▶ -prefix_iso "/BLT_q13" \n  ▶ ▶ ▶ ▶ ▶ -use_database_for_starter \n  ▶ ▶ ▶ ▶ ▶ -isomorph_testing \n  ▶ ▶ ▶ ▶ ▶ -solution_prefix "" \n  ▶ ▶ ▶ ▶ ▶ -base_fname "BLT_q13_graph" \n```

393
This last command results in three isomorphism types of BLT-sets of order 13.
Chapter 13

Graph Theory

13.1 Creating Graphs

Tables 13.1-13.2 show Orbiter commands to create graphs.

For instance, the command

```
Cycle_graph_13:
 $\$(ORBITER) -v 2 \n $\$ -define Gamma -graph \n $\$ -cycle 13 \n $\$ -end
```

creates the cycle graph of degree 13.

There are many ways to read graphs from file. One way is by means of an adjacency matrix stored as a csv file. Consider an example. The `load_csv_no_border` command can be used to create a graph from a csv file containing the adjacency matrix. The following command sequence uses a makefile variable to store the adjacency matrix of a graph. The matrix is then copied into a file and the graph is created from the file. Here is the makefile variable containing the adjacency matrix:

```
TRIANGLE_GRAPH="0,1,1\n1,0,1\n1,1,0"
```

And here is the command to create the csv file from the makefile variable and to create the graph from the csv file:

```
make_triangle_graph:
 echo $(TRIANGLE_GRAPH) >triangle_graph.csv
 $(ORBITER) -v 6 \n $\$ -define G -graph \n $\$ -load_csv_no_border \n $\$ triangle_graph.csv \n```
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-load</td>
<td>filename</td>
<td>Read a graph from file in Orbiter format.</td>
</tr>
<tr>
<td>-load_csv_no_border</td>
<td>filename</td>
<td>Read a graph from a csv file. Ignore the first row and first column.</td>
</tr>
<tr>
<td>-load_dimacs</td>
<td>filename</td>
<td>Read a graph from file in dimacs format.</td>
</tr>
<tr>
<td>-edge_list</td>
<td>$n$ list-of-edges</td>
<td>Create a graph on $n$ vertices from a list of edges as ranked pairs.</td>
</tr>
<tr>
<td>-edges_as_pairs</td>
<td>$n$ edges-as-pairs</td>
<td>Create a graph on $n$ vertices from a list of edges as pairs.</td>
</tr>
<tr>
<td>-cycle</td>
<td>$n$</td>
<td>Cycle graph on $n$ vertices.</td>
</tr>
<tr>
<td>-Hamming</td>
<td>$n$ $q$</td>
<td>Hamming graph $H(n, q)$</td>
</tr>
<tr>
<td>-Johnson</td>
<td>$n$ $k$ $s$</td>
<td>Johnson graph</td>
</tr>
<tr>
<td>-Paley</td>
<td>$q$</td>
<td>Paley graph</td>
</tr>
<tr>
<td>-Sarnak</td>
<td>$p$ $q$</td>
<td>Lubotzky-Phillips-Sarnak graph [46]</td>
</tr>
<tr>
<td>-Schlaefli</td>
<td>$q$</td>
<td>Schlaefli graph</td>
</tr>
<tr>
<td>-Shrikhande</td>
<td></td>
<td>Shrikhande graph</td>
</tr>
<tr>
<td>-Winnie_Li</td>
<td>$q$ $i$</td>
<td>Winnie-Li graph [45]</td>
</tr>
<tr>
<td>-Grassmann</td>
<td>$n$ $k$ $q$ $r$</td>
<td>Grassmann graph</td>
</tr>
<tr>
<td>-coll_orthogonal</td>
<td>$\epsilon$ $d$ $q$</td>
<td>Collinearity graph of $O^\epsilon(d, q)$</td>
</tr>
<tr>
<td>-triherdal_pair_</td>
<td></td>
<td>Triheral 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>
</tbody>
</table>

Table 13.1: Orbiter commands to define graphs (Part 1)
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-subset</td>
<td>label labeltex subset</td>
<td>Define vertex coloring with two colors based on a subset of vertices.</td>
</tr>
<tr>
<td>-disjoint_sets_graph</td>
<td>fname</td>
<td>Define a graph on a system of sets. Adjacency is when two sets are disjoint. The sets are taken from the given file.</td>
</tr>
<tr>
<td>-orbital_graph</td>
<td>G i</td>
<td>Define orbital graph from the $i$-th orbit of the group $G$ acting on pairs.</td>
</tr>
<tr>
<td>-collinearity_graph</td>
<td>inc-matrix</td>
<td>Collinearity graph of the given incidence matrix.</td>
</tr>
<tr>
<td>-chain_graph</td>
<td>P1 P2</td>
<td>Chain graph with respect to the partitions P1 and P2.</td>
</tr>
<tr>
<td>-Cayley_graph</td>
<td>G gens</td>
<td>Cayley graph with respect to group $G$ and generating set gens.</td>
</tr>
</tbody>
</table>

Table 13.2: Orbiter commands to define graphs (Part 2)

This will create the three-cycle graph.

The command

Chain_232:

```plaintext
$ (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:

```plaintext
$ (ORBITER) -v 2 \n ▶ ▶ -define Gamma -graph -Paley 13 -end \n```

creates the Paley graph on 13 vertices.

The command
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 \n  5 "0,1,0,2,0,3,0,4,1,3,1,4,2,4" \n  -end
```

creates a graph by listing the edges in pairs. In this case, the graph

![Graph diagram]

is created.

The command

```
petersen:
  $(ORBITER) -v 2 \n  -define G -graph -Johnson 5 2 0 -end
```

creates the Petersen graph.

The command
Johnson_6_2_0:
▷ $(ORBITER) -v 2 \n▷ ▷ -define G -graph -Johnson 6 2 0 -end

creates the Johnson graph $J(6, 2, 0)$.

The command

Hamming_graph_3:
▷ $(ORBITER) -v 2 \n▷ ▷ -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:

HJ_graph:
▷ $(ORBITER) -v 6 \n▷ ▷ -define G -graph \n▷ ▷ ▷ -load_csv_no_border \n▷ ▷ ▷ halljanko315.csv \n▷ ▷ -end

In Section 15.7, we will compute the automorphism group of the graph (of order 1209600). This will create a file halljanko315_gens.csv which we use here to create an orbital graph. An orbital graph is a graph whose adjacency matrix corresponds to an orbit of a permutation group in the action on pairs. The group is the automorphism group of the graph. The following command creates the third orbital graph:

HJ315_orbital_graph_3:
▷ $(ORBITER) -v 2 \n▷ ▷ -define gens -vector -file \n▷ ▷ ▷ halljanko315_gens.csv -end \n▷ ▷ -define G -permutation_group \n▷ ▷ ▷ -bsgs halljanko315 "File\_halljanko315" \n▷ ▷ ▷ 315 1209600 "0,1,2" 6 gens \n▷ ▷ -end \n▷ ▷ -define Gamma -graph \n▷ ▷ ▷ -orbital_graph G 3 \n▷ ▷ -end \n
399
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-complement</td>
<td></td>
<td>Complementary graph.</td>
</tr>
<tr>
<td>-distance_2</td>
<td></td>
<td>Distance two graph: Two vertices are adjacent if and only if they have distance two in the original graph.</td>
</tr>
</tbody>
</table>

Table 13.3: Orbiter commands to modify graphs

Table 13.3 shows some Orbiter commands to modify graphs. The commands replace the given graph by the graph obtained after applying the specified modification.

For a graph $\Gamma$, the distance 2 graph $\Delta$ has the same vertices as $\Gamma$, and two vertices in $\Delta$ are adjacent if and only if the distance in $\Gamma$ is two. The following command creates the distance 2 graph of the Cohen-Tits graph.

**HJ_d2.graph:**

```bash
$ (ORBITER) -v 6 \
$ -define G -graph \
$ -load_csv_no_border \
$ -define halljanko315.csv \
$ -distance -end
```

Let us look at some examples of Cayley graphs. The first graph has $G = \mathbb{Z}_{11}$ and connection set all elements congruent 1 mod 3. We create the group as a subgroup of the one-dimensional affine group over $\mathbb{F}_{11}$. This means that the map $x \mapsto ax + b \mod 11$ is encoded as a pair $(a,b)$.

**Cayley_Z11_1mod3:**

```bash
$ (ORBITER) -v 2 \
$ -define F -finite_field -q 11 -end \
$ -define S -vector -dense \
$ "1,1, 1,4, 1,7, 1,10" -end \
$ -define G -linear_group -AGL 1 F \
$ -subgroup_by_generators "Z11" 11 1 "1,1" \
$ -end \
$ -define Gamma -graph \
$ -Cayley_graph G S \
$ -end
```

The vertices of the Cayley graph are ordered. The ordering is determined by the stabilizer chain. This is a total ordering.
In the following example, we create a Cayley graph based on the symmetric group on 4 things. We take the Coxeter generators as connection set:

Cayley_Sym4_coxeter:
\[
\text{\$\{ORBITER\} -v 2 \}
\text{\> \> -define S -vector -dense "1,0,2,3, 0,2,1,3, 0,1,3,2" -end \}
\text{\> \> -define G -permutation_group -symmetric_group 4 \}
\text{\> \> -end \}
\text{\> \> -define Gamma -graph \}
\text{\> \> \> -Cayley_graph G S \}
\text{\> \> -end}
\]

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:
\[
\text{\$\{ORBITER\} -v 2 \}
\text{\> \> -define S -vector -dense "1,0,2,3, 2,1,0,3, 3,1,2,0" -end \}
\text{\> \> -define G -permutation_group -symmetric_group 4 \}
\text{\> \> -end \}
\text{\> \> -define Gamma -graph \}
\text{\> \> \> -Cayley_graph G S \}
\text{\> \> -end}
\]
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-find_cliques</td>
<td>options</td>
<td>Find all cliques. See Section 13.4.</td>
</tr>
<tr>
<td>-export_magma</td>
<td></td>
<td>Export to Magma [14].</td>
</tr>
<tr>
<td>-export_maple</td>
<td></td>
<td>Export to Maple [49].</td>
</tr>
<tr>
<td>-export_csv</td>
<td></td>
<td>Export to csv-file.</td>
</tr>
<tr>
<td>-export_graphviz</td>
<td></td>
<td>Export to graphviz-file.</td>
</tr>
<tr>
<td>-print</td>
<td></td>
<td>Print the graph.</td>
</tr>
<tr>
<td>-sort_by_colors</td>
<td></td>
<td>Sort the vertices by color classes.</td>
</tr>
<tr>
<td>-split</td>
<td>file</td>
<td>Split the graph into subgraphs.</td>
</tr>
<tr>
<td>-split_by_starters</td>
<td>file</td>
<td>Split the graph into subgraphs according to starters.</td>
</tr>
<tr>
<td>-split_by_clique</td>
<td>label clique</td>
<td>Compute the neighborhood graph of the given clique.</td>
</tr>
<tr>
<td>-save</td>
<td></td>
<td>Save the graph to file in binary format.</td>
</tr>
<tr>
<td>-automorphism_group</td>
<td></td>
<td>Compute the automorphism group and write a report. See Section 15.7.</td>
</tr>
<tr>
<td>-properties</td>
<td></td>
<td>Compute properties of the graph.</td>
</tr>
<tr>
<td>-eigenvalues</td>
<td></td>
<td>Compute the eigenvalues of the graph.</td>
</tr>
<tr>
<td>-draw</td>
<td></td>
<td>Draw the graph.</td>
</tr>
</tbody>
</table>

Table 13.4: Graph Theoretic Activities

### 13.2 Graph Theoretic Activities

Graph theoretic activities allow us to perform tasks on graphs. Table 13.4 shows the commands for graph theoretic activities. These are activities that can be applied to objects of type graph.

Continuing the example of the three-cycle, the command

```
triangle_graph_properties:
  ▶ echo $(TRIANGLE_GRAPH) >triangle_graph.csv
  ▶ $(ORBITER) -v 6 \
  ▶ ▶ -define G -graph \
  ▶ ▶ ▶ -load_csv_no_border \
  ▶ ▶ ▶ triangle_graph.csv \
  ▶ ▶ -end \
  ▶ ▶ -with G -do \
  ▶ ▶ ▶ -graph_theoretic_activity -properties \
  ▶ ▶ -end
```

402
computes the degree type of the graph. This is the distribution of degrees in the degree sequence of the graph. It prints the distribution of degree values in exponential notation. The multiplicities are indicated as exponent. For instance, the graph in this example has three vertices of degree 2, so the degree sequence is printed as $2^3$.

We can export the adjacency matrix and create a bitmap drawing like so:

```
Cycle_13.draw:
  $ (ORBITER) -v 2 \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:

```
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

Paley_13_draw:

\[
\text{\( \text{\texttt{(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{\texttt{\$(ORBITER) -v 2 -draw_matrix \ \\
\ -input_csv_file Paley_13.csv \ \\
\ -box_width 20 -bit_depth 8 -partition 4 13 13 -end \ \\
\ -do Tpng Paley_13.gv >Paley_13.png \ \\
\ -open Paley_13.draw.bmp}
\end{verbatim}

draws the Paley graph of order 13 created in Section 13.1 using the external tool graphviz.

Let us consider the Cayley graphs from Section 13.1. Here is a command that draws the first graph and computes the eigenvalues:

Cayley_Z11_1mod3_eigenvalues_and_draw:

\[
\text{\( \text{\texttt{(ORBITER) -v 2 \ \\
\ -draw_options -xin 2000000 \ \\
\ -yin 2000000 -embedded -radius 20000 -end \ \\
\ -define F -finite_field -q 11 -end \ \\
\end{verbatim}

404
The drawing is shown in Figure 13.1. Let us draw the Cayley graph of Sym(5) with respect to the Coxeter generators:

```latex
\begin{verbatim}
Cayley_Sym5.coxeter_draw:
  \$ (ORBITER) -v 2 \n  \$ -draw_options -xin 1000000 -yin 1000000 \n  \$ -embedded -radius 10000 -nodes_empty -end \n  \$ -define S -vector -dense \n  \$ "1,0,2,3,4, 0,2,1,3,4, 0,1,3,2,4, 0,1,2,4,3" -end \n  \$ -define G -permutation_group -symmetric_group 5 \n  \$ -end \n\end{verbatim}
```

405
Figure 13.2: The Cayley graph of Sym(5) w.r.t. the Coxeter generators

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:

\[ \text{PGO}_{5.2}\text{-collinearity_graph: 0.5.2\text{-incidence_matrix.csv}} \]

\[
\text{\$ (ORBITER) -v 3 \$
\text{\$ -define Inc -vector -file 0.5.2\text{-incidence_matrix.csv} -end \$
\text{\$ -define Gamma -graph -collinearity_graph Inc -end \$
\text{\$ -with Gamma -do \$
\text{\$ -graph\text{-theoretic_activity} -draw -end \$
\text{\$ pdflatex Cayley_graph_Perm5\text{-draw}.tex \$
\text{\$ open Cayley_graph_Perm5\text{-draw}.pdf \$}
\]
The command also computes properties of the graph. The graph has 15 vertices of degree 6. This is because in the geometry, each point lies on three lines, and hence is collinear with 6 other points.
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-girth</td>
<td>$d$</td>
<td>Girth at least $d$</td>
</tr>
<tr>
<td>-regular</td>
<td>$r$</td>
<td>Regular of degree $r$</td>
</tr>
<tr>
<td>-no_transmitter</td>
<td></td>
<td>Tournament without transmitter (requires -tournament)</td>
</tr>
</tbody>
</table>

Table 13.5: Options for classifying graphs

### 13.3 Classification of Graphs and Tournaments

Table 13.5 lists the Orbiter commands to classify graphs and tournaments. The following command classifies all graphs on 5 vertices:

```plaintext
graph_classify_5:
   $(ORBITER) -v 2 \
   -orбiter_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   -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   -radius 300 -nodes_empty \n   -line_width 1.5 \n   -scale 0.1 \n   -end \n   -draw_graphs_at_level 5 \n   -end \n   -print_symbols
   pdflatex graphs_v5_level_5_reps.tex
   open graphs_v5_level_5_reps.pdf
   pdflatex graphs_v5_poset.tex
   open graphs_v5_poset.pdf
```
After the classification, the graphs with 5 edges are shown. The file contains the following graph drawings:

![Graph drawings](image_url)

The next command classifies all tournaments on 4 vertices:

```plaintext
tournamentclassify_4:
  $ (ORBITER) -v 2
  -define GC -graphclassification
  -n 4 -tournament
  -posetclassificationcontrol
  -problem_label tournament_4
  -depth 6 -drawposet
  -drawoptions
  -radius 250 -embedded
  -end
  -end
  -with GC -do
  -graphclassificationactivity
  -drawoptions
  -radius 400
  -line_width 2 -scale 0.10
  -end
  -drawgraphs_at_level 6
  -end
  -print_symbols
  pdflatex tournament_4_level_6_reps.tex
  open tournament_4_level_6_reps.pdf
```

There are four tournaments. The following graph drawings are produced:
The next command classifies all cubic graphs on 8 vertices:

```bash
graph_classify_8_r3:
  $(ORBITER) -v 3 \
  -define GC -graph_classification \
  -n 8 -regular 3 \
  -poset_classification_control \
  -problem_label graphs_v8_r3 \
  -depth 12 -draw_poset \
  -draw_options -radius 250 \
  -line_width 0.2 -embedded \
  -end \
  -end \
  -with GC -do \
  -graph_classification_activity \
  -draw_options \
  -radius 400 \
  -line_width 2 -scale 0.10 \
  -end \
  -draw_graphs_at_level 12 \
  -end \
  -print_symbols
  #pdflatex graphs_v8_r3_poset_lvl_12.tex
  #open graphs_v8_r3_poset_lvl_12.pdf
```

There are six cubic graphs. The following graph drawings are produced:
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-rainbow</td>
<td></td>
<td>Find all rainbow cliques. The size of the cliques is the number of vertex colors.</td>
</tr>
<tr>
<td>-target_size</td>
<td>s</td>
<td>Find all cliques of size s.</td>
</tr>
<tr>
<td>-weighted</td>
<td>s</td>
<td>Find weighted cliques.</td>
</tr>
<tr>
<td>-Sajeeb</td>
<td></td>
<td>Use the implementation by Sajeeb Chowdhury.</td>
</tr>
<tr>
<td>-output_file</td>
<td>fname</td>
<td>Write cliques to the named file.</td>
</tr>
<tr>
<td>-restrictions</td>
<td>l r m</td>
<td>Restricted search: At level l, restrict to all branches congruent to r modulo m. Here, $0 \leq r &lt; m$.</td>
</tr>
</tbody>
</table>

Table 13.6: 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 13.4 can be used to find all cliques in a graph. Additional options for this command are shown in Table 13.6. For instance, the cliques of size 3 in the graph `graph_v5_e7.colored_graph` from Section 13.1 can be found using

```bash
small_graph_cliques: graph_v5_e7.colored_graph
    > $(ORBITER) -v 2 \
    >   -define G -graph -load graph_v5_e7.colored_graph -end \ 
    >   -with G -do \ 
    >   -graph_theoretic_activity \ 
    >   > -find_cliques -target_size 3 \ 
    >   > -end
```

This command finds three cliques of size 3.

It is also possible to classify all cliques under the automorphism group of the graph. This is a multi-step process, though. At first, the automorphism group of the graph has to be computed. Then, poset classification can be invoked to classify the cliques of a certain size. Here is an example. We wish to classify the cliques in the Paley graph of order 13. The first command creates the graph and computes the automorphism group:

```bash
Paley_13_aut:
    > $(ORBITER) -v 2 \
    >   -define Gamma -graph -Paley 13 -end \
```

412
The command writes a file Paley_13_group.makefile, shown below:

Paley_13:
\[
\text{\$\{ORBITER\_PATH\}orbiter.out -v 2 } \\text{\-define gens -vector -file Paley_13\_gens.csv -end } \\
\text{\-define G -permutation group } \\text{\-bsgs Paley_13 "Paley\_13" 13 78 "0,1" 3 gens -end }
\]

The group is given using a base and strong generating set. The base consists of the two points 0, 1. Three strong generators with respect to this base are given in a csv file. Using this group, the next command classifies all cliques of size at most 5 in the Paley graph of order 13 under the action of the automorphism group. It turns out that there are no 5-cliques, and that the largest cliques have size 3. The command shows that there is a unique orbit of 3-cliques:

Paley_13\_cliques\_classify:
\[
\text{\$\{ORBITER\} -v 4 } \\text{\-define Control -poset\_classification\_control } \\
\text{\-W } \\text{\-problem\_label Paley13\_cliques } \\
\text{\-clique\_test Gamma } \\text{\-depth 5 } \\
\text{\-end } \\text{\-define gens -vector -file Paley_13\_gens.csv -end } \\
\text{\-define G -permutation group } \\text{\-bsgs Paley_13 "Paley\_13" 13 78 "0,1" 3 gens -end } \\
\text{\-define Gamma -graph -Paley 13 -end } \\
\text{\-define Orb -orbits -group G } \\
\text{\-on\_subsets 5 Control } \\
\text{\-end }
\]

For comparison, the command

Paley_13\_cliques\_all:
\[
\text{\$\{ORBITER\} -v 10 } \\text{\-define Gamma -graph -Paley 13 -end } \\
\text{\-with Gamma -do } \\
\text{\-graph\_theoretic\_activity }
\]
finds all cliques of size 3. There are exactly 26 of them. Because of the previous command, 
we know that they are all equivalent under the automorphism group of the graph.

Let us consider the orbital graph created in Section 13.1. We wish to study the 5-cliques. 
We first determine the number of 5-cliques, and then the number of orbits of 5-cliques under 
the automorphism group. The following command computes all 5-cliques:

```
HJ64_cliques5:
  $(ORBITER) -v 6 \
  -define Gamma -graph \
  -load Group_Perms315_orbital_3.colored_graph \
  -end \
  -with Gamma -do \
  -graph_theoretic_activity \
  -find_cliques -target_size 5 -end \
  -end
```

It turns out that there are exactly 1008 5-cliques. Concerning the classification with respect 
to the automorphism group of the graph, we apply the following command:

```
HJ64_cliques5_classify:
  $(ORBITER) -v 6 \
  -define Control -poset_classification_control \
  -W \
  -problem_label HJ64_cliques \
  -clique_test Gamma \
  -depth 5 \
  -end \
  -define Gamma -graph \
  -load Group_Perms315_orbital_3.colored_graph \
  -end \
  -define gens -vector \
  -file halljanko315gens.csv \
  -end \
  -define G -permutation_group \
  -bsgs halljanko315 "File\halljanko315" \
  315 1209600 "0,1,42,95" 6 gens -end \
  -define Orb -orbits -group G \
  -on_subsets 5 Control \
  -end
```

414
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.clique: 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:

```plaintext
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 \
  -define C -combinatorial_objects \
```

417
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-set_of_points</code></td>
<td></td>
<td>A set consisting of points.</td>
</tr>
<tr>
<td><code>-set_of_lines</code></td>
<td></td>
<td>A set consisting of lines.</td>
</tr>
<tr>
<td><code>-set_of_points_and_lines</code></td>
<td></td>
<td>A set consisting of points and a second set consisting of lines.</td>
</tr>
<tr>
<td><code>-set_of_packing</code></td>
<td></td>
<td>A set of packings.</td>
</tr>
<tr>
<td><code>-file_of_points</code></td>
<td></td>
<td>A set consisting of points read from file.</td>
</tr>
<tr>
<td><code>-file_of_lines</code></td>
<td></td>
<td>A set consisting of lines read from file.</td>
</tr>
<tr>
<td><code>-file_of_packings</code></td>
<td></td>
<td>A set consisting of packings read from file.</td>
</tr>
<tr>
<td><code>-file_of_packings_through_spread_table</code></td>
<td></td>
<td>A file of packings.</td>
</tr>
<tr>
<td><code>-file_of_point_set</code></td>
<td></td>
<td>A file containing point sets.</td>
</tr>
<tr>
<td><code>-file_of_designs</code></td>
<td></td>
<td>A file containing designs or large sets.</td>
</tr>
<tr>
<td><code>-file_of_incidence_geometries</code></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><code>-file_of_incidence_geometries_by_row_ranks</code></td>
<td></td>
<td>A file describing incidence geometries defined by their row ranks.</td>
</tr>
<tr>
<td><code>-incidence_geometry</code></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><code>-incidence_geometry_by_row_ranks</code></td>
<td></td>
<td>An incidence geometry defined by row ranks.</td>
</tr>
<tr>
<td><code>-from_parallel_search</code></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 14.1: Defining Combinatorial Objects
-set_of_points H \  
-end

The next example creates the two hyperovals in PG(2,16). The hyperovals are stored in makefile variables:

HYPEROVAL_16_144="0, 1, 2, 3, 52, 67, 89, 106, 126, \ 141, 159, 176, 184, 199, 220, 235, 245, 262"

HYPEROVAL_16_16320="0, 1, 2, 3, 52, 70, 83, 109, 127, \ 139, 156, 174, 186, 199, 217, 229, 256, 264"

hyperoval_16_create:  
  $(ORBITER) -v 2 \  
  -define C -combinatorial_objects \  
  -set_of_points $(HYPEROVAL_16_16320) \  
  -set_of_points $(HYPEROVAL_16_144) \  
  -end \  

In the next example, we read the points of an elliptic curve from file (see Section 4.2):

EC_read: elliptic_curve_b1_c3_q11.txt  
  $(ORBITER) -v 4 \  
  -define C -combinatorial_objects \  
  -file_of_points elliptic_curve_b1_c3_q11.txt \  
  -end

In the next example, we read a packing, using a previously defined table of spreads, stored in a csv file.

PG_3_5_assume_31_read:  
  $(ORBITER) -v 2 \  
  -define C -combinatorial_objects \  
  -file_of_packings_through_spread_table \  
  H31_packings.csv \  
  SPREAD_TABLES_5_REG/spread_25_spreads.csv \  
  5 \  
  -end

The following command reads a file of large sets of designs:

419
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 \\
  ▶ ▶ ▶ ▶ 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 \\
  ▶ ▶ ▶ ▶ Desargues_path_lex_least.inc 10 10 3 \\
  ▶ ▶ ▶ -end

420
14.2 Encoding Combinatorial Objects

Combinatorial objects can be stored in text files. There can be any number of objects in one file. The objects themselves are coded. For instance, a set of points in projective space is given as a set of integers, using the Orbiter point ranks. Likewise, a set of lines is coded using Orbiter line ranks. For designs, there are several ways in which the object can be stored. One way is by listing the incidences in a numerical form. One number is one incidence. Another way is by describing the incidence matrix in a row-by-row fashion, using ranks of $k$-subsets. This assumes that the number of incidences per row is constant over all rows. Yet another way is by listing the columns of the incidence matrix, again using ranks of $k$-subsets. This version requires that the column sums of the incidence matrix are constant. Let us go over some of these file formats, using small examples to illustrate the ideas informally.

Suppose we want to work with the Pasch configuration. This is the configuration of 6 points and 4 lines shown in Figure 14.1. In the geometry, we have 4 lines, which we can identify with the index sets of the points as $\{0, 1, 2\}$, $\{0, 3, 4\}$, $\{1, 3, 5\}$ and $\{2, 4, 5\}$. The incidence matrix of the configuration is shown in Figure 14.2. Row labels are on the left, column labels are on top. The $(i, j)$-entry is one if $P_i$ lies on $\ell_j$, and it is zero otherwise. There are three ways to encode the incidence structure. One way encodes the flags of the geometry. This will be described next. The flag space is the set of all possible flags in the incidence matrix between the given number of points and lines. The space is totally ordered using the row-major ordering (see Figure 14.3). The Pasch configuration can now be coded as

$$\{0, 1, 4, 6, 8, 11, 13, 14, 17, 19, 22, 23\}.$$ 

The file `pasch.inc` contains:

```
6 4 12
```
The first line lists the number of rows and columns of the incidence matrix, and the number of incidences. The geometry is encoded on the next line. After that, a marker of -1 shows that this is the only geometry in this file (the file format allows for any number of incidence geometries, all with the same parameters). The final row is the order of the automorphism group of the geometry. This row is optional. In case that there are several geometries in the file, the orders will all be listed. In this case, the possible values will be collected with multiplicities, and listed in decreasing order. The command

```
geo_pasch_read:
 ▶ $(ORBITER) -v 10 \\n ▶ ▶ -define C -combinatorial_objects \\n ▶ ▶ ▶ -file_of_incidence_geometries \\n ▶ ▶ ▶ ▶ pasch.inc 6 4 12 \\n ▶ ▶ ▶ -end
```

reads the incidence geometry from the file `pasch.inc`. It is also possible to enter the incidence geometry directly from the command line. The following example uses the `-incidence_geometry` command to do so:
geo_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>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</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>

Table 15.1: Orbiter commands related to canonical labelings

The graph theory package Nauty [51] provides a canonical form algorithm for graphs. Using the Levi graph construction, this technique allows to solve the isomorphism problem for combinatorial objects in the more general sense just defined.

The technique of isomorphism testing can be lifted to combinatorial objects in projective spaces or other types of finite incidence geometries. For instance, arcs in projective planes have been classified this way (cf. [2]).

Table 15.1 list Orbiter commands related to canonical labelings of combinatorial objects.
15.2 Canonical Forms of Objects in Projective Space

Suppose we want to compute the stabilizer of an elliptic curve. In Section 4.1, we have created an elliptic curve over $F_{11}$ and stored the set of $F_q$-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 refer to the file containing the Orbiter point ranks of points on the curve.

```
EC_canon: elliptic_curve_b1_c3_q11.txt
\>$\text{(ORBITER) -v 3 \n\> -define C -combinatorial_objects \n\> -file_of_points elliptic_curve_b1_c3_q11.txt \n\> -end \n\> -define F -finite_field -q 11 -end \n\> -define P -projective_space -n 2 -field F -v 0 -end \n\> -with C -do \n\> -combinatorial_object_activity \n\> -canonical_form PG P \n\> -classification_prefix EC \n\> -label EC \n\> -save_ago \n\> -max_TDO_depth 4 \n\> -end \n\> -report \n\> -prefix EC \n\> -export_flag_orbits \n\> -show_TDO \n\> -show_TDA \n\> -dont_show_incidence_matrices \n\> -export_group_GAP \n\> -end \n\> -end
\> pdflatex EC_classification.tex
\> open EC_classification.pdf
\>$\text{(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,\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.c: Hirschfeld_surface_q4.txt
▷ $(ORBITER) -v 6 \n▷ ▷ -define C -combinatorial_objects \n▷ ▷ ▷ -file_of_points Hirschfeld_surface_q4.txt \n▷ ▷ ▷ -end \n▷ ▷ ▷ -define F -finite_field -q 4 -end \n▷ ▷ ▷ -define P -projective_space -n 3 -field F -v 0 -end \n▷ ▷ ▷ -with C -do \n▷ ▷ ▷ -combinatorial_object_activity \n▷ ▷ ▷ ▷ -canonical_form_PG P \n▷ ▷ ▷ ▷ ▷ -classification_prefix Hirschfeld_surface_q4 \n▷ ▷ ▷ ▷ ▷ -save_ago \n▷ ▷ ▷ ▷ ▷ -max_TDO_depth 10 \n▷ ▷ ▷ ▷ ▷ -end \n▷ ▷ ▷ ▷ -report \n▷ ▷ ▷ ▷ ▷ -prefix Hirschfeld_surface_q4 \n▷ ▷ ▷ ▷ ▷ -export_flag_orbits \n▷ ▷ ▷ ▷ ▷ -show_TDO \n▷ ▷ ▷ ▷ ▷ -show_TDA \n▷ ▷ ▷ ▷ ▷ -dont_show_incidence_matrices \n▷ ▷ ▷ ▷ ▷ -export_group_GAP \n▷ ▷ ▷ ▷ ▷ -end \n▷ ▷ ▷ ▷ -end
▷ ▷ ▷ pdflatex Hirschfeld_surface_q4_classification.tex
▷ ▷ open Hirschfeld_surface_q4_classification.pdf
In the next example, we compute the canonical form of the two hyperovals in PG(2, 16).

hyperoval_16_canonical_form:

```
$\text{(ORBITER)} -v 2 \n
-define C -combinatorial_objects \
  -set_of_points $\text{(HYPEROVAL}_16\text{.16320}) \n  -set_of_points $\text{(HYPEROVAL}_16\text{.144}) \n
-define F -finite_field -q 16 -end \n
-define P -projective_space -n 2 -field F -v 0 -end \n
-with C -do \n
-canonical_form PG P \n
-classification_prefix hyperoval_q16 \n
-label hyperoval_q16 \n
-save ago \n
-save transversal \n
-max_TDO_depth 10 \n
-end \n
-report \n
-prefix hyperoval_q16 \n
-export_flag_orbits \n
-show_TDO \n
-show_TDA \n
-dont_show_incidence_matrices \n
-export_group_GAP 
```
In the next example, we compute the set stabilizers of orbits of $\text{PGL}(4,2)$ on subsets of $\text{PG}(3,2)$, as computed earlier in Section 6.3, using the command $\text{PG}_3_2\text{-subsets}$. These orbits are relevant for Section 7.5. Concerning the work in Dickson [24] only subsets whose size is odd are relevant, so we restrict to those subsets:

**Dickson_sets.stabilizer:**

```bash
$\text{ORBITER}) -v 3 \n$\text{define C -combinatorial_objects} \n$\text{define F -finite_field -q 2 -end} \n$\text{define P -projective_space -n 3 -field F -v 0 -end} \n$\text{-with C -do} \n$\text{-combinatorial_object_activity} \n$\text{-canonical_form_PG P} \n```

430
There are two ovoids in $\text{PG}(3, 2)$. The classical ovoid is the elliptic quadric. It was created using the command `elliptic_quadric_ovoid_q8` in Section 4.10. The following command computes the stabilizer of the ovoid:

```
ovoid_q8_canon: ovoid_q8.txt
  $(ORBITER) -v 6 \n  \$ -define C -combinatorial_objects \n  \$ -file_of_points ovoid_q8.txt \n  \$ -end \n  \$ -define F -finite_field -q 8 -end \n  \$ -define P -projective_space -n 3 -field F -v 0 -end \n  \$ -with C -do \n  \$ -combinatorial_object_activity \n  \$ -canonical_form_PG P \n  \$ -classification_prefix 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_GAP \n  \$ -end \n  pdflatex ovoid_classification.tex
  open ovoid_classification.pdf
```

The other ovoid is the Suzuki Tits ovoid, which was created using the command `ovoid_ST_q8` in Section 4.10. The stabilizer of the Suzuki Tits ovoid is the Suzuki group. The following command computes this group for $q = 8$.

```
ovoid_ST_q8_canon: ovoid_ST_q8.txt
```

431
We can store the generators in a makefile variable as follows:

```
SUZUKI_8_GENERATORS="\n1,0,0,0,0,1,0,0,0,1,0,0,0,1,1, \n1,0,0,0,0,6,0,0,0,2,0,0,0,0,3,0, \n1,0,0,0,1,1,1,0,0,0,1,0,0,1,0, \n1,0,0,0,3,6,2,2,5,0,2,0,3,0,6,3,2, \n0,1,0,0,1,0,0,0,0,0,1,0,0,1,0,2"
```

We can now recover the Suzuki group using the command:

```
Suzuki_8:
  $(ORBITER) -v 6 \n  -define F -finite_field -q 8 -end \n  -define gens -vector -field F \n  -compact $(SUZUKI_8_GENERATORS) -end \n  -define G -linear_group -PGGL 4 8 \n  -subgroup_by_generators "Sz8" "87360" 5 gens \n  -end \n```
\begin{itemize}
\item \item -with G -do \item -group_theoretic_activity \item \item \item -report \item \item -end
\item pdflatex PGGL_4_8_Subgroup_Sz8_87360_report.tex
\item open PGGL_4_8_Subgroup_Sz8_87360_report.pdf
\end{itemize}
15.3 Canonical Forms of Incidence Geometries

Let us consider system of subsets. This subsets are chosen from the same set, which we call the underlying set. The elements of the group set are often called points. In many cases, there are conditions that restrict the way in which the sets can be chosen. There is a notion of isomorphism on such set systems. Two set systems are isomorphic 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 \( v \times b \times k \) 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 \( v \times r \) is used (the term symmetric is somewhat misleading because the incidence matrix of a symmetric configuration need not be symmetric). Orbiter can be used to classify incidence geometries. One of the important steps in this process is computing a canonical form of the incidence geometry.

We will also be producing drawings of the incidence matrices of geometries. In these drawings, flags are indicated as heavy squares while anti-flags are drawn as small squares. The coloring will indicate the orbits of the automorphism group on flags and anti-flags. Objects with the same color belong to the same orbit. For a flag-transitive geometry, there is only one color for the incidences.

The following command computes the canonical form and a report of the projective plane \( PG(2,2) \), which is a configuration \( 7_3 \).

```
geo_7_3.c:
▷ $(ORBITER) -v 10 \
▷ ▷ -draw_incidence_structure_description \n▷ ▷ ▷ -width 60 -with_10 6 -end \n```

434
define C -combinatorial_objects 
-file_of_incidence_geometries 7_3.inc 7 7 21 
-end 
-with C -do 
-combinatorial_object_activity 
-canonical_form 
-classification_prefix 7_3 
-label 7_3 
-save_ago 
-save_transversal 
-end 
-do 
-canonical_form 
-classification_prefix 7_3 
-label 7_3 
-save_ago 
-save_transversal 
-end 
-report 
-prefix 7_3 
-export_flag_orbits 
-show_incidence_matrices 
-export_group_GAP 
-end 
-end

pdflatex 7_3_classification.tex
open 7_3_classification.pdf
$(ORBITER) -v 2 -draw_matrix 
-input_csv_file 7_3_object0_TDA_flag_orbits.csv 
-secondary_input_csv_file 7_3_object0_TDA.csv 
-box_width 32 -bit_depth 24 
-end
$(ORBITER) -v 2 -draw_matrix 
-input_csv_file 7_3_object0_INP_flag_orbits.csv 
-secondary_input_csv_file 7_3_object0_INP.csv 
-box_width 32 -bit_depth 24 
-end
open 7_3_object0_INP_flag_orbits_draw.bmp

A bitmap drawing is produced, as shown in Figure 15.1. The command also produces the following report of the geometry:

<table>
<thead>
<tr>
<th>Rep</th>
<th># Ago</th>
<th>Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Ago : 168
Isomorphism type 0 / 1

Isomorphism type 0 / 1 is original object 0 and appears 1 times:
This isomorphism type appears 1 times, namely for the following 1 input objects:
{0}
incidence structure:
( 0, 1, 2, 7, 10, 11, 14, 19, 20, 22, 24, 26, 29, 32, 34, 37, 38, 41, 44, 46, 47 )
Column sets of the encoded object:
{ 0, 1, 2 }
{ 0, 3, 4 }
{ 0, 5, 6 }
{ 1, 3, 5 }
{ 1, 4, 6 }
{ 2, 3, 6 }
{ 2, 4, 5 }
Row sets of the encoded object:
{ 0, 1, 2 } = 0
{ 0, 3, 4 } = 9
{ 0, 5, 6 } = 14
{ 1, 3, 5 } = 20
{ 1, 4, 6 } = 23
{ 2, 3, 6 } = 27
{ 2, 4, 5 } = 28

Figure 15.1: Incidence matrix of the projective plane PG(2, 2)
Generators for the automorphism group:
The stabilizer of order 168 is generated by:
\[ g_1 = (3, 5)(4, 6)(8, 9)(12, 13) \] of order 2 and with 6 fixed points.
\[ g_2 = (3, 4)(5, 6)(10, 11)(12, 13) \] of order 2 and with 6 fixed points.
\[ g_3 = (1, 2)(5, 6)(10, 12)(11, 13) \] of order 2 and with 6 fixed points.
\[ g_4 = (1, 3)(2, 4)(7, 8)(11, 12) \] of order 2 and with 6 fixed points.
\[ g_5 = (0, 1)(4, 5)(8, 10)(9, 11) \] of order 2 and with 6 fixed points.

Canonical labeling:
canonical row = 6
canonical orbit number = 0
Flags : ( 0, 1, 2, 7, 10, 11, 14, 19, 20, 22, 24, 26, 30, 31, 34, 36, 39, 41, 44, 46, 47 )

Flag orbits:
orbit length : number of orbits of that length:

\[ 21 \quad 1 \]

Anti-Flag orbits:
orbit length : number of orbits of that length:

\[ 28 \quad 1 \]

The following command computes the canonical form and a report of the affine plane
AG(2, 3), which is a configuration 9_412_3.

AG_2_3.c: AG_2_3.inc
\( $(ORBITER) \ -v \ 2 \ \)
\( \ -define \ C \ -combinatorial_objects \ \)
\( \ -file_of_incidence_geometries \ \)
\( \ -AG_2_3.inc \ 9 \ 12 \ 36 \ \)
\( \ -end \ \)
\( \ -with \ C \ -do \ \)
\( \ -combinatorial_object_activity \ \)
\( \ -canonical_form \ \)
\( \ -classification_prefix \ AG_2_3 \ \)
\( \ -label \ AG_2_3 \ \)
Figure 15.2: The affine plane AG(2, 3) is a configuration $9_412_3$.

A bitmap drawing is produced, shown in Figure 15.2. Because the geometry is flag transitive, there is only one color being used for the incidence. In fact, all incidences are in black. The geometry is also anti-flag transitive. This can be seen from the fact that there is only one color in the picture for the smaller boxes, which represent anti-flags. Orbiter also produces the following report of the geometry:
Summary of Orbits

<table>
<thead>
<tr>
<th>Rep</th>
<th>#</th>
<th>Ago</th>
<th>Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>432</td>
</tr>
</tbody>
</table>

Ago :432

Isomorphism type 0 / 1

Isomorphism type 0 / 1 is original object 0 and appears 1 times:
This isomorphism type appears 1 times, namely for the following 1 input objects:
{0}

incidence structure:
( 0, 1, 2, 3, 12, 16, 17, 18, 24, 31, 32, 33, 37, 40, 43, 46, 49, 53, 56, 59, 62, 64, 69, 71, 74, 78, 80, 82, 87, 89, 93, 94, 99, 102, 103, 107 )

Generators for the automorphism group:
The stabilizer of order 432 is generated by:
\( g_1 = (3, 4)(5, 7)(6, 8)(11, 12)(13, 14)(16, 17)(19, 20) \) of order 2 and with 7 fixed points.
\( g_2 = (3, 5)(4, 6)(7, 8)(10, 11)(14, 15)(16, 18)(19, 20) \) of order 2 and with 7 fixed points.
\( g_3 = (1, 3)(2, 4)(7, 8)(9, 10)(14, 16)(15, 19)(18, 20) \) of order 2 and with 7 fixed points.
\( g_4 = (0, 1)(4, 5)(6, 7)(10, 13)(11, 14)(12, 15)(17, 18) \) of order 2 and with 7 fixed points.

Decomposition by combinatorial refinement:

\[
\begin{array}{c|c|c}
\rightarrow & 12_1 \\
\hline
9_0 & 4 \\
\downarrow & 12_1 \\
9_0 & 3 \\
\end{array}
\]
Decomposition by automorphism group:

Canonical labeling:
canonical row = 6
canonical orbit number = 0
Flags : ( 0, 1, 2, 3, 12, 16, 17, 18, 24, 31, 32, 33, 37, 40, 43, 46, 49, 53, 56, 59, 62,
64, 69, 71, 75, 78, 79, 83, 87, 89, 93, 94, 98, 102, 104, 106 )

Flag orbits:
orbit length : number of orbits of that length:

36 1

Anti-Flag orbits:
orbit length : number of orbits of that length:

72 1

It is possible to perform isomorph classification for configurations based on incidence files. Suppose we want to check that the configurations in 10_3 are in fact all non-isomorphic. 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 \textit{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:
\begin{verbatim}
  $(ORBITER) -v 10 \\
  -draw_incidence_structure_description \\
  -width 60 -with_10 6 -end \\
  -define C -combinatorial_objects \\
  -file_of_incidence_geometries 10_3.inc 10 10 30 \\
  -end \\
  -with C -do \\
  -combinatorial_object_activity \\
  -canonical_form \\
\end{verbatim}

440
The report is shown below. It is truncated for reasons of space. Only the first two geometries are shown. Note that the ordering of geometries in the report may be different from the ordering in the input file. This is because the classification program sorts the geometries according to the canonical form. Also, note that the report includes the incidence geometry in the form it is given as well as the tactical decomposition induced by the orbits of the automorphism group.
Summary of Orbits

<table>
<thead>
<tr>
<th>Rep</th>
<th>#</th>
<th>Ago</th>
<th>Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>9</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>1</td>
<td>24</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>1</td>
<td>120</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>1</td>
<td>6</td>
</tr>
</tbody>
</table>

Ago : 2, 3^2, 4^2, 6, 10, 12, 24, 120

Isomorphism type 0 / 10

Isomorphism type 0 / 10 is original object 9 and appears 1 times:
This isomorphism type appears 1 times, namely for the following 1 input objects:
{9}
incidence structure:
( 0, 1, 2, 10, 13, 14, 20, 25, 26, 31, 33, 35, 41, 44, 47, 52, 53, 58, 62, 66, 69, 74, 78, 79, 85, 87, 89, 96, 97, 98 )

Generators for the automorphism group:
The stabilizer of order 3 is generated by:
g_1 = (0, 1, 3)(2, 5, 4)(6, 7, 8)(10, 13, 11)(12, 14, 15)(16, 18, 17) of order 3 and with 2 fixed points.
Decomposition by automorphism group:

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:

```
10161119121317141815
```

Canonical labeling:
- canonical row = 0
- canonical orbit number = 0
- Flags : 0,1,2,15,18,19,24,26,29,33,37,39,40,43,44,50,55,56,61,67,69,72,75,77,82,84,88,91,93,96,

```
14121810131711161915
```

The following command computes the canonical form for the three triangle free configurations 24, found by Abdullah Alazemi. These configurations have 24 points, 24 lines, each line consists of 3 points and each point is on 3 lines.

```
FILE_24_3_TFC_INC="24 24 72\
\n0 1 2 24 27 28 48 53 54 73 79 80 97 105 106 122 131 \\n132 146 157 158 171 175 183 195 203 208 220 225 233 244 \\n258 259 269 272 282 293 300 308 318 324 327 342 354 357 \\
444
```

444
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:

-define F -finite_field -q 3 -end
-define D -design -field F -family PG_2_3 -end
-with D -do
-design_activity
-define C -combinatorial_objects
-file_of_incidence_geometries PG_2_3_inc.txt 13 13 52
-export_flag_orbits
-save_ago
-save_transversal
-report
-prefix PG_2_3
-export_flag_orbits
-show_incidence_matrices
-export_group_GAP
-end
-report
-prefix PG_2_3
-export_flag_orbits
-show_incidence_matrices
-export_group_GAP
-end
-pdflatex PG_2_3_classification.tex
-open PG_2_3_classification.pdf

$ORBITER) -v 2 -draw_matrix
-input_csv_file PG_2_3_object0_TDA_flag_orbits.csv
-secondary_input_csv_file PG_2_3_object0_TDA.csv
-box_width 32 -bit_depth 24
-end
-open PG_2_3_object0_TDA_flag_orbits_draw.bmp
The command

wreath_product_designs_n4_k2_c: wreath_product_designs_n4_k2_inc.txt

$\$(\text{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.GAP \n- -end \n- -end

\text{pdf}l\text{a}t\text{e}x \text{wreath}_4_2.classification.tex

\text{open \text{wreath}_4_2.classification.pdf}

computes the automorphism group of the design on 8 points created in Section 11.5. The group is $\text{Sym}(4) \rtimes \text{Sym}(2)$. The command

wreath_product_designs_n8_k6_c: wreath_product_designs_n8_k6_inc.txt

$\$(\text{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
448
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 \( \text{AG}(2, 3) \).

```
$($\text{ORBITER}$) -v 2 \
  -draw_incidence_structure_description \
  -width 20 -width_10 2 -end \
  -define C -combinatorial_objects \
  -file_of_designs \
  solutions.csv 9 84 3 12 \
  -end \
  -with C -do \
  -combinatorial_object_activity \
  -canonical_form \
  -save_ago \
  -save_transversal \
  -classification_prefix \( \text{LS\_AG\_2\_3} \) \
  -label \( \text{LS\_AG\_2\_3} \) \
  -max_TDO_depth 10 \
  -end \
  -report \
  -prefix \( \text{LS\_AG\_2\_3} \)
```
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:

```latex
\begin{verbatim}
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 -n 1 -field F -v 0 -end \$
  \$ -with P -do \$
  \$ -projective_space_activity \$
  \$ -canonical_form_of_code \$
  \$ -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
\end{verbatim}
```

The code has a semilinear automorphism group of order 6. The following report is written:

**Summary of Orbits**

<table>
<thead>
<tr>
<th>Rep</th>
<th># Ago</th>
<th>Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1 6</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:
\( g_1 = (1, 2) \) of order 2 and with 4 fixed points.
\( g_2 = (0, 1) \) of order 2 and with 4 fixed points.

Generators for the automorphism group as matrix group:
The stabilizer of order 6 is generated by:
\( g_1 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 10 \\ 11 \end{bmatrix} \) of order 2 and with 1 fixed points.
\( g_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 01 \\ 10 \end{bmatrix} \) of order 2 and with 1 fixed points.

Decomposition by combinatorial refinement:
Decomposition by automorphism group:

\[
\begin{array}{c|c}
\rightarrow & 2_1 \\
4_0 & 2 \\
\downarrow & 2_1 \\
4_0 & 3 \\
\end{array}
\]

Canonical labeling:
- canonical row = 3
- canonical orbit number = 1
- Flags : ( 0, 1, 2, 3, 4, 5, 7 )

Flag orbits:
- orbit length : number of orbits of that length:
  - 1 : 1
  - 3 : 2

Anti-Flag orbits:
- orbit length : number of orbits of that length:
  - 1 : 1

The command

```
CODE_RM_3_1_GENMA="
11111111
01010101
00110011
00001111"
```

```
RM_3_1_group:
  ▶ $(ORBITER) -v 2 \\
  ▶ ▶ -define F -finite_field -q 2 -end \\
  ▶ ▶ -define genma -vector -field F -format 4 \\
  ▶ ▶ ▶ -compact $(CODE_RM_3_1_GENMA) \\
  ▶ ▶ -end \\
```

453
computes the automorphism group of the Reed-Muller code, of order 1344. It is the affine group $AGL(3, 2)$. A report is created, showing the automorphism group and the action on $PG(3, 2)$, with the Reed-Muller code distinguished.

The following command creates a drawing of the incidence matrix between points and lines in $PG(3, 2)$, with the Reed-Muller code distinguished:

```bash
RM_3_1_group_and_diagram:
> $(ORBITER) -v 2 \
>   -define F -finite_field -q 2 -end \
>   -define genma -vector -field F -format 4 \
>   -compact $(CODE_RM_3_1_GENMA) \n>   -define P -projective_space -n 3 -field F -v 0 -end \
>   -projective_space_activity \
>   -canonical_form_of_code \
>   "RM_3_1" genma -save Ago -label "RM_3_1" \n>   -classification_prefix "RM_3_1" \n>   -end \
> -end
```

open RM_3_1_group_and_diagram.pdf
The command

```bash
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 -n 3 -field F -v 0 -end \\
  ▶ ▶ -with P -do \\
  ▶ ▶ -projective_space_activity \\
  ▶ ▶ ▶ -canonical_form_of_code \\
  ▶ ▶ ▶ ▶ "RS_6" genma -saveago -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,

\[
\rightarrow \begin{array}{c|c|c}
2850_{12} & l_2 \\
401_0 & 57 & 1 \\
\end{array}
\]

The command

GV_n15_k6_d5_group:

\begin{verbatim}
$\$(ORBITER) -v 20 \\
  -define F -finite_field -q 2 -end \\
  -define genma -vector -field F -format 6 \\
  -compact $(CODE_GV_N15_K6) \\
  -end \\
  -define P -projective_space -n 5 -field F -v 0 -end \\
  -with P -do \\
\end{verbatim}
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\_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:

```verbatim
$(ORBITER) -v 2 \
  define Gamma -graph -cycle 13 -end \n  with Gamma -do \n  graph\ntheoretic\nactivity\nautomorphism\ngroup \nend \n```

The output is two files: The first one, Cycle_13_group.makefile is a makefile containing an Orbiter command to create the automorphism group: The second file is Cycle_13_gens.csv, which contains the permutation representation of the group, and which is needed for the makefile.

The next command computes the automorphism group of the chain graph with respect to the partition $(2, 3, 2)$.

Chain_232_aut:

```verbatim
$(ORBITER) -v 2 \
  define P1 -vector -dense 2,3,2 -end \n  define P2 -vector -dense 2,3,2 -end \n  define Gamma -graph \n  chain_graph P1 P2 \n  with Gamma -do \n  graph\ntheoretic\nactivity\nautomorphism\ngroup \nend

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:

\[(12, 13),\]
\[(3, 4),\]
\[(2, 3),\]
Juntila and Kaski maintain a collection of graphs that can be used as test cases. The graphs are stored in Dimacs format and can be read in through the Orbiter \texttt{-load_dimacs} command. For instance, the following command computes the automorphism group of the Levi graph of the desarguesian projective plane of order 16:

\begin{verbatim}
JK_graph_pp16_1:
   $(ORBITER) -v 2 \
   -define Gamma -graph -load_dimacs \
   ../JUNTTILA_KASKI/benchmarks/pp/pp16-1 \
   -end \
   -with Gamma -do \
   -graph_theoretic_activity -save -end \
   -with Gamma -do \
   -graph_theoretic_activity -automorphism_group -end \
\end{verbatim}

The command shows a group of order 34217164800. As a measurement of the complexity, the number of backtrack nodes in Nauty is recorded:

\begin{verbatim}
nb_backtrack1 = 6
nb_backtrack2 = 134
nb_backtrack3 = 134
nb_backtrack4 = 1
\end{verbatim}

Here, \texttt{nb_backtrack1} is the number of calls to \texttt{firstpathnode}, \texttt{nb_backtrack2} is the number of calls to \texttt{othernode}, \texttt{nb_backtrack3} is the number of calls to \texttt{processnode},
nb_backtrack4 is the number of calls to firstterminal. These are the four recursive functions in Nauty.

Unfortunately, the complexity of graph isomorphism is not well-understood. We can see this here. While the first projective plane of order 16 can be handled relatively easily, the second one causes problems. The following command hardly finishes:

```bash
JK_graph_pp16_2:
  $ (ORBITER) -v 2 \
  -define Gamma -graph -load_dimacs \
  -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.

```bash
JK_graph_sts_13:
  $ (ORBITER) -v 2 \
  -define Gamma -graph -load_dimacs \
  -define Gamma -graph -load_dimacs ../JUNTTILA_KASKI/benchmarks/srg/sts-13 \
  -end \
  -with Gamma -do \
  -graph_theoretic_activity -save -end \
  -with Gamma -do \
  -graph_theoretic_activity -automorphism_group -end \
  make ORBITER_PATH=$(ORBITER_PATH) -f sts-13_group.makefile sts-13
```

The automorphism group has order 39 and is generated by:

\[
(1, 25, 16)(2, 18, 20)(3, 7, 15)(4, 13, 11)(5, 6, 17) \\
(9, 12, 19)(10, 14, 24)(21, 23, 22), \\
(0, 1, 2)(3, 4, 5)(7, 8, 9)(11, 12, 13)(14, 16, 18) \\
(15, 17, 19)(20, 22, 24)(21, 23, 25)
\]
Graphs can be created from groups by means of orbitals. An orbital is an orbit of a permutation group $G$ on the set of pairs. Here is an example. We start from the Coxeter-Tits graph on 315 vertices, whose automorphism group is the Hall-Janko group $J_2 : 2$. We first read the graph from file `halljanko315.csv` and compute the automorphism group using Nauty:

```bash
HJ_aut:
▶ $(ORBITER) -v 6 \\n▶ ▶ -define G -graph \\
▶ ▶ ▶ -load_csv_no_border \n▶ ▶ ▶ 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:

```bash
HJ_group_and_orbits:
▶ $(ORBITER) -v 2 \\
▶ ▶ -define Control -poset_classification_control \\
▶ ▶ ▶ -W \\
▶ ▶ ▶ -problem_label HJ_orbits \\
▶ ▶ ▶ -depth 2 \\
▶ ▶ -end \\
▶ ▶ -define gens -vector -file \\
▶ ▶ ▶ halljanko315_gens.csv -end \\
▶ ▶ -define G -permutation_group \\
▶ ▶ ▶ -bsgs halljanko315 "File\_halljanko315" \\
▶ ▶ ▶ 315 1209600 "0,1,2" 6 gens \\
▶ ▶ -end \\
▶ ▶ -define Orb -orbits -group G \\
▶ ▶ ▶ -on_subsets 2 Control \\
▶ ▶ -end
```

There are 4 orbits on pairs. We decide to pick the fourth orbit to create a new graph. Because indexing is zero-based, we give the orbit index of 3:

```bash
HJ_orbital_graph_3:
▶ $(ORBITER) -v 2 \\
▶ ▶ -define gens -vector -file \\
▶ ▶ ▶ halljanko315_gens.csv -end \\
```

462
The graph is regular of degree 64.

The next command computes the automorphism group of the collinearity graph of the $Q(4,2)$ quadric.

```
PGO_5_2_graph_group: 0_5_2.incidence_matrix.csv
```

```
$ (ORBITER) -v 3 \\
```

```
> -define Inc -vector -file 0_5_2.incidence_matrix.csv -end \\
```

```
> -define Gamma -graph -collinearity_graph Inc -end \\
```

```
> -with Gamma -do \\
```

```
> -graph.theoretic_activity \\
```

```
> -automorphism_group \\
```

```
> -end \\
```

```
> -with Gamma -do \\
```

```
> -graph.theoretic_activity \\
```

```
> -eigenvalues \\
```

```
> -end
```

\[ \text{pdf\LaTeX} \text{collinearity\_graph\_eigenvalues.tex} \]

```
open collinearity\_graph\_eigenvalues.pdf
```

The group is PGO(5,2) of order 720. The command creates the group as a permutation group on the 15 vertices of the graph. The group is no longer treated as a matrix group.
15.8 Canonical Forms of Quartic Curves

We wish to study the automorphism groups of certain quartic curves introduced by Edge. We start by creating a cheat sheet of the field $\mathbb{F}_{17}$

```bash
F_17_edge:
  $(ORBITER) -v 3 \n  -define F -finite_field -q 17 -end \n  -with F -do -finite_field_activity \n  -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.

```bash
Edge_curve_17_nauty:
  $(ORBITER) -v 3 \n  -define C -combinatorial_objects \n  -file_of_points Edge_q17.txt \n  -end \n  -define F -finite_field -q 17 -end \n  -define P -projective_space -n 2 -field F -v 0 -end \n  -with C -do \n  -combinatorial_object_activity \n  -canonical_form_PG P \n  -classification_prefix Edge_curve_q17 \n  -label Edge_curve_q17 \n  -save_ago \n  -save_transversal \n  -max_TDQ_depth 10 \n  -end \n  -report \n  -prefix Edge_curve_q17 \n  -export_flag_orbits \n  -show_TDQ \n  -show_TDA \n  -dont_show_incidence_matrices \n  -export_group_GAP \n  -end
  pdflatex Edge_curve_q17_classification.tex
  open Edge_curve_q17_classification.pdf
  $(ORBITER) -v 2 -draw_matrix \n  -input_csv_file Edge_curve_q17_object0_TDA_flag_orbits.csv
```

464
Using the generators that have just been computed, we can recreate the group of the quartic curve:

Edge\textunderscore curve\_17\textunderscore group: $(\text{ORBITER})-v\ 3$
\[ \text{define } \text{G} \ -\text{linearMixed} \ -\text{PGL} \ 3 \ 17 \]
\[ \text{subgroup} \text{by} \text{generators} \ "\text{Stab} \text{Edge}\" \ "24\" \ 3 \]
\[ "1,0,0,0,13,0,0,0,4" \]
\[ "1,0,0,0,0,16,0,16,0" \]
\[ "0,1,16,2,4,4,15,4,4" \]
\[ \text{end} \]
\[ \text{with} \ \text{G} \ -\text{do} \]
\[ \text{group} \text{theoretic} \text{activities} \]
\[ \text{print} \text{elements} \text{tex} \]
\[ \text{group} \text{table} \]
\[ \text{report} \]
\[ \text{end} \]

\texttt{pdflatex } \texttt{PGL\_3\_17\textunderscore Subgroup\_Stab\_Edge\_24\textunderscore report.tex}
\texttt{open PGL\_3\_17\textunderscore Subgroup\_Stab\_Edge\_24\textunderscore report.pdf}
Chapter 16

Interfaces

16.1 Graphical Output

Orbiter can produce graphical output in a variety of formats:

1. TikZ / Latex [67],  
2. Metapost [34],  
3. Bitmap files (.bmp) [70],  
4. Povray, see Section 16.2.

Bitmaps can be created using the \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>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-input_csv_file</td>
<td>csv-file</td>
<td>Specify the input csv file</td>
</tr>
<tr>
<td>-partition</td>
<td>$w \ R \ C$</td>
<td>Specify a partition $R$ of rows and $C$ of columns. Use separating lines of with $w$.</td>
</tr>
<tr>
<td>-box_width</td>
<td>$w$</td>
<td>Use $w$ pixels per matrix entry.</td>
</tr>
<tr>
<td>-bit_depth</td>
<td>$d$</td>
<td>Use color bit depth of $d$ bits ($d = 8$ or $d = 24$).</td>
</tr>
<tr>
<td>-invert_colors</td>
<td></td>
<td>Use an inverted color scheme.</td>
</tr>
</tbody>
</table>

Table 16.1: Commands to Create Bitmap Graphics

467
The finite field activity `-cheat_sheet_GF` creates the file

\[
\text{GF}_q7\text{addition_table.csv}
\]

which is used as the input for the second command. The file content is:

<table>
<thead>
<tr>
<th>Row</th>
<th>C0</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>C5</th>
<th>C6</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

The second command creates the diagram in Figure 16.1. The `-partition` command is used to define an outline of width 3 pixes. The all-in-one partition 7 is used as both row-partition and column-partition.

The planes PG(2, q) admit a cyclic automorphism group known as the Singer cycle. The command

PG.2.4_cyclic_incma:

\[
\text{PG}_2\text{4_cyclic_incma:}
\]

\[
\text{PG}_2\text{4_cyclic_incma:}
\]
The Singer cycle is the projectivity defined by the companion matrix of an irreducible polynomial. We may pick the irreducible polynomial $$x^2 + x + \omega$$ over $$\mathbb{F}_4$$. The associated Singer cycle is the projectivity defined by the companion matrix

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

The poset classification algorithm from Sections 6.3 and 6.4 computes partially ordered sets. The posets are created using the -draw_poset option in the poset classification control command package, see Table 6.2. 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

```plaintext
open PG_2.4_singer_incma_cyclic_draw.bmp
```

produces a cyclically ordered incidence matrix of the plane PG(2, 4), shown in Figure 16.2.
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-file</code></td>
<td>fname</td>
<td>Use the given file name for output files.</td>
</tr>
<tr>
<td><code>-xin</code></td>
<td>a</td>
<td>Assume input $x$-coordinates are in the interval $[0, a]$. Default value: 10000.</td>
</tr>
<tr>
<td><code>-yin</code></td>
<td>a</td>
<td>Assume input $y$-coordinates are in the interval $[0, a]$. Default value: 10000.</td>
</tr>
<tr>
<td><code>-xout</code></td>
<td>a</td>
<td>Assume output $x$-coordinates are in the interval $[0, a]$. Default value: 1000000.</td>
</tr>
<tr>
<td><code>-yout</code></td>
<td>a</td>
<td>Assume output $y$-coordinates are in the interval $[0, a]$. Default value: 1000000.</td>
</tr>
<tr>
<td><code>-spanning_tree</code></td>
<td></td>
<td>Place nodes according to a spanning tree. Default value: off.</td>
</tr>
<tr>
<td><code>-circle</code></td>
<td></td>
<td>Circle all nodes. Default value: on.</td>
</tr>
<tr>
<td><code>-corners</code></td>
<td></td>
<td>Draw corners at the outside of the drawing. Default value: off.</td>
</tr>
<tr>
<td><code>-rad</code></td>
<td>r</td>
<td>Use radius $r$ for drawing circles around nodes. Default value: 50.</td>
</tr>
<tr>
<td><code>-embedded</code></td>
<td></td>
<td>Create latex headers for stand-alone latex files. Default value: off.</td>
</tr>
<tr>
<td><code>-sideways</code></td>
<td></td>
<td>Create latex figure sideways. Default value: off.</td>
</tr>
<tr>
<td><code>-label_edges</code></td>
<td></td>
<td>Label the edges in Schreier trees. Default value: off.</td>
</tr>
<tr>
<td><code>-x_stretch</code></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><code>-y_stretch</code></td>
<td>s</td>
<td>Apply $y$-axis scaling by a factor of $s$. Default value: $s = 1.0$. This option does not affect the drawing of Schreier trees.</td>
</tr>
</tbody>
</table>

Table 16.2: Drawing Options for Layered Graph Files (Part 1)
Figure 16.2: A cyclic ordering of the incidence matrix of PG(2, 4)

<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-scale</td>
<td>$s$</td>
<td>Use tikz global scale-factor of $s$. Default value: $s = 0.45$.</td>
</tr>
<tr>
<td>-line_width</td>
<td>$s$</td>
<td>Set tikz line width to $s$. Default value: $s = 1.5$.</td>
</tr>
<tr>
<td>-nodes_empty</td>
<td></td>
<td>Draw nodes empty. Do not label. Default value: off.</td>
</tr>
<tr>
<td>-select_layers</td>
<td>$S$</td>
<td>Draw layers whose index is given in the list $S$ only.</td>
</tr>
<tr>
<td>-paths_in_between</td>
<td>$l_1$ $i_1$ $l_2$ $i_2$</td>
<td>Draw all paths from node $(l_1, i_1)$ to node $(l_2, i_2)$. Here, $(l, i)$ is the $i$-th node at layer $l$ (counting from zero). Delete all other edges between layers $l_1$ and $l_2$.</td>
</tr>
</tbody>
</table>

Table 16.3: Drawing Options for Layered Graph Files (Part 2)
Figure 16.3: The first basic orbit of PGL(4, 2) as a subgroup of PGO⁺(6, 2)

```
PGL_4_2_Wedge_4_0_graphical_output:
  $ (ORBITER) -v 4 \n  -define G -linear_group -PGL 4 2 \n  -wedge_detached \n  -end \n  -with G -do \n  -group_theoretic_activity \n  -report \n  -end \n  pdflatex PGL_4_2_Wedge_4_2_detached_report.tex \n  open PGL_4_2_Wedge_4_2_detached_report.pdf
```

produces a report about this group action. Figure 16.3 shows the first basic orbit in the stabilizer chain of the group in that action.

The command

```
schreier_tree_graphical_output:
  $ (ORBITER) -v 4 \n  -draw_options \n  -yout 500000 \n  -radius 15 -nodes_empty \n  -line_width 0.5 -y_stretch 0.25 \n  -embedded \n  -end \n  -define G -linear_group -PGL 4 2 -end \n  -define Orb -orbits -group G \n```

472
Figure 16.4: A 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 [59]. This is a multi step process: A 3D scene is defined through orbiter commands. Next, Orbiter produces Povray files. After that, the povray files are processed through povray, and turned into graphics files (png), called frames. The frames can be turned into a video by using tools like ffmpeg (see Section 16.3). By default, an rotational animation is produced.

The Orbiter Povray interface requires some general information about the animation, the camera position, the boundary box for clipping, the font size for text and others. Tables 16.4-16.5 list the commands to control the 3D-povray frontend. The main part in a 3D graphics is the scene description. This tells the system what will be in the picture. A scene is composed of objects. Various types of objects are available: points, lines, planes, faces, algebraic surfaces, reguli, 3D-text, and others. Some complex objects are predefined, for instance the Hilbert, Cohn-Vossen surface. Once the objects are defined, output commands can be invoked to draw them in various colors and with various options. At times, there are many objects in one scene. In order to make drawing easier, it is possible to group objects. All objects in a group must have the same type. One group of object can be drawn with one command. Tables 16.6 and 16.7 summarize the Orbiter commands to build objects of a 3D scene.

Building the scene itself does not create any graphical output. To this end, the commands in Table 16.8 are used. Each of these commands applies to a group of objects of the same kind. Groups of objects are created using the commands in Table 16.7 which start with `group_of`.

Here is a simple example which combines scene building and graphical output. The example creates a cube with vertices, edges and faces:

cube:

```
\$ (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"
```
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-do_not_rotate</td>
<td></td>
<td>No rotation. By default, the animation consists of a full rotation around a vertical axis.</td>
</tr>
<tr>
<td>-rotate_about_z_axis</td>
<td></td>
<td>Rotate around z-axis.</td>
</tr>
<tr>
<td>-rotate_about_111</td>
<td></td>
<td>Rotate around (1,1,1)-axis (default).</td>
</tr>
<tr>
<td>-rotate_about_custom_axis</td>
<td>axis</td>
<td>Rotate around a custom axis. The axis is specified as a vector of length 3.</td>
</tr>
<tr>
<td>-boundary_none</td>
<td></td>
<td>Remove the clipping.</td>
</tr>
<tr>
<td>-boundary_box</td>
<td></td>
<td>Clip using a box shape.</td>
</tr>
<tr>
<td>-boundary_sphere</td>
<td></td>
<td>Clip using a sphere (default).</td>
</tr>
<tr>
<td>-font_size</td>
<td>s</td>
<td>Set font size to s.</td>
</tr>
<tr>
<td>-stroke_width</td>
<td>s</td>
<td>Set text depth to s.</td>
</tr>
<tr>
<td>-omit_bottom_plane</td>
<td></td>
<td>Remove the bottom plane.</td>
</tr>
<tr>
<td>-W</td>
<td>w</td>
<td>Set output dimension to w pixels wide.</td>
</tr>
<tr>
<td>-H</td>
<td>h</td>
<td>Set output dimension to h pixels height.</td>
</tr>
<tr>
<td>-nb_frames</td>
<td>n</td>
<td>Set number of frames to n. One revolution around the axis is split into n frames.</td>
</tr>
<tr>
<td>-zoom</td>
<td>r a_s a_t c_s c_t</td>
<td>Set zoom angle and clipping with in round r to change from a_s to a_t and from c_s to c_t, respectively.</td>
</tr>
<tr>
<td>-pan</td>
<td>r F T C</td>
<td>In round r, pan the camera from location F to location T in a rotational movement with center at C. Each of F, T, C are three dimensional coordinates.</td>
</tr>
<tr>
<td>-pan_reverse</td>
<td>r F T C</td>
<td>Same as -pan, but camera movement is in opposite order.</td>
</tr>
<tr>
<td>-no_background</td>
<td></td>
<td>Remove background.</td>
</tr>
<tr>
<td>-no_bottom_plane</td>
<td>r</td>
<td>Remove bottom plane in round r.</td>
</tr>
<tr>
<td>-camera</td>
<td>r S C L</td>
<td>In round r, set camera location at C, sky at S and pointing towards L. Each of S, C, L are three-dimensional coordinate vectors.</td>
</tr>
</tbody>
</table>

Table 16.4: Options for Orbiter 3D-graphics (Part 1)
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-clipping</td>
<td>$r\ c$</td>
<td>In round $r$, set clipping radius to $c$.</td>
</tr>
<tr>
<td>-text</td>
<td>$r\ a\ \text{text}$</td>
<td>In round $r$, produce running text $\text{text}$ with sustain value $a$.</td>
</tr>
<tr>
<td>-label</td>
<td>$r\ s\ a\ g\ \text{text}$</td>
<td>In round $r$, produce running text $\text{text}$ with start value $s$, sustain $s$ and gravity $g$.</td>
</tr>
<tr>
<td>-latex</td>
<td>$r\ s\ a\ \text{praemable}\ g\ \text{text}\ l\ \text{fname}$</td>
<td>In round $r$, produce running latex text $\text{text}$ with start value $s$, sustain $s$ and gravity $g$. Put $\text{praemable}$ in the latex source code. Use $\text{fname}$ for the latex file names (no extension).</td>
</tr>
<tr>
<td>-global_picture_scale</td>
<td>$d$</td>
<td>Set scaling factor to $d$.</td>
</tr>
<tr>
<td>-picture</td>
<td>$r\ d\ \text{fname}\ \text{options}$</td>
<td>In round $r$, place picture $\text{fname}$ scaled by $d$ using options.</td>
</tr>
<tr>
<td>-picture</td>
<td>$r\ d\ \text{fname}\ \text{options}$</td>
<td>In round $r$, place picture $\text{fname}$ scaled by $d$ using options.</td>
</tr>
<tr>
<td>-look_at</td>
<td>$L$</td>
<td>Override camera look-at value to $L$. $L$ is a three-dimensional vector.</td>
</tr>
<tr>
<td>-default_angle</td>
<td>$a$</td>
<td>Set default camera angle to $a$.</td>
</tr>
<tr>
<td>-clipping_radius</td>
<td>$f$</td>
<td>Set default clipping radius to $f$.</td>
</tr>
<tr>
<td>-scale_factor</td>
<td>$s$</td>
<td>Set default scale factor to $s$.</td>
</tr>
<tr>
<td>-line_radius</td>
<td>$s$</td>
<td>Set default line radius to $s$.</td>
</tr>
</tbody>
</table>

Table 16.5: Options for Orbiter 3D-graphics (Part 2)
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-cubic_lex</td>
<td>coeffs</td>
<td>Cubic surface given by 20 coefficients in lexicographic ordering</td>
</tr>
<tr>
<td>-cubic_orbiter</td>
<td>coeffs</td>
<td>Cubic surface given by 20 coefficients in Orbiter ordering</td>
</tr>
<tr>
<td>-cubic_Goursat</td>
<td>A B C</td>
<td>Cubic surface with tetrahedral symmetry given by 3 Goursat coefficients as $Axyz + B(x^2 + y^2 + z^2) + C = 0$</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>i₁ i₂</td>
<td>Create a line through two points</td>
</tr>
<tr>
<td>-line_through_point_with_direction</td>
<td>x y z uₓ uᵧ uₗ</td>
<td>Create a line through a point ((x, y, z)) with a given direction ((uₓ, uᵧ, uₗ))</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\text{ points, }30\text{ edges, }12\text{ faces}))</td>
</tr>
<tr>
<td>-Hilbert_Cohn_Vossen_surface</td>
<td></td>
<td>Create the Hilbert, Cohn-Vossen surface ((1\text{ cubic surface, }45\text{ tritangent planes, }27\text{ lines}))</td>
</tr>
<tr>
<td>-obj_file</td>
<td>fname</td>
<td>Read points and faces from the given .obj file</td>
</tr>
<tr>
<td>-group_of_things</td>
<td>list</td>
<td>Create a group of things from the given list</td>
</tr>
<tr>
<td>-group_of_things_with_offset</td>
<td>list offset</td>
<td>Create a group of things from the given list, each value is increase by offset</td>
</tr>
<tr>
<td>-group_of_things_as_interval</td>
<td>a b</td>
<td>Create a group of things indexed by the interval (a, \ldots, a + b - 1)</td>
</tr>
<tr>
<td>-group_of_things_as_interval_with_exceptions</td>
<td>a b ex</td>
<td>Create a group of things indexed by the interval (a, \ldots, a + b - 1) with the exceptional elements in the list ex removed</td>
</tr>
<tr>
<td>-group_of_all_points</td>
<td></td>
<td>Create a group of things from all points currently defined</td>
</tr>
<tr>
<td>-group_of_all_faces</td>
<td></td>
<td>Create a group of things from all faces currently defined</td>
</tr>
<tr>
<td>-group_subset_at_random</td>
<td>i f</td>
<td>Create a group of things from the existing group (i) by picking a random subset with probability (f)</td>
</tr>
<tr>
<td>-create_regulus</td>
<td>i N</td>
<td>Create a regulus for quadric (i) with (N) lines</td>
</tr>
</tbody>
</table>

Table 16.7: Scene definition commands (part 2)
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-spheres</td>
<td>$i \ r \ \text{prop}$</td>
<td>For each element in point group $i$, create a sphere of radius $r$ with given Povray properties.</td>
</tr>
<tr>
<td>-cylinders</td>
<td>$i \ r \ \text{prop}$</td>
<td>For each element in edge group $i$, create a cylinder of radius $r$ with given Povray properties.</td>
</tr>
<tr>
<td>-prisms</td>
<td>$i \ d \ \text{prop}$</td>
<td>For each element in face group $i$, create a prism of half-thickness $d$ with given Povray properties.</td>
</tr>
<tr>
<td>-planes</td>
<td>$i \ \text{prop}$</td>
<td>For each element in plane group $i$, create a plane with given Povray properties.</td>
</tr>
<tr>
<td>-lines</td>
<td>$i \ r \ \text{prop}$</td>
<td>For each element in line group $i$, create a line of radius $r$ with given Povray properties.</td>
</tr>
<tr>
<td>-cubics</td>
<td>$i \ \text{prop}$</td>
<td>For each element in group $i$ of cubics, create a surface with given Povray properties.</td>
</tr>
<tr>
<td>-quadrics</td>
<td>$i \ \text{prop}$</td>
<td>For each element in group $i$ of quadrics, create a surface with given Povray properties.</td>
</tr>
<tr>
<td>-quartics</td>
<td>$i \ \text{prop}$</td>
<td>For each element in group $i$ of quartics, create a surface with given Povray properties.</td>
</tr>
<tr>
<td>-octics</td>
<td>$i \ \text{prop}$</td>
<td>For each element in group $i$ of octics, create a surface with given Povray properties.</td>
</tr>
<tr>
<td>-texts</td>
<td>$i \ d \ s \ \text{prop}$</td>
<td>For each element in group $i$ of labels, create a text element with half-thickness $d$ and size $s$ with given Povray properties.</td>
</tr>
</tbody>
</table>

Table 16.8: Graphical output commands
This command instructs Orbiter to create 30 povray files (extension .pov), one for each frame of a rotating scene. The scene contains a cube whose vertices are shown in chrome, whose edges are in red, and whose faces are yellow and transparent. The cube turns around a vertical axis of symmetry. Here is the first frame of the result:

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
v 1 -1 -1
v -1 1 -1
v 1 1 -1
v -1 -1 1
v 1 -1 1
v -1 1 1
```
The monkey saddle is a cubic surface, given by the equation

\[ z = x^3 - 3xy^2 \]

The next example plots the surface knowns as the monkey saddle. The tangent plane at \((0,0,0)\) is drawn as well. An animation is created by rotating the scene around the \(z\)-axis.

```
MONKEY_SADDLE_CUBIC="1,0,0,0,-3,0,0,0,0,0,0,0,0,0,0,0,-1,0"
```

```
monkey:
  $ (ORBITER) -v 2 -povray \
  -round 0 -nb_frames_default 30 \n  -output_mask monkey_%d_%03d.pov \n  -video_options -W 1024 -H 768 \n  -global_picture_scale 0.8 \n  -default_angle 75 \n  -clipping_radius 0.8 \n  -camera 0 "0,0,1" "1,1,0.5" "0,0,0" \n  -rotate_about_z_axis \n  -end \
  -scene_objects \
  -cubic_lex $(MONKEY_SADDLE_CUBIC) \
  -plane_by_dual_coordinates "0,0,1,0" \n  -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:

![Image of the Eckardt surface]

The Eckardt surface is given by the equation

\[
\frac{5}{2}xyz - (x^2 + y^2 + z^2) + 1 = 0.
\]

We use the following code to plot the surface and the lines on it. The Schläfli labeling of the lines is indicated.

Eckardt:

```plaintext
$($ORBITER) -v 2 -povray \
  -round 0 -nb_frames_default 30 \n  -output_mask Eckardt_%d_%03d.pov \n  -video_options -W 1024 -H 768 \n  -global_picture_scale 0.9 \n  -default_angle 75 \n  -clipping_radius 2.4 \n  -camera 0 "1,1,1" "-3,1,3" "0.12,0.12,0.12" \n  -end \n  -scene_objects \n  -Hilbert_Cohn_Vossen_surface \n  -group_of_things "0" \n  -cubics 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_with_exceptions 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  -label 0 "a1"
```

482
Figure 16.5 shows the final product.

The Endrass octic [26] is the algebraic surface given by the equation

\[ x^8 = 64 \left( -w^2 + x^2 \right) \left( -w^2 + y^2 \right) \left( (x+y)^2 - 2w^2 \right) \left( (x-y)^2 - 2w^2 \right) - \left( -4 \left( 1 + \sqrt{2} \right) \left( x^2 + y^2 \right)^2 + 8 \left( 2 + \sqrt{2} \right) z^2 + 2 \left( 2 + 7\sqrt{2} \right) w^2 \right) \left( x^2 + y^2 \right) - 16z^4 + 8 \left( 1 - 2\sqrt{2} \right) z^2w^2 - \left( 1 + 12\sqrt{2} \right) w^4 \]

The following Orbiter command creates a povray graphics of the octic, shown in Figure 16.6:
Figure 16.5: The Eckardt surface
endrass8:
  ▶ $(ORBITER) -v 2 -povray \\
  ▶ -round 0 -nb_frames_default 30 \\
  ▶ -output_mask endrass_octic_lex.%(d)_%(3d).pov \\
  ▶ -video_options -W 1024 -H 768 \\
  ▶ -global_picture_scale 0.75 \\
  ▶ -default_angle 75 \\
  ▶ -clipping_radius 3.7 \\
  ▶ -no_bottom_plane \\
  ▶ -camera 0 "1,1,1" "6,6,3" "0,0,0" \\
  ▶ -rotate_about_111 \\
  ▶ -end \\
  ▶ -scene_objects \\
  ▶ ▶ -line_through_two_points_recentered_from_csv_file \\
  ▶ ▶ ▶ coordinate_grid.csv \\
  ▶ ▶ ▶ -group_of_things "0" \\
  ▶ ▶ ▶ -group_of_things "1" \\
  ▶ ▶ ▶ -group_of_things "2" \\
  ▶ ▶ ▶ -group_of_things_as_interval 3 39 \\
  ▶ ▶ ▶ -lines 0 0.15 $(COLOR_RED_SHINY) \\
  ▶ ▶ ▶ -lines 1 0.15 $(COLOR_GREEN_SHINY) \\
  ▶ ▶ ▶ -lines 2 0.15 $(COLOR_BLUE_SHINY) \\
  ▶ ▶ ▶ -lines 3 0.05 $(COLOR_BLACK_SHINY) \\
  ▶ ▶ ▶ -octic_lex_165 $(ENDRASS_OCTIC_LEX_165) \\
  ▶ ▶ ▶ -plane_by_dual_coordinates "0,0,1,0" \\
  ▶ ▶ ▶ -group_of_things "0" \\
  ▶ ▶ ▶ -group_of_things "0" \\
  ▶ ▶ ▶ -octics 4 $(SURFACE_COLOR_SEETHROUGH) \\
  ▶ ▶ ▶ -planes 5 "texture{ pigment{ color Blue transmit 0.5 } \ finish { diffuse 0.9 phong 1 }}"
  ▶ ▶ -scene_objects_end \\
  ▶ ▶ -povray_end \\
  ▶ -rm -rf POV \\
  ▶ mkdir POV
Figure 16.6: The Endrass Octic

```
▶ mv endrass_octic_0_*.pov POV
▶ mv makefile_animation POV
```

This illustration includes coordinate axes and the $x,y$-plane.
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>s l mask</td>
<td>Specify the input file names by running a printf command with the given mask applied to the index $i$ where $i$ goes from $s$ to $s+l-1$. This option can be repeated.</td>
</tr>
<tr>
<td>-step</td>
<td>s</td>
<td>Increment the index in steps of size $s$.</td>
</tr>
<tr>
<td>-o</td>
<td>mask</td>
<td>Create the output file using the given mask.</td>
</tr>
<tr>
<td>-output_starts_at</td>
<td>i</td>
<td>Start output file indices at $i$ (default is 0).</td>
</tr>
</tbody>
</table>

Table 16.9: Prepare frames commands

### 16.3 Creating Animations

Orbiter can be used to create animations. This relies on the software ffmpeg. In a first step, all frames (i.e. individual graphics files) are created using Orbiter’s povray interface. After that, the frames are used to create the animation. In order to use ffmpeg, the frames should have a uniform file naming scheme, using a consecutive numbering to arrange the files in order. This is achieved by using a printf style mask, with %d representing the number of the current frame. In order to do so, Orbiter can be used to copy and rename files. A temporary directory can be used to collect the files. The Orbiter command `prepare_frames` can be used. For a list of commands, see Tables 16.9. For instance, the command

```
monkey_video:
▷ - rm -r FRAMES
▷ - mkdir FRAMES
▷ - rm monkey.mp4
▷ $(ORBITER) \\
▷   -prepare_frames \\
▷     -i 0 30 monkey_0_%03d.png \\
▷     -output_starts_at 0 \\
▷     -o FRAMES/frame%04d.png \\
▷     -end
▷ ffmpeg -r 5 -f image2 -i FRAMES/frame%04d.png \\
▷     -f mp4 -q:v 0 -vcodec mpeg4 monkey.mp4
```

creates a video `monkey.mp4` from a set of 30 files. The individual filenames are created using the printf format string `monkey_0_%03d.png`, with an integer index that is drawn from the interval $[0, 29]$. The part that starts with a percent sign and ends with a “d” character defines the way in which the integer is formatted. The number three before the “d” indicates that three characters will be printed. The zero indicates the use of leading zeros. So, the first file would be `monkey_0_000.png` and the very last file is `monkey_0_029.png`. The description of the printf format string can be found in the documentation of the C standard library [39].

487
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

\[ x = r \sin \left( at + c \right), \quad y = r \sin \left( bt \right), \quad a, b, c, r \in \mathbb{R}. \]

The terms 

\[ r \sin \left( at + c \right), \quad r \sin \left( bt \right) \]

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:

\[
\sin(x) \leftrightarrow \text{push } x \text{ sin}, \\
a + b \leftrightarrow \text{push } a \text{ push } b \text{ add}, \\
a \cdot b \leftrightarrow \text{push } a \text{ push } b \text{ mult}. 
\]

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:

\[
\text{push } t \text{ push } a \text{ mult push } c \text{ add sin push } r \text{ mult return } \]

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

\[-\text{const } a \ 3 \ b \ 2 \ c \ 1.57 \ r \ 7 \ -\text{const_end} \]
The input variable is defined using the line

```
-var t -var_end
```

The sequence

```
-smooth_curve "lissajous" 0.07 2000 15 0 18.85
```

defines the name of the output file, the fact that two consecutive points are never further than \( \epsilon = 0.07 \) away, the fact that points that are 15 or more away from the origin should be ignored, and the fact that the variable \( t \) loops over the range \([0, 18.85] \) with a default of 2000 steps. The evaluator automatically reduces the step-size if consecutive image points are more than \( \epsilon \) apart. The code to produce the plot is

```
lijasous_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
```

489
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 \( \alpha^0, \alpha^2, \alpha^4 \).

misc_select:
```
$ (ORBITER) -v 3 \\
> -define F -finite_field -q 7 -end \\
> -with F -do -finite_field_activity -cheat_sheet_GF -end \\
> $ (ORBITER) -v 4 -csv_file_select_rows_and_cols \\
> GF_q7_multiplication_table_reordered.csv \\
> "0,2,4" "0,2,4"
```

The even powers of \( \alpha \) create a multiplicative subgroup. Figure 17.1 shows the table of the multiplicative group \( \mathbb{F}_7^* \) and the subgroup of squares (compare with Figure 3.2 in Section 3.2). Here is the file `GF_q7_multiplication_table_reordered.csv`

<p>| 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      |</p>
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-csv_file_select_rows</code></td>
<td>fname $R$</td>
<td>Selects rows listed in $R$ from the csv-file fname.</td>
</tr>
<tr>
<td><code>-csv_file_select_cols</code></td>
<td>fname $R$</td>
<td>Selects columns listed in $R$ from the csv-file fname.</td>
</tr>
<tr>
<td><code>-csv_file_select_rows_and_cols</code></td>
<td>fname $R$ $C$</td>
<td>Selects rows listed in $R$ and columns listed in $C$ from the csv-file fname.</td>
</tr>
<tr>
<td><code>-csv_file_join</code></td>
<td>fname col-label</td>
<td>Joins csv file fname according to column with label col-label. This option is given once for each file that should be joined.</td>
</tr>
<tr>
<td><code>-csv_file_latex</code></td>
<td>fname</td>
<td>Produces a latex table from the given csv-file.</td>
</tr>
<tr>
<td><code>-store_as_csv_file</code></td>
<td>fname $m$ $n$ $L$</td>
<td>Stores the data in $L$ to a csv file. The data is an $m \times n$ matrix in row-major ordering.</td>
</tr>
</tbody>
</table>

Table 17.1: Miscellaneous Orbiter Commands

Figure 17.1: Cyclic multiplication table of $\mathbb{F}_7$ and subgroup of squares
and next the file that is created by selecting rows and columns 0, 2, 4:

<table>
<thead>
<tr>
<th>Row</th>
<th>&quot;C0&quot;</th>
<th>&quot;C2&quot;</th>
<th>&quot;C4&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>&quot;1&quot;</td>
<td>&quot;2&quot;</td>
<td>&quot;4&quot;</td>
</tr>
<tr>
<td>1</td>
<td>&quot;2&quot;</td>
<td>&quot;4&quot;</td>
<td>&quot;1&quot;</td>
</tr>
<tr>
<td>2</td>
<td>&quot;4&quot;</td>
<td>&quot;1&quot;</td>
<td>&quot;2&quot;</td>
</tr>
</tbody>
</table>

END
17.2 Limitations

Several limitations exist in Orbiter. Here is a list:

1. Field elements are encoded as int. This limits the size of fields that can be handled to $2^{8s-1}$ where $s = \text{sizeof(int)}$.

2. The ranks of elements in the permutation domain are encoded as long int. This limits the size of permutation domains that can be handled. The degree of a permutation group must be less than $2^{8s-1}$ where $s = \text{sizeof(long int)}$.

3. The finite field class builds tables for the addition and multiplication of field elements. This restricts the size of the fields that can be created.

4. The projective geometry class tries to build a bitmatrix for the adjacency matrix if the number of lines is less than MAX_NUMBER_OF_LINES_FOR_INCIDENCE_MATRIX which is defined in src/lib/foundations/geometry/projective_space.cpp. If the number of lines is too big, the table is not created. In this case, the projective geometry class may behave slower.

5. The projective geometry class tries to build a table for the lines if the number of points is less 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://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 \texttt{wsl --update}
  • Type \texttt{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
  • \texttt{wsl --status}
• To change WSL configuration type
  • \texttt{wsl --set-default-version 1}
  • \texttt{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 installation screen](image-url)
Ubuntu - update

- Ubuntu does not update automatically, to update run the command
  - `sudo apt update && sudo apt upgrade`
- You will be prompted to enter your password
- When update are ready to be installed the message will appear
  - Do you want to continue? [Y/n]
    - Y + enter

Ubuntu – g++ and make

- At this point, you have successfully installed and setup WSL, and now you can use the terminal as you would on Ubuntu
- Terminate and reboot Ubuntu
- Run the command in Ubuntu
  - `sudo apt install g++`
  - You can now compile C++ in WSL
- Run the command in Ubuntu
  - `sudo apt install make`
  - You can now use makefiles in WSL
Orbiter - installation

• The easiest way to run make is through the command prompt, not Ubuntu
• To run WSL commands in command prompt, use either
  • wsl <command>
  • wsl.exe <command>
• Open command prompt
• Change directory to Users\username
  • cd C:\Users\"your username"

Orbiter - installation

• In web, go to
  https://github.com/abetten/orbiter
• Click on the green icon “Code” that opens a drop-down menu
• You want to copy HTTPS URL
Orbiter - installation

- In command prompt, once you are in C: \Users\Joel type the command
  - `wsl.exe git clone https://github.com/abetten/orbiter.git`
  - Hit enter

- Now, orbiter will begin the cloning process

```
C:\Users\Joel> sure git clone https://github.com/abetten/orbiter.git
Cloning into 'orbiter'...
remote: Enumerating objects: 87873, done.
remote: Counting objects: 100% (8299/8299), done.
remote: Compressing objects: 100% (5987/5987), done.
remote: Total 87873 (delta 5245), reused 3408 (delta 1713), pack-reused 80844
Receiving objects: 100% (87873/87873), 1.55 GiB | 18.87 MiB/s, done.
Resolving deltas: 100% (47649/47649), done.
Updating files: 100% (1154/1154), done.
```

Orbiter - compile

- After cloning orbiter, run the command
  - `dir`

- You will find a new directory created called “orbiter”

- Change directory to “orbiter”
  - `cd orbiter`
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

![Image of wsl.exe make command output]

Makefile

• Now that orbiter has been successfully compiled, in the directory C:\Users\"your username"\orbiter
  • Change directory to C:\Users\"your username" and create a new directory
    • Ex: mkdir CPP_Workspace
• Change directory into CPP_Workspace
  • cd CPP_Workspace
• In C:\Users\"your username"\"new directory", run the command
  • wsl.exe vim makefile
• Vim (an IDE) will create the file “makefile”
• For Vim commands, go to https://vim.rtorr.com/
• Remember: all Ubuntu commands must begin with either
  • wsl or wsl.exe
Makefile

- To edit file in vim, click “i”
- You will see --insert-- in the lower left-hand corner
- The example to the right demonstrates a simple test to assure that orbiter is running correctly
- Assuming that orbiter directory is located in C:\Users\"your username" then the variable OP and ORBITER_PATH should work just fine
- Note were wsl.exe is inserted
- Makefile contains Ubuntu commands not windows commands

Running makefile

- Now that you have created the makefile,
  - Click “esc” to finish editing in vim
  - Run the command
    - :wa + enter
    - This saves & closes the makefile in vim
  - You will be returned to
    - C:\Users\"your username"\"new directory”
  - In the directory run,
    - wsl.exe make test
    - Hit “enter”
  - If everything runs correctly, you will see
Orbiter - notes

• Now that everything runs correctly, visit
  https://www.math.colostate.edu/~betten/orbiter/users_guide.pdf
• This is the Orbiter User’s guide
• Remember that you must use “wsl.exe make <target>” or “wsl make
  <target>” to run make correctly on Linux distribution
• Also, note how “wsl.exe” is used inside of the makefile
• Ubuntu commands are used in makefile

Orbiter - update

• To update orbiter, change directories to
  • C:\Users\"your username"\orbiter
• Run the commands
  • wsl.exe make clean ; wsl.exe make
• Good luck!
Chapter 19

The Makefile

19.1 The Makefile

1 #MY_PATH=../orbiter
2 MY_PATH="/DEV.22/orbiter"
3 #MY_PATH=/scratch/betten/COMPILE/orbiter
4
5 # uncomment exactly one of the following two lines.
6 # uncomment the first if you want to run orbiter through docker.
7 # uncomment the second if you have an installed copy of orbiter and you want to run it directly
8 #ORBITER_PATH=docker run -it --volume ${PWD}:/mnt -w /mnt abetten/orbiter
9 ORBITER_PATH=$(MY_PATH)/src/apps/orbiter/
10 ORBITER=$(ORBITER_PATH)orbiter.out
11 SANDBOX=$(MY_PATH)/src/apps/sandbox/sandbox.out
12
13 ###########################################################################
14 # additional configurations for when you want to compile automatically generated code
15 ###########################################################################
16
17 SRC=$(MY_PATH)/src
18 MY_CPP = g++
19 MY_CC = gcc
20 CPPFLAGS = -Wall -I../../DEV.22/orbiter/src/lib -std=c++14
21 LIB = $(SRC)/lib/liborbiter.a -lpthread
22 LFLAGS = -lm -Wl,-rpath -Wl,/usr/local/gcc-8.2.0/lib64
23
24 ###########################################################################
25 # End of configuration part
26 ###########################################################################
GINAC_PATH=$(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="\n1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,\n1,1,1,1,1,1,0,1,0,0,0,0,0,0,\n1,2,1,2,4,1,0,1,0,0,0,0,\n1,3,2,6,4,5,1,0,0,0,1,0,0,0,\n1,4,2,1,4,2,1,0,0,0,1,0,0,\n1,5,4,6,2,3,1,0,0,0,0,1,0,\n1,6,1,6,1,6,1,0,0,0,0,0,1"

DOILY="Row,C0,C1,C2\n\n0,0,12,13\n\n1,1,12,14\n\n2,8,9,12\n\n3,4,6,8\n\n4,6,10,14\n\n5,3,7,8\n\n6,7,10,13\n\n7,4,11,13\n\n8,3,11,14\n\n9,0,5,6"
CONWAY
GEN1="
1101110001000001010000
1111010111110100001011
0000001000000100010101
1111100110110001001110
0101010000000010011101
0000010000001000101011
1000100000001001010101
00100001100000111111
1110100100110100010011
0000000000000100010101
0000000000000100010101
0110111111101110111111
0110101001110100010011
0000000000001111010101
0000000000000100010101
0000000000000100010111
0000000000000100010001"

CONWAY
GEN2="
0101000010111010111111
0110101001111110111111
001101000111111010111
0001101110001011010011
0000000000000010100011
0000000000000101010101
0000000000000100010111
0000000000000100010001"
# large sets of PG(2,3):

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

GENERATORS_N5=""
0,1,2,3,4,9,5,6,7,8,10,11,12, \ 
0,1,2,3,4,5,6,7,8,9,10,12,11, \ 
0,4,3,2,1,5,9,8,7,6,10,11,12, \ 
0,2,4,1,3,5,7,9,6,8,10,11,12, \ 
0,1,2,3,4,5,6,7,8,9,11,10,12, \ 
1,2,3,4,0,6,7,8,9,5,10,11,12, \ 
5,9,8,7,6,0,4,3,2,1,10,11,12"

GENERATORS_C13="11, 0, 10, 12, 5, 3, 7, 4, 2, 8, 6, 9, 1"
# (0,11,9,8,2,10,6,7,4,5,3,12,1)

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

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

EC_11_EQUATION="1,0,3,0,0,0,10,1,0,0"
EDGE_CURVE_Q23_AS_POINTS="4, 25, 26, 47, 48, 71, 92, 95, 114, 119, \
136, 143, 158, 167, 180, 191, 202, 215, 224, 239, 246, 263, 268, \
287, 290, 311, 312, 334, 356, 359, 378, 383, 400, 407, 422, \
431, 444, 455, 466, 479, 488, 503, 510, 527, 530, 532, 551"

GEN_C13="1,2,3,4,5,6,7,8,9,10,11,12,0"
# (0,1,2,3,4,5,6,7,8,9,10,11,12)

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

GENERATORS_WEYL_GROUP_E8="\n-1,-1,-1,-1,0,0,0,0, \n0,0,0,1,0,0,0,0, \n1,0,0,0,0,0,0,0, \n0,0,1,0,0,0,0,0, \n1,0,1,1,0,0,0,0, \n0,0,0,0,1,0,0,0, \n0,0,0,0,0,1,0,0, \n0,0,0,0,0,0,1,1, \n-1,0,-1,-1,-1,-1,-1,-1, \n0,1,0,1,1,1,1,1, \n1,0,0,0,0,0,0,0, \n0,0,1,0,0,0,0,0, \n0,0,0,1,0,0,0,0, \n0,0,0,0,1,0,0,0, \n10,0,0,0,0,0,0,0, \n0,0,0,0,0,1,0,0, \n0,0,0,0,0,0,0,1,0"

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

Ree_gen2="16,3,11,5,16,22,20, 24,6,18,24,7,1,26, 9,23,17,18,23,20,13, \
9,7,2,15,17,5,11, 3,3,6,21,4,24,16, 25,8,6,24,21,12,7, \
2511
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"

HYPEROVAL_16_144="0, 1, 2, 3, 52, 67, 89, 106, 126, \n141, 159, 176, 184, 199, 220, 235, 245, 262"

HYPEROVAL_16_16320="0, 1, 2, 3, 52, 70, 83, 109, 127, \n139, 156, 174, 186, 199, 217, 229, 256, 264"

FILE_24_3_TFC_INC="24 24 72\n0 1 2 24 27 28 48 53 54 73 79 80 97 105 106 122 131 \n132 146 157 158 171 175 183 195 203 208 220 225 233 244 \n258 259 269 272 282 293 300 308 318 325 333 342 352 358 \n367 379 381 392 398 400 417 428 429 442 443 450 466 471 \n479 492 497 502 519 521 542 548 551 571 574 575 \n48"

\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 \n367 373 378 392 400 403 417 419 430 442 446 447 466 472 \n479 492 500 503 518 525 526 545 549 551 571 572 574 \n48"

\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 \n257 258 269 274 277 293 300 307 318 323 329 342 352 356 \n367 374 381 392 397 406 416 423 431 441 450 454 468 476 \n477 494 499 503 519 521 525 544 547 550 570 572 575 \n144"

\n-1 3"

ELEMENTARY_SYMMETRIC_3_1="x0 + x1 + x2"

ELEMENTARY_SYMMETRIC_3_2="x0*x1 + x0*x2 + x1*x2"

ELEMENTARY_SYMMETRIC_3_3="x0*x1*x2"

ELEMENTARY_SYMMETRIC_4_1="x0 + x1 + x2 + x3"
ELEMENTARY_SYMMETRIC_4_2="x0*x1 + x0*x2 + x0*x3 + x1*x2 + x1*x3 + x2*x3"

ELEMENTARY_SYMMETRIC_4_3="x0*x1*x2 + x0*x1*x3 + x0*x2*x3 + x1*x2*x3"

ELEMENTARY_SYMMETRIC_4_4="x0*x1*x2*x3"

CODE_5_2_3_CODEWORDS="0,7,25,30"

SURFACE_F7_15LINES_POINTS="0,1,2,3,5,6,7,8,9,10,12,18,20,26,28,34,36,42,44,45,50,53,58,59,60,61,62,63,64,67,77,80,90,93,103,107,115,118,122,125,127,142,147,155,157,162,165,170,172,204,208,219,229,240,244,246,251,253,259,277,278,281,286,298,300,302,303,310,312,316,340,343,351,354,358,365,369,372,373,379,384,386,388,393,399"

SURFACE_F7_15LINES_MCKEAN_POINTS="0,1,2,3,4,5,6,7,8,9,10,16,17,28,35,36,59,60,61,62,63,64,65,76,80,91,95,106,107,111,119,121,122,130,138,139,141,146,150,154,155,158,170,175,184,186,199,203,204,206,226,231,234,239,240,252,253,254,278,279,282,287,299,301,302,319,320,330,338,343,345,350,351,357,364,370,371,376,378,382,385,388,392,394,399"

SURFACE_MCKEAN_15_LINES="1,5,-1,16,-2,17,1,10,2,18,1,8,1,9,-1,11,-1,19,-1,14"

CODE_RS_6_4_7="\n621000 \n062100 \n006210 \n000621"

CODE_RS_10_8_11="\n7,2,1,0,0,0,0,0,0,0, \n0,7,2,1,0,0,0,0,0,0, \n0,0,7,2,1,0,0,0,0,0, \n0,0,0,7,2,1,0,0,0,0, \n0,0,0,0,7,2,1,0,0,0, \n0,0,0,0,0,0,7,2,1,0, \n0,0,0,0,0,0,0,7,2,1 "

# Normal form for cubic surfaces with 15 rational lines:
F_ALPHA_BETA_GAMMA_DELTA="beta*(gamma + 1)*x0*x0*x2  
+ (alpha*delta - beta*gamma + alpha - beta - delta - 1)*x0*x1*x2  
-1*(alpha*beta - alpha*delta + delta)*(gamma + 1)*x0*x1*x3  
+ (0-alpha*delta + alpha*gamma -beta*gamma -beta + delta - gamma)*x0*x2*x2  
- (alpha*delta + beta - delta)*(gamma +1)*x0*x2*x3  
- (delta + 1)*(alpha - 1)*x1*x1*x2  
- (delta + 1)*(alpha - 1)*x1*x1*x3  
+ (alpha*delta - alpha*gamma + beta*gamma + beta - delta + gamma)*x1*x2*x2  
+ (alpha*beta*gamma + alpha*beta + alpha*delta - alpha*gamma + beta*gamma + beta - delta + gamma)*x1*x3*x3"

# general normal form for cubic surfaces with 27 rational lines:

F_abcd_eqn="-(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*(b - d)*X0*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*a*c - a*a*d - a*b*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*c*c)*X1*X2*X3  
+ c*a*(a*d - b*c - a + b + c - d)*(b - d)*X1*X3*X3"

# general normal form for cubic surfaces with 27 rational lines:

F_abcd_eqn_with_exponents="-(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*(b - d)*X0^2*X2  
+ (a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*(a + b - c - d)*X0*X1*X2  
+ (a^2*c - a^2*d - a*c^2 + b*c^2 + a*d - b*c)*(b - d)*X0*X1*X3  
- (a*d - b*c)*(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*X0*X2*X2  
- (a^2*c*d - a*b*c^2 - a^2*d + a*b*d + b*c^2 - b*c*d)*(b - d)*X0*X2*X3  
- (a - c)*(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*X1*X1*X2  
- (a - c)*(a*a*c - a*a*d - a*b*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^2*b*c*d - a^2*b*d*d - (1+1)*a^2*c*d*d  
- (1+1)*a*b^2*c^2 + a*b^2*c*d + (1+1)*a*b*c^2*d + a*b*c*d^2  
- b*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*c*c)*X1*X2*X3  
+ c*a*(a*d - b*c - a + b + c - d)*(b - d)*X1*X3*X3"
\[-(1+1+1+1)ab + a^2d + b^2c^2)XX_1XX_2XX_3 - cda^2 \times (a + b + c - d)(b - d)XX_1XX_3^2\]

KNECHT.13.1.AS.PAIRS="1,0,1,1,2,12,9"
KNECHT.13.1.AS.VECTOR="1,1,1,0,0,0,0,0,0,0,0,0"
KNECHT.13.2.AS.PAIRS="1,0,1,1,2,8,9,10,8,11"
KNECHT.13.2.AS.VECTOR="1,1,1,0,0,0,0,0,0,0,0,0"

# coding theory
CRC4="1,4,1,2,1,1,1,0"
CRC7="1,7,1,3,1,0"
CRC8_ATM="1,8,1,2,1,1,1,0"
CRC16_CCITT="1,16,1,12,1,5,1,0"
CRC32.ETHERNET="1,32,1,26,1,23,1,22,1,16,1,12,1,11,1,10,1,9,8,1,7,\n1,5,1,4,1,2,1,1,1,0"
CRC32_CASTAGNOLI="1,32,1,28,1,27,1,26,1,25,1,23,1,22,1,20,1,19,1,\n18,1,14,1,13,1,11,1,10,1,9,1,8,1,6,1,0"
CRC64_ECMA182="1,64,1,62,1,57,1,55,1,54,1,53,1,52,1,47,1,46,1,45,\n1,40,1,39,1,38,1,37,1,35,1,33,1,32,1,31,1,29,1,27,1,24,1,23,1,22,\n1,21,1,19,1,17,1,13,1,12,1,10,1,9,1,7,1,4,1,1,1,0"
CRC64_ROCKSOFT="1,64,1,63,1,61,1,59,1,58,1,56,1,55,1,52,1,49,1,48,\n1,47,1,46,1,44,1,41,1,37,1,36,1,34,1,32,1,31,1,28,1,26,1,23,1,22,1,\n19,1,16,1,13,1,12,1,10,1,9,1,6,1,4,1,3,1,0"
GOLAY.23.COLUMN_RANKS_PROJECTIVELY="0,1,2,3,4,5,6,7,\n8,9,10,11,132,913,1460,1750,1898,2518,2787,2874,\n3320,3357,3662"
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 \
0005610 \
0000561"

CODE
RS 11_RREF="
1,0,0,0,0,0,0,0,7,2,\n0,1,0,0,0,0,0,0,8,3,\n0,0,1,0,0,0,0,0,1,2,\n0,0,0,1,0,0,0,0,8,8,\n0,0,0,0,1,0,0,0,10,3,\n0,0,0,0,0,1,0,0,1,4,\n0,0,0,0,0,0,1,0,5,4,\n0,0,0,0,0,0,0,1,5,8"
CODE_21_15_4="
11100100000000000000
11010001000000000000
10110001000000000000
01110000010000000000
11001000010000000000
10101000000100000000
01101000000100000000
10011000000010000000
00111000000010000000
111110000000010000
11001000000001000000
10101000000000100000
01101000000000010000
10010100000000000100"

# there are 5 [15,6,6]

# ago=12
CODE_15_6_6_A="
11111111110000
11111000001000
11100110000100
11010101000010
10101011000010
10110100100001"

# ago=12
CODE_15_6_6_B="
11111111110000
11111000001000
11100110000100
11010101000010
11100110000100
11010101000010
#ago=720:
CODE_15_6_6_C="\n111111111100000 \n111110000010000 \n110011000010000 \n110101001000100 \n101101001000010 \n100010111000001"

#ago=96:
CODE_15_6_6_D="\n111111111100000 \n111110000010000 \n110011000010000 \n110101001000100 \n101101001000010 \n011001011000001"

#ago=360
CODE_15_6_6_E="\n111111111100000 \n111110000010000 \n110011000010000 \n101101001000010 \n010101110000010 \n010110101000001"

BCH_21_15_PROJ=" 0, 1, 19, 37, 113, 420, 1651, 6577, \
26284, 105115, 420442, 1681753, 6727000, 26907991, \
107631958, 27874647, 111498582, 43341143, 173364566, \
156587350, 14 "

BCH_21_15_GENERATOR_MATRIX="1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 
0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 
0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1"
529
530

531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548

549
550

551
552

BCH
0,
0,
1,
1,
"

21 6 GENERATOR MATRIX=" 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0,
1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1,
0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1,
0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1,
0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1

POLY Q256 DEG30 SPARSE="1,0,26,1,210,2,24,3,\
138,4,148,5,160,6,58,7,108,8,199,9,95,10,56,\
11,9,12,205,13,194,14,193,15,3,16,248,17,110,\
18,150,19,24,20,169,21,192,22,212,23,112,24,\
144,25,97,26,109,27,174,28,253,29,1,30"
POLY Q256 DEG30 DENSE="1,26,210,24,138,148,\
160,58,108,199,95,56,9,205,194,193,3,248,110,\
150,24,169,192,212,112,144,97,109,174,253,1"

# created in the combinatorics section:
ELEMENTARY SYMMETRIC 8 1="x0 + x1 + x2 + x3 + x4 + x5 + x6 + x7"
ELEMENTARY SYMMETRIC 8 2="x0*x1 + x0*x2 + x0*x3 + x0*x4 + x0*x5 + x0*x6 + x0*x7 +
x1*x2 + x1*x3 + x1*x4 + x1*x5 + x1*x6 + x1*x7 + x2*x3 + x2*x4 + x2*x5 + x2*x6 +
x2*x7 + x3*x4 + x3*x5 + x3*x6 + x3*x7 + x4*x5 + x4*x6 + x4*x7 + x5*x6 + x5*x7 + x
6*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*x
3*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 + x
1*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*x
5 + x0*x1*x4*x6 + x0*x1*x4*x7 + x0*x1*x5*x6 + x0*x1*x5*x7 + x0*x1*x6*x7 + x0*x2*x
3*x4 + x0*x2*x3*x5 + x0*x2*x3*x6 + x0*x2*x3*x7 + x0*x2*x4*x5 + x0*x2*x4*x6 + x0*x
2*x4*x7 + x0*x2*x5*x6 + x0*x2*x5*x7 + x0*x2*x6*x7 + x0*x3*x4*x5 + x0*x3*x4*x6 + x

519


0*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 + 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 + 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 + x3*x4*x5*x6 + x3*x4*x5*x7 + x3*x4*x6
*x7 + x3*x5*x6*x7 + x4*x5*x6*x7"
553
554

555
556

557
558

559
560
561
562
563
564
565
566
567
568
569
570

ELEMENTARY SYMMETRIC 8 5="x0*x1*x2*x3*x4 + x0*x1*x2*x3*x5 + x0*x1*x2*x3*x6 + x0*x
1*x2*x3*x7 + x0*x1*x2*x4*x5 + x0*x1*x2*x4*x6 + x0*x1*x2*x4*x7 + x0*x1*x2*x5*x6 +
x0*x1*x2*x5*x7 + x0*x1*x2*x6*x7 + x0*x1*x3*x4*x5 + x0*x1*x3*x4*x6 + x0*x1*x3*x4*x
7 + x0*x1*x3*x5*x6 + x0*x1*x3*x5*x7 + x0*x1*x3*x6*x7 + x0*x1*x4*x5*x6 + x0*x1*x4*
x5*x7 + x0*x1*x4*x6*x7 + x0*x1*x5*x6*x7 + x0*x2*x3*x4*x5 + x0*x2*x3*x4*x6 + x0*x2
*x3*x4*x7 + x0*x2*x3*x5*x6 + x0*x2*x3*x5*x7 + x0*x2*x3*x6*x7 + x0*x2*x4*x5*x6 + x
0*x2*x4*x5*x7 + x0*x2*x4*x6*x7 + x0*x2*x5*x6*x7 + x0*x3*x4*x5*x6 + x0*x3*x4*x5*x7
+ x0*x3*x4*x6*x7 + x0*x3*x5*x6*x7 + x0*x4*x5*x6*x7 + x1*x2*x3*x4*x5 + x1*x2*x3*x
4*x6 + x1*x2*x3*x4*x7 + x1*x2*x3*x5*x6 + x1*x2*x3*x5*x7 + x1*x2*x3*x6*x7 + x1*x2*
x4*x5*x6 + x1*x2*x4*x5*x7 + x1*x2*x4*x6*x7 + x1*x2*x5*x6*x7 + x1*x3*x4*x5*x6 + x1
*x3*x4*x5*x7 + x1*x3*x4*x6*x7 + x1*x3*x5*x6*x7 + x1*x4*x5*x6*x7 + x2*x3*x4*x5*x6
+ x2*x3*x4*x5*x7 + x2*x3*x4*x6*x7 + x2*x3*x5*x6*x7 + x2*x4*x5*x6*x7 + x3*x4*x5*x6
*x7"
ELEMENTARY SYMMETRIC 8 6="x0*x1*x2*x3*x4*x5 + x0*x1*x2*x3*x4*x6 + x0*x1*x2*x3*x4*
x7 + x0*x1*x2*x3*x5*x6 + x0*x1*x2*x3*x5*x7 + x0*x1*x2*x3*x6*x7 + x0*x1*x2*x4*x5*x
6 + x0*x1*x2*x4*x5*x7 + x0*x1*x2*x4*x6*x7 + x0*x1*x2*x5*x6*x7 + x0*x1*x3*x4*x5*x6
+ x0*x1*x3*x4*x5*x7 + x0*x1*x3*x4*x6*x7 + x0*x1*x3*x5*x6*x7 + x0*x1*x4*x5*x6*x7
+ x0*x2*x3*x4*x5*x6 + x0*x2*x3*x4*x5*x7 + x0*x2*x3*x4*x6*x7 + x0*x2*x3*x5*x6*x7 +
x0*x2*x4*x5*x6*x7 + x0*x3*x4*x5*x6*x7 + x1*x2*x3*x4*x5*x6 + x1*x2*x3*x4*x5*x7 +
x1*x2*x3*x4*x6*x7 + x1*x2*x3*x5*x6*x7 + x1*x2*x4*x5*x6*x7 + x1*x3*x4*x5*x6*x7 + x
2*x3*x4*x5*x6*x7"
ELEMENTARY SYMMETRIC 8 7="x0*x1*x2*x3*x4*x5*x6 + x0*x1*x2*x3*x4*x5*x7 + x0*x1*x2*
x3*x4*x6*x7 + x0*x1*x2*x3*x5*x6*x7 + x0*x1*x2*x4*x5*x6*x7 + x0*x1*x3*x4*x5*x6*x7
+ x0*x2*x3*x4*x5*x6*x7 + x1*x2*x3*x4*x5*x6*x7"
ELEMENTARY SYMMETRIC 8 8="x0*x1*x2*x3*x4*x5*x6*x7"

PG 3 5 DESARGUESIAN SPREAD="0, 805, 36, 108, 72, 144, \
581, 509, 686, 415, 639, 758, 285, 722, 332, 343, 202, \
592, 473, 238, 675, 379, 166, 545, 249, 451"
# elements of order 2:
# conjugacy class reps:

520


# elt order, class size, centralizer order

## Class 2A
- Order: 2
- Class Size: 48960
- Centralizer Order: 40320
- Description: Baer involution

## Class 2B
- Order: 2
- Class Size: 5355
- Centralizer Order: 368640
- Description: One block of 10,11

## Class 2C
- Order: 2
- Class Size: 64260
- Centralizer Order: 30720
- Description: Two blocks of 10,11 (problem group)

---

# elements of order 2:

## Conjugacy Class Reps:

### Class 2A
- Centralizer of element: `1,0,0,0, 0,1,0,0, 0,0,1,0, 0,0,0,1, 1`
- Label: "2A"

### Class 2B
- Centralizer of element: `1,0,0,0, 1,1,0,0, 0,0,1,0, 0,0,0,1, 0`
- Label: "2B"

### Class 2C
- Centralizer of element: `1,0,0,0, 1,1,0,0, 0,0,1,0, 0,0,1,1, 0`
- Label: "2C"

---

# 3 classes of elements of order 3

# 4 classes of elements of order 4

---

# Baer involution:

```bash
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,
> 1,0,0,0,0,1,0,0,0,0,1,0,1,1,1,1,1,1,
> 1,0,0,0,0,1,0,0,0,0,1,0,1,1,1,1,1,0,
> 1,0,0,0,0,1,0,0,0,1,1,0,1,1,1,1,0, 
> 1,0,0,0,0,1,0,0,0,1,1,0,1,1,1,1,0, 
> 1,0,0,0,0,1,0,0,0,1,1,0,1,1,1,1,0, 
> 1,0,0,0,0,1,0,0,0,1,1,0,1,1,1,1,0, 
> 1,0,0,0,0,1,0,0,0,1,1,0,1,1,1,1,0, 
> 0,1,0,0,0,1,0,1,1,1,0,1,0,1,1,1,1

# 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, 0,0,1,1, 0"

PGGL_4_4_SUBGROUP_2C_NORMALIZER=-PGGL 4 4 \\
> -subgroup_by_generators "centralizer_2C" "30720" 9 \\
> "1,0,0,0, 1,1,0,0, 0,0,1,0, 0,0,1,1, 0"

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, 
> 1,0,0,0,1,0,0,0,0,1,0,0,0,0,2,0, 
> 1,0,0,0,1,0,0,0,0,1,0,0,0,0,2,0, 
> 1,0,0,0,1,0,0,0,0,1,0,0,0,0,2,0, 
> 1,0,0,0,2,2,0,0,0,0,1,0,0,0,0,1,1, 
> 0,1,0,0,3,3,0,0,0,0,2,0,0,0,0,2,0"
PGGL_4_4_SUBGROUP_5B=-PGGL 4 4 \n> -subgroup_by_generators "5B" 5 1 \n> "0,2,0,0,1,1,0,0,0,0,2,0,0,1,1,0" 

PGGL_4_4_SUBGROUP_5B_NORMALIZER=-PGGL 4 4 \n> -subgroup_by_generators "normalizer_5B" "81600" 6 
> "1,0,0,0,0,0,1,0,0,0,2,0,0,0,2,0, \n> 1,0,0,0,0,1,0,0,0,0,1,0,0,0,3,2,0, \n> 1,0,0,0,2,2,0,0,0,0,3,0,0,0,1,1,1, \n> 0,1,0,0,3,3,0,0,0,0,1,0,0,3,3,0, \n> 0,0,1,0,0,0,1,2,2,0,0,0,2,3,0,0,1 

PGGL_4_4_SUBGROUP_2Cx2_0=-PGGL 4 4 \n> -subgroup_by_generators "2Cx2_0" 4 2 
> "1,0,0,0,0,1,1,0,0,0,1,0,0,0,1,1,0 \n> 1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,1,0,0,1,0,1,0" 

PGGL_4_4_SUBGROUP_2Cx2_0_NORMALIZER=-PGGL 4 4 \n> -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 \n> 1,0,0,0,0,1,0,0,0,0,1,0,0,0,1,0,0,1,0,0,1,0,0,1,0,1,0 \n> 1,0,0,0,0,1,0,0,0,0,1,0,0,0,1,0,0,1,0,0,1,0,0,1,0,1,0 \n> 1,0,0,0,2,1,1,0,1,0,1,0,1,1,2,1,1 \n> 1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0 \n> 1,0,0,0,3,1,0,0,1,0,1,0,0,1,3,1,1" 

#PGL_4_5_SUBGROUP_3B=-PGL 4 5 
> -subgroup_by_generators "3B" 3 1 
> "1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1 \n> 1,0,0,0,0,1,0,0,0,0,0,0,1,0,2,0,0,1,0 \n> 1,0,0,0,1,0,0,1,0,0,1,0,1,0,3,1,0,1,1 \n> 1,0,0,0,0,1,0,0,0,0,1,0,0,0,1,0,0,1,0,0,1,0,0,1,0,1,0 \n> 1,0,0,0,2,1,1,0,1,0,1,0,1,1,2,1,1 \n> 1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0 \n> 1,0,0,0,3,1,0,0,1,0,1,0,0,1,3,1,1" 

#PGL_4_5_SUBGROUP_3B_NORMALIZER=-PGL 4 5 
> -subgroup_by_generators "normalizer_3B" "5760" 8 
> "1,0,0,0,0,4,0,0,0,0,1,0,0,0,0,1," \n> "1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,4," \n> "1,0,0,0,0,4,0,0,0,0,4,0,0,0,0,4," \n> "1,0,0,0,0,1,0,0,0,0,3,0,0,0,0,3," \n> "1,0,0,0,0,3,0,0,0,0,1,0,0,0,0,1," 

523
elementary abelian subgroups of order 4 with 3 elements of class 2C:

# nice generators, from Michael Epstein:

\[ \text{PGL}_4 5 \text{SUBGROUP}_3B \text{ME} = \text{PGL}_4 5 \]
\[ -\text{subgroup_by_generators} \text{"3B" 3 1} \]
\[ "1,0,0,0, 0,1,0,0, 0,0,2,2, 0,0,4,2" \]
\[ \text{PGL}_4 5 \text{SUBGROUP}_3B \text{ME NORMALIZER} = \text{PGL}_4 5 \]
\[ -\text{subgroup_by_generators} \text{"normalizer_3B" "5760" 8} \]
\[ "1,0,0,0,0,4,0,0,0,0,1,0,0,0,0,1, \]
\[ "1,0,0,0,0,4,0,0,0,0,4,0,0,0,0,4, \]
\[ "1,0,0,0,0,1,0,0,0,0,3,0,0,0,0,3, \]
\[ "1,0,0,0,0,3,0,0,0,0,1,0,0,0,0,1, \]
\[ "1,0,0,0,0,3,0,0,0,0,2,4,0,0,2,3, \]
\[ "1,0,0,0,4,4,0,0,0,0,1,0,0,0,0,1, \]
\[ "0,1,0,0,1,0,0,0,0,0,4,0,0,0,0,4," \]
\[ \text{PGL}_4 5 \text{SUBGROUP}_31 \text{ME} = \text{PGL}_4 5 \]
\[ -\text{subgroup_by_generators} \text{"31" 31 1} \]
\[ "1,0,0,0,0,3,4,3, 0,3,3,4, 0,3,2,3" \]
\[ \text{PGL}_4 5 \text{SUBGROUP}_31 \text{ME NORMALIZER} = \text{PGL}_4 5 \]
\[ -\text{subgroup_by_generators} \text{"normalizer_31" "372" 4} \]
\[ "1,0,0,0,0,4,0,0,0,0,4,0,0,0,0,4, \]
\[ "1,0,0,0,0,3,0,0,0,0,3,0,0,0,0,3, \]
\[ "1,0,0,0,0,4,0,0,0,0,2,1,0,3,2,4, \]
\[ "1,0,0,0,0,1,0,0,0,0,1,0,1,1,3," \]

# subgroup of order 31 for the construction of regular packings in PG_3.5:
-subgroup_by_generators "31" 31 1 \
"2,0,0,0, 0,0,1,0, 0,0,0,1, 0,3,0,4"

PGL_4_5_SUBGROUP_31_NORMALIZER=-PGL 4 5 \n-subgroup_by_generators "normalizer_31" "372" 4 \n"1,0,0,0,4,0,0,0,4,0,0,0,4, \
1,0,0,0,3,0,0,0,3,0,0,0,3, \
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,"

#372:
"1,0,0,0,4,0,0,0,4,0,0,0,4, " \
"1,0,0,0,3,0,0,0,3,0,0,0,3, " \
"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= 
[1 0 0 0]
[0 1 0 0]
[0 0 2 2]
[0 0 4 2]

#elt of order 31:
# the exterior square root of g is Z= 
[1 0 0 0]
[0 3 4 3]
[0 3 3 4]
[0 3 2 3]

#Michael

HAMMING_CODE_CODEWORDS="0, 67, 37, 102, 22, 85, \
51, 112, 15, 76, 42, 105, 25, 90, 60, 127"

SIMPLEX_CODE_GENERATOR="\n1,0,1,0,1,0,1, \

HAMMING_CODE_GENERATOR="\n1,0,0,0,0,1,1, \n0,1,0,1,0,1, \n0,0,1,0,1,0, \n0,0,0,1,1,1"\n
GOLAY23_CODE_GENERATOR="\n1,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,0,1,0,\n0,1,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,1,0,\n0,0,1,0,0,0,0,0,0,0,0,0,1,1,0,1,0,0,1,\n0,0,0,1,0,0,0,0,0,0,0,0,1,0,1,1,0,1,1,\n0,0,0,0,1,0,0,0,0,0,0,0,1,0,1,1,0,1,1,\n0,0,0,0,1,0,0,0,0,0,0,0,1,0,1,1,0,1,1,\n0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,1,0,1,0,\n0,0,0,0,0,0,1,0,0,0,0,0,1,0,1,1,0,1,0,\n0,0,0,0,0,0,0,1,0,0,0,0,1,0,1,1,0,1,0,\n0,0,0,0,0,0,0,0,1,0,0,0,1,0,1,1,0,1,0,\n0,0,0,0,0,0,0,0,0,1,0,0,1,0,1,1,0,1,1,\n0,0,0,0,0,0,0,0,0,0,1,0,1,0,1,1,0,1,1,\n0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,1,0,1,1,\n0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,\n0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,1,1,\n0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1"\n
HAMMING_CODE_ROWS_IN_BINARY_RANKS="67,37,22,15"\n
SIMPLEX_CODE_GENMA_CYCLIC="\n1,0,0,1,1,1,0, \n0,1,0,1,1,1, \n0,0,1,1,1,0,1"\n
CODE_GV_N15_K6="\n111111111100000\n111100000010000\n110011000010000\n110101010000100\n101010110000100\n101101001000010"\n
CODE_GV_N15_K6_CHECK="\n100000000111111\n010000000111100"
REED_MULLER_3_1_CODEWORDS="0,255,170,85,204,51,102,153,240,15,90,165,60,195,150,105"
REED_MULLER_3_1_BASIS_IN_BINARY="255,170,204,240"
REED_MULLER_4_1_COLUMNS_OF_PARTITY_CHECK="1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31"
#-nearest codeword "8,16,32,40,48,56,1,2,4,3,5,6,7,9,18,36,27,45,54,63,33,42,20,11,53,62,31"
AG_2_3_BLOCKS="0,13,22,27,35,41,47,53,55,59,71,76"
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"

PP4= -d1 1 -q1 3 -d2 1 -q2 7 -K 5 -search_control -W -end -problem_label PP4

PP4_GROUP1=-subgroup "1,1,1,1, " "21" -group_label "cyclic21"

PP4_MASK1=
  -nb_orbits_on_blocks 1 \
  -depth 5 \n  -mask.label "no_mask"

DELANDTSHEER_DOYEN_PROBLEM_COLBOURN_COLBOURN_7_13= -d1 1 -q1 7 -d2 1 -q2 13 -K 6 
  -search_control -W -end -problem_label DD_CC_7_13

DELANDTSHEER_DOYEN_PROBLEM_COLBOURN_COLBOURN_7_13_GROUP1=-subgroup "1,1,1,1, " "9
  1" -group_label "cyclic91"

DELANDTSHEER_DOYEN_PROBLEM_COLBOURN_COLBOURN_7_13_MASK1=
  -nb_orbits_on_blocks 3 \n  -depth 6 \n  -mask_label "no_mask"

DELANDTSHEER_DOYEN_PROBLEM_27_53= -d1 1 -q1 27 -d2 1 -q2 53 -K 11 -DDx 2 -DDy 1 -
  search_control -W -end

DELANDTSHEER_DOYEN_PROBLEM_27_53_GROUP1=-subgroup \n  "1,1,1,0, 1,3,1,0, 1,9,1,0, 1,0,1,1, -2,0,-4,0" "18603" -group_label "group1"

# mask 1:
#  XX.
# X.X+

DELANDTSHEER_DOYEN_PROBLEM_27_53_MASK1=
  -masktest 3 x ge 1 \n  -masktest 4 x+y ge 3 \n  -depth 4 \n  -mask_label "mask1"

DELANDTSHEER_DOYEN_PROBLEM_3_7= -d1 1 -q1 3 -d2 1 -q2 7 -K 5 -DDx 3 -DDy 1 -search_control -W -end
# consider the binary code with generator matrix:
# 1 0 1
# 0 1 1
CODEN3_K2_Q2_GENMA="1,0,1, 0,1,1"
CODEN6_K3_Q2_GENMA="111100"
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^2,4f,0,0)

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

EDGE_CURVE_Q17_AS_POINTS="4, 7, 16, 19, 20, 23, 32, 35, 89, 100, 244, 251"

FILE_Q17="orbit,curve,pts_on_curve,bitangents,go\n0,"$(EDGE_CURVE_Q17_EQUATION)"\n $(EDGE_CURVE_Q17_AS_POINTS)"\n\nEND"

DESARGUES_PATH_LEX_LEAST="10 10 3\n0\n1 0 15\n3 0 15 26\n4 0 15 26 46\n5 0 1 5 26 46 56\n6 0 15 26 46 56 72\n7 0 15 26 46 56 72 80\n8 0 15 26 46 56 72 80 93\n9 0 15 26 46 56 72 80 93 106\n10 0 15 26 46 56 72 80 93 106 119\n-1"

SPREADS_27_ISO_0=""

0, 33879, 1339, 2678, 3994, 7671, 10180, 5862, 9524, 6852, 22243, 
12745, 24295, 11062, 13615, 23894, 15056, 29367, 16429, 31521, 17726, 
31103, 18887, 26333, 19566, 28400, 21531, 27228"

SPREADS_27_ISO_1=""

0, 33879, 1339, 2678, 3994, 7671, 10182, 5761, 6796, 9327, 15339, 
31914, 24415, 12713, 22748, 11666, 13353, 23555, 30103, 16395, 17827, 

SPREADS_27_ISO_2=
0, 33879, 1339, 2678, 3994, 7671, 10182, 5817, 6796, 9276, 23891, \n15368, 11666, 22124, 12713, 24415, 13353, 29619, 15910, 31914, 17030, \n30931, 19213, 26422, 19905, 28112, 21217, 27545"

SPREADS_27_ISO_3=
0, 33879, 1339, 2678, 3994, 7671, 10625, 6590, 9476, 5576, 24688, \n23043, 10996, 22124, 12723, 13522, 15421, 29894, 16532, 32442, 17997, \n31015, 18311, 26109, 19807, 28113, 21220, 27195"

SPREADS_27_ISO_4=
0, 33879, 1339, 2678, 3994, 7674, 7051, 10666, 9327, 5419, 19806, \n21332, 22124, 13353, 24415, 12401, 11062, 23717, 15056, 29660, 16395, \n31950, 17873, 31153, 19212, 26221, 28515, 26708"

SPREADS_27_ISO_5=
0, 33879, 1339, 2678, 3994, 7988, 5333, 6672, 10666, 9327, 11062, \n22124, 13353, 15056, 16395, 17347, 31055, 27061, 24415, 12401, 23076, \n30103, 32394, 19041, 26109, 20380, 28400, 21332"

SPREADS_27_ISO_6=
0, 33879, 1339, 2679, 3992, 7672, 31140, 6913, 13513, 23167, 5653, \n10607, 9131, 11225, 22548, 13074, 24645, 15124, 29345, 16226, 31506, \n17684, 18732, 26116, 19458, 28361, 21571, 27680"

# Povray:
# povray colors:
POLISHED_CHROME_WHITE=

> "texture{ Polished_Chrome pigment{quick_color White} }

POLISHED_CHROME_WHITE=

> "texture{ pigment{ color Yellow transmit 0.7 } \

> finish {diffuse 0.9 phong 0.6} }

> "texture{ pigment{ color Red } \

> finish {diffuse 0.9 phong 0.6} }

> "texture{ pigment{ color Red } \

> finish {diffuse 0.9 phong 1}"
COLOR\_GREEN\_SHINY=
"texture\{ pigment\{ color Green \} \nfinish \{ diffuse 0.9 phong 1\}"\n
COLOR\_BLUE\_SHINY=
"texture\{ pigment\{ color Blue \} \nfinish \{ diffuse 0.9 phong 1\}"\n
COLOR\_YELLOW\_SHINY=
"texture\{ pigment\{ color Yellow \} \nfinish \{ diffuse 0.9 phong 1\}"\n
COLOR\_BLACK\_SHINY=
"texture\{ pigment\{ color Black \} \nfinish \{ diffuse 0.9 phong 1\}"\n
COLOR\_RED\_SEE\_THROUGH=
"texture\{ pigment\{ color Red transmit 0.5 \} \nfinish \{ diffuse 0.9 phong 1\}"\n
COLOR\_GREEN\_SEE\_THROUGH=
"texture\{ pigment\{ color Green transmit 0.5 \} \nfinish \{ diffuse 0.9 phong 1\}"\n
COLOR\_BLUE\_SEE\_THROUGH=
"texture\{ pigment\{ color Blue transmit 0.5 \} \nfinish \{ diffuse 0.9 phong 1\}"\n
COLOR\_YELLOW\_THICK=
"texture\{ pigment\{ color Yellow \} \nfinish \{ diffuse 0.9 phong 1\}"\n
COLOR\_BLACK\_NO\_SHADOW=
"texture\{ pigment\{Black\} no\_shadow\"\n
SURFACE\_COLOR=
"texture\{ pigment\{ White*0.5 \} \nfinish \{ambient 0.4 diffuse 0.5 roughness 0.001 \nreflection 0.1 specular .8\} \"\n
SURFACE\_COLOR\_SEE\_THROUGH=
"texture\{ pigment\{ White*0.5 transmit 0.5 \} \n
```
1119  > finish {ambient 0.4 diffuse 0.5 roughness 0.001 \
1120  > reflection 0.1 specular .8} }
1121
1122  COLOR_GOLD=\
1123  > "texture{ pigment{ Gold } finish \ 
1124  > {ambient 0.4 diffuse 0.5 roughness 0.001 \ 
1125  > reflection 0.1 specular .8} }
1126
1127  COLOR_TURQUOISE=\ 
1128  > "texture{ pigment{Cyan*1.3} \ 
1129  > finish {ambient 0.4 diffuse 0.6 roughness 0.001 \ 
1130  > reflection 0 specular .8} }
1131
1132  MONKEY_SADDLE_CUBIC="1,0,0,0,-3,0,0,0,0,0,0,0,0,0,0,0,0,0,-1,0"
1133
1134  ECKARDT_CUBIC_DEFORM1_LEX="0, 10, 0, -8, 10, 25, 2, 0, -20, -8, -20, -10, -24, 10 
1135  , -2, 12, 0, -8, 8, 16"
1136
1137  ECKARDT_CUBIC_DEFORM2_LEX="0, -5, 0, -5, -5, 10, -1, 0, 10, 4, 10, 5, 3, -5, 1, - 
1138  6, 0, -5, -4, 1"
1139
1140  KUMMER_QUARTIC_LEX_35="-2,0,0,0,2,0,0,2,0,2,0,0,\ 
1141  0,0,0,0,0,0,0,-2,0,0,2,0,2,0,0,0,0,-2,0,2,0,-2"
1142
1143  BEAUVILLE_QUINTIC_LEX_56="-44, 228, 400, 315, -396, -852, 
1144  -512, -553, -1050, -354, 284, 504, -62, -707, -1390, -1010, 
1145  281, -167, -1644, -1024, -72, -196, 192, 373, 322, 78, 150, 
1146  966, 1540, 348, -475, -492, 1063, 1550, 390, 0, 96, 3, -337, 
1147  -426, -66, 425, 673, -156, -216, -223, -60, 1543, 1998, 618, 
1148  263, -250, -919, 557, 1800, 741"
1149
1150  ENDRASS_OCTIC_LEX_165="-93.2548,0,0,0,-309.019,0,0,527.529,0,395.647,\ 
1151  0,0,0,0,0,0,0,0,0,-867.529,0,0,1582.59,0,1186.94,0,0,0,0,0,-1055.06,0, 
1152  -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,-309.019,\ 
1153  0,0,1582.59,0,1186.94,0,0,0,0,-2110.12,0,-3165.17,0,-1186.94,0,0,0,0,0, 
1154  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, 
1155  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,\ 
1156  0,0,0,0,0,-1055.06,0,-1582.59,0,-593.47,0,0,0,0,0,0,874.039,0,1560.63,0,\ 
1157  1677.92,0,343.362,0,0,0,0,0,0,0,-256,0,-468.077,0,-789.019,0,\ 
1158  156.726,0,0.941125"
1159
1160
1161
1162
1163  ###################################################################################################
```
# Chapter 2 - Getting Started

## Section 2.2: Orbiter Session

```bash
$ORBITER test:
```

## Section 2.3: Makefiles and Shell Scripts

## Section 2.4: Objects and Activities

```bash
example_set:
  $ORBITER -v 2
  -define S -set -here "2,3,5,7,11,13" -end
  -print_symbols

object_F.2:
  $ORBITER -v 3 -define F -finite_field -q 2 -end

object_PG.3.2:
  $ORBITER
  -define F -finite_field -q 2 -end
  -define P -projective_space -n 3 -field F -v 0 -end

vector_ex:
```
# Section 2.5: Mathematical Data

SECTION MATHEMATICAL_DATA:

create_BLT_5_1:

create_surface_4_0:

# Section 2.6: Set Builder

SECTION_SET_BUILDER:

set_of_primes:
1258
1259  set_interval:
1260  ▶  $(ORBITER) -v 2 -define S -set -loop 0 64 1 -end \n1261  ▶  ▶  -print_symbols
1262
1263
1264
1265
1266  # Section 2.7: Vector Builder
1267
1268  SECTION VECTOR_BUILDER:
1269
1270
1271
1272  vector_example1:
1273  ▶  $(ORBITER) -v 2 \n1274  ▶  ▶  -define F -finite_field -q 5 -end \n1275  ▶  ▶  -define v -vector -field F -dense "0,1,2,3,4" -end \n1276  ▶  ▶  -print_symbols
1277
1278
1279  vector_example2:
1280  ▶  $(ORBITER) -v 2 \n1281  ▶  ▶  -define F -finite_field -q 5 -end \n1282  ▶  ▶  -define v -vector -field F -format 2 -dense "0,1,2,3,4,0" -end \n1283  ▶  ▶  -print_symbols
1284
1285  vector_example_sparse:
1286  ▶  $(ORBITER) -v 2 \n1287  ▶  ▶  -define F -finite_field -q 5 -end \n1288  ▶  ▶  -define v -vector -field F -format 4 -sparse 20 "1,0,1,19" -end \n1289  ▶  ▶  -print_symbols
1290
1291  vector_example_repeat:
1292  ▶  $(ORBITER) -v 2 \n1293  ▶  ▶  -define v -vector -repeat "0,1,2,3" 11 -end \n1294  ▶  ▶  -print_symbols
1295
1296
1297  vector_example_all_one_11:
1298  ▶  $(ORBITER) -v 2 \n1299  ▶  ▶  -define v -vector -repeat 1 11 -end \n1300  ▶  ▶  -print_symbols
1301
1302
1303  matrix_example1:
1304  ▶  $(ORBITER) -v 2 \n
536
matrix_example_co_1:  

TEST_FORMULA="(-a+b*b)*x*x+a*b*x"

TEST_FORMULA="test\_formula" "test\_formula" ""

$\text{dot } -\text{Tpng test\_formula.gv >test\_formula.png}$

open test\_formula.png

formulatevaluate:  

$\text{dot } -\text{Tpng test\_formula.gv >test\_formula.png}$

open test\_formula.png

# should evaluate to 1, since (-2+3*3)*4*4+2*3*4=2+4=6=1 mod 5

# but: problem with the leading minus sign

# Chapter 3 – Basic Algebra
# Section 3.1: Basic Number Theory

SECTION_BASIC_NUMBER_THEORY:

PR29:

\[ \text{ORBITER} -v 1 -\text{smallest primitive root} \, 29 \]

PR31:

\[ \text{ORBITER} -v 1 -\text{smallest primitive root} \, 31 \]

PR37:

\[ \text{ORBITER} -v 1 -\text{smallest primitive root} \, 37 \]

PR100:

\[ \text{ORBITER} -v 1 -\text{smallest primitive root interval} \, 2 \, 100 \]

# randomized algo:

PR_915839:

\[ \text{ORBITER} -v 5 -\text{primitive root} \, 915839 \]

# a primitive root modulo 915839 is 43085

PR_915839_check:

\[ \text{ORBITER} -v 5 -\text{power mod} \, 43085 \, 49842 \, 915839 \]

# the power of 43085 to the 49842 mod 915839 is 487320

DL_915839:

\[ \text{ORBITER} -v 5 -\text{discrete log} \, 487320 \, 43085 \, 915839 \]

# The discrete log is 49842 since 487320 = 43085^49842 mod 915839, time: 0:22

IM_723:

\[ \text{ORBITER} -v 5 -\text{inverse mod} \, 723 \, 4060 \]

IM_3_19:

\[ \text{ORBITER} -v 5 -\text{inverse mod} \, 3 \, 19 \]
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 2 3 151
  $(ORBITER) -v 1 -csv_file latex 1 \n  $(ORBITER) -v 1 -csv_file latex 1 \n  $(ORBITER) -v 1 -csv_file latex 1 \n  pdflatex order_of_q_mod_n_q2_3_151.tex
  open order_of_q_mod_n_q2_3_151.pdf

Eulerfunction_150:
  $(ORBITER) -v 1 -eulerfunction.interval 1 150
  $(ORBITER) -v 1 -csv_file latex 1 \n  $(ORBITER) -v 1 -csv_file latex 1 \n  $(ORBITER) -v 1 -csv_file latex 1 \n  pdflatex table_eulerfunction_1_150.tex
  open table_eulerfunction_1_150.pdf

Eulerfunction_900:
1446  ▷ $(ORBITER) -v 1 -eulerfunction_interval 900 1150
1447  ▷ $(ORBITER) -v 1 -csv_file_latex 1 \
1448  ▷ ▷ table_eulerfunction_900_1150.csv
1449  ▷ pdflatex table_eulerfunction_900_1150.tex
1450  ▷ open table_eulerfunction_900_1150.pdf
1451
1452  Eulerfunction_10000:
1453  ▷ $(ORBITER) -v 1 -eulerfunction_interval 10000 10150
1454  ▷ $(ORBITER) -v 1 -csv_file_latex 1 \
1455  ▷ ▷ table_eulerfunction_10000_10150.csv
1456  ▷ pdflatex table_eulerfunction_10000_10150.tex
1457  ▷ open table_eulerfunction_10000_10150.pdf
1458
1459  PR_1000:
1460  ▷ $(ORBITER) -v 1 -smallest_primitive_root_interval 2 1000
1461  ▷ $(ORBITEER) -v 1 -csv_file_latex 1 \
1462  ▷ ▷ primitive_element_table_2_1000.csv
1463  ▷ pdflatex primitive_element_table_2_1000.tex
1464  ▷ open primitive_element_table_2_1000.pdf
1465
1466  PE_number_1000:
1467  ▷ $(ORBITER) -v 1 -number_of_primitive_roots_interval 2 1000
1468  ▷ $(ORBITEER) -v 1 -csv_file_latex 1 table_number_of_pe_2_1000.csv
1469  ▷ pdflatex table_number_of_pe_2_1000.tex
1470  ▷ open table_number_of_pe_2_1000.pdf
1471
1472  number_ofPrimitive_roots_10000:
1473  ▷ $(ORBITER) -v 1 -number_of_primitive_roots_interval 10000 10001
1474
1475  power_function_2_mod_11:
1476  ▷ $(ORBITER) -v 5 -power_function_mod_n 2 11
1477  ▷ $(ORBITEER) -v 1 -csv_file_latex 1 power_function_k2_n11.csv
1478  ▷ pdflatex power_function_k2_n11.tex
1479  ▷ open power_function_k2_n11.pdf
1480
1481  draw_mod_13:
1482  ▷ $(ORBITER) -v 2 \
1483  ▷ ▷ -draw_options -embedded -end \
1484  ▷ ▷ -draw_mod.n -n 13 -file mod_13 -power_cycle 2 -end
1485  ▷ pdflatex mod_13_draw.tex
1486  ▷ open mod_13_draw.pdf
1487
1488  Chinese_remainders_A:
1489
1490  ▷ $(ORBITER) -v 2 \<
\begin{verbatim}
1493 \> \> -define R -vector -dense "2,2,5" -end \ 
1494 \> \> -define M -vector -dense "5,6,7" -end \ 
1495 \> \> -Chinese_remainders R M \ 
1496 \ 
1497 Chinese_remainders_B: \ 
1498 \> \> $(ORBITER) -v 2 \ 
1499 \> \> \> -define R -vector -dense "38,2" -end \ 
1500 \> \> \> -define M -vector -dense "74,27" -end \ 
1501 \> \> \> -Chinese_remainders R M \ 
1502 \ 
1503 Chinese_remainders_C2: \ 
1504 \> \> $(ORBITER) -v 2 \ 
1505 \> \> \> -define R -vector -dense "2,3" -end \ 
1506 \> \> \> -define M -vector -dense "2147483647,5915587277" -end \ 
1507 \> \> \> -Chinese_remainders R M \ 
1508 \ 
1509 \ 
1510 # The solution is 5684294357108828365 modulo 12703626939758759219 (computed in longinteger) \ 
1511 \ 
1512 # checking with Maple: \ 
1513 # 5684294357108828365 mod 2147483647; \ 
1514 # \ 
1515 \ 
1516 # 5684294357108828365 mod 5915587277; \ 
1517 # \ 
1518 \ 
1519 \ 
1520 \ 
1521 Chinese_remainders_C3: \ 
1522 \> \> $(ORBITER) -v 2 \ 
1523 \> \> \> -define R -vector -dense "2,3,4" -end \ 
1524 \> \> \> -define M -vector -dense "2147483647,5915587277,3267000013" -end \ 
1525 \> \> \> -Chinese_remainders R M \ 
1526 \ 
1527 \ 
1528 # The solution is 31431541759324477327451572539 modulo 4150274937739016585336869847 (computed in longinteger) \ 
1529 \ 
1530 # checking with Maple: \ 
1531 # 31431541759324477327451572539 mod 2147483647; \ 
1532 # \ 
1533 \ 
1534 # 31431541759324477327451572539 mod 5915587277; \ 
1535 # \ 
1536 \ 
1537 # 31431541759324477327451572539 mod 3267000013; \ 
\end{verbatim}
# Section 3.2: Prime Fields

**SECTION**

Primes:

```bash
$ (ORBITER) -v 3 -define F -finite_field -q 2
$ with F -do -finite_field_activity -cheat_sheet_GF
$ pdflatex GF_2.tex
$ open GF_2.pdf

F_3:
$ (ORBITER) -v 3 -define F -finite_field -q 3
$ with F -do -finite_field_activity -cheat_sheet_GF
$ pdflatex GF_3.tex
$ open GF_3.pdf

F_5:
$ (ORBITER) -v 3 -define F -finite_field -q 5
$ with F -do -finite_field_activity -cheat_sheet_GF
$ pdflatex GF_5.tex
$ open GF_5.pdf

F_5_add_table:
$ (ORBITER) -v 3 -define F -finite_field -q 5
$ with F -do -finite_field_activity -cheat_sheet_GF
$ draw_matrix -input_csv_file GF_q5_addition_table.csv -box_width 40 -bit_depth 24 -partition 3 5 5

F_7:
$ (ORBITER) -v 3 -define F -finite_field -q 7
$ with F -do -finite_field_activity -cheat_sheet_GF
$ pdflatex GF_7.tex
$ open GF_7.pdf
```
F_127:
\[ (ORBITER) -v 3 \]
\[ \text{-define F -finite_field -q 127 -end} \]
\[ \text{-with F -do -finite_field_activity -cheat_sheet_GF -end} \]

F_11_product_of_all_nonzero_elements:
\[ (ORBITER) -v 3 \]
\[ \text{-define F -finite_field -q 11 -end} \]
\[ \text{-define S -vector -field F -loop 1 11 1 -end} \]
\[ \text{-with F -do -finite_field_activity -product_of S -end} \]

F_7_vandermonde:
\[ (ORBITER) -v 3 \]
\[ \text{-define F -finite_field -q 7 -end} \]
\[ \text{-with F -do -finite_field_activity} \]
\[ \text{-Vandermonde_matrix} \]
\[ \text{-end} \]

F_101_wo:
\[ (ORBITER) -v 3 \]
\[ \text{-define F -finite_field -q 101 -without_tables -end} \]
\[ \text{-with F -do -finite_field_activity -cheat_sheet_GF -end} \]
\[ \text{pdflatex GF_101.tex} \]
\[ \text{open GF_101.pdf} \]

F_1021_wo:
\[ (ORBITER) -v 3 \]
\[ \text{-define F -finite_field -q 1021 -without_tables -end} \]
\[ \text{-with F -do -finite_field_activity -cheat_sheet_GF -end} \]

# Section 3.3: Extension Fields

SECTION_EXTENSION_FIELDS:
\( \text{F}_4: \)
\[
\text{\$(ORBITE R) -v 3 \ \\
define F -finite_field -q 4 -end \ \\
with F -do -finite_field_activity -cheat_sheet_GF -end}
\]
\[
pdflatex GF_4.tex
open GF_4.pdf
\]

\( \text{F}_4 \text{tables}: \)
\[
\text{\$(ORBITE R) -v 3 \ \\
\define F -finite_field -q 4 -end \ \\
\with F -do -finite_field_activity -cheat_sheet_GF -end}
\]
\[
\text{\$pdflatex GF_4.tex} \ \\
\text{\$open GF_4.pdf}
\]

\( \text{F}_16: \)
\[
\text{\$(ORBITE R) -v 3 \ \\
\define F -finite_field -q 16 -end \ \\
\with F -do -finite_field_activity -cheat_sheet_GF -end}
\]
\[
pdflatex GF_16.tex
\]

\( \text{F}_16 \text{tables}: \)
\[
\text{\$(ORBITE R) -v 3 \ \\
\define F -finite_field -q 16 -end \ \\
\with F -do -finite_field_activity -cheat_sheet_GF -end}
\]
\[
\text{\$pdflatex GF_16.tex}
\]

544
# Section 3.4: Linear Algebra over Finite Fields

SECTION_LINEAR_ALGEBRA:

RREF:

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

RREF_V7:

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

nullspace:

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

classes_GL_3_2:
 $(ORBITER) -v 7 \\
-define F -finite_field -q 2 -end \\
-all_rational_normal_forms F 3 \\
pdflatex Class_reps.GL.3.2.tex \\
open Class_reps.GL.3.2.pdf

classes_GL_4_2:
 $(ORBITER) -v 7 \\
-define F -finite_field -q 2 -end \\
-all_rational_normal_forms F 4 \\
pdflatex Class_reps.GL.4.2.tex \\
open Class_reps.GL.4.2.pdf

# 252 classes

RREF.demo.4.4.q5:
 $(ORBITER) -v 2 \\
-define F -finite_field -q 5 -end \\
-with F -do \\
-finite_field_activity -RREF.demo 4 4 -end \\
pdflatex RREF.example.q5.4.4.tex \\
#open RREF_example.q5.4.4.pdf \\
gs -sDEVICE=png16 -dFIXEDMEDIA \\
-dDEVICEWIDTHPOINTS=500 -dDEVICEHEIGHTPOINTS=450 \\
-r240 -oRREF.example.q5.4.4.page%02d.png \\
RREF.example.q5.4.4.pdf

RREF.demo.4.6.q7:
```bash
$(ORBITER) -v 2 \n-define F -finite_field -q 7 -end \n-with F -do \n-finite_field_activity -RREF_random_matrix 4 6 -end \npdflatex RREF_example_q7_4_6.tex
gs -sDEVICE=png16 -dFIXEDMEDIA \n-dDEVICEWIDTHPOINTS=500 -dDEVICEHEIGHTPOINTS=450 \nr240 -oRREF_example_q7_4_6_page%02d.png \nRREF_example_q7_4_6.pdf
open RREF_example_q7_4_6.pdf
```

```bash
RREF_demo_4_8_q8:
$(ORBITER) -v 2 \n-define F -finite_field -q 8 -end \n-with F -do \n-finite_field_activity -RREF_random_matrix 4 8 -end \npdflatex RREF_example_q8_4_8.tex
#open RREF_example_q8_4_8.pdf
gs -sDEVICE=png16 -dFIXEDMEDIA \n-dDEVICEWIDTHPOINTS=500 -dDEVICEHEIGHTPOINTS=450 \nr240 -oRREF_example_q8_4_8_page%02d.png \nRREF_example_q8_4_8.pdf
open RREF_example_q8_4_8.pdf
```

```bash
RREF_RS_6_4_7:
$(ORBITER) -v 2 \n-define F -finite_field -q 7 -end \n-define v -vector -field F -format 4 \n-end \n-with F -do \n-finite_field_activity -RREF v -end \npdflatex RREF_example_q7_4_6.tex
open RREF_example_q7_4_6.pdf
gs -sDEVICE=png16 -dFIXEDMEDIA \n-dDEVICEWIDTHPOINTS=500 -dDEVICEHEIGHTPOINTS=450 \nr240 -oRREF_example_q7_4_6_page%02d.png \nRREF_example_q7_4_6.pdf
```

# Section 3.5: Advanced Topics in finite fields
SECTION_ADVANCED_TOPICS_IN_FINITE_FIELDS:

normal_basis_2_3:

normal_basis_2_6:

F8_over_F2_field_reduction:

F64_over_F8_field_reduction:
\begin{verbatim}
1867  -input_csv_file F64_over_F8.csv \\
1868  -box_width 40 -bit_depth 24 \\
1869  -partition 4 "2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2" -end \\
1870  open F64_over_F8.draw.bmp \\
1871  \#pdflatex field_reduction_Q64_q8_8_8.tex \\
1872  \#open field_reduction_Q64_q8_8_8.pdf \\
1873 \\
1874  F_64_over_F4_field_reduction: \\
1875  $(ORBITER) -v 2 \\
1876  -define F -finite_field -q 64 -end \\
1877  -define elts -vector -field F -loop 0 64 1 -end \\
1878  -with F -do \\
1879  -finite_field.activity \\
1880  \#field_reduction "F64_over_F4" 4 8 8 elts -end \\
1881  $(ORBITER) -v 2 -draw_matrix \\
1882  -input_csv_file F64_over_F4.csv \\
1883  -box_width 40 -bit_depth 24 \\
1884  -partition 4 "3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3" -end \\
1885  open F64_over_F4.draw.bmp \\
1886  \#pdflatex field_reduction_Q64_q4_8_8.tex \\
1887  \#open field_reduction_Q64_q4_8_8.pdf \\
1888 \\
1889 \\
1890  F_64_over_F2_field_reduction: \\
1891  $(ORBITER) -v 2 \\
1892  -define F -finite_field -q 64 -end \\
1893  -define elts -vector -field F -loop 0 64 1 -end \\
1894  -with F -do \\
1895  -finite_field.activity \\
1896  \#field_reduction "F64_over_F2" 2 8 8 elts -end \\
1897  $(ORBITER) -v 2 -draw_matrix \\
1898  -input_csv_file F64_over_F2.csv \\
1899  -box_width 40 -bit_depth 24 \\
1900  -partition 4 "6,6,6,6,6,6,6,6,6" "6,6,6,6,6,6,6,6,6" -end \\
1901  open F64_over_F2.draw.bmp \\
1902  \#pdflatex field_reduction_Q64_q2_8_8.tex \\
1903  \#open field_reduction_Q64_q2_8_8.pdf \\
1904 \\
1905 \\
1906 \\
1907 \\
1908 \\
1909  F_8_Nth_roots_21: \\
1910  $(ORBITER) -v 3 \\
1911  -define F -finite_field -q 8 -override_polynomial 11 -end \\
1912  -with F -do -coding_theoretic_activity \\
\end{verbatim}
F_8.vandermonde:
$\$(ORBITER) -v 3 \$
$\$ -define F -finite_field -q 8 -end \$
$\$ -with F -do -finite_field_activity \$
$\$ -Vandermonde_matrix \$
$\$ -end \$

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

rm Vandermonde_1024.csv
rm Vandermonde_inv_1024.csv

#User time: 0:46

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

F_17_NTTCompile: NTT_k4_q17.cpp
$\$(MY_CPP) NTT_k4_q17.cpp $(CPPFLAGS) \$
$\$ -o NTT_k4_q17.out \$

# Section 3.6: Basic Ring Theory

550
SECTION_BASIC_RING_THEORY:

Polynomial_ring:

```latex
\$\text{ORBITER} -v 3 \ $
\$\text{ORBITER} -d -define F -finite_field -q 4 -end \$
\$\text{ORBITER} -d -define R -polynomial_ring -field F \$
\$\text{ORBITER} -d -number_of_variables 4 \$
\$\text{ORBITER} -d -homogeneous_of_degree 3 \$
\$\text{ORBITER} -d -variables "x0,x1,x2,x3" "x.0,x.1,x.2,x.3" \$
\$\text{ORBITER} -end \$
```

SECTION_FINITE_PROJECTIVE_SPACES:

PG_3_2.easy:

```latex
\$\text{ORBITER} -v 2 \$
\$\text{ORBITER} -d -define F -finite_field -q 2 -end \$
\$\text{ORBITER} -d -define P -projective_space -n 3 -field F -end \$
```

PG_1_16:

```latex
\$\text{ORBITER} -v 2 \$
\$\text{ORBITER} -d -define F -finite_field -q 16 -end \$
\$\text{ORBITER} -d -define P -projective_space -n 1 -field F -v 0 -end \$
\$\text{ORBITER} -d -with P -do -projective_space_activity \$
\$\text{ORBITER} -d -cheat_sheet \$
\$\text{ORBITER} -d -end \$
```

```
pdflatex PG_1_16.tex
open PG_1_16.pdf
```
2008
2009
2010  PG_2.4:
2011  $(ORBITER) -v 2 \ 
2012  \>&  -define F -finite_field -q 4 -end \ 
2013  \>&  \>&  -define P -projective_space -n 2 -field F -v 0 -end \ 
2014  \>&  \>&  \>&  -with P -do -projective_space_activity \ 
2015  \>&  \>&  \>&  \>&  -cheat_sheet \ 
2016  \>&  \>&  \>&  \>&  \>&  -end
2017  pdflatex PG_2.4.tex
2018  open PG_2.4.pdf
2019
2020
2021
2022
2023  PG_2.13:
2024  $(ORBITER) -v 2 \ 
2025  \>&  -define F -finite_field -q 13 -end \ 
2026  \>&  \>&  -define P -projective_space -n 2 -field F -v 0 -end \ 
2027  \>&  \>&  \>&  -with P -do -projective_space_activity \ 
2028  \>&  \>&  \>&  \>&  -cheat_sheet \ 
2029  \>&  \>&  \>&  \>&  \>&  -end
2030  pdflatex PG_2.13.tex
2031  open PG_2.13.pdf
2032
2033
2034
2035
2036  PG_2.64:
2037  $(ORBITER) -v 2 \ 
2038  \>&  -define F -finite_field -q 64 -end \ 
2039  \>&  \>&  -define P -projective_space -n 2 -field F -v 0 -end \ 
2040  \>&  \>&  \>&  -with P -do -projective_space_activity \ 
2041  \>&  \>&  \>&  \>&  -cheat_sheet \ 
2042  \>&  \>&  \>&  \>&  \>&  -end
2043  pdflatex PG_2.64.tex
2044  open PG_2.64.pdf
2045
2046
2047
2048  PG_3.2:
2049  $(ORBITER) -v 2 \ 
2050  \>&  -define F -finite_field -q 2 -end \ 
2051  \>&  \>&  -define P -projective_space -n 3 -field F -v 0 -end \ 
2052  \>&  \>&  \>&  -with P -do -projective_space_activity \ 
2053  \>&  \>&  \>&  \>&  -cheat_sheet \ 
2054  \>&  \>&  \>&  \>&  \>&  -end
2055 \pdflatex PG_3_2.tex
2056 \open PG_3_2.pdf
2057
2058
2059 PG_3_4:
2060 \$(\text{ORBITER}) -v 2 \$
2061 \define F \texttt{-finite_field} -q 4 -end \$
2062 \define P \texttt{-projective_space} -n 3 -field F -v 0 -end \$
2063 \with P \texttt{-do -projective_space_activity} \$
2064 \texttt{-cheat_sheet} \$
2065 \texttt{-end}
2066 \pdflatex PG_3_4.tex
2067 \open PG_3_4.pdf
2068
2069 PG_3_5:
2070 \$(\text{ORBITER}) -v 2 \$
2071 \define F \texttt{-finite_field} -q 5 -end \$
2072 \define P \texttt{-projective_space} -n 3 -field F -v 0 -end \$
2073 \with P \texttt{-do -projective_space_activity} \$
2074 \texttt{-cheat_sheet} \$
2075 \texttt{-end}
2076 \pdflatex PG_3_5.tex
2077 \open PG_3_5.pdf
2078
2079
2080 PG_3_7:
2081 \$(\text{ORBITER}) -v 2 \$
2082 \define F \texttt{-finite_field} -q 7 -end \$
2083 \define P \texttt{-projective_space} -n 3 -field F -v 0 -end \$
2084 \with P \texttt{-do -projective_space_activity} \$
2085 \texttt{-cheat_sheet} \$
2086 \texttt{-end}
2087 \pdflatex PG_3_7.tex
2088 \open PG_3_7.pdf
2089
2090
2091
2092
2093 PG_3_8:
2094 \$(\text{ORBITER}) -v 2 \$
2095 \define F \texttt{-finite_field} -q 8 -end \$
2096 \define P \texttt{-projective_space} -n 3 -field F -v 0 -end \$
2097 \with P \texttt{-do -projective_space_activity} \$
2098 \texttt{-cheat_sheet} \$
2099 \texttt{-end}
2100 \pdflatex PG_3_8.tex
2101 \open PG_3_8.pdf
\$(ORBITER) -v 2 \ 
-define F -finite_field -q 16 -end \ 
-define P -projective_space -n 3 -field F -v 0 -end \ 
-with P -do -projective_space_activity \ 
- cheat_sheet \ 
-end 
\$pdflatex PG_3.16.tex 
\$open PG_3.16.pdf 

\$(ORBITER) -v 2 \ 
-define F -finite_field -q 25 -end \ 
-define P -projective_space -n 3 -field F -v 0 -end \ 
-with P -do -projective_space_activity \ 
- cheat_sheet \ 
-end 
\$pdflatex PG_3.25.tex 
\$open PG_3.25.pdf 

\$(ORBITER) -v 2 \ 
-define F -finite_field -q 3 -end \ 
-define P -projective_space -n 4 -field F -v 0 -end \ 
-with P -do -projective_space_activity \ 
- cheat_sheet \ 
-end 
\$pdflatex PG_4.3.tex 
\$open PG_4.3.pdf 

\$(ORBITER) -v 2 \ 
-define F -finite_field -q 2 -end \ 
-define P -projective_space -n 8 -field F -v 0 -end \ 
-with P -do -projective_space_activity \ 
- cheat_sheet \ 
-end 
\$pdflatex PG_8.2.tex
open PG_8_2.pdf

SECTION_INDEXING_POINTS:

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

PG_2_4_unrank_point:
\$(ORBITER) -v 2 \\
-define v -vector -dense "19,20" -end \\
-define F -finite_field -q 4 -end \\
-with F -do -finite_field_activity \\
-unrank_point_in_PG 2 v -end

elliptic_curve_b1_c3_q11.txt:
\$(ORBITER) -v 2 \\
-define F -finite_field -q 11 -end \\
-define P -projective_space -n 2 -field F -v 0 -end \\
-define EC -geometric_object P \\
-elliptic_curve 1 3 \\
-end \\
-with EC -do -combinatorial_object_activity -save \\
-end

PG_2_2.incidence_matrix:
\$(ORBITER) -v 2 \\
-define F -finite_field -q 2 -end \\
-define P -projective_space -n 2 -field F -v 0 -end \\
-with P -do -projective_space_activity \\
-export_point_line_incidence_matrix \\
-end
PG_2_4.incidence_matrix:

PG_2_8.incidence_matrix:
SECTION_FINEDESARGUESIANPROJECTIVEPLANES:

**PG\_2\_16:**

```latex
\textbackslash{}$(\text{ORBITER}) -v 2 \textbackslash{} -\text{draw}-\text{options} -\text{xin} 20000 -\text{yin} 20000 \textbackslash{} -\text{radius} 200 -\text{line}-\text{width} 0.3 -\text{nodes}-\text{empty} -\text{end} \textbackslash{} -\text{define} F -\text{finite}-\text{field} -q 16 -\text{end} \textbackslash{} -\text{define} P -\text{projective}-\text{space} -n 2 -\text{field} F -v 0 -\text{end} \textbackslash{} -\text{with} P -\text{do} -\text{projective}-\text{space}-\text{activity} \textbackslash{} -\text{cheat}-\text{sheet} \textbackslash{} -\text{end} \textbackslash{} \text{pdflatex} PG\_2\_16.tex \textbackslash{} open PG\_2\_16.pdf
```

**PG\_2\_4\_with\_decomposition:**

```latex
\textbackslash{}$(\text{ORBITER}) -v 2 \textbackslash{} -\text{define} F -\text{finite}-\text{field} -q 4 -\text{end} \textbackslash{} -\text{define} P -\text{projective}-\text{space} -n 2 -\text{field} F -v 0 -\text{end} \textbackslash{} -\text{with} P -\text{do} -\text{projective}-\text{space}-\text{activity} \textbackslash{} -\text{cheat}-\text{sheet} \text{for}\_\text{decomposition}\_\text{by}\_\text{element} PG \textbackslash{} 1 "0,1,0, 0,0,1, 2,1,1, 0" \textbackslash{} \text{pdflatex} PG\_2\_4\_singer.tex \textbackslash{} 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:
$\text{(ORBITER) -v 2 \n-\text{list\_arguments} \n-\text{define R -vector -repeat 1 21 -end} \n-\text{define C -vector -repeat 1 21 -end} \n-\text{draw\_matrix} \n-\text{input\_csv\_file PG_2.4_singer_incma_cyclic.csv} \n-\text{box\_width 40 -bit\_depth 24} \n-\text{partition 3 R C} \n-\text{end} \n\text{open PG_2.4_singer_incma_cyclic\_draw.bmp}$

PG_2.4_incma_singer_sub_3:
$\text{(ORBITER) -v 2 \n-\text{list\_arguments} \n-\text{define R -vector -repeat 3 7 -end} \n-\text{define C -vector -repeat 3 7 -end} \n-\text{draw\_matrix} \n-\text{input\_csv\_file PG_2.4_singer_incma_subgroup\_index\_3.csv} \n-\text{box\_width 40 -bit\_depth 24} \n-\text{partition 3 R C} \n-\text{end} \n\text{open PG_2.4_singer_incma_subgroup\_index\_3\_draw.bmp}$

PG_2.4_incma_singer_sub_7:
$\text{(ORBITER) -v 2 \n-\text{draw\_matrix} \n-\text{input\_csv\_file PG_2.4_singer_incma_subgroup\_index\_7.csv} \n-\text{box\_width 20 -bit\_depth 24} \n-\text{partition 3 3,3,3,3,3,3 3,3,3,3,3,3,3 -end} \n\text{open PG_2.4_singer_incma_subgroup\_index\_7\_draw.bmp}$

# Section 4.4: The Grassmannian
GRASSMANNIAN: 2339

GR3_2_2:

$\begin{align*}
\text{define } F & - \text{finite field } -q 2 \ - \text{end } \\
\text{with } F & \text{ do } - \text{finite field activity } \\
\text{do } & - \text{cheat sheet Gr 3 2 } - \text{end}
\end{align*}$

pdflatex Gr3_2_2.tex

open Gr3_2_2.pdf

PG3_3.rank_lines:

$\begin{align*}
\text{define } v1 & - \text{vector } - \text{format 3 } \\
\text{dense } & "1,0,2,2,0,1,1,2, 1,0,2,0,0,1,1,2, 1,0,2,2,0,1,2,1" \ \\
\text{end } & \\
\text{define } v2 & - \text{vector } - \text{format 3 } \\
\text{dense } & "1,0,0,0,0,1,0,0, 1,0,0,0,0,0,1, 0,1,0,0,0,0,2,1" \ \\
\text{end } & \\
\text{define } F & - \text{finite field } -q 3 \ - \text{end } \\
\text{define } P & - \text{projective space } -n 3 - \text{field } F \ - v 0 \ - \text{end } \\
\text{with } P & \text{ do } \\
\text{projective space activity } & \\
\text{rank lines in } & \text{PG v1 } \\
\text{end } & \\
\text{with } P & \text{ do } \\
\text{projective space activity } & \\
\text{rank lines in } & \text{PG v2 } \\
\text{end } & \\
\text{end } & \\
\end{align*}$

PG3_5.unrank_lines:

$\begin{align*}
\text{define } v & - \text{vector } \\
\text{dense } & "0,36,72,108,144,805" \ \\
\text{end } & \\
\text{define } F & - \text{finite field } -q 5 \ - \text{end } \\
\text{define } P & - \text{projective space } -n 3 - \text{field } F \ - v 0 \ - \text{end } \\
\text{with } P & \text{ do } \\
\text{projective space activity } & \\
\text{unrank lines in } & \text{PG v } \\
\text{end } & \\
\end{align*}$

559
planes_in_pencil:

$\text{ORBITER} -v 2 \$

-define F -finite_field -q 8 -end 

-define P -projective_space -n 3 -field F -v 0 -end 

-with P -do 

-projective_space_activity 

-planes_through_line 0 

-end

# Section 4.5: Algebraic Sets

SECTION_ALGEBRAIC_SETS:

EC_11.txt:

$\text{ORBITER} -v 2 \$

-define F -finite_field -q 11 -end 

-define R -polynomial_ring -field F 

-number_of_variables 3 

-homogeneous_of_degree 3 

-end 

-define P -projective_space -n 2 -field F -v 0 -end 

-define EC -geometric_object P 

-projective_variety R 

-"EC_11" "EC\_11" 

-$(EC_{11\_EQUATION}) 

-end 

-with EC -do -combinatorial_object_activity -save 

-end

Hirschfeld_surface_q4.txt:

$\text{ORBITER} -v 2 \$

-define F -finite_field -q 4 -end 

-define R -polynomial_ring -field F 

-number_of_variables 4 

-homogeneous_of_degree 3 

-end 

560
define P -projective_space -n 3 -field F -v 0 -end \ndefine H4 -geometric_object P \ndefine H16 -geometric_object P \ndefine F -finite_field -q 16 -end \ndefine F -finite_field -q 2 -end \nwith H4 -do -combinatorial_object_activity -save \nwith H16 -do -combinatorial_object_activity -save \n# creates Hirschfeld_surface_q4.txt

Hirschfeld_surface_q16.txt:

define F -finite_field -q 16 -end \ndefine R -polynomial_ring -field F \nnumber_of_variables 4 \nhomogeneous_of_degree 3 \ndefine P -projective_space -n 3 -field F -v 0 -end \ndefine H16 -geometric_object P \nwith H16 -do -combinatorial_object_activity -save \n# the coefficient vector is given as a list of pairs.
# 165 = binomial(11,3)

# Section 4.6: The Klein Quadric and Pluecker coordinates
SECTION_KLEIN_QUADRIC_AND_PLUECKER_COORDINATES:

GR_4_2_2:
$ (ORBITER) -v 2 \n$ (ORBITER) -v 2 \nwith F -do -finite_field_activity \n
# creates Hirschfeld_surface_q4.txt
Hirschfeld_surface_q16.txt:

define F -finite_field -q 2 -end \nwith F -do -finite_field_activity \n
# the coefficient vector is given as a list of pairs.
# 165 = binomial(11,3)

# Section 4.6: The Klein Quadric and Pluecker coordinates
SECTION_KLEIN_QUADRIC_AND_PLUECKER_COORDINATES:

GR_4_2_2:
# Section 4.7: Orthogonal spaces

SECTION

ORTOGONAL

SPACES:

Op.4.2:

$\text{pdflatex Gr 4.2.2.tex}$

$\text{open Gr 4.2.2.pdf}$

Op.5.2:

$\text{pdflatex 0.5.2.incidence_matrix.csv}$

$\text{open 0.5.2.incidence_matrix.draw.bmp}$

#0(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
\begin{verbatim}
2525 ▶ ▶ -define F -finite_field -q 2 -end \\
2526 ▶ ▶ -define O -orthogonal_space 1 6 F -without_group -end \\
2527 ▶ ▶ -with 0 -do -orthogonal_space_activity \ 
2528 ▶ ▶ ▶ -cheat_sheet_orthogonal -end \\
2529 ▶ pdflatex 0_1_6_2_report.tex \\
2530 ▶ open 0_1_6_2_report.pdf \\
2531  \\
2532  \\
2533 Op_6_2.incidence_matrix.csv: 
2534 ▶ $(ORBITER) -v 2 \ 
2535 ▶ ▶ -define F -finite_field -q 2 -end \\
2536 ▶ ▶ -define O -orthogonal_space 1 6 F -end \\
2537 ▶ ▶ -with 0 -do -orthogonal_space_activity \ 
2538 ▶ ▶ ▶ -export_point_line_incidence_matrix \ 
2539 ▶ ▶ ▶ -end \\
2540 ▶ ▶ $(ORBITER) -v 2 \ 
2541 ▶ ▶ ▶ -define all_one_r -vector -repeat 1 35 -end \\
2542 ▶ ▶ ▶ -define all_one_c -vector -repeat 1 105 -end \\
2543 ▶ ▶ ▶ -draw_matrix \ 
2544 ▶ ▶ ▶ ▶ -input_csv_file Op_6_2.incidence_matrix.csv \ 
2545 ▶ ▶ ▶ ▶ -box_width 20 -bit_depth 8 \ 
2546 ▶ ▶ ▶ ▶ -partition 2 \ 
2547 ▶ ▶ ▶ ▶ ▶ all_one_r all_one_c \ 
2548 ▶ ▶ ▶ ▶ ▶ -end \\
2549 ▶ open Op_6_2.incidence_matrix_draw.bmp \\
2550  \\
2551  \\
2552 Op_6_2.with_group: 
2553 ▶ $(ORBITER) -v 2 \ 
2554 ▶ ▶ -define F -finite_field -q 2 -end \\
2555 ▶ ▶ -define O -orthogonal_space 1 6 F -end \\
2556 ▶ ▶ -with 0 -do -orthogonal_space_activity \ 
2557 ▶ ▶ ▶ -cheat_sheet_orthogonal -end \\
2558 ▶ pdflatex 0_1_6_2_report.tex \\
2559 ▶ open 0_1_6_2_report.pdf \\
2560  \\
2561 # problem: 
2562 # error message: 
2563 #stabilizer_chain_base_data::allocate_base_data degree is too large 
2564  
2565  
2566 Op_6.8: 
2567 ▶ $(ORBITER) -v 2 \ 
2568 ▶ ▶ -define F -finite_field -q 8 -end \\
2569 ▶ ▶ -define O -orthogonal_space 1 6 F -end \\
2570 ▶ ▶ -with 0 -do -orthogonal_space_activity \ 
2571 ▶ ▶ ▶ -cheat_sheet_orthogonal \ 
\end{verbatim}
Op_8.2:
\$\text{(ORBITER) -v 2 \ -define F -finite_field -q 2 \end \ -define O -orthogonal_space 1 8 F -without_group \end \ -with O -do -orthogonal_space_activity \ -cheat_sheet_orthogonal \end \ \text{pdflatex 0.1_6_8_report.tex \ open 0.1_6_8_report.pdf}}$

Op_6.64:
\$\text{(ORBITER) -v 2 \ -define F -finite_field -q 64 \end \ -define O -orthogonal_space 1 6 F -without_group \end \ -with O -do -orthogonal_space_activity \ -cheat_sheet_orthogonal \end \ \text{pdflatex 0.1_6_64_report.tex \ open 0.1_6_64_report.pdf}}$

# problem, because we are trying to create $\text{PGL}(6,64)$:

Op_6.64_line_rank_problem:
\$\text{(ORBITER) -v 4 \ -define F -finite_field -q 64 \end \ -define O -orthogonal_space 1 6 F \end \ -with O -do -orthogonal_space_activity \ -unrank_line_through_two_points 15447347 15225451 \ -end}$

# use option -without_group to skip the group. This will work:

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

# this will create a basic report without the group:
elliptic_quadric_subspace:

BLT_database_7_1:

BLT_database_7_1_print:
2665 \> pdflatex S_set_report.tex
2666 \> open S_set_report.pdf
2667
2668 Doily_{W,2}:
2669 \> $(\text{ORBITER}) \- v \ 2 \$
2670 \> \> -define \ F \ -finite_field \ -q \ 2 \ -end \$
2671 \> \> \> -define \ 0 \ -orthogonal_space \ 0 \ 5 \ F \ -without_group \ -end \$
2672 \> \> \> -define \ W_2\_points \ -set \ -loop \ 0 \ 15 \ 1 \ -end \$
2673 \> \> \> -define \ W_2\_lines \ -set \ -loop \ 0 \ 15 \ 1 \ -end \$
2674 \> \> \> -with \ 0 \ -do \$
2675 \> \> \> -orthogonal_space_activity \$
2676 \> \> \> -print\_points \ W_2\_points \$
2677 \> \> \> -end \$
2678 \> \> \> -with \ 0 \ -do \$
2679 \> \> \> -orthogonal_space_activity \$
2680 \> \> \> -print\_lines \ W_2\_lines \$
2681 \> \> \> -end
2682 \> \> pdflatex W_2\_points_set_report.tex
2683 \> \> open W_2\_points_set_report.pdf
2684 \> \> pdflatex W_2\_lines_set\_of\_lines_report.tex
2685 \> \> open W_2\_lines_set\_of\_lines_report.pdf
2686
2687
2688 \> \# the output defines doily.csv
2689
2690
2691
2692
2693
2694
2695 \# Section 4.8: Hermitian varieties
2696
2697
2698 SECTION\_HERMITIAN\_VARIETIES:
2699
2700
2701
2702 \> H_{2,4}:
2703 \> \> $(\text{ORBITER}) \- v \ 2 \$
2704 \> \> \> -define \ F \ -finite_field \ -q \ 4 \ -end \$
2705 \> \> \> -with \ F \ -do \ -finite_field\_activity \$
2706 \> \> \> -cheat\_sheet\_hermitian \ 2 \ -end
2707 \> \> pdflatex H_{2,4}.tex
2708 \> \> open H_{2,4}.pdf
2709
2710
2711 \> H_{2,9}:
2712praak {ORBITER} -v 2 \\
2713↳↳ -define F -finite_field -q 9 -end \\
2714↳↳ -with F -do -finite_field_activity \\
2715↳↳↳ -cheat_sheet_hermitian 2 -end \\
2716↳↳ pdflatex H_2.9.tex \\
2717↳ open H_2.9.pdf \\
2718 \\
2719 # 28 points: 6, 11, 9, 7, 14, 19, 17, 15, 80, 75, 78, 74, 35, 30, 33, 29, 62, 57, 
2720 60, 56, 26, 21, 24, 3, 37, 82, 64, 46 \\
2721 \\
2722 H_3.4: \\
2723↳ $(ORBITER) -v 2 \\
2724↳↳ -define F -finite_field -q 4 -end \\
2725↳↳ -with F -do -finite_field_activity \\
2726↳↳↳ -cheat_sheet_hermitian 3 -end \\
2727↳↳ pdflatex H_3.4.tex \\
2728↳ open H_3.4.pdf \\
2729 \\
2730 # H_3.4 = the Hirschfeld surface \\
2731 \\
2732 # Section 4.9: Projective Space Advanced Topics \\
2733 \\
2734 SECTION_PROJECTIVE_SPACEADVANCED_TOPICS: \\
2735 \\
2736 fix_structure_2A: \\
2737↳ $(ORBITER) -v 2 \\
2738↳↳ -define F -finite_field -q 4 -end \\
2739↳↳ -define P -projective_space -n 3 -field F -v 0 -end \\
2740↳↳↳ -projective_space_activity \\
2741↳↳↳ -cheat_sheet_for_decomposition_by_element_PG 1 \\
2742↳↳↳↳ "1,0,0,0, 0,1,0,0, 0,0,1,0, 0,0,0,1, 1" \\
2743↳↳↳↳ fix_structure_2A \\
2744↳↳↳ -end \\
2745↳↳ pdflatex fix_structure_2A.tex \\
2746↳ open fix_structure_2A.pdf \\
2747 \\
2748 fix_structure_2B:
fix_structure_2C:
$(ORBITER) -v 2 \n$define F -finite_field -q 4 -end \ndefine P -projective_space -n 3 -field F -v 0 -end \n-with P -do \n-projective_space_activity \n-define f3 -formula "del_Pezzo_F13a" "del_Pezzo_F13ab_report:
$(ORBITER) -v 3 \n$define F -finite_field -q 13 -end \ndefine P -projective_space -n 3 -field F -v 0 -end \ndefine f3 -formula "del_Pezzo_F13a" "del_Pezzo_F13ab_report:
"x*x*x+x*y*y+y*z*z+8*x*x+y+y+8*x*x+z*z+8*y*y+z*z" \
-define f4 -formula "del\_Pezzo\_F13b" "x,y,z" \
"x*x*x+x+y*y+y*z*z+z*x*x*y*y" \
-define del\_Pezzo13 -collection "f3,f4" \
-with P -do \ 
-projective\_space\_activity \ 
analyze\_del\_Pezzo\_surface del\_Pezzo13 "" \ 
-end 

del\_Pezzo\_F13b\_report.tex
open del\_Pezzo\_F13b\_report.pdf

del\_Pezzo\_F13a\_points.txt:
$(ORBITER) -v 3 \ 
-define F -finite\_field -q 13 -end \ 
-define P -projective\_space -n 3 -field F -v 0 -end \ 
-define f1 -formula "del\_Pezzo\_F9" \
"x*x*x+x+y*y+y*z*z+z" \
-define f2 -formula "del\_Pezzo\_F11" \
"x,y,z" \
"x*x*x+x+y*y+y*z*z+z+x*x*y+y+x*x*z*z+y*y*z*z" \
-define f3 -formula "del\_Pezzo\_F13a" \
"x,y,z" \
"x*x*x+x+y*y+y*z*z+z*8*x*x*y*y+8*x*x+z*8*y*y+z*z" \
-define f4 -formula "del\_Pezzo\_F13b" \
"x,y,z" \
"x*x*x+x+y*y+y+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 "" \ 
-end 

del\_Pezzo\_F13a\_report.tex 
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
SECTION GEOMETRIC OBJECTS:

```plaintext
SECTION GEOMETRIC OBJECTS:

elliptic_quadric_ovoid_q8:
```

```
#ovoid_q8.txt
# 65 points

ovoid_ST_q8:
```

```plaintext
# Section 4.10: Geometric Objects
```

```plaintext
#ovoid_q8.txt
# 65 points
```
define P -projective_space -n 3 -field F -v 0 -end 

define O -geometric_object P 

-ovoid_ST 

-with O -do -combinatorial_object_activity -save 

end 

#ovoid_ST_q8.txt

Baer_PG_2.4:  

define P -projective_space -n 2 -field F -v 0 -end 

define O -geometric_object P 

-Baer_substructure 

-with O -do -combinatorial_object_activity -save 

end

Baer_PG_3.4:  

define P -projective_space -n 3 -field F -v 0 -end 

define O -geometric_object P 

-Baer_substructure 

-with O -do -combinatorial_object_activity -save 

end

BLT_database_5.0:  

define P -projective_space -n 4 -field F -v 0 -end 

define O -geometric_object P 

-BLT_database 0 

-with O -do -combinatorial_object_activity -save 

end

# writes BLT_5.0.txt

BLT_database_7.0:  

define P -projective_space -n 4 -field F -v 0 -end 

$ORBITER) -v 2 

define F -finite_field -q 4 -end 

define F -finite_field -q 5 -end 

define F -finite_field -q 7 -end 

$ORBITER) -v 2 

define F -finite_field -q 4 -end 

define F -finite_field -q 5 -end 

define F -finite_field -q 7 -end 

$ORBITER) -v 2
-print_symbols
$\text{ORBITER} -v 2 \
union doily.csv doily_cliques.txt doily_cliques_union.csv

# 80 cliques

Doily_disjoint_sets_graph_cliques_5:
-echo $(DOILY) >doily.csv
$\text{ORBITER} -v 2 \
define G -graph -disjoint_sets_graph doily.csv
-end \
-with G -do \
-graph-theoretic_activity \
-find_cliques \n-target_size 5 \n-output_file doily_cliques_5 \
-end \
-end 

# 6 cliques

# doily_cliques_5.csv

PG_3_2_test:
-define F -finite_field -q 2 -end \
-define P -projective_space -n 3 -field F -v 0 -end \
-with P -do -projective_space_activity \
-cheat_sheet \
-pdflatex PG_3_2.tex
-open PG_3_2.pdf

Edge_curve_17:
-define F -finite_field -q 17 -end \
-define R -polynomial_ring -field F \
-number_of_variables 3 \
-homogeneous_of_degree 4 \
-end \
-define P -projective_space -n 2 -field F -v 0 -end \
-define C -geometric_object P \
-projective_variety R \
-"Edge_q17" "Edge\_q17" \

573
Edge curve q17_line_type:
> echo $(FILE_Q17) >edge_q17.csv
> $(ORBITER) -v 2 
> -define F -finite_field -q 17 -end 
> -define R -polynomial_ring -field F 
> -number_of_variables 3 
> -homogeneous_of_degree 4 
> -end 
> -define P -projective_space -n 2 -field F -v 0 -end 
> -define C -geometric_object P 
> -projective_variety R 
> "Edge\q17" "Edge\q17" 
> $(EDGE_CURVE_Q17_EQUATION) 
> -end 
> -with C -do 
> -combinatorial_object_activity 
> -line_type 
> -end 
> -print_symbols 

#( 4^6, 2^30, 1^132, 0^139 )

Edge curve q23_line_type:
> $(ORBITER) -v 2 
> -define F -finite_field -q 23 -end 
> -define P -projective_space -n 2 -field F -v 0 -end 
> -define E -geometric_object P 
> -set $(EDGE_CURVE_Q23_AS_POINTS) 
> -end 
> -with E -do 
> -combinatorial_object_activity 

CHAPTER 5 - GROUP THEORY

SECTION 5.1: PERMUTATION GROUPS:

Cyclic_6:
```bash
$ (ORBITER) -v 3 \
```
```bash
define G permutation_group cyclic_group 6 -end \
```
```bash
with G -do \
```
```bash
-group_theoretic_activity \
```
```bash
-report \
```
```bash
-end \
```
```bash
pdflatex Perm6_report.tex \
```
```bash
open Perm6_report.pdf \
```
```bash
```
```
Cyclic_6_group_table:
```bash
$ (ORBITER) -v 3 \
```
```bash
define G permutation_group cyclic_group 6 -end \
```
```bash
with G -do \
```
```bash
-group_theoretic_activity \
```
```bash
-export_group_table \
```
3134 ▷ ▷ -end
3135 ▷ $(ORBITER) -v 2 \n3136 ▷ ▷ -define all_one_r -vector -repeat 1 6 -end \n3137 ▷ ▷ -define all_one_c -vector -repeat 1 6 -end \n3138 ▷ ▷ -draw_matrix \n3139 ▷ ▷ ▷ -input_csv_file Perm6_group_table.csv \n3140 ▷ ▷ ▷ -box_width 50 -bit_depth 24 \n3141 ▷ ▷ ▷ -partition 3 all_one_r all_one_c \n3142 ▷ ▷ -end
3143 ▷ open Perm6_group_table_draw.bmp
3144
3145
3146 Symmetric_3:
3147 ▷ $(ORBITER) -v 3 \n3148 ▷ ▷ -define G -permutation_group -symmetric_group 3 -end \n3149 ▷ ▷ -with G -do \n3150 ▷ ▷ -group_theoretic_activity \n3151 ▷ ▷ ▷ -report \n3152 ▷ ▷ -end
3153 ▷ pdflatex Perm3_report.tex
3154 ▷ open Perm3_report.pdf
3155
3156
3157 Symmetric_3_group_table:
3158 ▷ $(ORBITER) -v 3 \n3159 ▷ ▷ -define G -permutation_group -symmetric_group 3 -end \n3160 ▷ ▷ -with G -do \n3161 ▷ ▷ -group_theoretic_activity \n3162 ▷ ▷ ▷ -export_group_table \n3163 ▷ ▷ -end
3164 ▷ $(ORBITER) -v 2 \n3165 ▷ ▷ -define all_one_r -vector -repeat 1 6 -end \n3166 ▷ ▷ -define all_one_c -vector -repeat 1 6 -end \n3167 ▷ ▷ -draw_matrix \n3168 ▷ ▷ ▷ -input_csv_file Perm3_group_table.csv \n3169 ▷ ▷ ▷ -box_width 50 -bit_depth 24 \n3170 ▷ ▷ ▷ -partition 3 all_one_r all_one_c \n3171 ▷ ▷ -end
3172 ▷ open Perm3_group_table_draw.bmp
3173
3174 Symmetric_3_elements:
3175 ▷ $(ORBITER) -v 3 \n3176 ▷ ▷ -define G -permutation_group -symmetric_group 3 -end \n3177 ▷ ▷ -with G -do \n3178 ▷ ▷ -group_theoretic_activity \n3179 ▷ ▷ ▷ -save_elements_csv "Symmetric3_elts.csv" \n3180 ▷ ▷ -end

576
3181 $(ORBITER) -v 2 \
3182 -define Sym3_elts -vector -load_csv_data_column \
3183 Symmetric3_elts.csv 1 -end \
3184 -save_matrix_csv Sym3_elts 
3185 $(ORBITER) -v 2 \
3186 -define all_one_r -vector -repeat 1 6 -end \
3187 -define all_one_c -vector -repeat 1 3 -end \
3188 -draw_matrix \
3189 -input_csv_file Sym3_elts_matrix.csv \
3190 -box_width 50 -bit_depth 8 \
3191 -partition 3 \
3192 -all_one_r all_one_c \
3193 -end 
3194 open Sym3_elts_matrix.draw.bmp 
3195 Symmetric_3_long:
3196 $(ORBITER) -v 3 \
3197 -define G -permutation_group -symmetric_group 3 -end \
3198 -with G -do \
3199 -group_theoretic_activity \
3200 -export_orbiter \
3201 -end \
3202 -with G -do \
3203 -group_theoretic_activity \
3204 -export_orbiter \
3205 -print_elements.tex \
3206 -end \
3207 -with G -do \
3208 -group_theoretic_activity \
3209 -report \
3210 -end \
3211 -with G -do \
3212 -group_theoretic_activity \
3213 -save_elements_csv "Symmetric3_elts.csv" \
3214 -end
3215 $(ORBITER) -v 3 \
3216 -draw_options \
3217 -nodes \
3218 -embedded -radius 250 \
3219 -xin 10000 -yin 10000 \
3220 -xout 1000000 -yout 600000 \
3221 -scale 0.3 -line_width 1.0 \
3222 -end \
3223 -tree_draw -file Perm3_elements_tree.txt -end 
3224 $(ORBITER) -v 2 \
3225 -define M -vector -load_csv_data_column \
3226 Symmetric3_elts.csv 1 -end \
3227 -save_matrix_csv M
$\text{(ORBITER)} -v 2 \$

$\text{-define all_one_r -vector -repeat 1 6 -end} \$

$\text{-define all_one_c -vector -repeat 1 3 -end} \$

$\text{-draw_matrix} \$

$\text{-input_csv_file M\_matrix.csv} \$

$\text{-box_width 50 -bit_depth 8} \$

$\text{-partition 3} \$

$\text{all_one_r all_one_c} \$

$\text{-end} \$

$\text{pdflatex Perm3\_elements\_tree\_draw.tex} \$

$\text{open Perm3\_elements\_tree\_draw.pdf} \$

$\text{#pdflatex Perm3\_report.tex} \$

$\text{#open Perm3\_report.pdf} \$

$\text{Symmetric\_4:} \$

$\text{-define G -permutation\_group -symmetric\_group 4 -end} \$

$\text{-with G -do} \$

$\text{-group\_theoretic\_activity} \$

$\text{-report} \$

$\text{-end} \$

$\text{pdflatex Perm4\_report.tex} \$

$\text{open Perm4\_report.pdf} \$

$\text{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} \$

$\text{$(ORBITER)$ -v 2} \$

$\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} \$

$\text{Symmetric\_4\_long:} \$

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

578
-define G -permutation_group -symmetric_group 4 -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 \
-print_elements.tex \
-end \
-with G -do \
-group_theoretic_activity \
-save_elements_csv "Symmetric4_elts.csv" \
-end \
-with G -do \
-group_theoretic_activity \
-export_inversion_graphs "Symmetric4_inversion_graphs.csv" \
-end \
$(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 \ 

# Todo:

Symmetric_4stab:

Symmetric_4stab:

Symmetric_4stab:

Symmetric_4stab:
PGL_3_2:
> $(ORBITER) -v 2 \n> -define F -finite_field -q 2 -end \n> -define G -linear_group -PGL 3 F -end \n> -with G -do \n> -group_theoretic_activity \n> -report \n> -end
>
> pdflatex PGL_3_2.report.tex
> open PGL_3_2_report.pdf

PGL_4_2:
> $(ORBITER) -v 2 \n> -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
>
> pdflatex PGL_4_2_report.tex
> open PGL_4_2_report.pdf

PGL_4_2_export:
> $(ORBITER) -v 2 \n> -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

# created by PGL_4_2_export

PGL_4_2_generated:
> $(ORBITER) -v 2 \n> -define gens -vector -file PGL_4_2_gens.csv -end \n
581
-define G -permutation_group \ 
-bsgs PGL_4_2 "\{rm PGL\}(4,2)" 15 20160 "0,1,2,3" 6 gens -end \ 

L_5_3: 
$(ORBITER) -v 2 \ 
-define F -finite_field -q 3 -end \ 
-define G -linear_group -PSL 5 F -end \ 
-with G -do \ 
-group_theoretic_activity \ 
-report \ 
-end \ 
pdflatex PSL_5_3.report.tex \ 
open PSL_5_3_report.pdf \ 

PSL(5,3): Order 237783237120 = 121 * 120 * 117 * 108 * 81 * 16 \ 

L_4_5: 
$(ORBITER) -v 2 \ 
-define F -finite_field -q 5 -end \ 
-define G -linear_group -PSL 4 F -end \ 
-with G -do \ 
-group_theoretic_activity \ 
-report \ 
-end \ 
pdflatex PSL_4_5_report.tex \ 
open PSL_4_5_report.pdf \ 

PSL(4,5): Order 7254000000 \ 

PGL_4_5: 
$(ORBITER) -v 2 \ 
-define F -finite_field -q 5 -end \ 
-define G -linear_group -PGL 4 F -end \ 
-with G -do \ 
-group_theoretic_activity \ 
-report \ 

3463 ▷▷ -end
3464 ▷ pdflatex PGL_4.5_report.tex
3465 ▷ open PGL_4.5_report.pdf
3466
3467 PGGL_3.4:
3468 ▷ $(ORBITER) -v 2 \ 
3469 ▷ ▷ -define G -linear_group -PGGL 3 4 -end \ 
3470 ▷ ▷ -with G -do \ 
3471 ▷ ▷ -group_theoretic_activity \ 
3472 ▷ ▷ ▷ -report \ 
3473 ▷ ▷ ▷ -sylow \ 
3474 ▷ ▷ ▷ -classes \ 
3475 ▷ ▷ -end
3476 ▷ pdflatex PGGL_3.4_report.tex
3477 ▷ open PGGL_3.4_report.pdf
3478
3479
3480
3481 PGGL_3.8:
3482 ▷ $(ORBITER) -v 2 \ 
3483 ▷ ▷ -define G -linear_group -PGGL 3 8 -end
3484
3485
3486 PGGL_3.8_report:
3487 ▷ $(ORBITER) -v 3 \ 
3488 ▷ ▷ -define G -linear_group -PGGL 3 8 -end \ 
3489 ▷ ▷ -with G -do \ 
3490 ▷ ▷ -group_theoretic_activity \ 
3491 ▷ ▷ ▷ -report \ 
3492 ▷ ▷ ▷ -end
3493 ▷ pdflatex PGGL_3.8_report.tex
3494 ▷ open PGGL_3.8_report.pdf
3495
3496
3497 AGL_1.27:
3498 ▷ $(ORBITER) -v 2 \ 
3499 ▷ ▷ -define F -finite_field -q 27 -end \ 
3500 ▷ ▷ -define G -linear_group -AGL 1 F -end \ 
3501 ▷ ▷ -with G -do \ 
3502 ▷ ▷ -group_theoretic_activity \ 
3503 ▷ ▷ ▷ -report \ 
3504 ▷ ▷ ▷ -end
3505 ▷ pdflatex AGL_1.27_report.tex
3506 ▷ open AGL_1.27_report.pdf
3507
3508 #▷ ▷ -group_table \ 
3509

583
3510
3511 SP_4_2:
3512 ▶ $(ORBITER) -v 2 \n3513 ▶ ▶ -define F -finite_field -q 2 -end \n3514 ▶ ▶ -define G -linear_group -GL 4 F \n3515 ▶ ▶ ▶ -symplectic_group \n3516 ▶ ▶ ▶ -end \n3517 ▶ ▶ -with G -do \n3518 ▶ ▶ ▶ -group_theoretic_activity \n3519 ▶ ▶ ▶ ▶ -report \n3520 ▶ ▶ ▶ -end
3521 ▶ pdflatex GL_4_2_Sp_4_2_report.tex
3522 ▶ open GL_4_2_Sp_4_2_report.pdf
3523
3524 # order 720
3525
3526
3527 PSP_4_4:
3528 ▶ $(ORBITER) -v 2 \n3529 ▶ ▶ -define F -finite_field -q 4 -end \n3530 ▶ ▶ -define G -linear_group -PGL 4 F \n3531 ▶ ▶ ▶ -symplectic_group \n3532 ▶ ▶ ▶ -end \n3533 ▶ ▶ -with G -do \n3534 ▶ ▶ ▶ -group_theoretic_activity \n3535 ▶ ▶ ▶ ▶ -report \n3536 ▶ ▶ ▶ -end
3537 ▶ pdflatex PGL_4_4_Sp_4_4_report.tex
3538 ▶ open PGL_4_4_Sp_4_4_report.pdf
3539
3540 #order 979200
3541
3542
3543
3544 PGO_5_2:
3545 ▶ $(ORBITER) -v 2 \n3546 ▶ ▶ -define F -finite_field -q 2 -end \n3547 ▶ ▶ -define G -linear_group -PGO 5 F -end \n3548 ▶ ▶ -with G -do \n3549 ▶ ▶ -group_theoretic_activity \n3550 ▶ ▶ ▶ -report \n3551 ▶ ▶ ▶ -end
3552 ▶ pdflatex PGO_5_2_report.tex
3553 ▶ open PGO_5_2_report.pdf
3554
3555 PGGO_5_4:
3556 ▶ $(ORBITER) -v 2 \n
584
3557 $define F4 -finite_field -q 4 -end \n3558 $define G -linear_group -PGGO 5 F4 -end \n3559 $with G -do \n3560 $group_theoretic_activity \n3561 $report \n3562 $end
3563 pdflatex PGGO_5_4_report.tex
3564 open PGGO_5_4_report.pdf
3565
3566
3567
3568 PGOp_6_2:
3569 $(ORBITER) -v 2 \n3570 $define F -finite_field -q 2 -end \n3571 $define G -linear_group -PGOp 6 F -end \n3572 $with G -do \n3573 $group_theoretic_activity \n3574 $report \n3575 $end
3576 pdflatex PGOp_6_2_report.tex
3577 open PGOp_6_2_report.pdf
3578
3579 PGOm_6_2:
3580 $(ORBITER) -v 2 \n3581 $define F -finite_field -q 2 -end \n3582 $define G -linear_group -PGOm 6 F -end \n3583 $with G -do \n3584 $group_theoretic_activity \n3585 $report \n3586 $end
3587 pdflatex PGOm_6_2_report.tex
3588 open PGOm_6_2_report.pdf
3589
3590
3591
3592
3593
3594 # the following two groups are isomorphic:
3595
3596 PSP_6_2:
3597 $(ORBITER) -v 2 \n3598 $define F -finite_field -q 2 -end \n3599 $define G -linear_group -PGL 6 F \n3600 $symplectic_group \n3601 $end \n3602 $with G -do \n3603 $group_theoretic_activity \n3604
3585
# group order 1451520, acting on 63 points

PGO\_7\_2:

$(ORBITER) -v 2 \
-define F -finite_field -q 2 -end \
-define G -linear_group -PGO 7 F -end \
-with G -do \
-group.theoretic.activity \
-report \
-end \

pdflatex PGO\_7\_2\_report.tex \
open PGO\_7\_2\_report.pdf

---

# 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 \

C13\_as\_subgroup:
\拎\拎\拎 $\$(\texttt{ORBITER}) -v 2 \拎\拎\拎 
\拎\拎\拎 -define G -permutation\_group -symmetric\_group 13 \拎\拎\拎 
\拎\拎\拎 -subgroup\_by\_generators C13 13 1 $\$(\texttt{GEN\_C13}) -end \拎\拎\拎 
\拎\拎\拎 -with G -do \拎\拎\拎 
\拎\拎\拎 -with G -do \拎\拎\拎 
\拎\拎\拎 -group\_theoretic\_activity \拎\拎\拎 
\拎\拎\拎 -report \拎\拎\拎 
\拎\拎\拎 -end \拎\拎\拎 
\拎\拎\拎 -with G -do \拎\拎\拎 
\拎\拎\拎 -group\_theoretic\_activity \拎\拎\拎 
\拎\拎\拎 -save\_elements\_csv "C13\_elts.csv" \拎\拎\拎 
\拎\拎\拎 -end
\拎\拎\拎 #pdflatex Perm13\_Subgroup\_C13\_13\_report.tex
\拎\拎\拎 #open Perm13\_Subgroup\_C13\_13\_report.pdf

J1:
\拎\拎\拎 $\$(\texttt{ORBITER}) -v 2 \拎\拎\拎 
\拎\拎\拎 -define G -linear\_group -PGL 7 11 -Janko1 -end \拎\拎\拎
-with G -do \
-group_theoretic_activity \
-report \ 
-end

pdflatex PGL_7_11_Subgroup_Janko1_report.tex
open PGL_7_11_Subgroup_Janko1_report.pdf

PGL_3_11_singer:
$(ORBITER) -v 2 
-define G -linear_group -PGL 3 11 -singer 19 -end 
-with G -do 
-group_theoretic_activity 
-report \ 
-end

pdflatex PGL_3_11_Singer_3_11_19_report.tex
open PGL_3_11_Singer_3_11_19_report.pdf

PGL_3_11_singer_and_frobenius:
$(ORBITER) -v 2 
-define G -linear_group -PGL 3 11 -singer_and_frobenius 19 -end 
-with G -do 
-group_theoretic_activity 
-report \ 
-end

pdflatex PGL_3_11_Singer_and_Frob3_11_19_report.tex
open PGL_3_11_Singer_and_Frob3_11_19_report.pdf

PG_2_4_order_21:
$(ORBITER) -v 2 
-define G -linear_group -PGL 3 4 -end 
-with G -do 
-group_theoretic_activity 
-search_element_of_order 21 \ 
-end

quaternion:
$(ORBITER) -v 2 
-define G -linear_group -SL 2 3 \n-subgroup_by_generators "quaternion" "8" 3 \n "1,1,1,2, 2,1,1,1, 0,2,1,0" \n-end \n-with G -do 
-group_theoretic_activity 

588
cube_group:
3755 \$\text{(ORBITER)} \ -v \ 2 \$
3756 \ -define \ gens \ -vector \ -dense \\
3757 \ -define \ G \ -linear_group \ -GL \ 3 \ 3 \\
3758 \ -subgroup \ by \ generators \ "cube" \ "48" \ 3 \\
3759 \ gens \\
3760 \ -end \\
3761 \ -with \ G \ -do \\
3762 \ -group_theoretic_activity \\
3763 \ -print_elements.tex \\
3764 \ -report \\
3765 \ -end \\
3766 \ -do \\
3767 \ -with \ G \ -do \\
3768 \ -group_theoretic_activity \\
3769 \ -print_elements.tex \\
3770 \ -report \\
3771 \ -end \\
3772 \ pdflatex \ GL_2.3\_Subgroup\_quaternion\_8\_elements.tex \\
3773 \ open \ GL_2.3\_Subgroup\_quaternion\_8\_elements.pdf \\
3774 \ pdflatex \ GL_2.3\_Subgroup\_quaternion\_8\_report.tex \\
3775 \ open \ GL_2.3\_Subgroup\_quaternion\_8\_report.pdf \\
3776 \\
3777 tetra_group:
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

#Hesse group:
#1,0,0,0,0,1,0,0,0,0,1,0,3,2,3,2,0,
#1,0,0,0,0,1,0,0,3,1,2,0,1,0,1,3,0,
#1,0,0,0,0,1,0,0,0,1,1,0,0,0,0,1,1,
#1,0,0,0,2,2,0,0,2,0,0,0,0,1,0,
#1,0,0,0,2,3,1,0,2,0,1,0,3,1,3,1,0,
#0,1,1,0,2,0,0,0,1,3,2,0,2,1,1,2,1,
#1,1,0,0,0,0,3,0,3,2,2,0,1,2,3,3,1,

Weyl E8:
$(ORBITER) -v 3$
$-define gens -vector -dense$
$\$(GENERATORS_WEYL_GROUP_E8)\$
$-end$
$-define G -linear_group -GL 8 3$
$-subgroup_by_generators$
$\"Weyl_8E\" \"696729600\" 2 gens$
$-end$
$-with G -do$
$-group_theoretic_activity$
$-report$
$-end$
pdflatex GL_8_3_Subgroup_Weyl_E8_696729600_report.tex
open GL_8_3_Subgroup_Weyl_E8_696729600_report.pdf

# group generators from http://www.math.rwth-aachen.de/~Gabriele.Nebe/LATTICES/E8
test_subgroup:
$\text{(ORBITER)} -v 2$
-define F -finite_field -q 2 -end
-define G1 -linear_group -PGOp 6 F -end
-define G2 -linear_group -PGL 6 F
-define symplectic_group
-define G -linear_group -PGL 6 F
-define CR -vector_ge -action G2
-define CR -read_csv
 PGOp_6_2.coset_reps.csv Element
-define CR -end
SP_6_2.point_stab_subgroup:
\( -\text{symplectic group} \)
\( -\)end
\( -\text{define G0 -modified group -from G} \)
\( -\text{point stabilizer 0} \)
\( -\)end
\( -\text{with G0 -do} \)
\( -\text{group theoretic activity} \)
\( -\)report
\( -\)end
\( \text{pdflatex PGL.6_2_report.tex} \)
\( \text{open PGL.6_2_report.pdf} \)

# group of order 23040

\( -\text{PGOp.6_2_report:} \)
\( -\)$(\text{ORBITER}) -v 2 \)
\( -\text{define F -finite field -q 2 -end} \)
\( -\text{define G -linear group -PGOp 6 F -end} \)
\( -\text{with G -do} \)
\( -\text{group theoretic activity} \)
\( -\)report
\( -\)end
\( \text{pdflatex PGOp.6_2_report.tex} \)
\( \text{open PGOp.6_2_report.pdf} \)

# group order 40320

\( -\text{PGOp.6_2_point_stab_subgroup:} \)
\( -\)$(\text{ORBITER}) -v 2 \)
\( -\text{define F -finite field -q 2 -end} \)
\( -\text{define G -linear group -PGOp 6 F -end} \)
\( -\text{define G0 -modified group -from G} \)
\( -\text{point stabilizer 0} \)
\( -\)end
\( -\text{with G0 -do} \)
\( -\text{group theoretic activity} \)
\( -\)report
\( -\)end
\( \text{pdflatex PGOp.6_2_report.tex} \)
\( \text{open PGOp.6_2_report.pdf} \)

# group of order 1152
3932 PGOp.6.2_GENS="\n3933 1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,1,0,\n3934 1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,1,1,1,0,0,1,0,0,1,0,0,1,0,1,0,\n3935 1,0,0,0,0,0,1,0,0,0,0,0,0,1,1,0,0,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,1,0,\n3936 1,0,0,0,0,0,1,0,0,0,0,0,0,1,1,0,0,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,\n3937 1,0,0,0,0,0,1,0,0,0,0,0,0,1,1,0,0,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,1,\n3938 1,0,0,0,0,0,1,1,1,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,1,0,0,1,0,0,\n3939 1,0,0,0,0,0,1,0,0,1,0,0,1,0,1,0,0,1,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,\n3940 0,1,0,1,1,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,1,0,0,0,0,0,1,0,0,1,1,0,0,1,\n3941 0,0,1,1,1,1,1,0,1,0,0,0,0,1,1,1,1,1,1,1,1,0,0,0,1,0,1,0,1,0,1,0,1,0,1,\n3942 0,0,0,0,1,0,1,0,1,1,1,1,1,0,1,0,1,0,1,0,0,0,0,0,0,1,1,1,1,1,0,"\n3943
3944 PGOp.6.2_linear:
3945 $\$(ORBITER) -v 2 \n3946 -$define F -finite_field -q 2 -end \n3947 -$define G -linear_group -PGL 6 F \n3948 -$define G6 -modified_group -from G \n3949 -$define G6 -point_stabilizer 6 \n3950 -$end \n3951 -$with G -do \n3952 -$group_theoretic_activity \n3953 -$report \n3954 -$end \n3955 pdflatex PGL.6.2_Subgroup_PGOp.6.2.40320_report.tex \n3956 open PGL.6.2_Subgroup_PGOp.6.2.40320_report.pdf \n3957 PGL.6.2_report.tex \n3958 PGL.6.2_report.pdf \n3959
3960 PGOp.6.2_linear_stab_6:
3961 $\$(ORBITER) -v 2 \n3962 -$define F -finite_field -q 2 -end \n3963 -$define G -linear_group -PGL 6 F \n3964 -$define G6 -modified_group -from G \n3965 -$define G6 -point_stabilizer 6 \n3966 -$end \n3967 -$with G6 -do \n3968 -$group_theoretic_activity \n3969 -$report \n3970 -$end \n3971 pdflatex PGL.6.2_report.tex \n3972 open PGL.6.2_report.pdf \n3973 # group of order 1440, of index 28 in PGOp(6,2)
Section 5.4: Linear Groups, Advanced Topics

U₃₃:

```latex
\begin{verbatim}
\$\text{ORBITER} -v 3 \ \\
\text{define } F \text{ finite field } -q 9 \text{ override polynomial } "17" \text{ end } \ \\
\text{define } G \text{ linear group } -PGL 3 F \ \\
\text{subgroup by generators } "U₃₃" "6048" 2 \ \\
\text{end} \ \\
\text{with } G \text{ do } \ \\
\text{group theoretic activity} \ \\
\text{report} \ \\
\text{end}
\end{verbatim}
```

Co₃ from Conway et al., 1985 (ATLAS)
#order = 495766656000

#Co3 from the paper by Suleiman and Wilson 1997

Co3:

⊿ $(\text{ORBITER}) -v 2 \$

⊿ ⊿ -define F -finite_field -q 2 -end \n
⊿ ⊿ -define g1 -vector -field F -format 22 -compact $(\text{CONWAY\_GEN1}) -end \n
⊿ ⊿ -define g2 -vector -field F -format 22 -compact $(\text{CONWAY\_GEN2}) -end \n
⊿ ⊿ -define gens -vector -concatenate g1 -concatenate g2 -end \n
⊿ ⊿ -define G -linear_group -PGL 22 2 \n
⊿ ⊿ ⊿ -subgroup_by_generators "Co3" "495766656000" 2 gens \n
⊿ ⊿ ⊿ -with G -do \n
⊿ ⊿ ⊿ -group_theoretic_activity \n
⊿ ⊿ ⊿ ⊿ -report \n
⊿ ⊿ ⊿ -end

pdf\latex PGL\_22\_2\_Subgroup\_Co3\_495766656000\_report.tex

open PGL\_22\_2\_Subgroup\_Co3\_495766656000\_report.pdf

# needs a lot of memory to run!

Ree\_27:

⊿ $(\text{ORBITER}) -v 2 \$

⊿ ⊿ -define F -finite_field -q 27 -override_polynomial "34" -end \n
⊿ ⊿ -define g1 -vector -field F -format 7 -dense $(\text{Ree\_gen1}) -end \n
⊿ ⊿ -define g2 -vector -field F -format 7 -dense $(\text{Ree\_gen2}) -end \n
⊿ ⊿ -define gens -vector -concatenate g1 -concatenate g2 -end \n
⊿ ⊿ -define G -linear_group -PGL 7 F \n
⊿ ⊿ ⊿ -subgroup_by_generators "Ree\_27" "10073444472" 2 gens \n
⊿ ⊿ ⊿ -with G -do \n
⊿ ⊿ ⊿ -group_theoretic_activity \n
⊿ ⊿ ⊿ ⊿ -report \n
⊿ ⊿ ⊿ -end

# needs a lot of memory to run!

## Section 5.5: Induced Actions

SECTION_INDUCED_ACTIONS:

Symmetric_4_on_pairs:

⊿ $(\text{ORBITE}) -v 3 \$

# Section 5.5: Induced Actions
\begin{verbatim}
4073  ▶  ▶  -define G -permutation_group -symmetric_group 4 -end \ 4074  ▶  ▶  -define G_on_2subsets -modified_group -from G \ 4075  ▶  ▶  ▶  -on_k_subsets 2 \ 4076  ▶  ▶  ▶  -end \ 4077  ▶  ▶  ▶  -with G_on_2subsets -do \ 4078  ▶  ▶  ▶  -group_theoretic_activity \ 4079  ▶  ▶  ▶  ▶  -report \ 4080  ▶  ▶  ▶  -end \ 4081  ▶  pdflatex Perm4_on_2_subsets_report.tex 4082  ▶  open Perm4_on_2_subsets_report.pdf 4083  ▶ 4084 4085  T3_on_tensors: 4086  ▶  ▶  $(ORBITER) -v 2 \ 4087  ▶  ▶  ▶  -define G \ 4088  ▶  ▶  ▶  ▶  -linear_group -GL_d_q_wr_Sym_n 2 2 3 \ 4089  ▶  ▶  ▶  ▶  ▶  -on_tensors -end \ 4090  ▶  ▶  ▶  ▶  -with G -do \ 4091  ▶  ▶  ▶  ▶  -group_theoretic_activity \ 4092  ▶  ▶  ▶  ▶  ▶  -report \ 4093  ▶  ▶  ▶  ▶  -end \ 4094  ▶  pdflatex GL_2_2_wreath_Sym3_report.tex 4095  ▶  open GL_2_2_wreath_Sym3_report.pdf 4096  ▶ 4097 4098 4099  T3r1: 4100  ▶  ▶  $(ORBITER) -v 4 \ 4101  ▶  ▶  ▶  -define G \ 4102  ▶  ▶  ▶  ▶  -linear_group -GL_d_q_wr_Sym_n 2 2 3 \ 4103  ▶  ▶  ▶  ▶  ▶  -on_tensors -end \ 4104  ▶  ▶  ▶  ▶  ▶  -with G -do \ 4105  ▶  ▶  ▶  ▶  ▶  -group_theoretic_activity \ 4106  ▶  ▶  ▶  ▶  ▶  ▶  -report \ 4107  ▶  ▶  ▶  ▶  ▶  -end \ 4108  ▶  pdflatex GL_2_2_wreath_Sym3_report.tex 4109  ▶  open GL_2_2_wreath_Sym3_report.pdf 4110 4111 4112 4113 4114 4115  T4_on_tensors: 4116  ▶  ▶  $(ORBITER) -v 4 \ 4117  ▶  ▶  ▶  -define G \ 4118  ▶  ▶  ▶  ▶  -linear_group -GL_d_q_wr_Sym_n 2 2 4 \ 4119  ▶  ▶  ▶  ▶  ▶  -on_tensors -end \end{verbatim}
T4r1:

\$(\texttt{ORBITER}) -v 4 \$

-define G \\
-define linear_group -GL_d_q_wr_Sym_n 2 2 4 \\
-on rank one tensors -end \\
-with G -do \\
-group theoretic activity \\
-report \\
-end

pdflatex GL_2.2_wreath_Sym4_report.tex
open GL_2.2_wreath_Sym4_report.pdf

PGGL_2.8_on_conic:

\$(\texttt{ORBITER}) -v 4 \$

-define G \\
-define linear_group -PGGL 2 8 -PGL2OnConic -end \\
-with G -do \\
-group theoretic activity \\
-report \\
-end

pdflatex PGGL_2.2_wreath_Sym4_report.tex
open PGGL_2.2_wreath_Sym4_report.pdf

surface_q13_Eckardt:

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

-define F -finite_field -q 13 -end \\
-define P -projective_space -n 3 -field F -v 0 -end \\
-define S -cubic_surface -space P -arc_lifting "0,1,2,3,43,113" -end \\
-with S -do \\
-cubic_surface_activity \\
-report_with_group \\
-end \\
-with S -do \\
-cubic_surface_activity \\

597
4167  ▷ ▷ ▷ -export.tritangent.planes \\
4168  ▷ ▷ -end \\
4169  ▷ pdflatex surface_arc_lifting.trihedral_q13_arc0_1_2_3_43_113_with_group.tex \\
4170  ▷ open surface_arc_lifting.trihedral_q13_arc0_1_2_3_43_113_with_group.pdf \\
4171 \\
4172 \\
4173  SURFACE_Q13_STAB_GENS="1,0,0,0,9,12,0,0,10,0,12,0,9,0,0,12, \\
4174  1,0,0,0,0,12,12,6,6,0,0,7,1,0,2,0, \\
4175  0,1,1,7,3,9,9,11,2,10,10,3,9,9,1,11"
4176 \\
4177  surface_q13.Eckardt_on_tritangent.planes:
4178  ▷ $(ORBITER) -v 2 \\
4179  ▷ ▷ -orbiter_path $(ORBITER_PATH) \\
4180  ▷ ▷ -draw_options -embedded -end \\
4181  ▷ ▷ -define F -finite_field -q 13 -end \\
4182  ▷ ▷ -define gens -vector -field F -dense $(SURFACE_Q13_STAB_GENS) -end \\
4183  ▷ ▷ -define TriP -set -file \\
4184  ▷ ▷ ▷ -export.tritangent.planes.csv \\
4185  ▷ ▷ ▷ -end \\
4186  ▷ ▷ ▷ -define G -linear_group -PGL 4 F \\
4187  ▷ ▷ ▷ -subgroup_by_generators "stab" \\
4188  ▷ ▷ ▷ -define G_on_planes -modified_group -from G \\
4189  ▷ ▷ ▷ ▷ -on_k_subspaces 3 \\
4190  ▷ ▷ ▷ ▷ -end \\
4191  ▷ ▷ ▷ ▷ -define G_on_planes -modified_group -from G_on_planes \\
4192  ▷ ▷ ▷ ▷ ▷ -restricted_action TriP \\
4193  ▷ ▷ ▷ ▷ ▷ -end \\
4194  ▷ ▷ ▷ ▷ ▷ -with Gr -do \\
4195  ▷ ▷ ▷ ▷ ▷ -group_theoretic_activity \\
4196  ▷ ▷ ▷ ▷ ▷ ▷ -report \\
4197  ▷ ▷ ▷ ▷ ▷ ▷ -end \\
4198  ▷ ▷ ▷ ▷ ▷ ▷ -define Orb -orbits -group Gr \\
4199  ▷ ▷ ▷ ▷ ▷ ▷ ▷ -on_points \\
4200  ▷ ▷ ▷ ▷ ▷ ▷ ▷ -end \\
4201  ▷ ▷ ▷ ▷ ▷ ▷ ▷ -with Orb -do -orbits_activity \\
4202  ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ -report \\
4203  ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ -end \\
4204  ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ -with Orb -do -orbits_activity \\
4205  ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ -draw_tree 0 \\
4206  ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ -end \\
4207  ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ -with Orb -do -orbits_activity \\
4208  ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ -draw_tree 1 \\
4209  ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ -end \\
4210  ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ -with Orb -do -orbits_activity \\
4211  ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ -draw_tree 1 \\
4212  ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ -end \\
4213  ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ ▷ -with Orb -do -orbits_activity \\
598
PGL\_4.2\_wd:

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

\$\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} \$

PGL\_4.2\_wd\_reverse:

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

\$\text{-linear\_group -PGL 4 2 -wedge\_detached -end} \$

\$\text{-group\_theoretic\_activity} \$

\$\text{-reverse\_isomorphism\_exterior\_square} \$

\$\text{-end} \$

#Section 5.6: Group Theoretic Activities

SECTION\_GROUP\_THEORETIC\_ACTIVITIES:
PGL\_3\_2\_elements:

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

\$-\text{define G -linear\_group -PGL 3 2 -end} \$

\$-\text{with G -do} \$

\$-\text{group\_theoretic\_activity} \$

\$-\text{save\_elements\_csv \"PGL\_3\_2\_elements.csv\"} \$

\$-\text{end} \$

# creates PGL\_3\_2\_elements.csv

Sym\_3\_elements:

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

\$-\text{define G -permutation\_group -symmetric\_group 3 -end} \$

\$-\text{with G -do} \$

\$-\text{group\_theoretic\_activity} \$

\$-\text{print\_elements\_tex} \$

\$-\text{end} \$

\$-\text{draw\_options} \$

\$-\text{nodes} \$

\$-\text{embedded -radius 250} \$

\$-\text{xin 10000 -yin 10000} \$

\$-\text{xout 1000000 -yout 600000} \$

\$-\text{scale 0.3 -line\_width 1.0} \$

\$-\text{tree\_draw -file Perm3\_elements\_tree.tex -end} \$

\$-\text{pdflatex Perm3\_elements\_tree\_draw.tex} \$

\$-\text{open Perm3\_elements\_tree\_draw.pdf} \$

Cycle\_13\_power:

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

\$-\text{define G -permutation\_group -symmetric\_group 13 -end} \$

\$-\text{with G -do} \$

\$-\text{group\_theoretic\_activity} \$

\$-\text{-consecutive\_powers} \$

\$-\text{"1,2,3,4,5,6,7,8,9,10,11,12,0\" 13} \$

\$-\text{end} \$

\$-\text{pdflatex Perm13\_all\_powers.tex} \$

\$-\text{open Perm13\_all\_powers.pdf} \$

Cycle\_12\_power:

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

600
define G -permutation_group -symmetric_group 12 -end \
with G -do \
-group_theoretic_activity \
-consecutive_powers \
"1,2,3,4,5,6,7,8,9,10,11,0" 12 \
-end 

pdflatex Perm12_all_powers.tex 
open Perm12_all_powers.pdf 

PGL_3_4_singer: 
$\text{(ORBITER)} -v 5 \$

define G -linear_group -PGL 3 4 -end \
with G -do \
-group_theoretic_activity \
-find_singer_cycle \ 
-end 

GL_2_8_multiply: 
$\text{(ORBITER)} -v 5 \$

define G -linear_group -GL 2 8 -end \
with G -do \
-group_theoretic_activity \
multiply "0,1,2,3" "4,5,6,7" \
-end 

pdflatex GL_2_8_mult.tex 
open GL_2_8_mult.pdf 

GL_2_7_multiply: 
$\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 

pdflatex GL_2_7_mult.tex 
open GL_2_7_mult.pdf 

GL_2_7_inv: 
$\text{(ORBITER)} -v 5 \$

define G -linear_group -GL 2 7 -end \
with G -do \

GL_2.7_power:
$\text{GL}_2 7 \text{.inv.tex}$

$\text{open GL}_2 7 \text{.inv.pdf}$

GL_2.7_power:
$\text{GL}_2 7 \text{.power.tex}$

$\text{open GL}_2 7 \text{.power.pdf}$

PGL_3.2_classes:
$\text{PGL}_3 2 \text{.classes.tex}$

$\text{open PGL}_3 2 \text{.classes.pdf}$

#PGL_3.2_classes

PGL_4.2_classes_based_on_normal_form:
$\text{PGL}_4 2 \text{.classes.tex}$

$\text{open PGL}_4 2 \text{.classes.pdf}$

PGL_10.2_classes_based_on_normal_form:
$\text{PGL}_10 2 \text{.classes.tex}$

$\text{open PGL}_10 2 \text{.classes.pdf}$
normal_forms_PGL_4_4:

PGL_4_4_2A_rank:

PGL_4_4_2A_unrank:

PGL_4_5_3B_rank:
PGL$_4$ 3B_unrank:

normal_forms_PGL$_4$:

# related to planes_in_pencil:
# we are computing the action on the planes through the line 0.

on_planes:

# pdflatex PGL$_4$ 8 Gr$_4$ 3 apply.tex
Section 5.7: Group Theoretic Activities Based on Magma

SECTION_GROUP_THEORETIC_ACTIVITIES_BASED_ON_MAGMA:

PGL_2_4_classes:
-define G -linear_group -PGGL 2 4
-end
-with G -do
-group_theoretic_activity
-classes
-end

PGL_7_2_classes:
-define G -linear_group -PGL 7 2
-end
-with G -do
-group_theoretic_activity
-classes
-end

$(MAGMA_PATH)magma PGL_2_4_classes.magma
$(MAGMA_PATH)magma PGL_7_2_classes.magma
-linear_group -PGL 7 2
-end
-with G -do
-group_theoretic_activity
-classes
-end

PGL_8_2_classes:
$(ORBITER) -v 3
define G
-linear_group -PGL 8 2
-end
-with G -do
-group_theoretic_activity
-classes
-end

$(MAGMA_PATH)magma PGL_8_2_classes.magma

PGL_10_2_classes:
$(ORBITER) -v 3
define G
-linear_group -PGL 10 2
-end
-with G -do
-group_theoretic_activity
-classes
-end

$(MAGMA_PATH)magma PGL_10_2_classes.magma
4590
4591
4592 PGGL_2_4_cent_2A:
4593 ▶ $(ORBITER) -v 3 \ 
4594 ▶ ▶ -define G \ 
4595 ▶ ▶ ▶ -linear_group -PGGL 2 4 -end \ 
4596 ▶ ▶ ▶ -with G -do \ 
4597 ▶ ▶ ▶ -group_theoretic_activity \ 
4598 ▶ ▶ ▶ ▶ -centralizer_of_element "2A" "1,0, 0,1, 1" \ 
4599 ▶ ▶ ▶ ▶ -report \ 
4600 ▶ ▶ ▶ -end
4601 ▶ $(MAGMA_PATH)magma element_2A_centralizer.magma
4602 ▶ $(ORBITER) -v 6 \ 
4603 ▶ ▶ -define G \ 
4604 ▶ ▶ ▶ -linear_group -PGGL 2 4 -end \ 
4605 ▶ ▶ ▶ -with G -do \ 
4606 ▶ ▶ ▶ -group_theoretic_activity \ 
4607 ▶ ▶ ▶ ▶ -centralizer_of_element "2A" "1,0, 0,1, 1" \ 
4608 ▶ ▶ ▶ ▶ -report \ 
4609 ▶ ▶ ▶ -end
4610 ▶ pdflatex PGGL_2_4_elt_2A_centralizer.tex
4611 ▶ open PGGL_2_4_elt_2A_centralizer.pdf
4612 ▶
4613
4614
4615 Normalizer_of_H5:
4616 ▶ $(ORBITER) -v 2 \ 
4617 ▶ ▶ -define G -permutation_group -symmetric_group 13 \ 
4618 ▶ ▶ ▶ -subgroup_by_generators H5 5 1 \ 
4619 ▶ ▶ ▶ ▶ $(GENERATORS_H5) -end \ 
4620 ▶ ▶ ▶ -with G -do \ 
4621 ▶ ▶ ▶ -group_theoretic_activity \ 
4622 ▶ ▶ ▶ ▶ -normalizer \ 
4623 ▶ ▶ ▶ -end
4624 ▶ pdflatex Perm13_Subgroup_H5_5_normalizer.tex
4625 ▶ open Perm13_Subgroup_H5_5_normalizer.pdf
4626
4627
4628
4629
4630 PGGL_3_4_classes:
4631 ▶ $(ORBITER) -v 3 \ 
4632 ▶ ▶ -define G \ 
4633 ▶ ▶ ▶ -linear_group -PGGL 3 4 \ 
4634 ▶ ▶ ▶ -end \ 
4635 ▶ ▶ ▶ -with G -do \ 
4636 ▶ ▶ ▶ -group_theoretic_activity \ 

607
$(ORBITER) -v 3 -define G -linear_group -PGGL 4 4 -end -group_theoretic_activity -classes \

# group order 1974067200 = 2^13 * 3^4 * 5^2 * 7 * 17

# the -find_subgroup command is too specialized

subgroups_PGL_4_5:

$(ORBTER) -v 6 -define G -linear_group -PGL 4 5 -end -group_theoretic_activity -find_subgroup 3 -end

pdflatex PGL_4_5_report.tex

open PGL_4_5_report.pdf

classes_PGL_4_5:

$(ORBTER) -v 6 -define G -linear_group -PGL 4 5 -end -group_theoretic_activity \
# 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
# of order 3 and with 6 fixed points.
#0,0,0,1,2,3,0,1,0,3,4,4,0,1,2,1,

PGL_4_5_3B_class_again:

search_primitive_poly_q5_deg3:

#OK, we found an irreducible and primitive polynomial X^{3} + X^{2} + 2

GL_3_5_singer_power:

PGL_4_5_norm_31:
Normalizer of $Z_{22}$ in $PGL_{2,9}$:

$$(ORBITER) -v 4$$

$$(ORBITER) -v 2$$

$(ORBITER) -v 2$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$

$(ORBITER) -v 4$
(-with Orb -do -orbits_activity \ 
-draw_tree 0 \ 
-end
pdflatex PGL_4_2.orbit_0.tree.tex
open PGL_4_2.orbit_0.tree.pdf

orbits_PGL_4_2.on_points_export_trees:
$(ORBITER) -v 4 \ 
-draw_options -embedded -end \ 
-define G -linear_group -PGL 4 2 -end \ 
-define Orb -orbits -group G \ 
-define _on_points \ 
-end
-define G -linear_group -GL_d_q_wreath_Sym n 2 2 3 \ 
-define _on_rank_one_tensors -end \ 
-define Orb -orbits -group G \ 
-define _on_points \ 
-end \ 
-with Orb -do -orbits_activity \ 
-report \ 
-end

T3r1_orbits:
$(ORBITER) -v 4 \ 
-define G \ 
-linear_group -GL_d_q_wreath_Sym n 2 2 3 \ 
-on_rank_one_tensors -end \ 
-define Orb -orbits -group G \ 
-on_points \ 
-end \ 
-with Orb -do -orbits_activity \ 
-report \ 
-end \ 
-with Orb -do -orbits_activity \ 
-draw_tree 0 \ 
-end
pdflatex GL_2_2.wreath_Sym3.orbit_0.tree.tex
open GL_2_2.wreath_Sym3.orbit_0.tree.pdf
T3r1_orbits_draw:

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

$\text{-draw\_layered\_graph} \$

$\text{GL}_2^2 \text{wreath Sym3 \_res27\_0.layered\_graph} \$

$\text{-radius 500 \_spanning\_tree \_embedded} \$

$\text{-line\_width 1.1 \_x\_stretch 1.4 \_scale 0.25} \$

$\text{-end} \$

$\#pdflatex GL_2^2 \text{wreath Sym3 \_report.tex}$

$\#open GL_2^2 \text{wreath Sym3 \_report.pdf}$

$\#pdflatex GL_2^2 \text{wreath Sym3 \_res27\_0.draw.tex}$

$\#open GL_2^2 \text{wreath Sym3 \_res27\_0.draw.pdf}$

$\#write GL_2^2 \text{wreath Sym3 \_res27\_0.layered\_graph}$

$\text{2C\_orbit\_under\_PGGL_4\_4\_elements\_coded.csv:}$

$\text{\$\text{(ORBITER)} -v 6 \$}$

$\text{\$\text{-define G \_linear\_group \_PGGL 4 4 \_end} \$}$

$\text{\$\text{-with G \_do} \$}$

$\text{\$\text{-group\_theoretic\_activity} \$}$

$\text{\$\text{-conjugacy\_class\_of\_element} \$}$

$\text{\$\text{\"2C\" \"1,0,0,0, 1,1,0,0, 0,0,1,0, 0,0,1,1, 0\"} \$}$

$\text{\$\text{-end} \$}$

$\text{\# class of size 64260}$

$\text{\# creates:}$

$\text{\# 2C\_orbit\_under\_PGGL_4\_4.csv}$

$\text{\# 2C\_orbit\_under\_PGGL_4\_4.txt}$

$\text{\# 2C\_orbit\_under\_PGGL_4\_4\_elements\_coded.csv}$

$\text{\# 2C\_orbit\_under\_PGGL_4\_4\_transporter.csv}$

$\text{\# 1:33 on Mac}$

$\text{\#User time: 2:59 on Mac}$

$\text{\text{PGGL}_4\_4\_subgroups\_of\_type\_2C\_2C: 2C\_orbit\_under\_PGGL_4\_4\_elements\_coded.csv}$

$\text{\$\text{(ORBITER)} -v 6 \$}$

$\text{\$\text{-define G \_linear\_group \_PGGL 4 4} \$}$

$\text{\$\text{-subgroup\_by\_generators \"centralizer\_2C\" \"30720\" 9} \$}$

$\text{\$\text{\"1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1,1\"} \$}$

$\text{\$\text{\"1,0,0,0,0,1,0,0,0,0,2,0,0,0,0,2,1,1\"} \$}$

$\text{\$\text{\"1,0,0,0,0,1,0,0,0,0,1,0,2,0,3,1,0\"} \$}$

$\text{\$\text{\"1,0,0,0,0,1,0,0,0,0,1,0,3,1,0,0\"} \$}$

$\text{\$\text{\"1,0,0,0,0,1,0,0,0,1,0,1,1,1,1,1\"} \$}$

$\text{\$\text{\"1,0,0,0,1,1,0,0,0,0,1,0,0,0,0,1,1,0\"} \$}$
#The distribution of orbit lengths is: (1, 2, 15, 20, 24^3, 30, 40, 240, 256, 480, 512, 960^2, 1280, 1920^2, 2560^4, 3840, 5120, 6144^3, 7680, 10240)

#group_theoretic_activity::do orbits on group elements under conjugation after Cl
asses.compute_all_point_orbits

#found 29 conjugacy classes

#User time: 0:57

orbits.on_conics.q13:

orbits.cubic_curves.q2:

orbits.cubic_curves.q2_with_draw_tree:
Todo: problem in report

poly_orbits_d3_n3_q2.csv:
$((ORBITER) \,-v \, 4 \) 
$((ORBITER) \,-draw_options \,-yout \, 500000 \,-radius \, 15 \,-nodes_empty \) 
$((ORBITER) \,-line_width \, 0.5 \,-y\,stretch \, 0.25 \,-embedded \) 
$((ORBITER) \,-define G \,-linear_group \,-PGL \, 4 \, 2 \) 
$((ORBITER) \,-define Orb \,-orbits \,-group G \) 
$((ORBITER) \,-on\,polynomials \, 3 \) 
$((ORBITER) \,-end \) 
$((ORBITER) \,-with Orb \,-do \,-orbits\,activity \) 
$((ORBITER) \,-report \) 
$((ORBITER) \,-end \) 
$((ORBITER) \,-with Orb \,-do \,-orbits\,activity \) 
$((ORBITER) \,-draw_tree \, 6 \) 
$((ORBITER) \,-end \) 

poly_orbits_d3_n3_q2_get_ranks:
$((ORBITER) \,-v \, 4 \) 
$((ORBITER) \,-csv\,file\,select\,cols\,poly_orbits_d3_n3_q2.csv \, 0 \) 
#latex\,poly_orbits_d3_n3_q2.tex 
#open poly_orbits_d3_n3_q2.pdf

T4_orbits:
$((ORBITER) \,-v \, 4 \) 
$((ORBITER) \,-define G \) 
$((ORBITER) \,-linear_group \,-GL\,d\,q\,wr\,Sym\,n \, 2 \, 2 \, 4 \) 
$((ORBITER) \,-on\,tensors \,-end \) 
$((ORBITER) \,-define Orb \,-orbits \,-group G \) 
$((ORBITER) \,-on\,points \) 
$((ORBITER) \,-end \) 

pdflatex GL_2_2\,wreath\,Sym4\,res65535\,orbits.tex 
open GL_2_2\,wreath\,Sym4\,res65535\,orbits.pdf
\begin{verbatim}
T4r1_orbits:
\$\mathrm{ORBITER}\ -v 4 \$
\> define G -linear\_group -GL\_d\_q\_wr\_Sym\_n 2 2 4 \\
\> on_rank\_one\_tensors -end \\
\> define Orb -orbits -group G \\
\> on_points \\
\> end \\
\> with Orb -do -orbits\_activity \\
\> export\_trees \\
\> end

T4r1_orbits\_draw:
\$\mathrm{ORBITER}\ -v 3 \$
\> draw\_layered\_graph \\
\> orbit\_GL\_2 2\_wreath\_Sym\_n\_res81\_0.layered\_graph \\
\> radius 400 -spanning\_tree -embedded \\
\> line\_width 1.1 -x\_stretch 2.5 -scale 0.15 \\
\> end \\
\> pdf\_latex orbit\_GL\_2 2\_wreath\_Sym\_n\_res81\_0.draft.tex

T4r1_orbits\_4:
\$\mathrm{ORBITER}\ -v 4 \$
\> orbiter\_path $(\mathrm{ORBITER}\_PATH)$ \\
\> define Control -poset\_classification\_control -problem\_label T4r1 -W \\
\> bit\_depth 4 -draw\_options -end -draw\_poset -report -end \\
\> end \\
\> define G -linear\_group -GL\_d\_q\_wr\_Sym\_n 2 2 4 \\
\> on_rank\_one\_tensors -end \\
\> define Orb -orbits -group G \\
\> on_subsets 4 Control \\
\> end \\
\> pdf\_latex T4r1\_poset.tex

PGGL\_2 8\_on\_conic\_orbits:
\$\mathrm{ORBITER}\ -v 4 \$
\end{verbatim}
# example from the Fining manual, page 107:

PGGL_7_8_orbits:

```
> $(ORBITER) -v 4 \
> -define G \n> -linear_group -PGGL 7 8 -end \n> -define Orb -orbits -group G \n> -on_points \n> -end
```

# 1 min 31 sec on Mac

### Section 6.2: Poset Classification

poset of 4 subsets:

```
> $(ORBITER) -v 3 \
> -orbiter_path $(ORBITER_PATH) \
> -define Control -poset_classification_control \
> -problem_label poset_4 \
> -W -depth 4 \
> -draw_options -radius 200 -embedded -end \
> -report -end \
> -draw_poset \
> -end
```

```
> -define G -linear_group -PGL 2 3 -identity_group -end \
> -define Orb -orbits -group G \
> -on_subsets 4 Control \n> -end
```

```
> #pdflatex poset_4_poset_lvl_4_draw.tex
> #open poset_4_poset_lvl_4_draw.pdf
```

poset_of_4subsets_draw:

```
```
$(\text{ORBITER}) -v 3$
- draw_layered_graph
- poset_4_poset_lvl_4.layered_graph
- radius 300 -embedded -line_width 1.1
- y_stretch 0.9 -scale 0.25
-end
pdflatex poset_4_poset_lvl_4.draw.tex
open poset_4_poset_lvl_4.draw.pdf

poset_of_5_subsets:
$(\text{ORBITER}) -v 3$
- orbiter_path $(\text{ORBITER\_PATH})$
- define Control -poset_classification_control
- problem_label poset_5
- W -depth 5 -draw_options -radius 150 -end
- report -end -draw_poset
-end
- define G -linear_group -PGL 2 4 -identity_group -end
- define Orb -orbits -group G
- on_subsets 5 Control
-end
pdflatex poset_5_poset.tex
open poset_5_poset.pdf

poset_of_5_subsets_draw:
$(\text{ORBITER}) -v 3$
- 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
pdflatex poset_5_poset_lvl_5.draw.tex
open poset_5_poset_lvl_5.draw.pdf

Symmetric_4_on_pairs_poset:
$(\text{ORBITER}) -v 3$
- orbiter_path $(\text{ORBITER\_PATH})$
- define Control -poset_classification_control
- problem_label Sym4_on2
- W -depth 6 -draw_options -radius 150 -end
- report -end -draw_poset
-end
- define G -permutation_group -symmetric_group 4 -end

Symmetric_4 on pairs poset:
$\text{Symmetric}_4$ on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
Symmetric_4 on pairs poset:
-define G_on_2subsets -modified_group -from G \\
-define G_on_ksubsets 2 \\
-end \\
-define Orb_orbits -group G_on_2subsets \\
-on_subsets 6 Control \\
-end \\
pdflatex Sym4_on2.poset.tex  \\
open Sym4_on2.poset.pdf  \\

V_3_2_trivial:  \\
$(ORBITER) -v 5 \ \\
-orbiter_path $(ORBITER_PATH) \ \\
(define Control -poset_classification.control \ \\
-problem_label V_3_2_trivial \ \\
-W -depth 3 -node_label_is_element \ \\
-draw_options \ \\
-radius 200 -embedded \ \\
-end \ \\
-report -end \ \\
-draw_poset \ \\
-end \ \\

V_4_2_trivial:  \\
$(ORBITER) -v 5 \ \\
-orbiter_path $(ORBITER_PATH) \ \\
(define Control -poset_classification.control \ \\
-problem_label V_4_2_trivial \ \\
-W -depth 3 -node_label_is_element \ \\
-draw_options \ \\
-radius 200 -embedded \ \\
-end \ \\
-report -end \ \\
-draw_poset \ \\
-end \ \\

define G -linear_group -PGL 3 2 -identity_group -end \ \\
define Orb_orbits -group G \ \\
on_subspaces 3 Control \ 
-end \\

define G -linear_group -PGL 4 2 -identity_group -end \ \\
define Orb_orbits -group G \ 
on_subspaces 4 Control \ 
-end
# Section 6.3: Orbits on Subsets

SECTION ORBITS ON SUBSETS:

PG_2_2_subsets:

[$(ORBITER) -v 3 \ -orbiter_path $(ORBITER_PATH) \ -define Control -poset_classification_control \ -problem_label PGL_3_2 \ -depth 7 \ -draw.options \ -radius 200 -embedded \ -end \ -report -end \ -draw_poset \ -end \ -define F -finite_field -q 2 -end \ -define G -linear_group -PGL 3 F -end \ -define Orb -orbits -group G \ -on_subsets 7 Control \ -end \ pdflatex PGL_3_2_poset_lvl_7.tex \ open PGL_3_2_poset_lvl_7.pdf \ pdflatex PGL_3_2_poset.tex \ open PGL_3_2_poset.pdf \ #pdflatex PGL_3_2_poset_detailed_lvl_7.tex \ #open PGL_3_2_poset_detailed_lvl_7.pdf

PG_3_2_subsets:

[$(ORBITER) -v 3 \ -orbiter_path $(ORBITER_PATH) \ -define Control -poset_classification_control \ -problem_label PGL_4_2 \ -depth 15 \ -draw.options \ -radius 200 -embedded \
5198 \$-end \$
5199 \$-report -end \$
5200 \$-draw_poset \$
5201 \$-end \$
5202 \$-define F -finite_field -q 2 -end \$
5203 \$-define G -linear_group -PGL 4 F -end \$
5204 \$-define Orb -orbits -group G \$
5205 \$-on_subsets 15 Control \$
5206 \$-end \$
5207 \$pdflatex PGL_4_2_poset.tex \$
5208 \$open PGL_4_2_poset.pdf \$
5209
5210 PGL_3_2_singer:
5211 \$-(ORBITER) -v 3 \$
5212 \$-orbiter_path $(ORBITER_PATH) \$
5213 \$-define Control -poset_classification_control \$
5214 \$-problem_label PGL_3_2_singer_1 -W -depth 7 \$
5215 \$-draw_poset \$
5216 \$-report -end \$
5217 \$-end \$
5218 \$-define G -linear_group -PGL 3 2 -singer 1 -end \$
5219 \$-define Orb -orbits -group G \$
5220 \$-on_subsets 7 Control \$
5221 \$-end \$
5222 \$pdflatex PGL_3_2_singer_1_poset.tex \$
5223 \$open PGL_3_2_singer_1_poset.pdf \$
5224
5225
5226
5227 PGL_3_2_on_lines:
5228 \$-(ORBITER) -v 3 \$
5229 \$-orbiter_path $(ORBITER_PATH) \$
5230 \$-define Control -poset_classification_control \$
5231 \$-problem_label PGL_3_2_lines -W -depth 7 \$
5232 \$-draw_poset \$
5233 \$-report -end \$
5234 \$-end \$
5235 \$-define G -linear_group -PGL 3 2 -end \$
5236 \$-define G_on_lines -modified_group -from G \$
5237 \$-on_k_subspaces 2 \$
5238 \$-end \$
5239 \$-define Orb -orbits -group G_on_lines \$
5240 \$-on_subsets 7 Control \$
5241 \$-end \$
5242 \$pdflatex PGL_3_2_lines_poset.tex \$
5243 \$open PGL_3_2_lines_poset.pdf \$
5244
620
PGL_2.5_on_subsets:
$(\text{ORBITER}) -v 5 \$
  -orbiter_path $(\text{ORBITER\_PATH}) \\
  -define Control -poset\_classification\_control \\
  -problem_label PGL_2.5 -W -depth 6 \\
  -draw_poset \\
  -draw_options -radius 200 -end \\
  -report -end \\
  -end \\
  -define G -linear_group -PGL 2.5 -end \\
  -define Orb -orbits -group G \\
  -on_subsets 6 Control \\
  -end \\
  pdflatex PGL_2.5_poset.tex \\
  open PGL_2.5_poset.pdf \\

PGL_2.7_on_subsets:
$(\text{ORBITER}) -v 10 \$
  -orbiter_path $(\text{ORBITER\_PATH}) \\
  -define Control -poset\_classification\_control \\
  -problem_label PGL_2.7 -W -depth 8 \\
  -draw_poset \\
  -draw_options -radius 200 -end \\
  -report -end \\
  -end \\
  -define G -linear_group -PGL 2.7 -end \\
  -define Orb -orbits -group G \\
  -on_subsets 8 Control \\
  -end \\
  pdflatex PGL_2.7_poset.tex \\
  open PGL_2.7_poset.pdf \\

PGGL_2.8_on_subsets:
$(\text{ORBITER}) -v 10 \$
  -orbiter_path $(\text{ORBITER\_PATH}) \\
  -define Control -poset\_classification\_control \\
  -problem_label PGGL_2.8 -W -depth 9 \\
  -draw_poset \\
  -draw_options -radius 200 -end \\
  -report -end \\
  -end \\
  -define G -linear_group -PGGL 2.8 -end \\
  -define Orb -orbits -group G \\
  -on_subsets 9 Control \\
  -end
PGGL_2.9_on_subsets:
$(ORBITER) -v 10 \n-orbiter_path $(ORBITER_PATH) \n(define Control -poset_classification_control \n-problem_label PGGL_2.9 -W -depth 10 \n-draw_poset \n-draw_options -radius 200 -end \n-report -end \n-end \n-define G -linear_group -PGGL 2 9 -end \n-define Orb -orbits -group G \n-on_subsets 10 Control \n-end
pdflatex PGGL_2.9_poset.tex
open PGGL_2.9_poset.pdf

PGGL_2.11_on_subsets:
$(ORBITER) -v 10 \n-orbiter_path $(ORBITER_PATH) \n(define Control -poset_classification_control \n-problem_label PGGL_2.11 -W -depth 12 \n-draw_poset \n-draw_options -radius 200 -end \n-report -end \n-end \n-define G -linear_group -PGL 2 11 -end \n-define Orb -orbits -group G \n-on_subsets 12 Control \n-end
pdflatex PGL_2.11_poset.tex
open PGL_2.11_poset.pdf

PGGL_2.16_on_subsets:
$(ORBITER) -v 3 \n-orbiter_path $(ORBITER_PATH) \n(define Control -poset_classification_control \n-problem_label PGGL_2.16 -W -depth 10 \n-draw_poset \n-report -end \n-end \n
5339 ▶ ▶ -define G -linear_group -PGGL 2 16 -end \
5340 ▶ ▶ -define Orb -orbits -group G \
5341 ▶ ▶ ▶ -on_subsets 10 Control \
5342 ▶ ▶ -end 
5343 ▶ pdflatex PGGL_2_16_poset.tex 
5344 ▶ open PGGL_2_16_poset.pdf 
5345 5346 5347 PGGL_2_32_on_subsets: 
5348 ▶ $(ORBITER) -v 3 \ 
5349 ▶ ▶ -orbiter_path $(ORBITER_PATH) \ 
5350 ▶ ▶ -define Control -poset_classification_control \ 
5351 ▶ ▶ ▶ -problem_label PGGL_2_32 -W -depth 8 \ 
5352 ▶ ▶ ▶ -draw_poset \
5353 ▶ ▶ ▶ -report -end \ 
5354 ▶ ▶ -end \ 
5355 ▶ ▶ -define G -linear_group -PGGL 2 32 -end \ 
5356 ▶ ▶ -define Orb -orbits -group G \ 
5357 ▶ ▶ ▶ -on_subsets 8 Control \ 
5358 ▶ ▶ -end 
5359 ▶ pdflatex PGGL_2_32_poset.tex 
5360 ▶ open PGGL_2_32_poset.pdf 
5361 5362 5363 PG_3_4_subsets: 
5364 ▶ $(ORBITER) -v 3 \ 
5365 ▶ ▶ -orbiter_path $(ORBITER_PATH) \ 
5366 ▶ ▶ -define Control -poset_classification_control \ 
5367 ▶ ▶ ▶ -problem_label PGGL_4_4 \ 
5368 ▶ ▶ ▶ -depth 5 \ 
5369 ▶ ▶ ▶ -draw_poset \ 
5370 ▶ ▶ ▶ -draw_options \ 
5371 ▶ ▶ ▶ ▶ -radius 200 \ 
5372 ▶ ▶ ▶ -end \ 
5373 ▶ ▶ ▶ -report -end \ 
5374 ▶ ▶ -end \ 
5375 ▶ ▶ -define G -linear_group -PGGL 4 4 -end \ 
5376 ▶ ▶ -define Orb -orbits -group G \ 
5377 ▶ ▶ ▶ -on_subsets 5 Control \ 
5378 ▶ ▶ -end 
5379 ▶ pdflatex PGGL_4_4_poset.tex 
5380 ▶ open PGGL_4_4_poset.pdf 
5381 5382 5383 PGGL_2_9_orbits: 
5384 ▶ $(ORBITER) -v 3 \ 
5385 ▶ ▶ -orbiter_path $(ORBITER_PATH) \
5386 ⧢ ⧢ -define Control -poset_classification.control \n5387 ⧢ ⧢ ⧢ -problem_label PGGL_2_9 -W -depth 5 \n5388 ⧢ ⧢ ⧢ ⧢ -report -end \n5389 ⧢ ⧢ ⧢ -draw_poset \n5390 ⧢ ⧢ ⧢ ⧢ -draw_options -radius 200 -end \n5391 ⧢ ⧢ -end \n5392 ⧢ ⧢ -define G -linear_group -PGGL 2 9 -end \n5393 ⧢ ⧢ ⧢ -define Orb -orbits -group G \n5394 ⧢ ⧢ ⧢ ⧢ -on_subsets 5 Control \n5395 ⧢ ⧢ ⧢ -end \n5396 ⧢ ⧢ pdflatex PGGL_2_9_poset.tex \n5397 ⧢ ⧢ open PGGL_2_9_poset.pdf \n5398 \n5399 \n5400 \n5401 \n5402 PG0_5_2_on_subsets: \n5403 ⧢ ⧢ ⧢ ⧢ $(ORBITER) -v 3 \n5404 ⧢ ⧢ ⧢ ⧢ ⧢ -orbiter_path $(ORBITER_PATH) \n5405 ⧢ ⧢ ⧢ ⧢ ⧢ -define Control -poset_classification.control \n5406 ⧢ ⧢ ⧢ ⧢ ⧢ ⧢ -problem_label PG0_5_2 \n5407 ⧢ ⧢ ⧢ ⧢ ⧢ ⧢ -depth 15 \n5408 ⧢ ⧢ ⧢ ⧢ ⧢ ⧢ -report -end \n5409 ⧢ ⧢ ⧢ ⧢ ⧢ -draw_poset \n5410 ⧢ ⧢ ⧢ ⧢ ⧢ -w \n5411 ⧢ ⧢ ⧢ ⧢ -end \n5412 ⧢ ⧢ ⧢ ⧢ -define F -finite_field -q 2 -end \n5413 ⧢ ⧢ ⧢ ⧢ -define G -linear_group -PGO 5 F -end \n5414 ⧢ ⧢ ⧢ ⧢ -define Orb -orbits -group G \n5415 ⧢ ⧢ ⧢ ⧢ ⧢ ⧢ -on_subsets 15 Control \n5416 ⧢ ⧢ ⧢ ⧢ -end \n5417 ⧢ ⧢ pdflatex PGO_5_2_poset.tex \n5418 ⧢ ⧢ open PGO_5_2_poset.pdf \n5419 \n5420 \n5421 \n5422 \n5423 # Section 6.4: Orbits on Subspaces
5424 \n5425 \n5426 SECTION_ORBITS_ON_SUBSPACES:
5427 ⧢
5428 \n5429 subspaces_0p_4_2:
5430 ⧢ ⧢ $(ORBITER) -v 5 \n5431 ⧢ ⧢ ⧢ -orbiter_path $(ORBITER_PATH) \n
624
PGL.4_2.on_subspaces:

PGL.4_2.singer_on_subspaces:
PGL_8_2_singer_on_subspaces:
$ $(ORBITER) -v 5 $
$ -orbiter_path $(ORBITER_PATH) $
$ -define Control -poset_classification_control $
$ -node_label_is_element $
$ -draw_poset $
$ -draw_options -radius 150 -end $
$ -problem_label PGL_8_2_singer $
$ -W -depth 8 -report -end $
$ -end $
$ -define G -linear_group -PGL 8 2 -singer 1 -end $

# May 7, 2020: 16 sec on Mac
# 1643 orbits in total

Op_6_2_orbits_on_subspaces:
$ $(ORBITER) -v 5 $
$ -orbiter_path $(ORBITER_PATH) $
$ -define Control -poset_classification_control $
$ -node_label_is_element $
$ -draw_poset $
$ -draw_options -radius 200 -end $
$ -problem_label Op_6_2 -W $
$ -depth 6 -report -end $
$ -end $
$ -define G -linear_group -PGL 6 2 -orthogonal 1 -end $

Op_6_3_orbits_on_subspaces:
$ $(ORBITER) -v 5 $
$ -orbiter_path $(ORBITER_PATH) $
$ -define Control -poset_classification_control $
$ -node_label_is_element $
$ -draw_poset $
$ -draw_options -radius 200 -end $
$ -problem_label Op_6_3 -W $
$ -depth 6 -report -end $
$ -end $
$ -define G -linear_group -PGL 6 3 -orthogonal 1 -end $

626
5527  \texttt{define Orb -orbits -group G} \\
5528  \texttt{-on_subspaces 6 Control} \\
5529  \texttt{-end} \\
5530 \\
5531 \# June 3, 2020 on Mac: 0 sec \\
5532 \\
5533 \\
5534 5535 \texttt{Op}_{6,11}\texttt{.orbits_on_subspaces:} \\
5536  \texttt{\$\texttt{(ORBITER) -v 5 \}} \\
5537  \texttt{\texttt{-orbiter_path \texttt{$\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{(ORBITER_PATH))}}}}}}}}}}}}}}}}}}}}}} \\
5538  \texttt{\texttt{-define Control -poset_classification_control \}} \\
5539  \texttt{\texttt{-node_label_is_element \}} \\
5540  \texttt{\texttt{-draw_poset \}} \\
5541  \texttt{\texttt{-draw_options -radius 200 -end \}} \\
5542  \texttt{\texttt{-problem_label Op}_{6,11} -W \}} \\
5543  \texttt{\texttt{-depth 6 -report -end \}} \\
5544  \texttt{\texttt{-end \}} \\
5545  \texttt{\texttt{-draw_options -nodes_empty -end \}} \\
5546  \texttt{\texttt{-define G -linear_group -PGL 6 11 -orthogonal 1 -end \}} \\
5547  \texttt{\texttt{-define Orb -orbits -group G \}} \\
5548  \texttt{\texttt{-on_subspaces 6 Control \}} \\
5549  \texttt{\texttt{-end \}} \\
5550 \\
5551 \\
5552 \# June 3, 2020 on Mac: 12 sec \\
5553 \\
5554 \\
5555 5555 \texttt{Op}_{8,2}\texttt{.orbits_on_subspaces:} \\
5556  \texttt{\$\texttt{(ORBITER) -v 5 \}} \\
5557  \texttt{\texttt{-orbiter_path \texttt{$\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{(ORBITER_PATH))}}}}}}}}}}}}}}}}}}}}}} \\
5558  \texttt{\texttt{-define Control -poset_classification_control \}} \\
5559  \texttt{\texttt{-node_label_is_element \}} \\
5560  \texttt{\texttt{-draw_poset -draw_options -radius 200 -end \}} \\
5561  \texttt{\texttt{-problem_label Op}_{8,2} -W -depth 8 -report -end \}} \\
5562  \texttt{\texttt{-end \}} \\
5563  \texttt{\texttt{-define G -linear_group -PGL 8 2 -orthogonal 1 -end \}} \\
5564  \texttt{\texttt{-define Orb -orbits -group G \}} \\
5565  \texttt{\texttt{-on_subspaces 8 Control \}} \\
5566  \texttt{\texttt{-end \}} \\
5567 \\
5568 \\
5569 \texttt{PGO}_{7,2}\texttt{.on_subspaces:} \\
5570  \texttt{\$\texttt{(ORBITER) -v 20 \}} \\
5571  \texttt{\texttt{-orbiter_path \texttt{$\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{(ORBITER_PATH))}}}}}}}}}}}}}}}}}}}}}} \\
5572  \texttt{\texttt{-define Control -poset_classification_control \}} \\
5573  \texttt{\texttt{-node_label_is_element \}}
### Section 6.5: Orbits on set partitions

**C6 on partition:**

```plaintext
  > $(ORBITER) -v 5
  > -orbit_path $(ORBITER_PATH)
  > -define Control -poset_classification_control
  > -problem_label C6
  > -depth 2
  > -W
  > -draw_options
  > -radius 200 -embedded
  > -end
  > -end
  > -define G -permutation_group -cyclic_group 6
  > -define Orb -orbits -group G
  > -on_partition 2 Control
  > -end
```

**PGL 2.17 on partition:**

```plaintext
  > $(ORBITER) -v 5
  > -define Control -poset_classification_control
  > -problem_label PGL_2.17
  > -depth 6
  > -W
  > -end
  > -define G -linear_group -PGL 2 17
```

---

**W -depth 7**

**-report -end**

**-draw poset**

**-define F -finite_field -q 2 -end**

**-define G -linear_group -PGL 7 F -orthogonal 0 -end**

**-define Orb -orbits -group G**

**-on_subspaces 7 Control**

**-end**
# Section 6.6: Arcs and Caps in Projective Spaces

---

**PGL\_3\_27:**

- \$\texttt{ORBITER}\$ -v 5 \\
- \$\texttt{define G}\$
- \$\texttt{linear_group -PGL 3 27 -end}\$
- \$\texttt{with G -do}\$
- \$\texttt{group_theoretic_activity}\$
- \$\texttt{report}\$
- \$\texttt{end}\$

- \texttt{pdflatex PGL\_3\_27_report.tex}
- \texttt{open PGL\_3\_27_report.pdf}

---

**AGGL\_2\_27:**

- \$\texttt{ORBITER}\$ -v 5 \\
- \$\texttt{define G}\$
- \$\texttt{linear_group -AGGL 2 27 -end}\$
- \$\texttt{with G -do}\$
- \$\texttt{group_theoretic_activity}\$
- \$\texttt{report}\$
- \$\texttt{end}\$

- \texttt{pdflatex AGGL\_2\_27_report.tex}
- \texttt{open AGGL\_2\_27_report.pdf}

---

**hyperoval\_4.classify:**

- \$\texttt{ORBITER}\$ -v 4 \\
- \$\texttt{define F -finite_field -q 4 -end}\$
- \$\texttt{define P -projective_space -n 2 -field F -v 0 -end}\$
- \$\texttt{with P -do}\$
- \$\texttt{projective_space_activity}\$

629
classify arcs

poset_classification

problem_label hyperoval_q4

W -depth 6

report -end

end

draw -end

target size 6

d -2

end

draw options

radius 200

draw -end

end

target size 10

d -2

end

frame_stabilizer_PGGL:

define G

-linear_group -PGGL 3 8 -end \
frame_stabilizer_PGL:
$$(\text{ORBITER}) \ -v \ 4 \ \backslash$$
-define G \ns
-linear_group -PGL 3 8 -end \ns
-with G -do
-group_theoretic_activity \ns
-poset_classification_control \ns
-problem_label frame_q8 -W -depth 4 \
s
-draw_options -end\ns
-report -end\ns
-end\ns
-classify_arcs\ns
-target_size 4 \ns
-q 8 \s
-n 3 \s
-d 2 \s
-end \s
-end

pdflatex frame_q8_poset.tex
open frame_q8_poset.pdf

hyperoval.16.classify:
$$(\text{ORBITER}) \ -v \ 4 \ \backslash$$
-orbiter_path $(\text{ORBITER}\_\text{PATH}) \backslash$$
-define F -finite_field -q 16 -end \s
-define P -projective_space -n 2 -field F -v 0 -end \s
-with P -do \s
-projective_space_activity \s

-classify_arcs \
-poset_classification_control \
-problem_label hyperoval_q16 -W -depth 18 \
-report -end \
-target_size 18 \
-d 2 \
-end \
-pdflatex hyperoval_q16_poset.tex 
-open hyperoval_q16_poset.pdf 

#-draw_poset -draw_options -end \

hyperoval_16_1_conic_type: 
$\$(ORBITER) -v 2 \ 
-def F -finite_field -q 16 -end \ 
-def P -projective_space -n 2 -field F -v 0 -end \ 
-def H_16_1 -geometric_object P \ 
-set $(HYPEROVAL_16_144) \ 
-end \ 
-with H_16_1 -do \ 
-combinatorial_object_activity \ 
-save \ 
-end \ 
-with H_16_1 -do \ 
-combinatorial_object_activity \ 
-conic_type 6 \ 
-end \ 
-print_symbols 

hyperoval_16_1_nonconical_type: 
$\$(ORBITER) -v 2 \ 
-def F -finite_field -q 16 -end \ 
-def P -projective_space -n 2 -field F -v 0 -end \ 
-def H_16_1 -geometric_object P \ 
-set $(HYPEROVAL_16_144) \ 
-end \ 
-with H_16_1 -do \ 
-combinatorial_object_activity \ 
-save \ 
-end \ 
-with H_16_1 -do \ 
-combinatorial_object_activity \
We found 17028 non-conical 6 subsets

#Eckardt point number distribution: $13^{252}, 9^{720}, 5^{2304}, 3^{13752}$

We found 6188 = \binom{17}{5} non-conical 6 subsets

#Eckardt point number distribution: $45^{68}, 13^{2040}, 5^{4080}$

#neighbors of 0 with 4 removed.csv
Row,C0,C1,C2,C3
0,2,3,9,10
1,1,3,7,8
2,10,12,13,15
3,1,5,10,11
4,3,5,6,13
5,8,9,11,12
6,7,11,13,17
7,7,10,14,16
8,1,9,13,16
9,2,8,13,14
10,1,2,15,17
11,6,8,10,17
12,6,7,9,15
13,2,6,11,16
14,5,9,14,17
hyperoval

$\diamondsuit$

-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

634
5902  # nonconical 6-arcs are used for classifying cubic surfaces:
5903
5904
5905
5906
5907
5908  nc_arcs_16:
5909    $(ORBITER) -v 4 \
5910    -define F -finite_field -q 16 -end \
5911    -define P -projective_space -n 2 -field F -v 0 -end \n5912    -with P -do \n5913    -projective_space_activity \n5914    -classify_arcs \n5915    -poset.classification_control \n5916    -problem_label nc_arcs_q16_d2 \n5917    -W -depth 6 \n5918    -report -end \n5919    -target_size 6 \n5920    -d 2 \n5921    -conic_test \n5922    -end \n5923    -end
5924
5925    pdflatex nc_arcs_q16_d2_poset.tex
5926    open nc_arcs_q16_d2_poset.pdf
5927
5928
5929  #User time: 0:00
5930
5931
5932
5933  nc_arcs_32_E13:
5934    $(ORBITER) -v 4 \
5935    -oribter_path $(ORBITER_PATH) \n5936    -define F -finite_field -q 32 -end \
5937    -define P -projective_space -n 2 -field F -v 0 -end \n5938    -with P -do \n5939    -projective_space_activity \n5940    -classify_arcs \n5941    -poset.classification_control \n5942    -problem_label nc_arcs_q32_d2 \n5943    -W -depth 6 \n5944    -draw_poset -draw_options -end \n5945    -report -end \n5946    -target_size 6 \n5947    -test_nb_Eckardt_points 13 \
5948
\begin{verbatim}
F64_work:
  $(ORBITER) -v 3 \\
  -define F -finite_field -q 64 -end \\
  -define f -formula "f" "f" "a*a+a" \\
  -with F -do -finite_field_activity \\
  -evaluate f "a=2" -end

F64_frob:
  $(ORBITER) -v 3 \\
  -define F -finite_field -q 64 -end \\
  -define f -formula "f" "f" "a*a*a+a*a*a+a" \\
  -with F -do -finite_field_activity \\
  -evaluate f "a=61" -end

# surfaces with 13 Eckardt points have OCN=0.98,99

surface_64_0:
  $(ORBITER) -v 3 \\
  -define F -finite_field -q 64 -end \\
  -define P -projective_space -n 3 -field F -v 0 -end \\
  -define S -cubic_surface -space P -catalogue 0 -end \\
  -with S -do \\
  -cubic_surface_activity \\
  -report \\
  -report_with_group \\
  -end

# makes it slow:
  #test_nb_Eckardt_points 13 \\
\end{verbatim}
nc_arcs_128:
$ $(ORBITER) -v 4 \\
define F -finite_field -q 128 -end \\
define P -projective_space -n 2 -field F -use_projectivity_subgroup -v 0 -end \\
with P -do \\
projective_space_activity \\
classify_arcs \\
poset_classification_control \\
problem_label nc_arcs_q128_d2 -W -depth 6 \\
report -select_orbits_by_level 6 \\
select_orbits_by_stabilizer_order_multiple_of 24 \\
end \\
end \\
target_size 6 \\
c_RUN_TIME: 0:52
nc_arcs_256_E13:
$ $(ORBITER) -v 8 \\
define F -finite_field -q 256 -end \\
define P -projective_space -n 2 -field F -use_projectivity_subgroup -v 0 -end \\
with P -do \\
projective_space_activity \\
classify_arcs \\
poset_classification_control \\
problem_label nc_arcs_q256_d2 -W -depth 6 \\
report -select_orbits_by_level 6 \\
select_orbits_by_stabilizer_order_multiple_of 24 \\
end \\
end \\
target_size 6 \\
c_RUN_TIME: 0:52
Example_F64:

```
$\texttt{ORBITER} -v 3 \\
define F -finite_field -q 64 -end \\
define P -projective_space -n 3 -field F -v 0 -end \\
define S64_abcd_52_8_8_52 -cubic_surface -space P -family_general_abcd 52 8 8 52 -end \\
with S64_abcd_52_8_8_52 -do \\
cubic_surface_activity \\
report \\
end
```

```
pdflatex surface_family_general_abcd_q64_a52_b8_c8_d52_report.tex
```

###

```
six_arcs_4_nbE13:
$\texttt{ORBITER} -v 3 \\
define F -finite_field -q 4 -end \\
define P -projective_space -n 2 -field F -v 0 -end \\
with P -do \\
projective_space_activity \\
control_six_arcs -problem_label sixarcs_q4 -end \\
six_arcs_not_on_conic -filter_by_nb_Eckardt_points 13 -end
```

```
pdflatex six_arcs_q4_report.tex
```

```
six_arcs_8_nbE13:
$\texttt{ORBITER} -v 3 \\
define F -finite_field -q 8 -end \\
define P -projective_space -n 2 -field F -v 0 -end \\
with P -do \\
projective_space_activity \\
control_six_arcs -problem_label sixarcs_q8 -end \\
six_arcs_not_on_conic -filter_by_nb_Eckardt_points 13 -end
```

```
pdflatex six_arcs_q8_report.tex
```

```
six_arcs_16_nbE13:
$\text{(ORBITER)} -v 3 \$
-define F -finite_field -q 16 -end \n-define P -projective_space -n 2 -field F -v 0 -end \n-with P -do \n-projective_space_activity \n-control six_arcs -problem_label sixarcs_q16 -end \n-six_arcs_not_on_conic -filter_by_nb_Eckardt_points 13 -end

six_arcs_32_nbE13:
$\text{(ORBITER)} -v 3 \$
-define F -finite_field -q 32 -end \n-define P -projective_space -n 2 -field F -v 0 -end \n-with P -do \n-projective_space_activity \n-control six_arcs -problem_label sixarcs_q32 -end \n-six_arcs_not_on_conic -filter_by_nb_Eckardt_points 13 -end

six_arcs_64_nbE13:
$\text{(ORBITER)} -v 3 \$
-define F -finite_field -q 64 -end \n-define P -projective_space -n 2 -field F -v 0 -end \n-with P -do \n-projective_space_activity \n-control six_arcs -problem_label sixarcs_q64 -end \n-six_arcs_not_on_conic -filter_by_nb_Eckardt_points 13 -end

#User time: 0:7
# 9 arcs: ago: 4, 8, 24^5, 48^2

six_arcs_128_nbE13:
$\text{(ORBITER)} -v 3 \$
-define F -finite_field -q 128 -end \n-define P -projective_space -n 2 -field F -v 0 -end \n-with P -do \n-projective_space_activity \n-control six_arcs -problem_label sixarcs_q128 -end \n-six_arcs_not_on_conic -filter_by_nb_Eckardt_points 13 -end

# 1 min 39 sec
# 12 arcs, ago: 4^3, 24^9
six_arcs_256_nbE13:
$\texttt{(ORBITER)} -v 3$

\[\texttt{-define F -finite_field -q 256 -end}\]

\[\texttt{-define P -projective_space -n 2 -field F -v 0 -end}\]

\[\texttt{-with P -do}\]

\[\texttt{-projective_space_activity}\]

\[\texttt{-control_six_arcs -problem_label sixarcs_q256 -end}\]

\[\texttt{-six_arcs_not_on_conic -filter_by_nb_Eckardt_points 13 -end}\]

# 27 minutes on ripoff

# User time: 29:11 on ripoff 7/30/21

five_arcs_q13:
$\texttt{(ORBITER)} -v 4$

\[\texttt{-define F -finite_field -q 13 -end}\]

\[\texttt{-define P -projective_space -n 2 -field F -v 0 -end}\]

\[\texttt{-with P -do}\]

\[\texttt{-projective_space_activity}\]

\[\texttt{-classify_arcs}\]

\[\texttt{-poset_classification_control}\]

\[\texttt{-problem_label five_arcs_q13 -W -depth 5}\]

\[\texttt{-report -end}\]

\[\texttt{-end}\]

\[\texttt{-target_size 5}\]

\[\texttt{-d 2}\]

\[\texttt{-end}\]

\[\texttt{-end}\]

\[\texttt{pdflatex five_arcs_q13_poset.tex}\]

\[\texttt{open five_arcs_q13_poset.pdf}\]

# Section 6.7: Cubic Curves

SECTION_CUBIC_CURVES:

cubic_curves_PG_2.4:
$\texttt{(ORBITER)} -v 3$
> > -orbiter_path $(ORBITER_PATH) \\
> > -define F -finite_field -q 3 -end \\
> > -define P -projective_space -n 2 -field F -v 0 -end \\
> > -with P -do \\
> > -projective_space_activity \\
> > -define F -finite_field -q 4 -end \\
> > -define P -projective_space -n 2 -field F -v 0 -end \\
> > -with P -do \\
> > -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 -n 2 -field F -v 0 -end \\
> > -with P -do \\
> > -projective_space_activity \\
> > -define F -finite_field -q 8 -end \\
> > -define P -projective_space -n 2 -field F -v 0 -end \\
> > -with P -do \\
> > -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 \\
> > 641
cubic_curves_PG.2.8.draw:

$\$(ORBITER) -v 3 \$

$\$ -draw_layered_graph \$

$\$ cc_8_poset_lvl_9.layered_graph \$

$\$ -radius 2 -embedded -line_width 0.01 \$

$\$ -y.Stretch 1.3 -scale 0.5 \$

$\$ -paths_in_between 6 7 9 1 \$

$\$ -end \$

$\$ pdflatex cc_8-poset_lvl_9.draw.tex \$

$\$ open cc_8-poset_lvl_9.draw.pdf \$

# Chapter 7 - Cubic Surfaces

SECTION_CUBIC_SURFACES_CREATION:

surface_4.0:

$\$(ORBITER) -v 3 \$

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

$\$ -define P -projective_space -n 3 -field F -v 0 -end \$

$\$ -define S -cubic_surface -space P -catalogue 0 -end \$

642
\begin{verbatim}
6274 \> \> -with S -do \\
6275 \> \> -cubic_surface_activity \\
6276 \> \> \> -report \\
6277 \> \> \> -report_with_group \\
6278 \> \> -end \\
6279 \> \> -with S -do \\
6280 \> \> -cubic_surface_activity \\
6281 \> \> \> -export_activity "points" \\
6282 \> \> -end \\
6283 \> \> -with S -do \\
6284 \> \> -cubic_surface_activity \\
6285 \> \> \> -export_activity "points off" \\
6286 \> \> -end \\
6287 \> \> -with S -do \\
6288 \> \> -cubic_surface_activity \\
6289 \> \> \> -export_activity "Hesse_planes" \\
6290 \> \> -end \\
6291 \> \#pdflatex surface_catalogue_q4.iso0_report.tex \\
6292 \> \open surface_catalogue_q4.iso0_report.pdf \\
6293 \> \pdflatex surface_catalogue_q4.iso0_with_group.tex \\
6294 \> \open surface_catalogue_q4.iso0_with_group.pdf \\
6295 \\
6296 \# points:
6297 \#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 \\
6298 \\
6299 \# points off:
6300 \#15,16,17,18,19,20,21,22,24,25,28,29,32,33,36,37,40,41,43,44,45,46,49,50,55,56,57 ,58,63,64,65,66,71,72,73,74,77,78,83,84 \\
6301 \\
6302 HIRSCHFELD_SURFACE_POINTS_OFF="15,16,17,18,19,20,21,22,24,25,28,29,32,33,36,37,40, \\
41,43,44,45,46,49,50,55,56,57,58,63,64,65,66,71,72,73,74,77,78,83,84" \\
6303 \\
6304 \#Hesse planes:
6305 \#7,8,11,13,14,16,17,19,28,29,32,34,35,37,38,40,42,43,44,45,47,48,52,54,56,57,60,6 \\
1,63,64,65,66,68,69,73,75,77,78,81,82 \\
6306 \\
6307 HIRSCHFELD_SURFACE_HESSE_PLANES="7,8,11,13,14,16,17,19,28,29,32,34,35,37,38,40,42, \\
43,44,45,47,48,52,54,56,57,60,61,63,64,65,66,68,69,73,75,77,78,81,82" \\
6308 \\
6309 Hirschfeld_surface.get_incidence_matrix.40.40: \\
6310 \> $(ORBITER) -v 3 \ \\
6311 \> \> -define points -vector -dense $(HIRSCHFELD_SURFACE_POINTS_OFF) -end \ \\
6312 \> \> -define planes -vector -dense $(HIRSCHFELD_SURFACE_HESSE_PLANES) -end \ \\
6313 \> \> -define F -finite_field -q 4 -end \ \\
6314 \> \> -define P -projective_space -n 3 -field F -v 0 -end \ \\
6315 \> \> -with P -do \ \\
643
\end{verbatim}
6316 ▷ ▷ -projective_space_activity \\
6317 ▷ ▷ ▷ -restricted_incidence_matrix 1 3 points planes "H_incma_40_40" \\
6318 ▷ ▷ ▷ -end \\
6319 6320 6321 6322 Hirschfeld_surface_incma_40_40_c:
6323 ▷ $(ORBITER) -v 10 \\
6324 ▷ ▷ -draw_incidence_structure_description \\
6325 ▷ ▷ ▷ -width 60 -with 10 6 -end \\
6326 ▷ ▷ -define C -combinatorial_objects \\
6327 ▷ ▷ ▷ -file_of_incidence_geometries H_incma_40_40.inc 40 40 480 \\
6328 ▷ ▷ ▷ -end \\
6329 ▷ ▷ ▷ -with C -do \\
6330 ▷ ▷ ▷ ▷ -combinatorial_object_activity \\
6331 ▷ ▷ ▷ ▷ -canonical_form \\
6332 ▷ ▷ ▷ ▷ ▷ -classification_prefix H_incma_40_40 \\
6333 ▷ ▷ ▷ ▷ ▷ -label H_incma_40_40 \\
6334 ▷ ▷ ▷ ▷ ▷ -save_agoo \\
6335 ▷ ▷ ▷ ▷ ▷ -save_transversal \\
6336 ▷ ▷ ▷ ▷ ▷ -end \\
6337 ▷ ▷ ▷ ▷ ▷ -report \\
6338 ▷ ▷ ▷ ▷ ▷ ▷ -prefix H_incma_40_40 \\
6339 ▷ ▷ ▷ ▷ ▷ ▷ -export_flag_orbits \\
6340 ▷ ▷ ▷ ▷ ▷ ▷ -show_incidence_matrices \\
6341 ▷ ▷ ▷ ▷ ▷ ▷ -export_group_GAP \\
6342 ▷ ▷ ▷ ▷ ▷ ▷ -end \\
6343 ▷ ▷ ▷ ▷ ▷ -end \\
6344 ▷ ▷ ▷ ▷ ▷ $(ORBITER) -v 2 -draw_matrix \\
6345 ▷ ▷ ▷ ▷ ▷ -input_csv_file H_incma_40_40_object0_TDA_flag_orbits.csv \\
6346 ▷ ▷ ▷ ▷ ▷ -secondary_input_csv_file H_incma_40_40_object0_TDA.csv \\
6347 ▷ ▷ ▷ ▷ ▷ -box_width 32 -bit_depth 24 \\
6348 ▷ ▷ ▷ ▷ ▷ -end \\
6349 ▷ ▷ ▷ ▷ ▷ pdflatex H_incma_40_40_classification.tex \\
6350 ▷ ▷ ▷ ▷ ▷ open H_incma_40_40_classification.pdf \\
6351 6352 6353 6354 surface_7_0:
6355 ▷ $(ORBITER) -v 3 \\
6356 ▷ ▷ -define F -finite_field -q 7 -end \\
6357 ▷ ▷ -define P -projective_space -n 3 -field F -v 0 -end \\
6358 ▷ ▷ -define S -cubic_surface -space P -catalogue 0 -end \\
6359 ▷ ▷ ▷ -with S -do \\
6360 ▷ ▷ ▷ -cubic_surface_activity \\
6361 ▷ ▷ ▷ ▷ -report \\
6362 ▷ ▷ ▷ ▷ ▷ -report_with_group \\

644
Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_general_F7:

Family_ge...
surface_8_0_catalogue:
\$\text{ORBITER} -v 3 \$

\$\text{define F -finite_field -q 8 -end}$
\$\text{define P -projective_space -n 3 -field F -v 0 -end}$
\$\text{define S8_0 -cubic_surface -space P -catalogue 0 -end}$
\$\text{with S8_0 -do}$
\$\text{-cubic_surface_activity}$

\$\text{-report}$
\$\text{-report_with_group}$
\$\text{-end}$

pdflatex surface_catalogue_q8_iso0_report.tex
open surface_catalogue_q8_iso0_report.pdf
pdflatex surface_catalogue_q8_iso0_with_group.tex
open surface_catalogue_q8_iso0_with_group.pdf

# clean equation for Tekirdag-1:

surface_8_0b:
\$\text{ORBITER} -v 3 \$

\$\text{define F -finite_field -q 8 -end}$
\$\text{define P -projective_space -n 3 -field F -v 0 -end}$
\$\text{define S8_0 -cubic_surface -space P -catalogue 0}$
\$\text{-select_double_six "15,11,22,19,24,5,16,10,23,20,25,4"}$
\$\text{-select_double_six "3,2,1,0,5,4,9,8,7,6,11,10"}$
\$\text{-transform_inverse "1,4,4,0,6,0,0,6,2,0,1,7,0,4,0,0"}$
\$\text{-transform_inverse "2,0,0,0,0,2,0,0,0,2,0,1,1,2,3,0"}$
\$\text{-end}$

pdflatex surface_catalogue_q8_iso0_report.tex
open surface_catalogue_q8_iso0_report.pdf
-select_double_six "3,2,1,0,5,4,9,8,7,6,11,10" \
-transform "1,0,0,0,0,1,0,6,0,0,1,6,0,0,0,1,0" \
-transform_inverse "3,1,1,0,0,1,0,0,0,0,1,0,0,0,0,1,0" \
-transform_inverse "2,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0" \
-end \n-with S8_0 -do \n-cubic_surface_activity \
-report \n-report_with_group \n-end 
-pdflatex surface_catalogue_q8_is0_with_group.tex 
-open surface_catalogue_q8_is0_with_group.pdf 

# writes tangents.txt 

# 13.0 has 4 Eckardt points 
# 13.1 has 6 Eckardt points 
# 13.2 has 9 Eckardt points 
# 13.3 has 18 Eckardt points 

Eckardt_13: 
$(ORBITER) -v 3 \n-define F -finite_field -q 13 -end \n-define P -projective_space -n 3 -field F -v 0 -end \n-define Eckardt_3.1 -cubic_surface -space P -family_Eckardt 3 1 -end \n-with Eckardt_3.1 -do \n-cubic_surface_activity \
-report \n-report_with_group \n-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 \n-define F -finite_field -q 13 -end \n-define P -projective_space -n 3 -field F -v 0 -end \n-define S13_0 -cubic_surface -space P -catalogue 0 -end \n
# clean equation for Tekirdag-2:

surface_16_0:

$\text{define } F \text{ -finite_field -q 16 -end }$

$\text{define } P \text{ -projective_space -n 3 -field } F \text{ -v 0 -end }$

$\text{define } S_{16} \text{ -cubic_surface -space } P \text{ -catalogue 0 }$

$\text{transform } "1,0,0,0,1,0,12,0,0,0,0,1,0"$

$\text{transform } "15,11,4,0,0,12,0,0,12,0,0,0,0,0,1,3"$

$\text{end }$

$\text{with } S_{16} \text{ -do }$

# rank of lines ( 66591, 26737, 4093, 69904, 28376, 26208, 5 847, 396, 32230, 529, 30293, 70668, 2178, 261, 28666, 8575, 105, 31694, 0, 51784, 25209, 22193, 49862, 274 )

# Rank of points on Klein quadric: ( 29181, 4677, 29950, 33, 62496, 429, 1, 9205, 37, 29964, 29364, 21501, 4656, 54735, 5425, 30105, 754, 6680, 13354, 758, 30106, 0, 29209, 48736, 25595, 33780, 4657 )

# ai: 29181, 4677, 29950, 33, 62496, 429

# bi: 1, 9205, 37, 29964, 29364, 21501
6546  G13_8:
6547  > $(ORBITER) -v 3 \
6548  > -define F -finite_field -q 8 -end \
6549  > -define P -projective_space -n 3 -field F -v 0 -end \
6550  > -define T1 -cubic_surface -space P -family_G13 2 -end \
6551  > -with T1 -do \ 
6552  > -cubic_surface_activity \
6553  > -report \ 
6554  > -report_with_group \ 
6555  > -end
6556  > pdflatex surface_family_G13_q8_a2_with_group.tex
6557  > open surface_family_G13_q8_a2_with_group.pdf
6558
6559
6560  F13_8:
6561  > $(ORBITER) -v 3 \
6562  > -define F -finite_field -q 8 -end \
6563  > -define P -projective_space -n 3 -field F -v 0 -end \
6564  > -define T1 -cubic_surface -space P -family_F13 2 -end \
6565  > -with T1 -do \ 
6566  > -cubic_surface_activity \
6567  > -report \ 
6568  > -report_with_group \ 
6569  > -end
6570  > pdflatex surface_family_F13_q8_a2_with_group.tex
6571  > open surface_family_F13_q8_a2_with_group.pdf
6572
6573
6574
6575
6576  # Tekirdag-2:
6577
6578  F13_16:
6579  > $(ORBITER) -v 3 \
6580  > -define F -finite_field -q 16 -end \
6581  > -define P -projective_space -n 3 -field F -v 0 -end \
6582  > -define T2 -cubic_surface -space P -family_F13 2 -end \
6583  > -with T2 -do \ 
6584  > -cubic_surface_activity \
6585  > -report \ 
6586  > -report_with_group \ 
6587  > -end
6588  > pdflatex surface_family_F13_q16_a2_with_group.tex
6589  > open surface_family_F13_q16_a2_with_group.pdf
6590
6591
6592  # Tekirdag-3:
F13_32:

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

-define F -finite_field -q 32 -end

-define P -projective_space -n 3 -field F -v 0 -end

-define T3 -cubic_surface -space P -family_F13 2 -end

-with T3 -do

-cubic_surface_activity

-report

-report_with_group

-end

pdflatex surface_family_F13_q32_a2_with_group.tex

open surface_family_F13_q32_a2_with_group.pdf

# Kapadokya-1:

F13_64a:

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

-define F -finite_field -q 64 -end

-define P -projective_space -n 3 -field F -v 0 -end

-define K1 -cubic_surface -space P -family_F13 2 -end

-with K1 -do

-cubic_surface_activity

-report

-report_with_group

-end

# Kapadokya-2:

F13_64b:

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

-define F -finite_field -q 64 -end

-define P -projective_space -n 3 -field F -v 0 -end

-define K2 -cubic_surface -space P -family_F13 18 -end

-with K2 -do

-cubic_surface_activity

-report

-report_with_group

-end

Colorado1:

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

-define F -finite_field -q 128 -end

-define P -projective_space -n 3 -field F -v 0 -end
define CO-1 -cubic_surface -space P -catalogue 0 \
-define CO-1 -cubic_surface_activity \n-with CO-1 -do \n-cubic_surface_activity \n-report \n-report_with_group \n-end 

# recognize the arcs from Colorado-1,2,3: 

Colorado2: 

define F -finite_field -q 128 -end 
.define P -projective_space -n 3 -field F -v 0 -end 
.define CO-2 -cubic_surface -space P -catalogue 926 \
-transform_inverse "1,0,0,0,0,1,0,96,0,0,1,96,0,0,0,1,0" \
-end 
-with CO-2 -do \n-cubic_surface_activity \n-report \n-report_with_group \n-end 

Colorado3: 

define F -finite_field -q 128 -end 
.define P -projective_space -n 3 -field F -v 0 -end 
.define CO-3 -cubic_surface -space P -catalogue 928 \
-transform_inverse "1,0,0,0,0,1,0,32,0,0,1,32,0,0,0,1,0" \
-end 
-with CO-3 -do \n-cubic_surface_activity \n-report \n-report_with_group \n-end 

# Colorado-1: 

F13_128a: 

define F -finite_field -q 128 -end 
.define P -projective_space -n 3 -field F -v 0 -end 

651
define CO-1 -cubic_surface -space P -family_F13 2 -end \
-with CO-1 -do \n-cubic_surface_activity \n-report \n-report_with_group \n-end 

# Colorado-2: 
F13_128b: 
$ (ORBITER) -v 3 \n-define F -finite_field -q 128 -end \n-define P -projective_space -n 3 -field F -v 0 -end \n-define CO-2 -cubic_surface -space P -family_F13 6 -end 
-with CO-2 -do \n-cubic_surface_activity \n-report \n-report_with_group \n-end 

# Colorado-3: 
F13_128c: 
$ (ORBITER) -v 3 \n-define F -finite_field -q 128 -end \n-define P -projective_space -n 3 -field F -v 0 -end \n-define CO-3 -cubic_surface -space P -family_F13 14 -end 
-with CO-3 -do \n-cubic_surface_activity \n-report \n-report_with_group \n-end 

move_two_lines: 
$ (ORBITER) -v 5 \n-define F -finite_field -q 8 -end \n-with F -do -finite_field_activity \n-move_two_lines_in_hyperplane_stabilizer \n-65 4680 72 657 \n-end 

F_alpha_beta_gamma_delta: 
$ (ORBITER) -v 3 \n
define F -finite_field -q 7 -end \ 
with F -do -finite_field_activity \ 
parse_and_evaluate \ 
"F_alpha_beta_gamma_delta" "x0,x1,x2,x3" \ 
$(F_ALPHA_BETA_GAMMA_DELTA) \ 
"alpha=2, beta=3, gamma=4, delta=5" \ 
end 
dot -Tpng F_alpha_beta_gamma_delta.gv >F_alpha_beta_gamma_delta.png

F_abcd_Eckardt_q31:
$\texttt{ORBITER} -v 3 \ 
define F -finite_field -q 31 -end \ 
define P -projective_space -n 3 -field F -v 0 -end \ 
define F_abcd -cubic_surface -space P \ 
by_equation "F_abcd" \ 
$(F_abcd_eqn) \ 
"a=2,b=30,c=30,d=2" \ 
"d=2,b=30,c=30,d=2\" \ 
end \ 
with F_abcd -do \ 
cubic_surface_activity \ 
-report \ 
end 

surface_F_abcd:
SURFACE $\texttt{ORBITER} -v 3 \ 
define F -finite_field -q 7 -end \ 
with F -do \ 
finite_field_activity \ 
parse_and_evaluate "Fabcd" "X0,X1,X2,X3" \ 
$(F_abcd_eqn) "a=4,b=2,c=2,d=4" \ 
end 

#dot -Tpng F_alpha_beta_gamma_delta.gv >F_alpha_beta_gamma_delta.png

F_abcd_sweep_4.27.q7:
$\$(\text{ORBITER}) -v 3 \$

-define F -finite_field -q 7 -end \
-define P -projective_space -n 3 -field F -v 0 -end \
-with P -do \
-projective_space_activity \
-sweep 4.27 sweep 4.27.q7 -q 7 -by_equation "F_{abcd}" \
-"\$F_{\{a,b,c,d\}}" "X0,X1,X2,X3" \
-define F -finite_field -q 7 -end \
-projective_space_activity \
-sweep 4.27 q7 -by_equation "F_{abcd}" \
-"\$F_{\{a,b,c,d\}}" "X0,X1,X2,X3" \
-end \
-end

\begin{align*}
F_{\alpha\beta\gamma\delta} & \text{ sweep 4} \\
\text{with } F_{\alpha\beta\gamma\delta} & \text{ do } \\
\text{cubic_surface_activity } \\
\text{report } \\
\text{report with group } \\
\end{align*}

# cubic surfaces with 15 lines:

\begin{align*}
F_{\alpha\beta\gamma\delta} & \text{ sweep 4} \\
\text{with } F_{\alpha\beta\gamma\delta} & \text{ do } \\
\text{cubic_surface_activity } \\
\text{report } \\
\text{report with group } \\
\end{align*}

\begin{align*}
\text{pdflatex surface_equation\_F_{\alpha\beta\gamma\delta}q7\_report.tex} \\
\text{open surface_equation\_F_{\alpha\beta\gamma\delta}q7\_report.pdf} \\
\text{pdflatex surface_equation\_F_{\alpha\beta\gamma\delta}q7\_with\_group.tex} \\
\text{open surface_equation\_F_{\alpha\beta\gamma\delta}q7\_with\_group.pdf} \\
\end{align*}
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) "\DF_{\{\alpha,\beta,\gamma,\delta}\}D" \\
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 -n 3 -field F -v 0 -end \\
-define S -cubic_surface -space P \\
-by_equation "F_alpha_beta_gamma_delta" \\
"\DF_{\{alpha,\beta,\gamma,\delta}\}D" "x0,x1,x2,x3" \\
$(F\alpha\beta\gamma\delta) "\DF_{\{\alpha,\beta,\gamma,\delta}\}D" \\
alpha=6,\beta=4,\gamma=2,\delta=2" \\
"\D\alpha=6,\beta=4,\gamma=2,\delta=2\D" \\
-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 -n 3 -field F -v 0 -end \\
-with P -do \\
-projective_space_activity \\
-sweep_4_15_lines sweep_4_q7 -q 7 \\
-by_equation \\
"F_alpha_beta_gamma_delta" \\
"\DF_{\{alpha,\beta,\gamma,\delta}\}D" \\
"x0,x1,x2,x3" \\

# 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:

F alpha beta gamma delta -v 2 \n
-define F -finite_field -q 49 -end \n
-define P -projective_space -n 3 -field F -v 0 -end \n
-with P -do \n
-projective_space_activity \n
-classify_surfaces_with_double_sixes Surf -W -end \n
-end \n
-with Surf -do \n
-classification_of_cubic_surfaces_with_double_sixes_activity \n
-recognize \n
-q 49 \n
-by_equation "F alpha beta gamma delta" \n
"DF\{alpha,\beta,\gamma,\delta\}\{x0,x1,x2,x3\}" \n
$(F ALPHA_BETA_GAMMA_DELTA) \n
"alpha=2,\beta=1,\gamma=1,\delta=2" \n
-end \n
-end \n
-end

surf49 recognize:

$ (ORBITER) -v 3 \n
-define F -finite_field -q 49 -end \n
-define P -projective_space -n 3 -field F -v 0 -end \n
-with P -do \n
-projective_space_activity \n
-classify_surfaces_with_double_sixes Surf27 -W -end \n
-end \n
-with Surf27 -do \n
-classification_of_cubic_surfaces_with_double_sixes_activity \n

recognize 
-q 49 
-by_coefficients "2,5,1,16,4,10,1,18,4,8,4,9,3,11,4,14" 
-end 
-end 
-end 
-print_symbols 

McKean_15lines_q7:

$\text{(ORBITER) -v 3 \n-define F -finite_field -q 7 -end \n-define P -projective_space -n 3 -field F -v 0 -end \n-define S -cubic_surface -space P -by_coefficients $(\text{SURFACE_MCKEAN_15_LINES})$ -end \n-with S -do \n-cubic_surface_activity \n-report \n-end \n}$

#pdflatex surface by_coefficients_q7_report.tex
#open surface by_coefficients_q7_report.pdf

# has 4 Eckardt points

F_4.4.3.3_q7:

$\text{(ORBITER) -v 3 \n-define F -finite_field -q 7 -end \n-define P -projective_space -n 3 -field F -v 0 -end \n-define S -cubic_surface -space P \n-by_equation \n"F.alpha_beta_gamma_delta" \n"\\DF_{\{alpha,\beta,\gamma,\delta\}}D" \n"x0,x1,x2,x3" \n$(F\_\alpha\_\beta\_\gamma\_\delta) \n"alpha=4,\beta=4,\gamma=3,\delta=3" \n"\D\alpha=4,\beta=4,\gamma=3,\delta=3\D" \n-end \n-with S -do \n-cubic_surface_activity \n-report \n-end \n}$

#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
# Section 7.2: Cubic Surfaces and Quartic Curves

SECTION_CUBIC_SURFACES_AND_QUARTIC_CURVES:

quartic_curve_9.0_report:

```
$ (ORBITER) -v 3 \
  -define F -finite_field -q 9 -end \
  -define P -projective_space -n 2 -field F -v 0 -end \
  -define C -quartic_curve -space P -catalogue 0 -end \
  -with C -do \
  -quartic_curve_activity \ 
  -report \ 
  -end \
```

open quartic_curve_catalogue_q9_iso0_report.pdf

quartic_curve_13.0_report:

```
$ (ORBITER) -v 3 \
  -define F -finite_field -q 13 -end \
  -define P -projective_space -n 2 -field F -v 0 -end \
  -define C -quartic_curve -space P -catalogue 0 \
  -transform "10,4,1,11,5,11,4,1,1" \
  -transform_inverse "9,1,0,12,9,0,2,10,11" \ 
  -end \
  -with C -do \ 
  -quartic_curve_activity \ 
  -report \ 
  -end \
  -with C -do \ 
  -quartic_curve_activity \ 
  -extract_orbit_on_bitangents_by_length 4 \ 
  -end \
```

open quartic_curve_catalogue_q13_iso0_report.pdf
PG_2.13_rank_lines:
  $(ORBITER) -v 2 \-
  -define v -vector -format 4 \-
  -dense "1,0,0, 0,1,0, 1,0,0, 0,0,1, 1,1,1, 0,1,0, 1,1,1, 0,0,1" \-
  -end \-
  -define F -finite_field -q 23 -end \-
  -define P -projective_space -n 2 -field F -v 0 -end \-
  -with P -do \-
  -projective_space_activity \-
  -rank_lines_in_PG v \-
  -end \-

PG_2.13_orbits_on_lines:
  $(ORBITER) -v 5 \-
  -orbiter_path $(ORBITER_PATH) \-
  -define Control -poset_classification_control \-
  -problem_label PGL_3_13 \-
  -depth 4 \-
  -report -end \-
  -draw_poset \-
  -draw_options -radius 200 -end \-
  -recognize "170, 111, 140, 2" \-
  -recognize "0,23,24,47" \-
  -end \-
  -define G -linear_group -PGL 3 13 -end \-
  -define G_on_lines -modified_group -from G \-
  -on_k_subspaces 2 \-
  -end \-
  -define Orb -orbits -group G_on_lines \-
  -on_subsets 4 Control \-
  -end \-
  #pdflatex PGL_3_13.poset.tex \-
  #open PGL_3_13.poset.pdf \-

# stabilizer of {0,23,24,47} \-
#1,0,0,7,9,0,9,5,3, \-
#1,3,0,1,12,0,10,9,2, \-
#1,1,11,7,9,6,10,5,2, \-
#1,4,11,12,1,8,10,4,2, \-

quartic_curve_13.1_report:
  $(ORBITER) -v 3 \-
  -define F -finite_field -q 13 -end \-


nb_quartic_curves_q19=14

quartic_curves_19_report:

$(ORBITER) -v 3 \n-define F -finite_field -q 19 -end \n-define P -projective_space -n 2 -field F -v 0 -end \n-loop L 0 $(nb_quartic_curves_q19) 1 \n-define C -quartic_curve -space P -catalogue %L -end \n-with C -do \n-define quartic_curve_activity \n-report \n-end \n-end_loop

pdflatex quartic_curve_catalogue_q19_iso0_report.tex

pdflatex quartic_curve_catalogue_q19_iso1_report.tex

pdflatex quartic_curve_catalogue_q19_iso2_report.tex

pdflatex quartic_curve_catalogue_q19_iso3_report.tex

pdflatex quartic_curve_catalogue_q19_iso4_report.tex

pdflatex quartic_curve_catalogue_q19_iso5_report.tex

pdflatex quartic_curve_catalogue_q19_iso6_report.tex

pdflatex quartic_curve_catalogue_q19_iso7_report.tex

pdflatex quartic_curve_catalogue_q19_iso8_report.tex

pdflatex quartic_curve_catalogue_q19_iso9_report.tex

pdflatex quartic_curve_catalogue_q19_iso10_report.tex

pdflatex quartic_curve_catalogue_q19_iso11_report.tex

pdflatex quartic_curve_catalogue_q19_iso12_report.tex

pdflatex quartic_curve_catalogue_q19_iso13_report.tex

#open quartic_curve_catalogue_q19_iso4_report.pdf

surface_4.0_quartic_curves:

$(ORBITER) -v 3 \n-define F -finite_field -q 4 -end \n-define P -projective_space -n 3 -field F -v 0 -end \n-define S4_0 -cubic_surface -space P -catalogue 0 -end \n-with S4_0 -do \n-cubic_surface_activity \n
NB CUBIC SURFACES Q7=1

quartic_curves_q7:

$(ORBITER_PATH)orbiter.out -v 3 \
-board -list_arguments \
-board -draw_options -end \
-board -define F -finite_field -q 7 -end \
-board -define P -projective_space -n 3 -field F -end \
-board -loop L 0 $(NB_CUBIC_SURFACES_Q7) 1 \
-board -define S_%L -cubic_surface -space P -catalogue %L -end \
-board -end_loop \
-board -print_symbols \
-board -loop L 0 $(NB_CUBIC_SURFACES_Q7) 1 \
-board -with S_%L -do \
-board -cubic_surface_activity \
-board -export_all_quartic_curves \
-board -end \
-board -end_loop \
-board -print_symbols

NB CUBIC_SURFACES_Q13=4

quartic_curves_q13:

$(ORBITER_PATH)orbiter.out -v 3 \
-board -list_arguments \
-board -draw_options -end \
-board -define F -finite_field -q 13 -end \
-board -define P -projective_space -n 3 -field F -end \
-board -loop L 0 $(NB_CUBIC_SURFACES_Q13) 1 \
-board -define S_%L -cubic_surface -space P -catalogue %L -end \
-board -end_loop \
-board -print_symbols

661
quartic_curves_q13_combine:

$($ORBITER_PATH)orbiter.out -v 3 \n-csv_file_concatenate_from_mask $(NB_CUBIC_SURFACES_Q13) \n-surface_catalogue_q13_iso%ld_quartics.csv \n-quartics_q13.csv

quartic_curves_q13_classify:

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

quartic_curves_q19:

$($ORBITER_PATH)orbiter.out -v 3 \n-list_arguments \n-draw_options -end \n-define F -finite_field -q 19 -end \n-define P -projective_space -n 3 -field F -end \n-loop L 0 $(NB_CUBIC_SURFACES_Q19) 1 \n-define S.%L -cubic_surface -space P -catalogue %L -end \n-end_loop \n-print_symbols \n-loop L 0 $(NB_CUBIC_SURFACES_Q19) 1 \n-with S.%L -do \n-cubic_surface_activity \n-export_all_quartic_curves

NB_CUBIC_SURFACES_Q19=10
7203 ▶ ▶ ▶ -end \n7204 ▶ ▶ -end_loop \n7205 ▶ ▶ -print_symbols
7206
7207 7208 quartic_curves_q19_combine:
7209 ▶ $(ORBITER_PATH)orbirter.out -v 3 \n7210 ▶ ▶ -csv_file_concatenate_from_mask $(NB_CUBIC_SURFACES_Q19) \n7211 ▶ ▶ ▶ surface_catalogue_q19_iso%ld_quartics.csv \n7212 ▶ ▶ ▶ quartics_q19.csv
7213
7214 7215 7216 quartic_curves_q19_classify:
7217 ▶ $(ORBITER) -v 3 \n7218 ▶ ▶ -list_arguments \n7219 ▶ ▶ ▶ -define F -finite_field -q 19 -end \n7220 ▶ ▶ ▶ ▶ -define P -projective_space -n 2 -field F -v 0 -end \n7221 ▶ ▶ ▶ ▶ -with P -do \n7222 ▶ ▶ ▶ ▶ -projective_space_activity \n7223 ▶ ▶ ▶ ▶ ▶ -classify_quartic_curves_with_substructure \n7224 ▶ ▶ ▶ ▶ ▶ ▶ quartics_q19.csv \n7225 ▶ ▶ ▶ ▶ ▶ ▶ 1 4 4 quartics_q19.csv \n7226 ▶ ▶ ▶ ▶ -end \n7227 ▶ ▶ ▶ -print_symbols
7228
7229 7230 7231 7232 7233 7234 7235 # Section 7.3: Classification of Cubic Surfaces with 27 lines
7236
7237 7238 SECTION_CLASSIFICATION_OF_CUBIC_SURFACES_WITH_27_LINES:
7239
7240
7241 7242 7243 surface_classify_q4:
7244 ▶ $(ORBITER) -v 5 \n7245 ▶ ▶ -define F -finite_field -q 4 -end \n7246 ▶ ▶ -define P -projective_space -n 3 -field F -v 0 -end \n7247 ▶ ▶ -with P -do \n7248 ▶ ▶ -projective_space_activity \n7249 ▶ ▶ ▶ -classify_surfaces_with_double_sixes Surf27 -W -end \n
663
surface_classify_q4.arc.lifting_two_lines:
    $(ORBITER) -v 10 \
    -define F -finite_field -q 4 -end \
    -define P -projective_space -n 3 -field F -v 0 -end \
    -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

surface_classify_q7:
    $(ORBITER) -v 5 \
    -define F -finite_field -q 7 -end \
    -define P -projective_space -n 3 -field F -v 0 -end \
    -with P -do \
    -projective_space_activity \
    -classify_surfaces_with_double_sixes 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 \

664
surface_classify_q13:

surface_recognize_q7_abcd_2_3_3_4:
7344  ▶ ▶ ▶  -classify_surfaces_with_double_sixes Surf -W -end \n7345  ▶ ▶   -end \n7346  ▶ ▶   -with Surf -do \n7347  ▶ ▶ ▶  -classification_of_cubic_surfaces_with_double_sixes_activity \n7348  ▶ ▶ ▶  -recognize \n7349  ▶ ▶ ▶  ▶  ▶ -q 7 \n7350  ▶ ▶ ▶  ▶  ▶ -family_general_abcd 2 3 3 4 \n7351  ▶ ▶ ▶  ▶  ▶ -end \n7352  ▶ ▶ ▶  ▶ -end \n7353  ▶ ▶   -end \n7354
7355
7356  surface_isomorph_16: \n7357  ▶ $(ORBITER) -v 3 \n7358  ▶ ▶   -define F -finite_field -q 16 -end \n7359  ▶ ▶   -define P -projective_space -n 3 -field F -v 0 -end \n7360  ▶ ▶   -with P -do \n7361  ▶ ▶   -projective_space_activity \n7362  ▶ ▶ ▶  -classify_surfaces_with_double_sixes Surf27 -W -end \n7363  ▶ ▶   -end \n7364  ▶ ▶   -with Surf27 -do \n7365  ▶ ▶ ▶  -classification_of_cubic_surfaces_with_double_sixes_activity \n7366  ▶ ▶ ▶  -isomorphism_testing \n7367  ▶ ▶ ▶  ▶  ▶ -q 16 -by.coefficients \n7368  ▶ ▶ ▶  ▶  ▶ "1,5,1,8,1,9,1,10,1,11,1,12,6,14,6,15,7,18,7,19" -end \n7369  ▶ ▶ ▶  ▶  ▶ -q 16 -by.coefficients \n7370  ▶ ▶ ▶  ▶  ▶ "13,6,3,8,3,11,13,13,1,19" -end \n7371  ▶ ▶ ▶   -end \n7372  ▶ ▶   -end \n7373  ▶ ▶ ▶ -print_symbols \n7374
7375
7376
7377  # 1 min 8 sec on Mac from scratch (with all data files removed) \n7378
7379
7380  surface_recognize_8: \n7381  ▶ $(ORBITER) -v 3 \n7382  ▶ ▶   -define F -finite_field -q 8 -end \n7383  ▶ ▶   -define P -projective_space -n 3 -field F -v 0 -end \n7384  ▶ ▶   -with P -do \n7385  ▶ ▶   -projective_space.activity \n7386  ▶ ▶ ▶  -classify_surfaces_with_double_sixes Surf27 -W -end \n7387  ▶ ▶   -end \n7388  ▶ ▶   -with Surf27 -do \n7389  ▶ ▶ ▶  -classification_of_cubic_surfaces_with_double_sixes_activity \n7390  ▶ ▶ ▶  -recognize \n
666
7400 surface_recognize_F13.q4:
7401 $(ORBITER) -v 3 \n7402 -define F -finite_field -q 4 -end \n7403 -define P -projective_space -n 3 -field F -v 0 -end \n7404 -with P -do \n7405 -projective_space_activity \n7406 -classify_surfaces_with_double_sixes Surf27 -W -end \n7407 -end \n7408 -with Surf27 -do \n7409 -classification_of_cubic_surfaces_with_double_sixes_activity \n7410 -identify_F13 \n7411 -end \n7412 -print_symbols
7413
7414 surface_sweep_Cayley_13:
7415 $(ORBITER) -v 3 \n7416 -define F -finite_field -q 13 -end \n7417 -define P -projective_space -n 3 -field F -v 0 -end \n7418 -with P -do \n7419 -projective_space_activity \n7420 -classify_surfaces_with_double_sixes Surf27 -W -end \n7421 -end \n7422 -with Surf27 -do \n7423 -classification_of_cubic_surfaces_with_double_sixes_activity \n7424 -sweep_Cayley \n7425 -end \n7426 -print_symbols
7427
7428
7429
7430 F_sweep_15.q7:
7431 $(ORBITER) -v 20 \n7432 -define F -finite_field -q 7 -end \n7433 -define P -projective_space -n 3 -field F -v 0 -end \n7434 -with P -do \n7435 -projective_space_activity \n7436 -sweep_4_15_lines sweep_4_15_lines_q7 -q 7 \n7437
-by_equation "F_alpha_beta_gamma_delta" \\
"DF{\alpha,\beta,\gamma,\delta}D" "x0,x1,x2,x3" \\
$F_{\alpha,\beta,\gamma,\delta}$ \\
"DF=2,\beta=1,\gamma=2,\delta=3" \\
"DF=2,\beta=1,\gamma=2,\delta=3\D" \\
-end \\
-end

0:29

# Section 7.5: Cubic Surfaces of Dickson type

SECTION_CUBIC_SURFACES_DICKSON:

orbits_cubic_surfaces_q3:

$(ORBITER) -v 4 \\
-define G -linear_group -PGL 4 3 -end \\
-define Orb -orbits -group G \\
-on_polynomials 3 \\
-end

# 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 \\
-define Orb -orbits -group G \\
-on_polynomials 3 \\
-end

#pdflatex poly_orbits_d3_n2_q2.tex
7485 # open poly_orbits_d3_n2_q2.pdf
7486
7487 orbits_cubic_curves_q3:
7488 $\text{\texttt{(ORBITER)} } -v 4 \$
7489 $\text{\texttt{define G}}$
7490 $\text{\texttt{-linear_group -PGL 3 3}}$
7491 $\text{\texttt{-end}}$
7492 $\text{\texttt{define Orb -orbits -group G}}$
7493 $\text{\texttt{on polynomials 3}}$
7494 $\text{\texttt{-end}}$
7495 $\text{\texttt{#pdflatex poly_orbits_d3_n2_q3.tex}}$
7496 $\text{\texttt{#open poly_orbits_d3_n2_q3.pdf}}$
7497 7498 7499 # compute and analyze properties over F2
7500
7501 poly_orbits_d3_n3_q2_F2.csv: poly_orbits_d3_n3_q2.csv
7502 $\text{\texttt{(ORBITER)} } -v 4 \$
7503 $\text{\texttt{define F -finite_field -q 2 -end}}$
7504 $\text{\texttt{define P -projective_space -n 3 -field F -v 0 -end}}$
7505 $\text{\texttt{with P -do}}$
7506 $\text{\texttt{table_of_cubic_surfaces_compute_properties}}$
7507 $\text{\texttt{poly_orbits_d3_n3_q2.csv 2 0}}$
7508 $\text{\texttt{-end}}$
7509 7510
7511 Dickson_q2_analyze: poly_orbits_d3_n3_q2_F2.csv
7512 $\text{\texttt{(ORBITER)} } -v 4 \$
7513 $\text{\texttt{define F -finite_field -q 2 -end}}$
7514 $\text{\texttt{define P -projective_space -n 3 -field F -v 0 -end}}$
7515 $\text{\texttt{with P -do}}$
7516 $\text{\texttt{projective_space_activity}}$
7517 $\text{\texttt{cubic_surface_properties_analyze}}$
7518 $\text{\texttt{poly_orbits_d3_n3_q2_F2.csv 2}}$
7519 $\text{\texttt{-end}}$
7520 $\text{\texttt{pdflatex poly_orbits_d3_n3_q2_F2_report.tex}}$
7521 $\text{\texttt{open poly_orbits_d3_n3_q2_F2_report.pdf}}$
7522 7523 # compute and analyze properties over F4
7524
7525 poly_orbits_d3_n3_q2_F4.csv: poly_orbits_d3_n3_q2.csv
7526 $\text{\texttt{(ORBITER)} } -v 4 \$
7527 $\text{\texttt{define F -finite_field -q 4 -end}}$
7528 $\text{\texttt{define P -projective_space -n 3 -field F -v 0 -end}}$
7529 $\text{\texttt{with P -do}}$
7530 7531
# compute and analyze properties over $F_4$

```
Dickson_q4_analyze: poly_orbits_d3_n3_q2_F4.csv
$(ORBITER) -v 4
-define F -finite_field -q 4 -end
-define P -projective_space -n 3 -field F -v 0 -end
-with P -do
-projective_space_activity
-cubic_surface_properties_analyze
-poly_orbits_d3_n3_q2_F4.csv 2
-end

pdflatex poly_orbits_d3_n3_q2_F4_report.tex
open poly_orbits_d3_n3_q2_F4_report.pdf
```

# compute and analyze properties over $F_8$

```
poly_orbits_d3_n3_q2_F8.csv: poly_orbits_d3_n3_q2.csv
$(ORBITER) -v 4
-define F -finite_field -q 8 -end
-define P -projective_space -n 3 -field F -v 0 -end
-with P -do
-projective_space_activity
-cubic_surface_properties_analyze
-poly_orbits_d3_n3_q2_F8.csv 2
-end

pdflatex poly_orbits_d3_n3_q2_F8_report.tex
open poly_orbits_d3_n3_q2_F8_report.pdf
```

# compute and analyze properties over $F_{16}$

```
poly_orbits_d3_n3_q2_F16.csv: poly_orbits_d3_n3_q2.csv
$(ORBITER) -v 4
-define F -finite_field -q 16 -end
```

define P -projective_space -n 3 -field F -v 0 -end \\
with P -do \\
-projective_space_activity \\
-table_of_cubic_surfaces_compute_properties \\
poly_orbits_d3_n3_q2.csv 2 0 \\
-end \\
Dickson_q16.analyze: poly_orbits_d3_n3_q2_F16.csv \\
$(ORBITER) -v 4 \\
$define F -finite_field -q 16 -end \\
$define P -projective_space -n 3 -field F -v 0 -end \\
-with P -do \\
-projective_space_activity \\
-cubic_surface_properties_analyze \\
poly_orbits_d3_n3_q2_F16.csv 2 \\
-end \\
pdflatex poly_orbits_d3_n3_q2_F16_report.tex \\
open poly_orbits_d3_n3_q2_F16_report.pdf \\
Dickson_q16.analyze: poly_orbits_d3_n3_q2_F16.csv \\
$(ORBITER) -v 4 \\
$define F -finite_field -q 16 -end \\
$define P -projective_space -n 3 -field F -v 0 -end \\
-with P -do \\
-projective_space_activity \\
-cubic_surface_properties_analyze \\
poly_orbits_d3_n3_q2_F16.csv 2 \\
-end \\
pdflatex poly_orbits_d3_n3_q2_F16_report.tex \\
open poly_orbits_d3_n3_q2_F16_report.pdf 

# Section 7.6: Cubic Surfaces - ATLAS and Tables

SECTION CUBIC SURFACES ATLAS AND TABLES:

MAKE_TABLE_OF_CUBIC_SURFACES=-define \\
P -projective_space -n 3 -field F -v 0 -end \\
-with P -do \\
-projective_space_activity \\
-table_of_cubic_surfaces \\
-end \\
cubic_surfaces_tables_17: 
$(ORBITER) -v 3 \\
$define F -finite_field -q 17 -end \\
$(MAKE_TABLE_OF_CUBIC_SURFACES) \\
cubic_surfaces_tables_17: 
$(ORBITER) -v 3 \\
$csv_file_latex 1 \\
-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_SURFACE S)

$ (ORBITER) -v 3 -define F -finite_field -q 7 -end $(MAKE_TABLE_OF_CUBIC_SURFACE S)

$ (ORBITER) -v 3 -define F -finite_field -q 8 -end $(MAKE_TABLE_OF_CUBIC_SURFACE S)

$ (ORBITER) -v 3 -define F -finite_field -q 9 -end $(MAKE_TABLE_OF_CUBIC_SURFACE S)

$ (ORBITER) -v 3 -define F -finite_field -q 11 -end $(MAKE_TABLE_OF_CUBIC_SURFACE S)

$ (ORBITER) -v 3 -define F -finite_field -q 13 -end $(MAKE_TABLE_OF_CUBIC_SURFACE S)

$ (ORBITER) -v 3 -define F -finite_field -q 16 -end $(MAKE_TABLE_OF_CUBIC_SURFACE S)

$ (ORBITER) -v 3 -define F -finite_field -q 17 -end $(MAKE_TABLE_OF_CUBIC_SURFACE S)

$ (ORBITER) -v 3 -define F -finite_field -q 19 -end $(MAKE_TABLE_OF_CUBIC_SURFACE S)

$ (ORBITER) -v 3 -define F -finite_field -q 23 -end $(MAKE_TABLE_OF_CUBIC_SURFACE S)

$ (ORBITER) -v 3 -define F -finite_field -q 25 -end $(MAKE_TABLE_OF_CUBIC_SURFACE S)

$ (ORBITER) -v 3 -define F -finite_field -q 27 -end $(MAKE_TABLE_OF_CUBIC_SURFACE S)

$ (ORBITER) -v 3 -define F -finite_field -q 29 -end $(MAKE_TABLE_OF_CUBIC_SURFACE S)

$ (ORBITER) -v 3 -define F -finite_field -q 31 -end $(MAKE_TABLE_OF_CUBIC_SURFACE S)

$ (ORBITER) -v 3 -define F -finite_field -q 32 -end $(MAKE_TABLE_OF_CUBIC_SURFACE S)

$ (ORBITER) -v 3 -define F -finite_field -q 37 -end $(MAKE_TABLE_OF_CUBIC_SURFACE S)

$ (ORBITER) -v 3 -define F -finite_field -q 41 -end $(MAKE_TABLE_OF_CUBIC_SURFACE S)

$ (ORBITER) -v 3 -define F -finite_field -q 43 -end $(MAKE_TABLE_OF_CUBIC_SURFACE S)

$ (ORBITER) -v 3 -define F -finite_field -q 47 -end $(MAKE_TABLE_OF_CUBIC_SURFACE S)

$ (ORBITER) -v 3 -define F -finite_field -q 49 -end $(MAKE_TABLE_OF_CUBIC_SURFACE S)
7653 $(ORBITER) -v 3 -define F -finite_field -q 53 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
7654 $(ORBITER) -v 3 -define F -finite_field -q 59 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
7655 $(ORBITER) -v 3 -define F -finite_field -q 61 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
7656 $(ORBITER) -v 3 -define F -finite_field -q 64 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
7657 $(ORBITER) -v 3 -define F -finite_field -q 67 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
7658 $(ORBITER) -v 3 -define F -finite_field -q 71 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
7659 $(ORBITER) -v 3 -define F -finite_field -q 73 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
7660 $(ORBITER) -v 3 -define F -finite_field -q 79 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
7661 $(ORBITER) -v 3 -define F -finite_field -q 81 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
7662 $(ORBITER) -v 3 -define F -finite_field -q 83 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
7663 $(ORBITER) -v 3 -define F -finite_field -q 89 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
7664 $(ORBITER) -v 3 -define F -finite_field -q 97 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
7665 $(ORBITER) -v 3 -define F -finite_field -q 101 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
7666 $(ORBITER) -v 3 -define F -finite_field -q 103 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
7667 $(ORBITER) -v 3 -define F -finite_field -q 107 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
7668 $(ORBITER) -v 3 -define F -finite_field -q 109 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
7669 $(ORBITER) -v 3 -define F -finite_field -q 113 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
7670 $(ORBITER) -v 3 -define F -finite_field -q 121 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
7671 $(ORBITER) -v 3 -define F -finite_field -q 127 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
7672 $(ORBITER) -v 3 -define F -finite_field -q 128 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

7673 cubic_surfaces_tables_latex:
7674 $(ORBITER) -v 3 -csv_file_latex 1 test.csv
7675 $(ORBITER) -v 3 -csv_file_latex 0 table_of_cubic_surfaces_q4_info.csv
7676 $(ORBITER) -v 3 -csv_file_latex 0 table_of_cubic_surfaces_q7_info.csv
7677 $(ORBITER) -v 3 -csv_file_latex 0 table_of_cubic_surfaces_q8_info.csv

673
cubic_surfaces_tables latex big:

- $(ORBITER) -v 3 -csv_file latex 0 table_of_cubic_surfaces_q9_info.csv
- $(ORBITER) -v 3 -csv_file latex 0 table_of_cubic_surfaces_q11_info.csv
- $(ORBITER) -v 3 -csv_file latex 0 table_of_cubic_surfaces_q13_info.csv
- $(ORBITER) -v 3 -csv_file latex 0 table_of_cubic_surfaces_q16_info.csv
- $(ORBITER) -v 3 -csv_file latex 0 table_of_cubic_surfaces_q17_info.csv
- $(ORBITER) -v 3 -csv_file latex 0 table_of_cubic_surfaces_q19_info.csv

#$(ORBITER) -v 3 -csv_file latex 1 quartic_curves q9_info.csv

- pdflatex quartic_curves_q13_info.tex
- open quartic_curves_q13_info.pdf
- ~/bin/tth quartic_curves q13_info.tex
- open quartic_curves q13_info.html

surface_table:

- $(ORBITER) -v 3 -make_table_of_surfaces
- pdflatex surfaces_report.tex
- open surfaces_report.pdf

surface_atlas:

- $(ORBITER) -v 3 -create_surface_atlas 97
- ~/bin/tth surface_atlas.tex

surface_reports:

- $(ORBITER) -v 3 \ -orbiter_path $(ORBITER_PATH) -create_surface_reports 4,7,8,9,11
quartic_curve_tables_19:
$\text{(ORBITER)} -v 3$
-define F -finite_field -q 19 -end 
-define P -projective_space -n 2 -field F -v 0 -end 
-with P -do 
-projective_space_activity 
-table_of_quartic_curves 
-end

quartic_curve_tables_23:
$\text{(ORBITER)} -v 3$
-define F -finite_field -q 23 -end 
-define P -projective_space -n 2 -field F -v 0 -end 
-with P -do 
-projective_space_activity 
-table_of_quartic_curves 
-end

quartic_curve_tables:
$\text{(ORBITER)} -v 3$
-define F -finite_field -q 9 -end 
-define P -projective_space -n 2 -field F -v 0 -end 
-with P -do 
-projective_space_activity 
-table_of_quartic_curves 
-end

$\text{(ORBITER)} -v 3$
-define F -finite_field -q 13 -end 
-define P -projective_space -n 2 -field F -v 0 -end 
-with P -do 
-projective_space_activity 
-table_of_quartic_curves 
-end

$\text{(ORBITER)} -v 3$
-define F -finite_field -q 17 -end 
-define P -projective_space -n 2 -field F -v 0 -end 
-with P -do 
-projective_space_activity 
-table_of_quartic_curves 
-end

$\text{(ORBITER)} -v 3$
-define F -finite_field -q 19 -end 
-define P -projective_space -n 2 -field F -v 0 -end 
-with P -do 

-projective_space_activity \ 
-table_of_quartic_curves \ 
-end

$\text{ORBITER) -v 3 \ 
-def F -finite_field -q 23 -end \ 
-def P -projective_space -n 2 -field F -v 0 -end \ 
-with P -do \ 
-projective_space_activity \ 
-table_of_quartic_curves \ 
-end

$\text{ORBITER) -v 3 \ 
-def F -finite_field -q 25 -end \ 
-def P -projective_space -n 2 -field F -v 0 -end \ 
-with P -do \ 
-projective_space_activity \ 
-table_of_quartic_curves \ 
-end

$\text{ORBITER) -v 3 \ 
-def F -finite_field -q 27 -end \ 
-def P -projective_space -n 2 -field F -v 0 -end \ 
-with P -do \ 
-projective_space_activity \ 
-table_of_quartic_curves \ 
-end

$\text{ORBITER) -v 3 \ 
-def F -finite_field -q 29 -end \ 
-def P -projective_space -n 2 -field F -v 0 -end \ 
-with P -do \ 
-projective_space_activity \ 
-table_of_quartic_curves \ 
-end

$\text{ORBITER) -v 3 \ 
-def F -finite_field -q 31 -end \ 
-def P -projective_space -n 2 -field F -v 0 -end \ 
-with P -do \ 
-projective_space_activity \ 
-table_of_quartic_curves \ 
-end

#quartic_curves_q9.info.csv
#quartic_curves_q13.info.csv
#quartic_curves_q17.info.csv
#quartic_curves_q19.info.csv
#quartic_curves_q23.info.csv

676
### Chapter 8 - Ring Theory

#### Section 8.1: Polynomials over Finite Fields

# check which polynomials are irreducible and which are primitive:

```
sift_polynomials_deg3.q2:
```

```bash
$ (ORBITER) -v 2 \
```
sift polynomials deg4 q2:

poly_division:

poly_division2:

poly_gcd:

poly_mult_mod1:

poly_mult_mod2:
poly_mult_mod_F4:

$($(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" -end

$($(ORBITER) -v 2 \n-define F -finite_field -q 2 -end \n-with F -do \n-finite_field_activity \n-polynomial_mult_mod "0,1" "0,1,1" -end

mult_polynomials_2_5_7:

$($(ORBITER) -v 2 \n-define F -finite_field -q 2 -end \n-with F -do \n-finite_field_activity -mult_polynomials 5 7 -end

pdflatex polynomial_mult_5_7.tex
open polynomial_mult_5_7.pdf

polynomial_division_ranked_2_27_13:

$($(ORBITER) -v 2 \n-define F -finite_field -q 2 -end \n-with F -do \n-finite_field_activity \n-polynomial_division_ranked 27 13 \n-end

pdflatex polynomial_division_27_13.tex
open polynomial_division_27_13.pdf
mult.polynomials_2.8.15:
$\text{(ORBITER)} -v 2 \$
$\text{define } F -\text{finite field} -q 2 -\text{end} \$
$\text{with } F -\text{do} \$
$\text{finite field activity} -\text{mult.polynomials 8 15 -end} \$
pdflatex polynomial_mult_8_15.tex
open polynomial_mult_8_15.pdf

polynomial_division_ranked_2.120.25:
$\text{(ORBITER)} -v 2 \$
$\text{define } F -\text{finite field} -q 2 -\text{end} \$
$\text{with } F -\text{do} \$
$\text{finite field activity} \$
$\text{polynomial division ranked 120 25 }\$
$\text{end} \$
pdflatex polynomial_division_120_25.tex
open polynomial_division_120_25.pdf

# the answer is 5

polynomial_mult_7.7:
$\text{(ORBITER)} -v 2 \$
$\text{define } F -\text{finite field} -q 2 -\text{end} \$
$\text{with } F -\text{do} \$
$\text{finite field activity} \$
pdflatex polynomial_mult_7_7.tex
open polynomial_mult_7_7.pdf

polynomial_mult_4.6:
$\text{(ORBITER)} -v 2 \$
$\text{define } F -\text{finite field} -q 2 -\text{end} \$
$\text{with } F -\text{do} \$
$\text{finite field activity} \$
pdflatex polynomial_mult_4_6.tex
open polynomial_mult_4_6.pdf

polynomial_division_ranked_2.24.13:
$\text{(ORBITER)} -v 2 \$
$\text{define } F -\text{finite field} -q 2 -\text{end} \$
$\text{with } F -\text{do} \$

680
mult.polynomials_1024_999_997:

polynomial_division_ranked_2_349147_1033:

mult.polynomials_1024_999_997_check:

mult.polynomials_17_12:
```plaintext
pdflatex polynomial_mult_17_12.tex
open polynomial_mult_17_12.pdf

# gives 204

polynomial_division_ranked_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 \ncrc32 "123456789"

Berlekamp_matrix_crc32:
$(ORBITER) -v 2 \ndefine F -finite_field -q 2 -end \ndefine v -vector -field F -sparse 33 $(CRC32_SPARSE) -end \nwith F -do \nfinite_field_activity \nBerlekamp_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
```

682
8103  \( \text{TWO\_TO\_THE\_32\_MINUS\_2} = 4294967294 \)
8104
8105  \text{power\_mod\_inverse:}
8106 \> \$\text{(ORBITER)} -v 2 \>
8107 \> \> -define F -finite_field -q 2 -end \>
8108 \> \> -define M -vector -field F -sparse 33 \$(\text{CRC32\_SPARSE}) -end \>
8109 \> \> -define A -vector -field F -sparse 2 "1,1" -end \>
8110 \> \> -with F -do \>
8111 \> \> -finite_field_activity \>
8112 \> \> -polynomial\_power\_mod A \$(\text{TWO\_TO\_THE\_32\_MINUS\_2}) M \>
8113 \> \> -end
8114
8115  \text{INVERSE\_SPARSE=}"1,31,1,25,1,22,1,21,1,15,\
8116 1,11,1,10,1,9,1,7,1,6,1,4,1,3,1,1,1,0"
8117
8118  \#A(X)=X^{-31} + X^{-25} + X^{-22} + X^{-21} + X^{-15}
8119  \#* X^{-11} + X^{-10} + X^{-9} + X^{-7} + X^{-6} + X^{-4} + X^{-3} + X + 1
8120
8121  \text{mult\_mod\_to\_get\_one:}
8122 \> \$\text{(ORBITER)} -v 2 \>
8123 \> \> -define F -finite_field -q 2 -end \>
8124 \> \> -define M -vector -field F -sparse 33 \$(\text{CRC32\_SPARSE}) -end \>
8125 \> \> -define A -vector -field F -sparse 2 "1,1" -end \>
8126 \> \> -define B -vector -field F -sparse 33 \$(\text{INVERSE\_SPARSE}) -end \>
8127 \> \> -with F -do \>
8128 \> \> -finite_field_activity \>
8129 \> \> \> -polynomial\_mult\_mod A B M \>
8130 \> \> -end
8131
8132  \#C(X)=1
8133
8134
8135
8136
8137
8138
8139
8140  \text{Berlekamp\_matrix\_2,3:}
8141 \> \$\text{(ORBITER)} -v 2 \>
8142 \> \> -define F -finite_field -q 2 -end \>
8143 \> \> -define v -vector -field F -dense "1,1,0,1" -end \>
8144 \> \> -with F -do \>
8145 \> \> -finite_field_activity \>
8146 \> \> \> -Berlekamp\_matrix v -end
8147
8148  \# the polynomial X^3+X+1 is irreducible over GF(2) because the rank of the Berlekamp matrix is 2.
Berlekamp matrix 2.4:

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

# the polynomial $X^4+X+1$ is irreducible over GF(2) because the rank of the Berlekamp matrix is 3.

Berlekamp matrix 4.3a:

```bash
$ (ORBITER) -v 2 \n-define F -finite_field -q 4 -end \n-define v -vector -field F -dense "1,3,0,1" -end \n-with F -do \n-finite_field_activity \n-Berlekamp_matrix v -end
```

Berlekamp matrix 4.3b:

```bash
$ (ORBITER) -v 2 \n-define F -finite_field -q 4 -end \n-define v -vector -field F -dense "1,3,1,1" -end \n-with F -do \n-finite_field_activity \n-Berlekamp_matrix v -end
```

find roots a:

```bash
$ (ORBITER) -v 2 \n-define F -finite_field -q 19 -end \n-define v -vector -field F -dense "18,1,1" -end \n-with F -do \n-finite_field_activity \n-polynomial_find_roots v -end
```

find roots b:

```bash
$ (ORBITER) -v 2 \n-define F -finite_field -q 19 -end \n-define v -vector -field F -dense "1,3,1" -end \n-with F -do \n-finite_field_activity \n-polynomial_find_roots v -end
```
find_roots_c:
find_roots_d:
find_roots_e:
roots_over_F2:
roots_over_F8:
# degree and then order of the field of coefficients:

irred_2:

$\$(ORBITER) -v 3 \\
-define F -finite_field -q 2 -end \\
-with F -do \\
-finite_field_activity \\
-make_table_of_irreducible_polynomials 3 -end

pdflatex Irred_q2_d3.tex
open Irred_q2_d3.pdf

# 3 polys

irred_3:

$\$(ORBITER) -v 3 \\
-define F -finite_field -q 2 -end \\
-with F -do \\
-finite_field_activity \\
-make_table_of_irreducible_polynomials 4 -end

pdflatex Irred_q2_d4.tex
open Irred_q2_d4.pdf

# 6 polys

irred_4:

$\$(ORBITER) -v 3 \\
-define F -finite_field -q 2 -end \\
-with F -do \\
-finite_field_activity \\
-make_table_of_irreducible_polynomials 5 -end

pdflatex Irred_q2_d5.tex
open Irred_q2_d5.pdf

# 9 polys

irred_5:

$\$(ORBITER) -v 3 \\
-define F -finite_field -q 2 -end \\
-with F -do \\
-finite_field_activity \\
-make_table_of_irreducible_polynomials 6 -end

pdflatex Irred_q2_d6.tex
open Irred_q2_d6.pdf

# 9 polys

irred_7:

$\$(ORBITER) -v 3 \
8289 " define F -finite_field -q 2 -end "
8290 " with F -do "
8291 " finite_field_activity "
8292 " make_table_of_irreducible_polynomials 7 -end
8293 " pdflatex Irred_q2_d7.tex
8294 " open Irred_q2_d7.pdf
8295
8296 # 18 polys
8297
8298 irred_8.2:
8299 " $(ORBITER) -v 3 "
8300 " define F -finite_field -q 2 -end "
8301 " with F -do "
8302 " finite_field_activity "
8303 " make_table_of_irreducible_polynomials 8 -end
8304 " pdflatex Irred_q2_d8.tex
8305 " open Irred_q2_d8.pdf
8306
8307 # 30 polys
8308
8309 irred_9.2:
8310 " $(ORBITER) -v 3 "
8311 " define F -finite_field -q 2 -end "
8312 " with F -do "
8313 " finite_field_activity "
8314 " make_table_of_irreducible_polynomials 9 -end
8315 " pdflatex Irred_q2_d9.tex
8316 " open Irred_q2_d9.pdf
8317
8318 # 56 polys
8319
8320 irred_10.2:
8321 " $(ORBITER) -v 3 "
8322 " define F -finite_field -q 2 -end "
8323 " with F -do "
8324 " finite_field_activity "
8325 " make_table_of_irreducible_polynomials 10 -end
8326 " pdflatex Irred_q2_d10.tex
8327 " open Irred_q2_d10.pdf
8328
8329 # 99 polys
8330
8331 irred_2.4:
8332 " $(ORBITER) -v 3 "
8333 " define F -finite_field -q 4 -end "
8334 " with F -do "
8335 " finite_field_activity "

687
\begin{verbatim}
8336 \texttt{\textdagger \textdagger -make_table_of_irreducible_polynomials 2 -end}
8337 \texttt{pdflatex Irred_q4_d2.tex}
8338 \texttt{open Irred_q4_d2.pdf}
8339
8340 \texttt{# 6 polys}
8341
8342 \texttt{irred_3_4:}
8343 \texttt{\textdagger \textdagger \$(ORBITER) -v 6 \textbackslash}
8344 \texttt{\textdagger \textdagger \textdagger -define F -finite_field -q 4 -end \textbackslash}
8345 \texttt{\textdagger \textdagger \textdagger -with F -do \textbackslash}
8346 \texttt{\textdagger \textdagger \textdagger -finite_field_activity \textbackslash}
8347 \texttt{\textdagger \textdagger \textdagger -make_table_of_irreducible_polynomials 3 -end}
8348 \texttt{pdflatex Irred_q4_d3.tex}
8349 \texttt{open Irred_q4_d3.pdf}
8350
8351 \texttt{# 20 polys}
8352
8353
8354
8355
8356
8357
8358
8359 \texttt{search_primitive_poly_2:}
8360 \texttt{\textdagger \textdagger \$(ORBITER) -v 3 \textbackslash}
8361 \texttt{\textdagger \textdagger \textdagger -search_for_primitive_polynomial_in_range 2 2 2 10 #! grep //}
8362
8363 \texttt{# stuck in factoring 2^61-1 (which is prime)}
8364
8365 \texttt{search_primitive_poly_3:}
8366 \texttt{\textdagger \textdagger \$(ORBITER) -v 6 \textbackslash}
8367 \texttt{\textdagger \textdagger \textdagger -search_for_primitive_polynomial_in_range 3 3 2 60}
8368 \texttt{\textdagger \textdagger \textdagger}
8369
8370 \texttt{search_primitive_poly_4:}
8371 \texttt{\textdagger \textdagger \$(ORBITER) -v 6 \textbackslash}
8372 \texttt{\textdagger \textdagger \textdagger -search_for_primitive_polynomial_in_range 4 4 2 30}
8373 \texttt{\textdagger \textdagger \textdagger}
8374 \texttt{search_primitive_poly_5:}
8375 \texttt{\textdagger \textdagger \$(ORBITER) -v 6 \textbackslash}
8376 \texttt{\textdagger \textdagger \textdagger -search_for_primitive_polynomial_in_range 5 5 2 30}
8377 \texttt{\textdagger \textdagger \textdagger}
8378
8379 \texttt{search_primitive_poly_7:}
8380 \texttt{\textdagger \textdagger \$(ORBITER) -v 6 \textbackslash}
8381 \texttt{\textdagger \textdagger \textdagger -search_for_primitive_polynomial_in_range 7 7 2 20}
8382 \texttt{\textdagger \textdagger \textdagger}
\end{verbatim}
8383
8384 search_primitive_poly_8:
8385 \> $(ORBITER) -v 6 \n8386 \> \> \> -search_forPrimitive_polynomial_in_range 8 8 2 20
8387 \>
8388 search_primitive_poly_9:
8389 \> $(ORBITER) -v 6 \n8390 \> \> \> -search_forPrimitive_polynomial_in_range 9 9 2 15
8391 \>
8392 search_primitive_poly_11:
8393 \> $(ORBITER) -v 6 \n8394 \> \> \> -search_forPrimitive_polynomial_in_range 11 11 2 15
8395 \>
8396 search_primitive_poly_13:
8397 \> $(ORBITER) -v 6 \n8398 \> \> \> -search_forPrimitive_polynomial_in_range 13 13 2 15
8399 \>
8400 search_primitive_poly_degree_16:
8401 \> $(ORBITER) -v 6 \n8402 \> \> \> -search_forPrimitive_polynomial_in_range 2 2 16 16
8403 \>
8404 search_primitive_poly_32:
8405 \> $(ORBITER) -v 6 \n8406 \> \> \> -search_forPrimitive_polynomial_in_range 32 32 2 10
8407 \>
8408
8409
8410
8411
8412
8413 # Section 8.2: Multivariate Polynomials
8414
8415
8416
8417
8418 SECTION_MULTIVARIATE_POLYNOMIALS:
8419
8420
8421 CREMONA_MAP_Y0="3*y0*y0*y0*y0*y0*y2+4*y0*y0*y0*y1*y1*y2+
8422 +2*y0*y0*y0*y2*y2*y2+y0*y1*y1*y1*y1*y2+
8423 +6*y0*y1*y1*y2*y2*y2+9*y0*y2*y2*y2*y2*y2"
8424
8425 CREMONA_MAP_Y1="y0*y0*y0*y0*y0*y1+5*y0*y0*y1*y1*y1+
8426 +12*y0*y0*y1*y1*y2*y2+3*y0*y0*y1*y1*y1*y1+
8427 +5*y0*y1*y1*y2*y2*y0*y1*y2*y2*y2*y2"
8428
8429 CREMONA_MAP_Y2="10*y0*y0*y0*y0*y0*y0+11*y0*y0*y0*y0*y0*y1*y1"
CREMONA_MAP_Y3="0"

Cremona_map:

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

$\text{define F} -\text{finite field} -q 13 -\text{end} \$

$\text{define P} -\text{projective space} -n 2 -\text{field F} -v 0 -\text{end} \$

$\text{define R} -\text{polynomial ring} \$

$\text{field F} \$

$\text{number of variables 3} \$

$\text{homogeneous of degree 6} \$

$\text{monomial ordering lex} \$

$\text{variables "y0,y1,y2" "y_0,y_1,y_2"} \$

$\text{end} \$

$\text{define Y0} -\text{formula} \$

$\text{define Y1} -\text{formula} \$

$\text{define Y2} -\text{formula} \$

$\text{define Cremona} -\text{collection "Y0,Y1,Y2"} \$

$\text{with P} -\text{do} \$

$\text{projective space activity} \$

$\text{map R Cremona} "" \$

$\text{end} \$

arcs_5_2_q11:

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

$\text{define F} -\text{finite field} -q 11 -\text{end} \$

$\text{define P} -\text{projective space} -n 2 -\text{field F} -v 0 -\text{end} \$

$\text{with P} -\text{do} \$

$\text{projective space activity} \$

$\text{classify arcs} \$

$\text{poset classification control} \$

$\text{problem label arcs_5_2_q11} \$

$\text{W -depth 5} \$

$\text{report -end} \$

690
# 2 orbits:
# 0 1 2 3 37
# 0 1 2 3 49

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 \\
\$(ORBITER) -v 2 \\
-define C -combinatorial_objects \\
-file_of_points arcs_5_2_q11_lvl5 \\
-end \\
-with C -do \\
-ideal R \\
-end

#( 0, 1, 2, 3, 37 )
#generator 0 / 1 is 7*x0*x1 + 5*x0*x2 + 10*x1*x2
#We found 12 points on the generator of the ideal
#They are : ( 0, 1, 2, 3, 37, 54, 74, 80, 93, 105, 121, 128 )

#( 0, 1, 2, 3, 49 )
#generator 0 / 1 is 4*x0*x1 + 8*x0*x2 + 10*x1*x2
#looping over all generators of the ideal:
#generator 0 / 1 is ( 0, 4, 8, 0, 10, 0 ) :
#We found 12 points on the generator of the ideal
#They are : ( 0, 1, 2, 3, 41, 49, 58, 77, 83, 95, 109, 130 )
surface_9lines_4E_ideal:

g$(ORBITER) -v 2 \n-define Pts -vector -dense \n-define F -finite_field -q 3 -end \n-define R -polynomial_ring \n-define F -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 "surf_eqn" "surf_eqn" Pts \n-end

#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 2 2 0
#x0*x1 + 2*x0*x1*x1 + 2*x0*x1*x3
#2*x2*x2*x3 + 2*x2*x3*x3

SURFACE_F_9="x0*x1 - x0*x1*x1 -x0*x1*x3 -x2*x2*x3 - x2*x3*x3"

F_9_q7:

g$(ORBITER) -v 3 \n-define F -finite_field -q 7 -end \n-define P -projective_space -n 3 -field F -v 0 -end \n-define F_9 -cubic_surface -space P \n-by_equation "F_9" \n"\DF_9\D" "x0,x1,x2,x3" \n$(SURFACE_F_9) \n"" \n"Dno parameters\D" \n-end \n-with F_9 -do \n-cubic_surface_activity \n-report \n
# we create 20 5-subsets of PG(2,11) at random. Note that PG(2,11) has 133 points.

**random_k_subsets_PG_2_11:**

```bash
$(ORBITER) -v 4 \
-create_random_k_subsets 133 5 20
```

```bash
#random_k_subsets_n133_k5_nb20.csv
```

# We compute the line intersections:

**line_type_in_PG_2_11:**

```bash
$(ORBITER) -v 3 \
-orbiter_path $(ORBITER_PATH) \
-define F -finite_field -q 11 -end \
-define P -projective_space -n 2 -field F -v 0 -end \
-define C -combinatorial_objects \
-file_of_points random_k_subsets_n133_k5_nb20.csv \
-end \
-with C -do \
-combinatorial_object_activity \
-do -line_type P random_sets \
```

# the second one is an arc: 3,33,40,83,102

**random_arc_5_2_q11_ideal:**

```bash
$(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" \
```

693
define C:combinatorial_objects

set_of_points: "3,33,40,83,102"

with C:do

combinatorial_object:activity

ideal R:

#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)

Endrass_F7.txt:

define F:finite_field -q 7

define R:polynomial_ring -field F

number_of_variables: 4

homogeneous_of_degree: 8

define eqn:vector -field F -sparse 165

(ENDRASS_SPARSE):

define P:projective_space -n 3 -field F -v 0

define Endrass_F7:geometric_object P

projective_variety R:

"Endrass_F7":

eqn:

- with Endrass_F7:do

combinatorial_object:activity:save

# we created a set of 33 points, called Endrass_F7.txt

octic_prepare:

define A:vector -format 1 -dense "1,1,1,1"

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)

#############################################################################
# Chapter 9 - Applications
#############################################################################
# Section 9.1: Number Theory

SECTION_NUMBER_THEORY:

inverse_mod_26.99:

inverse_mod_a:

jacobi_35.41:

jacobi_33.41:
representation_on_polynomials_of_degree_3:
$\text{ORBITER} -v 4 \$
$\text{-define G -linear_group -PGL 4 3 -end} \$
$\text{-with G -do} \$
$\text{-group_theoretic_activity} \$
$\text{-representation_on_polynomials 3} \$
$\text{-end} \$
$\text{ORBITER} -v 2 \$
$\text{-loop L 0 9 1 -draw_matrix} \$
$\text{-input_csv_file PGL_4_3_rep_3_%L.csv} \$
$\text{-box_width 40 -bit_depth 24 -partition 3 20 20 -end} \$
$\text{-end_loop} \$
representation_tetrahedral_group_on_polynomials_of_degree_3:
$\text{ORBITER} -v 4 \$
$\text{-define G -linear_group -GL 3 3} \$
$\text{-subgroup_by_generators "tetra" "12" 2} \$
$\text{"0,1,0,0,0,1,1,0,0, 0,0,1,2,0,0,0,2,0"} \$
$\text{-end} \$

# Section 9.2: Representation Theory

SECTION_REPRESENTATION_THEORY:
# Section 9.3: Cryptography

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

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

EC\_points\_13:

```bash
$(ORBITER) -v 2
define F -finite_field -q 13 -end
with F -do
finite_field_activity
EC\_points "EC\_2.5\_q13" 2 5 -end
```
EC_points_199:
$(ORBITER) -v 2
  -define F -finite_field -q 199 -end
  -with F -do
  -finite_field_activity
  -EC_points "EC_5_7_q199" 5 7 -end
$(ORBITER) -v 2
  -draw_matrix -input_csv_file EC_5_7_q199_points_xy.csv
  -box_width 10 -bit_depth 24
  -partition 2 199 199 -end

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

EC_bsgs:
$(ORBITER) -v 2
  -define F -finite_field -q 199 -end
  -with F -do
  -finite_field_activity
  -EC_bsgs 5 7 "147,164" 212
  "172,158,45,195,50,22,10,103,55,33,50,22,145,105,31,74,73,155,67,60,25,6" 
  -end

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

NTRU N=7

698
NTRU P=3
NTRU Q=41
NTRU D=2
NTRU XN1="-1,0,0,0,0,0,1,"

# 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"

NTRU Alice1:

⊿ $(ORBITER) -v 2 \\
⊿ ⊿ -define F -finite_field -q $(NTRU Q) -end \\
⊿ ⊿ -with F -do \\
⊿ ⊿ -finite_field_activity \\
⊿ ⊿ -extended_gcd_for_polynomials \\
⊿ ⊿ -extended_gcd_for_polynomials \\
⊿ ⊿ -end

#F q(x) = 8X^{6} + 26X^{5} + 31X^{4} + 21X^{3} + 40X^{2} + 2X + 37
ALICE PRIVATE FQ="37,2,40,21,31,26,8"

NTRU Alice2:

⊿ $(ORBITER) -v 2 \\
⊿ ⊿ -define F -finite_field -q $(NTRU P) -end \\
⊿ ⊿ -with F -do \\
⊿ ⊿ -finite_field_activity \\
⊿ ⊿ -extended_gcd_for_polynomials \\
⊿ ⊿ -end

#F p(x) = X^{6} + 2X^{5} + X^{3} + X^{2} + X + 1
ALICE PRIVATE FP="1,1,1,0,1,2,1"

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) \\

699
#C(X) = 20X^6 + 40X^5 + 2X^4 + 38X^3 + 8X^2 + 26X + 30
ALICE_PUBLIC_KEY = "30,26,8,38,2,40,20"

BOB_MESSAGE = "1,-1,1,1,0,-1"

BOB_ONE_TIME_KEY = "-1,1,0,0,0,-1,1"

NTRU encrypt:

# E(X) = 31X^6 + 19X^5 + 4X^4 + 2X^3 + 40X^2 + 3X + 25
BOB_ENCRYPT = "25,3,40,2,4,19,31"

NTRU decrypt1:

# A(X) = X^6 + 10X^5 - 8X^4 - X^3 - X^2 + X - 1
ALICE_C2 = "-1,1,-1,-1,-8,10,1"

NTRU decrypt2:

NTRU decrypt3:
```plaintext
# define F - finite_field -q $(NTRU_P) -end 
# with F -do 
# finite_field_activity 
# polynomial_reduce_mod p $(ALICE_C2) -end 

#A(X)=X^6 + X^5 + X^4 + 2X^3 + 2X^2 + X + 2
ALICE_C3="2,1,2,1,1,1"

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) 
-end

#C(X)=2X^5 + X^3 + X^2 + 2X + 1
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

#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"
RSA

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

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 \
```
9039  ▶ ▶ -RSA 1061 25330309 3 "19019931,1619805,740498,2671344"
9040
9041
9042 #*****************************************************************************
9043 # 7253\ast8171 = 59264263
9044 # 7252\ast8170 = 59248840
9045
9046
9047 im3:
9048 ▶ ▶ $(ORBITER) -v 2 \n9049 ▶ ▶ -inverse_mod 2909 59248840
9050 ▶
9051 #the inverse of 2909 mod 59248840 is 4358629
9052
9053 RSA_e3:
9054 ▶ ▶ $(ORBITER) -v 2 \n9055 ▶ ▶ -RSA_encrypt_text 2909 59264263 3 encrypted
9056
9057 RSA_d3:
9058 ▶ ▶ $(ORBITER) -v 2 \n9059 ▶ ▶ -RSA 4358629 59264263 3 "35270141,9642524,49091707"
9060
9061 #51403,182516,200504 = encrypted
9062
9063
9064 ####
9065 # 7879 \ast 7901 = 62251979
9066 # 7878 \ast 7900 = 62236200
9067
9068 # e =
9069
9070 im4:
9071 ▶ ▶ $(ORBITER) -v 2 -inverse_mod 583 62236200
9072
9073 # the inverse of 583 mod 62236200 is 32559247
9074
9075 RSA_e4:
9076 ▶ ▶ $(ORBITER) -v 2 \n9077 ▶ ▶ -RSA_encrypt_text 583 62251979 3 venividivici
9078
9079 #-RSA_encrypt_text 583 62251979 venividivici
9080 #40513610,53979973,56449676,35068535
9081
9082 RSA_d4:
9083 ▶ ▶ $(ORBITER) -v 2 \n9084 ▶ ▶ -RSA 32559247 62251979 "40513610,53979973,56449676,35068535"
9085
```
# 7369 * 7127 = 52518863
# 7368 * 7126 = 52504368

im5:

```bash
$ (ORBITER) -v 2 -inverse_mod 173 52504368
```

# the inverse of 173 mod 52504368 is 38543669

```bash
$(ORBITER) -v 2
```

```bash
-RSA encrypt_text 38543669 52518863 3 fascinating
```

```bash
#-RSA_encrypt_text 38543669 52518863 fascinating
```

```bash
#31526751,8962078,51045732,51894467
```

```bash
$(ORBITER) -v 2
```

```bash
-RSA 173 52518863 "31526751,8962078,51045732,51894467"
```

```bash
$(ORBITER) -v 2
```

```bash
-RSA 47177497 55040413 "28702119,48926559"
```

```bash
$(ORBITER) -v 2
```

```bash
-sift_smooth 100000 100 "2,3,5,7,11,13,17,19"
```

```bash
$(ORBITER) -v 2
```

# 1999 * 7907 = 15806093
# 1998 * 7906 = 15796188

im7:

```bash
$ (ORBITER) -v 2 -inverse_mod 3221 15796188
```

# the inverse of 3221 mod 15796188 is 10048553
$(ORBITER) -v 2 -RSA_encrypt.text 10048553 15806093 3 beachandfun

# 7853 * 7673 = 60256069
# 7852 * 7672 = 60240544

im8:
$(ORBITER) -v 2 -inverse_mod 9017 60240544

# the inverse of 9017 mod 60240544 is 14430473

RSA_e8:
$(ORBITER) -v 2 -RSA_encrypt.text 9017 60256069 3 strawberry

sqrt_big:
$(ORBITER) -v 2 -square_root 1002001

sqrt_mod_33_41:
$(ORBITER) -v 2 -square_root_mod 33 41

quadratic_sieve:
$(ORBITER) -v 5 -quadratic_sieve 31 500 1

pseudoprime3:
$(ORBITER) -v 5 -seed 2531011 -find_pseudoprime 3 5 0 0
\pdflatex pseudoprime3.tex
\open pseudoprime3.pdf

pseudoprime10:
$(ORBITER) -v 5 -seed 2531011 -find_pseudoprime 10 5 5 5
\pdflatex pseudoprime10.tex
\open pseudoprime10.pdf

# 4460190157

PR10_test1:
pseudoprime11:

```bash
$ (ORBITER) -v 5 \n$ (ORBITER) -v 5 -power_mod 1293 2230095078 4460190157
$ (ORBITER) -v 5 -power_mod 9865 2230095078 4460190157
$ (ORBITER) -v 5 -power_mod 19645 2230095078 4460190157
$ (ORBITER) -v 5 -power_mod 974586571 2230095078 4460190157
$ (ORBITER) -v 5 -power_mod 974586571 1486730052 4460190157
$ (ORBITER) -v 5 -power_mod 974586571 15222492 4460190157
$ (ORBITER) -v 5 -power_mod 974586571 284796 4460190157
```

# 63814633367

# product is 284625399616057168619

pseudoprime20:

```bash
$ (ORBITER) -v 5 \n$ (ORBITER) -v 5 -find_pseudoprime 11 5 5 5
$ pdflatex pseudoprime_11.tex
$ open pseudoprime_11.pdf
```

PR10:

```bash
$ (ORBITER) -v 5 -primitive_root 4460190157
```

# mistake! long integer overflow

# a primitive root modulo 165222861 is 1293

pseudoprime50:

```bash
$ (ORBITER) -v 5 \n$ (ORBITER) -v 5 -find_pseudoprime 50 5 0 0
$ pdflatex pseudoprime_50.tex
$ open pseudoprime_50.pdf
```

#91322792878581218181431392170986926262336688354473
pseudoprime51:
  $(ORBITER) -v 5 \n  -seed 2531011 -find_pseudoprime 51 5 5 5
  pdflatex pseudoprime_51.tex
  open pseudoprime_51.pdf

#7546007277468344704721408970249004944659715367045417

# product 6891224596605081960619999442326431573233529532400658436661744403244049
  57291497379904326661586100241

pseudoprime30:
  $(ORBITER) -v 5 \n  -seed 2531011 -find_pseudoprime 30 5 5 5
  pdflatex pseudoprime_30.tex
  open pseudoprime_30.pdf

# 286525565474504516914595596387

pseudoprime31:
  $(ORBITER) -v 5 \n  -seed 2531011 -find_pseudoprime 31 5 5 5
  pdflatex pseudoprime_31.tex
  open pseudoprime_31.pdf

# 2514911323283298698837184692002835573476743643265896783515097

# maybe 2 seconds

pseudoprime33:
  $(ORBITER) -v 5 \n  -seed 2531011 -find_pseudoprime 33 5 5 5
  pdflatex pseudoprime_33.tex
  open pseudoprime_33.pdf

#37167419949829534554363004459891

pseudoprime34:
  $(ORBITER) -v 5 \n  -seed 2531011 -find_pseudoprime 34 5 5 5
  pdflatex pseudoprime_34.tex
  open pseudoprime_34.pdf
pseudoprime35:

```
▷ $(ORBITER) -v 5 \
▷ -seed 2531011 -find_pseudoprime 35 5 5 5
▷ pdflatex pseudoprime_35.tex
▷ open pseudoprime_35.pdf
```

# 81329557792505271120435930267680203

pseudoprime36:

```
▷ $(ORBITER) -v 5 \
▷ -seed 2531011 -find_pseudoprime 36 5 5 5
▷ pdflatex pseudoprime_36.tex
▷ open pseudoprime_36.pdf
```

# 162624680891993404333363207561599139

MATH360_hw2:

```
▷ $(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
▷ $(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
▷ $(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
▷ $(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
```

-parse_and_evaluate "test" "a*a+b*b" "a=5,b=7" -end

F_256_Rijndahl:
$ (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:
$ (ORBITER) -v 3 -all_square_roots_mod_n 1075922 1549411

power_mod_211:
$ (ORBITER) -v 3 -power_mod_n 2 211
$ (ORBITER) -v 3 \-plot_function power_mod_n_a2_n211.csv
$ (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:
$ (ORBITER) -v 3 -power_mod_n 2 31
$ (ORBITER) -v 3 \-plot_function power_mod_n_a2_n31.csv
$ (ORBITER) -v 2 -draw_matrix \-input_csv_file power_mod_n_a2_n31_graph.csv \-box_width 10 -bit_depth 8 -partition 3 31 31 -end

power_mod_3_31:
$ (ORBITER) -v 3 -power_mod_n 3 31
$ (ORBITER) -v 3 \-plot_function power_mod_n_a3_n31.csv
$ (ORBITER) -v 2 -draw_matrix \-input_csv_file power_mod_n_a3_n31_graph.csv \-box_width 10 -bit_depth 8 -partition 3 31 31 -end

# Chapter 10 - Coding Theory
# Section 10.1: Coding Theory

SECTION CODING THEORY INTRODUCTION:

Allen Gates noise 1 percent:
```bash
$ (ORBITER) -v 3 -random_noise_in_bitmap_file allen.Gates.bmp
```

Hamming space 4 2 distance matrix:
```bash
$ (ORBITER) -Hamming_space_distance_matrix 4 2
```

Hamming space 4 2 distance matrix draw:
```bash
$ (ORBITER) -v 2 -draw_matrix -input_csv_file Hamming_n4_q2.csv -box_width 20 -bit_depth 24 -partition 4 16 16 -end
```

open Hamming_n4_q2.draw.bmp

Hamming code macwilliams:
```bash
$ (ORBITER) -v 2 -make_macwilliams_system 7 4 2
```

pdflatex MacWilliams_n7_k4_q2.tex
open MacWilliams_n7_k4_q2.pdf

code 5 2 3 diagram:
```bash
$ (ORBITER) -v 2 -define F -finite_field -q 2 -end -with F -do -coding_theoretic_activity -code_diagram "code_5_2_3"
```

710
Hamming_5_2.graph:

```bash
$ (ORBITER) -v 2 \
define G -graph -Hamming 5 2 -end \
with G -do \
-graph_theoretic_activity -export_csv -end \
with G -do \
-graph_theoretic_activity -export_graphviz -end \
with G -do \
-graph_theoretic_activity -save -end
```

```
$(ORBITER) -v 2 -draw_matrix \
-input_csv_file code_5_2_3_diagram_01_5_4.csv \
-box_width 25 -bit_depth 24 \
-partition 4 8 4 \
-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_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
9461
9462
9463 code_6:
9464 ▶ $(ORBITER) -v 2 \
9465 ▶ ▶ -define F -finite_field -q 2 -end \
9466 ▶ ▶ -with F -do -coding_theoretic_activity \
9467 ▶ ▶ ▶ -general_code_binary 6 "0,60,50,41,14,21,27,39" \
9468 ▶ ▶ -end
9469 ▶ $(ORBITER) -v 2 -draw_matrix \
9470 ▶ ▶ -input_csv_file code_matrix_8_8.csv \
9471 ▶ ▶ -box_width 20 -bit_depth 24 \
9472 ▶ ▶ -partition 2 "1,1,1,1,1,1,1,1" "1,1,1,1,1,1,1,1" \
9473 ▶ ▶ -end
9474 ▶ pdflatex code_6_8.tex
9475 ▶ open code_6_8.pdf
9476 ▶ open code_matrix_8_8_draw.bmp
9477
9478
9479 # linear code with generator matrix
9480 # 111100
9481 # 110010
9482 # 101001
9483
9484
9485
9486 # Section 10.2: Linear codes
9487
9488
9489 SECTION_CODING_THEORY_LINEAR_CODES:
9490
9491
9492 RM_3_1:
9493 ▶ $(ORBITER) -v 2 \
9494 ▶ ▶ -define F -finite_field -q 2 -end \
9495 ▶ ▶ -define C -code -field F \n9496 ▶ ▶ ▶ -first_order_Reed_Muller 3 \
9497 ▶ ▶ -end \
9498 ▶ ▶ -with C -and F -do -coding_theoretic_activity \n9499 ▶ ▶ ▶ -export_magma RM_3_1.magma \
9500 ▶ ▶ -end
9501
9502
9503
9504
9505
9506 simplex_code:
9507 ▶ $(ORBITER) -v 2 \

712
```plaintext
9508  \triangleright \triangleright -define F -finite_field -q 2 -end \n9509  \triangleright \triangleright -define v -vector -field F -format 3 \n9510  \triangleright \triangleright \triangleright -dense $(SIMPLEX_CODE_GENERATOR) \n9511  \triangleright \triangleright -end \n9512  \triangleright \triangleright -define C -code -field F \n9513  \triangleright \triangleright \triangleright -linear_code_through_generator_matrix v \n9514  \triangleright \triangleright -end
9515
9516
9517
9518  Hamming_generator:
9519  \triangleright $(ORBITER) -v 2 \n9520  \triangleright \triangleright -define F -finite_field -q 2 -end \n9521  \triangleright \triangleright -define v -vector -field F -format 3 \n9522  \triangleright \triangleright \triangleright -dense $(SIMPLEX_CODE_GENERATOR) \n9523  \triangleright \triangleright -end \n9524  \triangleright \triangleright -with F -do \n9525  \triangleright \triangleright -finite_field_activity \n9526  \triangleright \triangleright \triangleright -nullspace v \n9527  \triangleright \triangleright -end
9528  \triangleright pdflatex nullspace_3_7.tex
9529  \triangleright open nullspace_3_7.pdf
9530
9531  # basis in binary:
9532  # 67, 37, 22, 15
9533  # normalize from the right \n9534
9535
9536
9537
9538  Hamming_code:
9539  \triangleright $(ORBITER) -v 2 \n9540  \triangleright \triangleright -define F -finite_field -q 2 -end \n9541  \triangleright \triangleright -define v -vector -field F -format 3 \n9542  \triangleright \triangleright \triangleright -dense $(SIMPLEX_CODE_GENERATOR) \n9543  \triangleright \triangleright -end \n9544  \triangleright \triangleright -define C -code -field F \n9545  \triangleright \triangleright \triangleright -linear_code_through_generator_matrix v \n9546  \triangleright \triangleright \triangleright -dual \n9547  \triangleright \triangleright -end \n9548  \triangleright \triangleright -with C -do -coding_theoretic_activity \n9549  \triangleright \triangleright \triangleright -export_magma Hamming.magma \n9550  \triangleright \triangleright -end
9551
9552  # writes Hamming.magma
9553
9554
713```
9555
9556  RM_3_1_and_codewords:
9557  ▶ $(ORBITER) -v 2 \\
9558  ▶ ▶ -define F -finite_field -q 2 -end \\
9559  ▶ ▶ -define C -code -field F -first_order_Reed_Muller 3 -end \\
9560  ▶ ▶ -with C -and F -do -coding_theoretic_activity \\
9561  ▶ ▶ ▶ -export_magma RM_3_1.magma \\
9562  ▶ ▶ ▶ -end \\
9563  ▶ ▶ -with C -and F -do -coding_theoretic_activity \\
9564  ▶ ▶ ▶ ▶ -export_codewords RM_3_1_codewords.csv \\
9565  ▶ ▶ ▶ ▶ -end \\
9566  ▶ ▶ -with C -and F -do -coding_theoretic_activity \\
9567  ▶ ▶ ▶ ▶ -export_genma RM_3_1_genma.csv \\
9568  ▶ ▶ ▶ ▶ -end
9569
9570
9571  RM_3_1_from_generator_matrix:
9572  ▶ $(ORBITER) -v 2 \\
9573  ▶ ▶ -define F -finite_field -q 2 -end \\
9574  ▶ ▶ -define genma -vector -format 8 -field F \\
9575  ▶ ▶ ▶ -compact $(CODE_RM_3_1_GENMA) \\
9576  ▶ ▶ ▶ -end \\
9577  ▶ ▶ -define C -code -field F \\
9578  ▶ ▶ ▶ -linear_code_through_generator_matrix genma \\
9579  ▶ ▶ ▶ -end
9580  ▶ #pdflatex code_n8_k4_q2.tex
9581  ▶ #open code_n8_k4_q2.pdf
9582
9583  #Codewords: (0,255,170,85,204,51,102,153,240,15,90,165,60,195,150,105)
9584
9585
9586  RM_4_1_and_codewords:
9587  ▶ $(ORBITER) -v 2 \\
9588  ▶ ▶ -define F -finite_field -q 2 -end \\
9589  ▶ ▶ -define C -code -field F -first_order_Reed_Muller 4 -end \\
9590  ▶ ▶ -with C -and F -do -coding_theoretic_activity \\
9591  ▶ ▶ ▶ -export_magma RM_4_1.magma \\
9592  ▶ ▶ ▶ -end \\
9593  ▶ ▶ -with C -and F -do -coding_theoretic_activity \\
9594  ▶ ▶ ▶ ▶ -export_codewords RM_4_1_codewords.csv \\
9595  ▶ ▶ ▶ ▶ -end \\
9596  ▶ ▶ -with C -and F -do -coding_theoretic_activity \\
9597  ▶ ▶ ▶ ▶ ▶ -export_genma RM_4_1_genma.csv \\
9598  ▶ ▶ ▶ ▶ ▶ -end
9599
9600  RM_5_1_and_codewords:
9601  ▶ $(ORBITER) -v 2 \\

714
-define F -finite_field -q 2 -end \n-define C -code -field F -first_order_Reed_Muller 5 -end \n-with C -and F -do -coding_theoretic_activity \n-export magma RM_5_1.magma \n-end \n-with C -and F -do -coding_theoretic_activity \n-export codewords RM_5_1.codewords.csv \n-end \n-with C -and F -do -coding_theoretic_activity \n-export genma RM_5_1.genma.csv \n-end

# ToDo:

Hamming_code_words_old:
$\texttt{ORBITER} -v 2 \n-def v -vector -dense $\texttt{HAMMING\_CODE\_ROWS\_IN\_BINARY\_RANKS}$ -end \n-linear_code_through_basis 7 v
\n\n\nHamming_weightenumerator:
$\texttt{ORBITER} -v 2 \n-def F -finite_field -q 2 -end \n-def v -vector -field F -format 4 \n-dense $\texttt{HAMMING\_CODE\_GENERATOR}$ \n-end \n-def C -code -field F \n-linear_code_through_generator_matrix v \n-end \n-with C -do \n-coding_theoretic_activity \n-weight Enumerator \n-end

Hamming_minimum_distance:
$\texttt{ORBITER} -v 2 \n-def F -finite_field -q 2 -end \n-def v -vector -field F -format 4 \n-dense $\texttt{HAMMING\_CODE\_GENERATOR}$ \n-end \n-with F -do \n-coding_theoretic_activity \n
Golay23\_minimum\_distance:  
\>$\text{ORBITER} -v 2 \end  
\>$\text{define } F -\text{finite\_field} -q 2 -\text{end}  
\>$\text{define } v -\text{vector -field } F -\text{format} 12  
\>$\text{dense } $(\text{GOLAY23\_CODE\_GENERATOR})  
\>$\text{end}  
\>$\text{with } F -\text{do}  
\>$\text{coding\_theoretic\_activity}  
\>$\text{minimum\_distance } v  
\>$\text{end}  

#d=7 in 0 sec  

Hamming\_code\_diagram:  
\>$\text{ ORBITER} -v 2 \end  
\>$\text{define } F -\text{finite\_field} -q 2 -\text{end}  
\>$\text{with } F -\text{do} -\text{coding\_theoretic\_activity}  
\>$\text{code\_diagram } "\text{Hamming\_7\_4}"  
\>$\text{hamming\_code\_codewords} 7  
\>$\text{metric\_balls} 1  
\>$\text{end}  
\>$\text{ ORBITER} -v 2 \end  
\>$\text{draw\_matrix}  
\>$\text{input\_csv\_file } \text{Hamming\_7\_4\_diagram\_01\_7\_16.csv}  
\>$\text{box\_width} 25 -\text{bit\_depth} 24  
\>$\text{partition} 4 16 8  
\>$\text{end}  
\>$\text{ ORBITER} -v 2 \end  
\>$\text{draw\_matrix}  
\>$\text{input\_csv\_file } \text{Hamming\_7\_4\_diagram\_7\_16.csv}  
\>$\text{box\_width} 25 -\text{bit\_depth} 14  
\>$\text{partition} 4 16 8  
\>$\text{end}  
\>$\text{ ORBITER} -v 2 -\text{define } F -\text{finite\_field} -q 2 -\text{end} -\text{with } F -\text{do} -\text{coding\_theoretic\_activity} -\text{code\_diagram } "\text{Hamming\_7\_4\_word\_0}" "0" 7 -\text{metric\_balls} 1 -\text{end}  
\>$\text{ ORBITER} -v 2 -\text{define } F -\text{finite\_field} -q 2 -\text{end} -\text{with } F -\text{do} -\text{coding\_theoretic\_activity} -\text{code\_diagram } "\text{Hamming\_7\_4\_word\_1}" "67" 7 -\text{metric\_balls} 1 -\text{end}  
\>$\text{ ORBITER} -v 2 -\text{define } F -\text{finite\_field} -q 2 -\text{end} -\text{with } F -\text{do} -\text{coding\_theoretic\_activity} -\text{code\_diagram } "\text{Hamming\_7\_4\_word\_2}" "37" 7 -\text{metric\_balls} 1 -\text{end}  
\>$\text{ ORBITER} -v 2 -\text{define } F -\text{finite\_field} -q 2 -\text{end} -\text{with } F -\text{do} -\text{coding\_theoretic\_activity} -\text{code\_diagram } "\text{Hamming\_7\_4\_word\_3}" "102" 7 -\text{metric\_balls} 1 -\text{end}  
\>$\text{ ORBITER} -v 2 -\text{define } F -\text{finite\_field} -q 2 -\text{end} -\text{with } F -\text{do} -\text{coding\_theoretic\_activity} -\text{code\_diagram } "\text{Hamming\_7\_4\_word\_4}" "22" 7 -\text{metric\_balls} 1 -\text{end}
$(ORBITER) -v 2 -define F -finite_field -q 2 -end -with F -do -coding_theoretic_activity -code_diagram "Hamming_7_4_word_5" "85" 7 -metric_balls 1 -end
$(ORBITER) -v 2 -define F -finite_field -q 2 -end -with F -do -coding_theoretic_activity -code_diagram "Hamming_7_4_word_6" "51" 7 -metric_balls 1 -end
$(ORBITER) -v 2 -define F -finite_field -q 2 -end -with F -do -coding_theoretic_activity -code_diagram "Hamming_7_4_word_7" "112" 7 -metric_balls 1 -end
$(ORBITER) -v 2 -define F -finite_field -q 2 -end -with F -do -coding_theoretic_activity -code_diagram "Hamming_7_4_word_8" "15" 7 -metric_balls 1 -end
$(ORBITER) -v 2 -define F -finite_field -q 2 -end -with F -do -coding_theoretic_activity -code_diagram "Hamming_7_4_word_9" "76" 7 -metric_balls 1 -end
$(ORBITER) -v 2 -define F -finite_field -q 2 -end -with F -do -coding_theoretic_activity -code_diagram "Hamming_7_4_word_10" "42" 7 -metric_balls 1 -end
$(ORBITER) -v 2 -define F -finite_field -q 2 -end -with F -do -coding_theoretic_activity -code_diagram "Hamming_7_4_word_11" "105" 7 -metric_balls 1 -end
$(ORBITER) -v 2 -define F -finite_field -q 2 -end -with F -do -coding_theoretic_activity -code_diagram "Hamming_7_4_word_12" "25" 7 -metric_balls 1 -end
$(ORBITER) -v 2 -define F -finite_field -q 2 -end -with F -do -coding_theoretic_activity -code_diagram "Hamming_7_4_word_13" "90" 7 -metric_balls 1 -end
$(ORBITER) -v 2 -define F -finite_field -q 2 -end -with F -do -coding_theoretic_activity -code_diagram "Hamming_7_4_word_14" "60" 7 -metric_balls 1 -end
$(ORBITER) -v 2 -define F -finite_field -q 2 -end -with F -do -coding_theoretic_activity -code_diagram "Hamming_7_4_word_15" "127" 7 -metric_balls 1 -end
$(ORBITER) -v 2 -loop L 0 16 1 -draw_matrix \input_csv_file Hamming_7_4_word_%L_diagram_7_1.csv \box_width 25 -bit_depth 8 -partition 4 16 8 -end \end_loop

# DRAW_PATH/draw_matrix.out -read code_matrix_256_256.csv -box_width 10

# 1 0 0 0 0 1 1 = 67
# 0 1 0 0 1 0 1 = 37
# 0 0 1 0 1 1 0 = 22
Hamming_RREF:

\[(\text{ORBITER}) -v 2 \]
\[\define F -\text{finite\_field} -q 2 -\text{end} \]
\[\define v -\text{vector} -\text{format} 4 -\text{field} F \]
\[\text{dense} \$(\text{HAMMING\_CODE\_GENERATOR}) \]
\[\text{end} \]
\[\text{with} F -\text{do} \]
\[\text{finite\_field\_activity} \]
\[\text{RREF v -end} \]
\[\text{pdflatex} \text{RREF\_example.q2.4.7.tex} \]
\[\text{gs} -sDEVICE=png16 -d\text{FIXEDMEDIA} \]
\[\text{nullspace} v \]
\[\text{normalize\_from\_the\_right} \]
\[\text{pdflatex} \text{nullspace.4.7.tex} \]
\[\text{open} \text{nullspace.4.7.pdf} \]

Hamming_nullspace:

\[(\text{ORBITER}) -v 2 \]
\[\define F2 -\text{finite\_field} -q 2 -\text{end} \]
\[\define v -\text{vector} -\text{format} 4 -\text{field} F2 \]
\[\text{dense} \$(\text{HAMMING\_CODE\_GENERATOR}) \]
\[\text{end} \]
\[\text{with} F2 -\text{do} \]
\[\text{finite\_field\_activity} \]
\[\text{nullspace} v \]
\[\text{normalize\_from\_right} \]
\[\text{pdflatex} \text{nullspace.4.7.tex} \]
\[\text{open} \text{nullspace.4.7.pdf} \]

#check equations of the Hamming code:

\# a4+a5+a6+a7 =1+0+1+0=0 \ modulo 2 \ OK.
\# a2+a3+a6+a7 =0+1+1+0=0 \ modulo 2 \ OK.
\# a1+a3+a5+a7 =1+1+0+0=0 \ modulo 2 \ OK.
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\_L.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 singer:
$(ORBITER) -v 3 
-define G -linear\_group -PGL 3 2 -singer 1 -end 
-define Orb -orbits -group G 
-on\_points 
-end
#pdflatex PGL\_3.2\_Singer\_3.2.1\_report.tex
#open PGL\_3.2\_Singer\_3.2.1\_report.pdf

# cycle is 0,1,2,5,3,4,6
#1001110
#0100111
#0011101

#with G -do 
#group\_theoretic\_activity 
#-report 
#-orbits\_on\_points 
#-end

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 

Hamming cyclic long:
$\text{ORBITER} -v 2 -long\_code 7 4 \$

$0,4,6$ 

$1,4,5,6$ 

$2,4,5$ 

$3,5,6$

$(\text{ORBITER}) -v 2 -loop L 0 16 1 -draw\_matrix \$

$\text{input}\_csv\_file long\_code\_genma\_n7\_k4\_codeword\_\%L.csv \$

$\text{-box\_width 25 -bit\_depth 8 -partition 3 4 2 -end} \$

$(\text{ORBITER}) -v 2 -\text{define v -vector -dense } 69,39,22,11$ -end 

$\text{ORBITER} -v 2 -\text{linear}\_code\_through\_basis 7 v$

$\text{input}\_csv\_file code\_matrix\_16.8.csv \$

$\text{-box\_width 25 -bit\_depth 8 -partition 2 16 8 -end} \$

open code\_matrix\_16.8\_draw.bmp

pdflatex code\_7.16.tex

open code\_7.16.pdf

Hamming cyclic clean:

$\text{ORBITER} -v 2 \$

$\text{-define F -finite\_field -q 2 -end} \$

$\text{-define v -vector -format 3 -field F} \$

$\text{-dense } (\text{SIMPLEX\_CODE\_GENMA\_CYCLIC}) \$

$\text{-end} \$

$\text{-with F -do -finite\_field\_activity} \$

$\text{-nullspace v} \$

$\text{-normalize\_from\_the\_right} \$

$\text{-end} \$

pdflatex nullspace\_4.7.tex

open nullspace\_4.7.pdf
Hamming cyclic clean long:

```bash
$(ORBITER) -v 2 -long_code 7 4 \
0,2,3" \n"1,3,4" \n"2,4,5" \n"3,5,6"
```

```bash
$(ORBITER) -v 2 \
-loop L 0 16 1 -draw_matrix \
-input_csv_file \
long_code_genma_n7_k4.codeword_%L.csv \
-box_width 25 -bit_depth 8 \n-partition 3 4 2 -end \n-end_loop
```

# Section 10.3: Coding Theory - Golay codes

SECTION CODING THEORY GOLAY CODES:

Golay23 code words:

```bash
$(ORBITER) -v 2 \
$define v -vector -dense $(GOLAY_23_COLUMN_RANKS_PROJECTIVELY) -end \n$define F -finite_field -q 2 -end \n```
$\triangledown \triangledown \triangledown$

-define C -code -field F
-linear_code_from_from_projective_set 12 v -end
-with C -and F -do -coding_theoretic_activity
-export_magma Golay23.magma
-end
-with C -and F -do -coding_theoretic_activity
-export_codewords Golay23_codewords.csv
-end
-with C -and F -do -coding_theoretic_activity
-export_genma Golay23_genma.csv
-end

#pdflatex code
n23
k12
q2.tex

#open code
n23
k12
q2.pdf

#-with F -do
-coding_theoretic_activity
-linear_code_through_columns_of_parity_check_projectively
-12 v
-end

Golay23_code_diagram:

$\triangledown\triangledown$$(ORBITER) -v 2$

-define F -finite_field -q 2 -end
-with F -do
-coding_theoretic_activity
-linear_code_through_columns_of_parity_check_projectively
-12 v
-end

>Golay23_code_diagram_from_file "Golay_23"
-codewords_n23_k12_q2.csv 23
-enhance 4
-end

#-metric_balls 3

Golay23_code_diagram_draw:

$\triangledown\triangledown$$(ORBITER) -v 2$
-draw_matrix
-input_csv_file Golay23_diagram_01.23.4096.csv
-box_width 4 -bit_depth 8
-partition 20 4096 2048
-end

###############################################################################
encode_text_5bits:
 $(ORBITER) -encode_text_5bits
 "Hithere" "text.csv"
 $(ORBITER) -v 2
 -define F -finite_field -q 2 -end
 -with F -do
 -coding_theoretic_activity
 -polynomial_division_from_file
 text.csv 13 -end
 pdflatex polynomial_division_file_13.tex
 open polynomial_division_file_13.pdf

encode_text_5bits_check:
 $(ORBITER) -v 2
 -define F -finite_field -q 2 -end
 -with F -do
 -coding_theoretic_activity
 -polynomial_division_from_file
 text_with_1error.csv 13
 -end
 pdflatex polynomial_division_file_13.tex
 open polynomial_division_file_13.pdf

encode_text_5bits_1error:
 $(ORBITER) -encode_text_5bits
 "Hithere" "text.csv"
 $(ORBITER) -v 2
 -define F -finite_field -q 2 -end
 -with F -do
 -coding_theoretic_activity
 -polynomial_division_from_file_all_k_bit_error_patterns
 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
CRC_3_128_10:
$(ORBITER) -v 1 \\  
-define F -finite_field -q 2 -end \\  
-with F -do -coding_theoretic_activity \\  
-find_CRC_polynomials 3 128 10 \  
-end

crc32_test:
$(ORBITER) -v 3 \\  
-define F -finite_field -q 2 -end \\  
-with F -do -coding_theoretic_activity \\  
-crc32 "123456789" \\  
-end

crc32_test_hexdata:
$(ORBITER) -v 3 \\  
-define F -finite_field -q 2 -end \\  
-with F -do -coding_theoretic_activity \\  
-crc32_hexdata "7BD11C4010" \\  
-end

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

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

# alfa:

CRC_F256_BCH_code_d2:
# degree of polynomial = 3
10068 # dense "214,167,1"
10069 # sparse "214,0,167,1,1,2"
10070
10071 CRC_F256_BCH_code.d2.old:
10072 $(ORBITER) -v 3 \
10073 -define F -finite_field -q 256 -end \n10074 -with F -do -coding_theoretic_activity \n10075 -make_BCH_code 771 2 \n10076 -end
10077 pdflatex BCH_codes_q256_n771_d2.tex
10078 open BCH_codes_q256_n771_d2.pdf
10079
10080
10081 CRC_POLY_Q256_DEG2_DENSE="214,167,1"
10082
10083
10084 CRC_F256_BCH_write_code_for_division.d2:
10085 $(ORBITER) -v 2 \
10086 -define F -finite_field -q 256 -end \n10087 -define A -vector -field F -sparse 772 "1,771,1,0" -end \n10088 -define B -vector -field F -dense $(CRC_POLY_Q256_DEG2_DENSE) -end \n10089 -with F -do \n10090 -coding_theoretic_activity \n10091 -write_code_for_division \n10092 alfa A B \n10093 -end
10094 g++ crcalfa.cpp -o crcalfa.out
10095 ./crcalfa.out
10096
10097
10098
10099
10100 # bravo:
10101
10102 # degree of polynomial = 4
CRC_F256_BCH_code_d3:
1007  $(ORBITER) -v 2 \n1008  -define F -finite_field -q 256 -end \n1009  -define C -code -field F \n1010  -BCH 771 3 \n1011  -end \n1012  -with C -and F -do -coding_theoretic_activity \n1013  -export magma BCH_lq8_n771_d3.magma \n1014  -end \n1015  pdflatex BCH_codes_q256_n771_d3.tex
1016  open BCH_codes_q256_n771_d3.pdf
1017
1018 CRC_POLY_BRAVO_DENSE="1,23,27,213,1"
1019
1020 CRC_F256_BCH_write_code_for_division_Bravo:
1023  $(ORBITER) -v 2 \n1024  -define F -finite_field -q 256 -end \n1025  -define A -vector -field F -sparse 772 "1,771,1,0" -end \n1026  -define B -vector -field F -dense $(CRC_POLY_BRAVO_DENSE) -end \n1027  -with F -do \n1028  -coding_theoretic_activity \n1029  -write_code_for_division \n1030  -bravo A B \n1031  -end \n1032  g++ crc_bravo.cpp -o crc_bravo.out
1033  ./crc_bravo.out
1034
1035 # Charlie
1036 CRC_F256_BCH_code_d7:
1039  $(ORBITER) -v 2 \n1040  -define F -finite_field -q 256 -end \n1041  -define C -code -field F \n1042  -BCH 771 7 \n1043  -end \n1044  -with C -and F -do -coding_theoretic_activity \n1045  -export magma BCH_lq8_n771_d7.magma \n1046  -end \n1047  pdflatex BCH_codes_q256_n771_d7.tex
1048  open BCH_codes_q256_n771_d7.pdf
1049
# polynomial of degree 12:

- dense "1,126,25,1,196,209,244,3,121,126,35,65,1"
- sparse "1,0,126,1,25,2,1,3,196,4,209,5,244,6,3,7,121,8,126,9,35,10,65,11,1,12"

CRC_POLY_CHARLIE_DENSE="1,126,25,1,196,209,244,3,121,126,35,65,1"

CRC_POLY_CHARLIE_DENSE="1,126,25,1,196,209,244,3,121,126,35,65,1"

CRC_F256_BCH_write_code_for_division_Chipotle:

```
g++ $(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_CHARLIE_DENSE) -end \  
```

```
-with F -do \  
```

```
-coding_theoretic_activity \  
```

```
write_code_for_division \  
```

```
charlie A B \  
```

```
-end \  
```

```
g++ crc_charlie.cpp -o crc_charlie.out \  
```

```
./crc_charlie.out \  
```

```
F256_BCH_code_d16: \  
```

```
g++ $(ORBITER) -v 3 \  
```

```
-define F -finite_field -q 256 -end \  
```

```
-with F -do -coding_theoretic_activity \  
```

```
-make_BCH_code 771 16 \  
```

```
-end \  
```

```
pdflatex BCH_codes_q256_n771_d16.tex \  
```

```
open BCH_codes_q256_n771_d16.pdf \  
```

```
#generator polynomial is X^{30} + 253X^{29} + 174X^{28} + 109X^{27} + 97X^{26} +  
```

```
144X^{25} + 112X^{24} + 212X^{23} + 192X^{22} + 169X^{21} + 24X^{20} + 150X^{19} +  
```

```
110X^{18} + 248X^{17} + 3X^{16} + 193X^{15} + 194X^{14} + 205X^{13} + 9X^{12} +  
```

```
56X^{11} + 95X^{10} + 199X^{9} + 108X^{8} + 58X^{7} + 160X^{6} + 148X^{5} + 138X^{4} +  
```

```
24X^{3} + 210X^{2} + 26X + 1 \  
```

POLY_Q256_DEG30_SPARSE="1,0,26,1,210,2,24,3,\  

```
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,\  
```

727
POLY_Q256
DEG30
DENSE="1,26,210,24,138,148,
160,58,108,199,95,56,9,205,194,193,3,248,110,
150,24,169,192,212,112,144,97,109,174,253,1"

F256
BCH
write code for division d16:

F256
BCH
code d16 division:

F256
BCH
code d16 error:
10240 #CRC_FILE=allen_Gates
10241 CRC_FILE=javad-allahyari-Fs1E2JXM3Gc-unsplash
10242
10243 CRC_FILE EXTENSION=bmp
10244
10245 crc_encode_16:
10246 $ (ORBITER) -v 3 \
10247 $ -define F -finite_field -q 2 -end \
10248 $ -with F -do \
10249 $ -coding_theoretic_activity \
10250 $ -crc_encode_file_based $(CRC_FILE).$(CRC_FILE EXTENSION) $(CRC_FILE) crc1 
10251 6.bin crc16 771 \ 
10252 $ -end
10253
10254 #-rw-r--r-- 1 betten staff 646576 Aug 24 14:35 allen_Gates_crc16.bin
10255 #-rw-r--r-- 1 betten staff 21656232 Aug 24 15:35 javad-allahyari-Fs1E2JXM3Gc-unsplash_crc16.bin
10256
10257
10258
10259 crc_encode_32:
10260 $ (ORBITER) -v 3 \
10261 $ -define F -finite_field -q 2 -end \
10262 $ -with F -do \
10263 $ -coding_theoretic_activity \
10264 $ -crc_encode_file_based $(CRC_FILE).$(CRC_FILE EXTENSION) $(CRC_FILE) crc3 
10265 2.bin crc32 771 \ 
10266 $ -end
10267
10268 #-rw-r--r-- 1 betten staff 648262 Aug 24 14:34 allen_Gates_crc32.bin
10269
10270
10271
10272 #crc_encode_new:
10273 $ (ORBITER) -v 3 \
10274 $ -define F -finite_field -q 256 -end \
10275 $ -with F -do \
10276 $ -coding_theoretic_activity \
10277 $ -crc_new_file_based $(CRC_FILE).$(CRC_FILE EXTENSION) \
10278 $ -end
10279
10280
10281 introduce_errors_16.500000:
10282 $ (ORBITER) -v 3 \
10283 $ -introduce_errors \
10284

729
$\text{input $(CRC\_FILE)$} \\
$\text{output $(CRC\_FILE)$} \\
\text{block\_based\_error\_generator} \\
\text{block\_length 771} \\
\text{threshold 500000} \\
\text{file\_based\_error\_generator 500000} \\
\text{nb\_repeats 30} \\
\text{end} \\
\text{introduce\_errors\_32\_100000:} \\
\text{$(ORBITER) -v 3$} \\
\text{check\_errors\_16:} \\
\text{$(ORBITER) -v 3$} \\
\text{check\_errors\_32:} \\
\text{$(ORBITER) -v 3$} \\
\text{extract\_block:} \\
\text{$(ORBITER) -v 3$}
 SECTION CODING THEORY REED MULLER CODES:

RM 3 1 Hamming space diagram:

RM 3 1 draw:

# Section 10.5: Coding Theory - Reed-Muller codes

# creates RM 3 1 8-16.tex

# creates RM 3 1 diagram_8_16.csv

# creates RM 3 1 holes_8_16.csv

# creates RM 3 1 draw:

# creates RM 3 1 diagram_01_8_16.csv

# creates RM 3 1 holes_8_16.csv

# creates RM 3 1 draw:

<additional code snippets>
10378 ▶ ▶ ▶ -input_csv_file RM_3_1_diagram_01_8_16.csv \\
10379 ▶ ▶ ▶ -box_width 25 -bit_depth 8 \\
10380 ▶ ▶ ▶ -partition 4 16 16 \\
10381 ▶ ▶ -end \\
10382 ▶ ▶ $\$(ORBITER) -v 2 \\
10383 ▶ ▶ -draw_matrix \\
10384 ▶ ▶ ▶ -input_csv_file RM_3_1_diagram_8_16.csv \\
10385 ▶ ▶ ▶ -box_width 25 -bit_depth 8 \\
10386 ▶ ▶ ▶ -partition 4 16 16 \\
10387 ▶ ▶ -end \\
10388 ▶ ▶ open RM_3_1_diagram_8_16_draw.bmp \\
10389 \\
10390 \\
10391 RM_3_1.split: \\
10392 ▶ ▶ $\$(ORBITER) -split_by_values RM_3_1_holes_8_16.csv \\
10393 \\
10394 RM_3_1_holes_draw: \\
10395 ▶ ▶ $\$(ORBITER) -v 2 \\
10396 ▶ ▶ ▶ -loop L 0 3 1 \\
10397 ▶ ▶ ▶ ▶ -draw_matrix \\
10398 ▶ ▶ ▶ ▶ ▶ -input_csv_file RM_3_1_holes_8_16_value%L.csv \\
10399 ▶ ▶ ▶ ▶ ▶ ▶ -box_width 25 -bit_depth 8 -partition 5 16 16 \\
10400 ▶ ▶ ▶ ▶ -end \\
10401 ▶ ▶ ▶ -end_loop \\
10402 \\
10403 RM_3_1_hole0: \\
10404 ▶ ▶ $\$(ORBITER) -v 3 \\
10405 ▶ ▶ ▶ -define F -finite_field -q 2 -end \\
10406 ▶ ▶ ▶ -with F -do -finite_field_activity \\
10407 ▶ ▶ ▶ ▶ -algebraic_normal_form \\
10408 ▶ ▶ ▶ ▶ RM_3_1_holes_8_16_value0.csv 8 \\
10409 ▶ ▶ ▶ -end \\
10410 \\
10411 # E_0 = X_0X_8^{^7} + X_1X_8^{^7} + X_2X_8^{^7} + X_3X_8^{^7} + X_4X_8^{^7} + X_5X_8^{^7} + X_6X_8^{^7} + X_7X_8^{^7} \\
10412 \\
10413 \\
10414 RM_3_1_hole1: \\
10415 ▶ ▶ $\$(ORBITER) -v 3 \\
10416 ▶ ▶ ▶ -define F -finite_field -q 2 -end \\
10417 ▶ ▶ ▶ -with F -do -finite_field_activity \\
10418 ▶ ▶ ▶ ▶ -algebraic_normal_form \\
10419 ▶ ▶ ▶ ▶ RM_3_1_holes_8_16_value1.csv 8 \\
10420 ▶ ▶ ▶ -end \\
10421 \\
10422 #E_1 = X_0X_8^{^7} + X_1X_8^{^7} + X_2X_8^{^7} + X_3X_8^{^7} + X_4X_8^{^7} + X_5X_8^{^7} + X_6X_8^{^7} + X_7X_8^{^7} \\
10423 \\
732
10424 RM_3_1_hole2:
10425 ▷ $(\text{ORBITER}) -v 3 \$
10426 ▷ ▷ -define F -finite_field -q 2 -end \n10427 ▷ ▷ -with F -do -finite_field_activity \n10428 ▷ ▷ -algebraic_normal_form \n10429 ▷ ▷ RM_3_1_holes_8_16_value2.csv 8 \n10430 ▷ ▷ -end
10431
10432 #X0*X1*X8^6 + X0*X2*X8^6 + X0*X3*X8^6 + X0*X4*X8^6 + X0*X5*X8^6 + X0*X6*X8^6 + X0*
X7*X8^6 + X1*X2*X8^6 + X1*X3*X8^6 + X1*X4*X8^6 + X1*X5*X8^6 + X1*X6*X8^6 + X1*X7*
X8^6 + X2*X3*X8^6 + X2*X4*X8^6 + X2*X5*X8^6 + X2*X6*X8^6 + X2*X7*X8^6 + X3*X4*X8*
^6 + X3*X5*X8^6 + X3*X6*X8^6 + X3*X7*X8^6 + X4*X5*X8^6 + X4*X6*X8^6 + X4*X7*X8^6 +
X5*X6*X8^6 + X5*X7*X8^6 + X6*X7*X8^6 + X0*X1*X2*X8^5 + X0*X1*X3*X8^5 + X0*X1*X4*X8^5 +
X0*X1*X5*X8^5 + X0*X1*X6*X8^5 + X0*X1*X7*X8^5 + X0*X2*X3*X8^5 + X0*X2*X4*X8^5 + X0*X2*
X5*X8^5 + X0*X2*X6*X8^5 + X0*X2*X7*X8^5 + X0*X3*X4*X8^5 + X0*X3*X5*X8^5 + X0*X3*X6*X8^5 +
X0*X3*X7*X8^5 + X0*X4*X5*X8^5 + X0*X4*X6*X8^5 + X0*X4*X7*X8^5 + X0*X5*X6*X8^5 + X0*
X5*X7*X8^5 + X0*X6*X7*X8^5 + X0*X7*X8^5 + X1*X2*X3*X8^5 + X1*X2*X4*X8^5 + X1*X2*X5*X8^5 +
X1*X2*X6*X8^5 + X1*X2*X7*X8^5 + X1*X3*X4*X8^5 + X1*X3*X5*X8^5 + X1*X3*X6*X8^5 + X1*X3*
X7*X8^5 + X1*X4*X5*X8^5 + X1*X4*X6*X8^5 + X1*X4*X7*X8^5 + X1*X5*X6*X8^5 + X1*X5*X7*X8^5 +
X1*X6*X7*X8^5 + X1*X7*X8^5 + X2*X3*X4*X8^5 + X2*X3*X5*X8^5 + X2*X3*X6*X8^5 + X2*X3*X7*X8^5 +
X2*X4*X5*X8^5 + X2*X4*X6*X8^5 + X2*X4*X7*X8^5 + X2*X5*X6*X8^5 + X2*X5*X7*X8^5 + X2*X6*
X7*X8^5 + X2*X7*X8^5 + X3*X4*X5*X8^5 + X3*X4*X6*X8^5 + X3*X4*X7*X8^5 + X3*X5*X6*X8^5 +
X3*X5*X7*X8^5 + X3*X6*X7*X8^5 + X3*X7*X8^5 + X4*X5*X6*X8^5 + X4*X5*X7*X8^5 + X4*X6*
X7*X8^5 + X4*X7*X8^5 + X5*X6*X7*X8^5 + X5*X6*X8^5 + X5*X7*X8^5 + X6*X7*X8^5 + X0*X1*
X2*X3*X8^4 + X0*X1*X2*X4*X8^4 + X0*X1*X2*X5*X8^4 + X0*X1*X2*X6*X8^4 + X0*X1*X2*X7*X8^4 +
X0*X1*X3*X4*X8^4 + X0*X1*X3*X5*X8^4 + X0*X1*X3*X6*X8^4 + X0*X1*X3*X7*X8^4 + X0*X1*X4*
X5*X8^4 + X0*X1*X4*X6*X8^4 + X0*X1*X4*X7*X8^4 + X0*X1*X5*X6*X8^4 + X0*X1*X5*X7*X8^4 +
X0*X1*X6*X7*X8^4 + X0*X1*X6*X8^4 + X0*X1*X7*X8^4 + X0*X2*X3*X4*X8^4 + X0*X2*X3*X5*X8^4 +
X0*X2*X3*X6*X8^4 + X0*X2*X3*X7*X8^4 + X0*X2*X4*X5*X8^4 + X0*X2*X4*X6*X8^4 + X0*X2*X4*
X7*X8^4 + X0*X2*X5*X6*X8^4 + X0*X2*X5*X7*X8^4 + X0*X2*X6*X7*X8^4 + X0*X2*X6*X8^4 +
X0*X2*X7*X8^4 + X0*X3*X4*X5*X8^4 + X0*X3*X4*X6*X8^4 + X0*X3*X4*X7*X8^4 + X0*X3*X5*
X6*X8^4 + X0*X3*X5*X7*X8^4 + X0*X3*X6*X7*X8^4 + X0*X3*X6*X8^4 + X0*X3*X7*X8^4 + X0*
X4*X5*X6*X8^4 + X0*X4*X5*X7*X8^4 + X0*X4*X6*X7*X8^4 + X0*X4*X6*X8^4 + X0*X4*X7*
X8^4 + X0*X5*X6*X7*X8^4 + X0*X5*X6*X8^4 + X0*X5*X7*X8^4 + X0*X6*X7*X8^4 + X0*X6*X8^4 +
X0*X7*X8^4 + X1*X2*X3*X4*X8^4 + X1*X2*X3*X5*X8^4 + X1*X2*X3*X6*X8^4 + X1*X2*X3*X7*X8^4 +
X1*X2*X4*X5*X8^4 + X1*X2*X4*X6*X8^4 + X1*X2*X4*X7*X8^4 + X1*X2*X5*X6*X8^4 + X1*X2*
X5*X7*X8^4 + X1*X2*X6*X7*X8^4 + X1*X2*X6*X8^4 + X1*X2*X7*X8^4 + X1*X3*X4*X5*X8^4 +
X1*X3*X4*X6*X8^4 + X1*X3*X4*X7*X8^4 + X1*X3*X5*X6*X8^4 + X1*X3*X5*X7*X8^4 + X1*X3*
X6*X7*X8^4 + X1*X3*X6*X8^4 + X1*X3*X7*X8^4 + X1*X4*X5*X6*X8^4 + X1*X4*X5*X7*X8^4 +
X1*X4*X6*X7*X8^4 + X1*X4*X6*X8^4 + X1*X4*X7*X8^4 + X1*X5*X6*X7*X8^4 + X1*X5*X6*
X8^4 + X1*X5*X7*X8^4 + X1*X6*X7*X8^4 + X1*X6*X8^4 + X1*X7*X8^4 + X2*X3*X4*X5*X8^4 +
X2*X3*X4*X6*X8^4 + X2*X3*X4*X7*X8^4 + X2*X3*X5*X6*X8^4 + X2*X3*X5*X7*X8^4 + X2*X3*X6*
X7*X8^4 + X2*X3*X6*X8^4 + X2*X3*X7*X8^4 + X2*X4*X5*X6*X8^4 + X2*X4*X5*X7*X8^4 +
X2*X4*X6*X7*X8^4 + X2*X4*X6*X8^4 + X2*X4*X7*X8^4 + X2*X5*X6*X7*X8^4 + X2*X5*X6*
X8^4 + X2*X5*X7*X8^4 + X2*X6*X7*X8^4 + X2*X6*X8^4 + X2*X7*X8^4 + X3*X4*X5*X6*X8^4 +
X3*X4*X5*X7*X8^4 + X3*X4*X6*X7*X8^4 + X3*X4*X6*X8^4 + X3*X4*X7*X8^4 + X3*X5*X6*X8^4 +
X3*X5*X7*X8^4 + X3*X6*X7*X8^4 + X3*X6*X8^4 + X3*X7*X8^4
10433
10434 # E_2 + E_3 + E_4
10435
10436
10437
10438 RM_4_1:
10439 ▷ $(\text{ORBITER}) -v 2 \$
10440 ▷ ▷ -define F -finite_field -q 2 -end \n10441 ▷ ▷ -define C -code -field F -first_order_Reed_Muller 4 -end \n10442 ▷ ▷ -with C -and F -do -coding_theoretic_activity \n733
RM₄₁_old:

```plaintext
$(ORBITER) -v 2
```

```plaintext
define F -finite_field -q 2 -end
```

```plaintext
with F -do
```

```plaintext
-coding_theoretic_activity
```

```plaintext
linear_code_through_columns_of_parity_check 5
```

```plaintext
(REED_MULLER_4₁_COLUMNS_OF_PARITY_CHECK)
```

```plaintext
-end
```

```plaintext
pdflatex code_n16_k5_q2.tex
```

```plaintext
open code_n16_k5_q2.pdf
```

```plaintext
# codewords_n16_k5_q2.csv
```

```plaintext
RM₄₁_diagram:
```

```plaintext
$(ORBITER) -v 2
```

```plaintext
define F -finite_field -q 2 -end
```

```plaintext
with F -do
```

```plaintext
-coding_theoretic_activity
```

```plaintext
code_diagram_from_file "RM_4₁"
```

```plaintext
codewords_n16_k5_q2.csv 16
```

```plaintext
-end
```

```plaintext
#-enhance 4
```

```plaintext
#-metric_balls 3
```

```plaintext
RM₄₁_diagram_draw:
```

```plaintext
$(ORBITER) -v 2
```

```plaintext
-draw_matrix
```

```plaintext
-input_csv_file RM_4₁_diagram_01_16_32.csv
```

```plaintext
-box_width 25 -bit_depth 8
```

```plaintext
-partition 10 256 256
```

```plaintext
-end
```

```plaintext
open RM_4₁_diagram_01_16_32_draw.bmp
```

```plaintext
RM₄₁_split:
```

```plaintext
$(ORBITER) -split_by_values RM_4₁_holes_16_32.csv
```

734
RM_4_1_diagram_draw_holes:
$ (ORBITER) -v 2 \
-define F -finite_field -q 2 -end 
-with F -do 
-coding_theoretic_activity 
-with F -do -finite_field_activity 
-algebraic_normal_form 
-RM_4_1_holes_16_32_value0.csv 16 
-end 
-loop L 0 7 1 
-box_width 5 -bit_depth 8 
-partition 10 256 256 
-end 
-input_csv_file RM_4_1_holes_16_32.csv 
-box_width 5 -bit_depth 8 
-partition 10 256 256 
-end 
-end_loop

RM_4_1_diagram_metric_balls:
$ (ORBITER) -v 2 \
-define F -finite_field -q 2 -end 
-with F -do 
-coding_theoretic_activity 
-code_diagram_from_file "RM_4_1" 
-codewords_n16_k5_q2.csv 16 
-metric_balls 3 
-end 
-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
Section 10.6: Coding Theory - BCH Codes

SECTION CODING THEORY BCH CODES:

10547 draw_cyclotomic_mod_21_q8:
10548  $\text{ORBITER \ -v 2 \ -draw_options \ -radius 100 \ -line_width 1.0 \ -embedded \ -draw_mod_n \ -n 21 \ -file mod\_21\_cyclotomic \ -cyclotomic_sets 8 \ "1,2,4,5,7,10,13" \ -end \ -pdflatex \ mod\_21\_cyclotomic\_draw.tex \ -open mod\_21\_cyclotomic\_draw.pdf}$

10559 $\text{F}_8$ BCH code $d_3$:
10560  $\text{ORBITER \ -v 3 \ -define F \ -finite_field \ -q 8 \ -override_polynomial 11 \ -end \ -with F \ -do \ -coding_theoretic_activity \ -make_BCH_code 21 3 \ -end \ -pdflatex \ BCH\_codes\_q8\_n21\_d3.tex \ -open BCH\_codes\_q8\_n21\_d3.pdf}$

10569 #generator polynomial is $X^4 + 4X^3 + 4X^2 + 3X + 4$

10571 $\text{F}_8$ BCH code $d_4$:
10572  $\text{ORBITER \ -v 3 \ -define F \ -finite_field \ -q 8 \ -override_polynomial 11 \ -end \ -with F \ -do \ -coding_theoretic_activity \ -make_BCH_code 21 4 \ -end}$

10579 #generator polynomial is $X^5 + 6X^4 + 7X^3 + 2X + 3$

10582 $\text{F}_8$ BCH code $d_5$:
10583  $\text{ORBITER \ -v 3 \ -define F \ -finite_field \ -q 8 \ -override_polynomial 11 \ -end \ -with F \ -do \ -coding_theoretic_activity \ -make_BCH_code 21 5 \ -end}$

10578 #generator polynomial is $X^5 + 6X^4 + 7X^3 + 2X + 3$
\begin{verbatim}
define F -finite_field -q 8 -override_polynomial 11 -end \\
with F -do \\
coding_theoretic_activity \\
\end{verbatim}
10630 F_8_BCH_code_d5_minimum_distance:
10631 ▷ $\$(ORBITER) -v 2 \n10632 ▷ ▷ -define F -finite_field -q 8 -override_polynomial 11 -end \n10633 ▷ ▷ -define v -vector -format 14 -field F \n10634 ▷ ▷ ▷ -compact $(CODE_BCH_F8_N21_D5_GENMA_OVERRIDE_POLYNOMIAL11) \n10635 ▷ ▷ ▷ -end \n10636 ▷ ▷ -with F -do \n10637 ▷ ▷ ▷ -coding_theoretic_activity \n10638 ▷ ▷ ▷ ▷ -minimum_distance v \n10639 ▷ ▷ ▷ -end
10640 # important: use the same polynomial as when creating the code.
10641 #
10642 # d=5
10643
10644
10645
10646 F_8_BCH_code_d7:
10647 ▷ $\$(ORBITER) -v 3 \n10648 ▷ ▷ -define F -finite_field -q 8 -override_polynomial 11 -end \n10649 ▷ ▷ -with F -do \n10650 ▷ ▷ ▷ -coding_theoretic_activity \n10651 ▷ ▷ ▷ ▷ -make_BCH_code 21 7 \n10652 ▷ ▷ ▷ -end
10653
10654
10655
10656
10657 F8_BCH_code_n63_d43:
10658 ▷ $\$(ORBITER) -v 3 \n10659 ▷ ▷ -define F -finite_field -q 8 -override_polynomial 11 -end \n10660 ▷ ▷ -with F -do \n10661 ▷ ▷ ▷ -coding_theoretic_activity \n10662 ▷ ▷ ▷ ▷ -make_BCH_code 63 43 \n10663 ▷ ▷ ▷ -end
10664 ▶ pdflatex BCH_codes_q8_n63_d43.tex
10665 ▶ open BCH_codes_q8_n63_d43.pdf
10666
10667
10668
10669
10670 CODE_BCH_F8_N63_K9_D43_GENMA="\n10671 4,5,2,2,4,5,6,2,4,2,6,0,0,7,1,5,3,7,1,0,0,5,6,4,7,6,7,7,1,2,6,3,1,6,0,0,6,6,6,4,7,7,0,0,3,4,7,5,6,2,5,1,4,4,1,0,0,0,0,0,0,\n10672 0,4,5,2,2,4,5,6,2,4,2,6,0,0,7,1,5,3,7,1,0,0,5,6,4,7,6,7,7,1,2,6,3,1,6,0,0,6,6,6,4,7,7,0,0,3,4,7,5,6,2,5,1,4,4,1,0,0,0,0,0,\n10673 0,0,4,5,2,2,4,5,6,2,4,2,6,0,0,7,1,5,3,7,1,0,0,5,6,4,7,6,7,7,1,2,6,3,1,6,0,0,6,6,6,4,7,7,0,0,3,4,7,5,6,2,5,1,4,4,1,0,0,0,0,0,0,\n"
The minimum distance is \( d = 45 \), computed in 0 days, 0 hours, 1 minutes, 32 seconds.

#coding_theory_domain::do_minimum_distance

The minimum distance is \( d = 45 \), computed in 0 days, 0 hours, 1 minutes, 32 seconds.
10714 \> \> -define F -finite_field -q 7 -end \  10715 \> \> -with F -do \  10716 \> \> \> -coding_theoretic_activity \  10717 \> -make_BCH_code 6 3 \  10718 \> \> -end  10719  10720  10721 F_64_again:  10722 \> $\text{(ORBITER)} -v 3 \  10723 \> -define F -finite_field -q 64 -end \  10724 \> -with F -do \  10725 \> \> -finite_field_activity \  10726 \> \> -cheat_sheet_GF \  10727 \> \> -end  10728 \> pdflatex GF_64.tex  10729 \> open GF_64.pdf  10730  10731 10732 BCH_255_5_evaluate_elementary_symmetric_functions_1:  10733 \> $\text{(ORBITER)} -v 3 -define F -finite_field -q 256 -end \  10734 \> -define e1 -formula "e1" "e1" "" $(ELEMENTARY_SYMMETRIC_8.1) \  10735 \> -define e2 -formula "e2" "e2" "" $(ELEMENTARY_SYMMETRIC_8.2) \  10736 \> -define e3 -formula "e3" "e3" "" $(ELEMENTARY_SYMMETRIC_8.3) \  10737 \> -define e4 -formula "e4" "e4" "" $(ELEMENTARY_SYMMETRIC_8.4) \  10738 \> -define e5 -formula "e5" "e5" "" $(ELEMENTARY_SYMMETRIC_8.5) \  10739 \> -define e6 -formula "e6" "e6" "" $(ELEMENTARY_SYMMETRIC_8.6) \  10740 \> -define e7 -formula "e7" "e7" "" $(ELEMENTARY_SYMMETRIC_8.7) \  10741 \> -define e8 -formula "e8" "e8" "" $(ELEMENTARY_SYMMETRIC_8.8) \  10742 \> -define E8 -collection "e1,e2,e3,e4,e5,e6,e7,e8" \  10743 \> -with F -do -finite_field_activity \  10744 \> \> -evaluate E8 "x0=2,x1=4,x2=16,x3=29,x4=76,x5=157,x6=95,x7=133" -end  10745 10746 BCH_255_5_evaluate_elementary_symmetric_functions_2:  10747 \> $\text{(ORBITER)} -v 3 -define F -finite_field -q 256 -end \  10748 \> -define e1 -formula "e1" "e1" "" $(ELEMENTARY_SYMMETRIC_8.1) \  10749 \> -define e2 -formula "e2" "e2" "" $(ELEMENTARY_SYMMETRIC_8.2) \  10750 \> -define e3 -formula "e3" "e3" "" $(ELEMENTARY_SYMMETRIC_8.3) \  10751 \> -define e4 -formula "e4" "e4" "" $(ELEMENTARY_SYMMETRIC_8.4) \  10752 \> -define e5 -formula "e5" "e5" "" $(ELEMENTARY_SYMMETRIC_8.5) \  10753 \> -define e6 -formula "e6" "e6" "" $(ELEMENTARY_SYMMETRIC_8.6) \  10754 \> -define e7 -formula "e7" "e7" "" $(ELEMENTARY_SYMMETRIC_8.7) \  10755 \> -define e8 -formula "e8" "e8" "" $(ELEMENTARY_SYMMETRIC_8.8) \  10756 \> -define E8 -collection "e1,e2,e3,e4,e5,e6,e7,e8" \  10757 \> -with F -do -finite_field_activity \  10758 \> \> -evaluate E8 "x0=8,x1=4,x2=16,x3=29,x4=76,x5=157,x6=95,x7=133" -end  10759  10760
BCH15:
$\texttt{\#(ORBITER)}} \texttt{-BCH 15 2 3}$
$\texttt{\$(ORBITER)}} \texttt{-define F -finite_field -q 2 -end \}$
$\texttt{\$(ORBITER)}} \texttt{-with F -do \}$
$\texttt{\$(ORBITER)}} \texttt{-coding_theoretic_activity \}$
$\texttt{\$(ORBITER)}} \texttt{-BCH 15 2 5 \}$
$\texttt{\$(ORBITER)}} \texttt{-BCH 15 2 7}$
$\texttt{\$(ORBITER)}} \texttt{-BCH 15 2 9}$

draw_mod_31:
$\texttt{\$(ORBITER)}} \texttt{-v 2 \}$
$\texttt{\$(ORBITER)}} \texttt{-draw_options -embedded -end \}$
$\texttt{\$(ORBITER)}} \texttt{-draw_mod_n 31 \}$
$\texttt{\$(ORBITER)}} \texttt{-file mod_31 \}$
$\texttt{\$(ORBITER)}} \texttt{-draw_mod_n_power_cycle 2 \}$
$\texttt{\$(ORBITER)}} \texttt{-end}$
$\texttt{pdflatex mod_31.draw.tex}$
$\texttt{open mod_31.draw.pdf}$

PR127:
$\texttt{\$(ORBITER)}} \texttt{-v 5 -primitive_root 127}$

draw_mod_127_power:
$\texttt{\$(ORBITER)}} \texttt{-v 2 \}$
$\texttt{\$(ORBITER)}} \texttt{-draw_options -scale 0.4 -embedded -end \}$
$\texttt{\$(ORBITER)}} \texttt{-draw_mod_n 127 mod_127 -draw_mod_n_power_cycle 3}$
$\texttt{pdflatex mod_127.draw.tex}$
$\texttt{open mod_127.draw.pdf}$

draw_mod_251:
$\texttt{\$(ORBITER)}} \texttt{-v 2 \}$
$\texttt{\$(ORBITER)}} \texttt{-draw_options -nodes_empty -radius 10 -embedded -end \}$
$\texttt{\$(ORBITER)}} \texttt{-draw_mod_n 251 mod_251}$
$\texttt{pdflatex mod_251.draw.tex}$
$\texttt{open mod_251.draw.pdf}$
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

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

BCH_21.poly_mult_mod_F4:

$(ORBITER) -v 2 

-define F -finite_field -q 4 -end 

-with F -do 

-finite_field_activity 

743
10902 \triangleright \triangleright \triangleright \text{-polynomial_mult_mod "1,0,2,1" "1,0,3,1"} \\
10903 \triangleright \triangleright \triangleright \text{"1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1"} \\
10904 \triangleright \triangleright \text{-end} \\
10905 \\
10906 \#C(X)=X^{-6} + X^{-5} + X^{-4} + X^{-2} + 1 \\
10907 \\
10908 \# poly 1,0,1,0,1,1,1 \\
10909 \\
10910 \\
10911 \text{BCH}_21\text{.poly_division}_a: \\
10912 \triangleright \$(ORBITER) -v 2 \ \\
10913 \triangleright \triangleright \text{-define F -finite_field -q 4 -end} \ \\
10914 \triangleright \triangleright \text{-with F -do} \ \\
10915 \triangleright \triangleright \text{-finite_field_activity} \ \\
10916 \triangleright \triangleright \text{-polynomial_division} \ \\
10917 \triangleright \triangleright \text{"1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1"} \ \\
10918 \triangleright \triangleright \text{"1,0,2,1"} \ \\
10919 \triangleright \triangleright \text{-end} \ \\
10920 \\
10921 \text{BCH}_21\text{.poly_division}_b: \\
10922 \triangleright \$(ORBITER) -v 2 \ \\
10923 \triangleright \triangleright \text{-define F -finite_field -q 4 -end} \ \\
10924 \triangleright \triangleright \text{-with F -do} \ \\
10925 \triangleright \triangleright \text{-finite_field_activity} \ \\
10926 \triangleright \triangleright \text{-polynomial_division} \ \\
10927 \triangleright \triangleright \text{"1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1"} \ \\
10928 \triangleright \triangleright \text{"1,0,3,1"} \ \\
10929 \triangleright \triangleright \text{-end} \ \\
10930 \\
10931 \\
10932 \text{BCH}_21\text{.poly_division}_ab: \\
10933 \triangleright \$(ORBITER) -v 2 \ \\
10934 \triangleright \triangleright \text{-define F -finite_field -q 4 -end} \ \\
10935 \triangleright \triangleright \text{-with F -do} \ \\
10936 \triangleright \triangleright \text{-finite_field_activity} \ \\
10937 \triangleright \triangleright \text{-polynomial_division} \ \\
10938 \triangleright \triangleright \text{"1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1"} \ \\
10939 \triangleright \triangleright \text{"1,0,1,0,1,1,1"} \ \\
10940 \triangleright \triangleright \text{-end} \ \\
10941 \\
10942 \text{BCH}_21\text{.generator_matrix}: \\
10943 \triangleright \$(ORBITER) -v 2 \ \\
10944 \triangleright \triangleright \text{-define F -finite_field -q 4 -end} \ \\
10945 \triangleright \triangleright \text{-with F -do} \ \\
10946 \triangleright \triangleright \text{-coding_theoretic_activity} \ \\
10947 \triangleright \triangleright \triangleright \text{-generator_matrix_cyclic_code} \ \\
10948 \triangleright \triangleright \triangleright \triangleright 21 "1,0,1,0,1,1,1"
BCH\_21\_15\_weight\_enumerator:

```plaintext
$(\text{ORBITER}) -v 2 \\
\text{define } F \text{ -finite_field -q 4 -end} \\
\text{define } v \text{ -vector -field F -format 15} \\
\text{dense } $(\text{BCH\_21\_15\_GENERATOR\_MATRIX}) \\
\text{end} \\
\text{define } C \text{ -code -field F} \\
\text{linear_code_through_generator_matrix } v \\
\text{end} \\
\text{with } C \text{ -do} \\
\text{coding\_theoretic\_activity} \\
\text{weight\_enumerator} \\
\text{end} \\
\text{too slow!}
```

BCH\_21\_15\_dual:

```plaintext
$(\text{ORBITER}) -v 2 \\
\text{define } F \text{ -finite_field -q 4 -end} \\
\text{define } v \text{ -vector -field F -format 15} \\
\text{dense } $(\text{BCH\_21\_15\_GENERATOR\_MATRIX}) \text{ -end} \\
\text{with } F \text{ -do -finite_field\_activity} \\
\text{nullspace } v \\
\text{normalize\_from\_the\_right} \\
\text{end} \\
```

BCH\_21\_6\_weight\_enumerator:

```plaintext
$(\text{ORBITER}) -v 2 \\
\text{define } F \text{ -finite_field -q 4 -end} \\
\text{define } v \text{ -vector -field F -format 6 -field F} \\
\text{dense } $(\text{BCH\_21\_6\_GENERATOR\_MATRIX}) \\
\text{end} \\
\text{define } C \text{ -code -field F} \\
\text{linear_code_through_generator_matrix } v \\
\text{end} \\
\text{with } F \text{ -do} \\
\text{coding\_theoretic\_activity} \\
\text{weight\_enumerator} \\
\text{end} \\
```

```
y^{21} + 63x^8y^{13} + 294x^{12}y^9 + 756x^{14}y^7 + 1890x^{16}y^5 + 1092x^{18}y^{17} + 585x^{20}y^9 + 108x^{22}y^7 + 7x^{24}y^5 + x^{26}y^3 + x^{28}y + 1 
```
10996 \#y^3
10997 \#( 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 294, 0, 0, 0, 756, 0, 1890, 0, 1092, 0, 0, 0 )
10998
10999
11000
11001 BCH_21.6.4_macwilliams:
11002 \$ (ORBITER) -v 2 \$
11003 \$ -make_macwilliams_system 21 6 4 $
11004 \$ pdflatex MacWilliams_n21_k6_q4.tex $
11005 \$ open MacWilliams_n21_k6_q4.pdf $
11006
11007
11008
11009 \#ww := [1, 0, 0, 84, 252, 1575, 10080, 58032, 319662, 1411116, 5133744, 15282792, 37951620, 79336530, 135622080, 190615824, 213273081, 188911548, 125744304, 59721732, 17767512, 2580255]
11010
11011
11012 BCH_21.15.4_field_reduction:
11013 \$ (ORBITER) -v 2 \$
11014 \$ -define F -finite_field -q 4 -end \$
11015 \$ -with F -do \$
11016 \$ -finite_field_activity \$
11017 \$ -field_reduction "BCH_21.15" 2 15 21 $(BCH_21.15) \$
11018 \$ -end \$
11019 \$ $(ORBITER) -v 2 \$
11020 \$ -draw_matrix \$
11021 \$ -input_csv_file BCH_21.15.4.csv \$
11022 \$ -box_width 20 -bit_depth 24 \$
11023 \$ -partition 4 "30" "42" \$
11024 \$ -end \$
11025 \$ pdflatex field_reduction_Q4_q2.15.21.tex \$
11026 \$ open BCH_21.15.4_draw.bmp \$
11027 \$ #open field_reduction_Q4_q2.15.21.pdf \$
11028
11029 \# poly of degree 12: 1,0,1,0,0,0,1,0,0,0,1,0,1
11030
11031 BCH_21.poly_division_c:
11032 \$ (ORBITER) -v 2 \$
11033 \$ -define F -finite_field -q 2 -end \$
11034 \$ -with F -do \$
11035 \$ -finite_field_activity \$
11036 \$ -polynomial_division \$
11037 \$ -"1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1" \$
11038 \$ -"1,0,1,0,0,0,1,0,0,0,1" \$
746
F16_roots_5:
$(ORBITER) -v 3 -define F -finite_field -q 2 -end -with F -do -coding_theoretic_activity -nth_roots 5 -end
pdflatex Nth_roots_q2_n5.tex
open Nth_roots_q2_n5.pdf

F64_roots_21:
$(ORBITER) -v 3 -define F -finite_field -q 2 -end -with F -do -coding_theoretic_activity -nth_roots 21 -end
pdflatex Nth_roots_q2_n21.tex
open Nth_roots_q2_n21.pdf

BCH_F256_roots_771:
$(ORBITER) -v 3 -define F -finite_field -q 256 -end -with F -do -coding_theoretic_activity -nth_roots 771 -end
pdflatex BCH_codes_q256_n771_d16.tex
open BCH_codes_q256_n771_d16.pdf

#generator polynomial is X^{30} + 253X^{29} + 174X^{28} + 109X^{27} + 97X^{26} + 144X^{25} + 112X^{24} + 212X^{23} + 192X^{22} + 169X^{21} + 24X^{20} + 150X^{19} + 110X^{18} + 248X^{17} + 3X^{16} + 193X^{15} + 194X^{14} + 205X^{13} + 9X^{12} + 56X^{11} + 95X^{10} + 199X^{9} + 108X^{8} + 58X^{7} + 160X^{6} + 148X^{5} + 138X^{4} + 126X^{3} + 7X^{2} + 44X^{1} + 1$
\[ T^4 + 24T^3 + 210T^2 + 26T + 1 \]

11083
11084
11085
11086
11087
11088
11089
11090
11091

# Section 10.7: Coding Theory - Reed Solomon codes

11092
11093
11094
11095
11096
11097
11098
11099

F_7.BCH.code.n6:

11100
11101
11102
11103
11104
11105
11106
11107
11108

RREF_RS_6_4.7.weightenumerator:

11109
11110
11111
11112
11113
11114
11115
11116
11117
11118
11119
11120
11121
11122

#1y^6 + 120x^3y^3 + 360x^4y^2 + 972x^5y + 948x^6

11123
11124
11125
11126
11127
11128
\begin{verbatim}
11129
11130  Code_RS_11:
11131  $(ORBITER) -v 2 \\
11132  -define F -finite_field -q 11 -end \\
11133  -define v -vector -format 8 -field F \\
11134  -compact $(CODE_RS_10_8_11) \\
11135  -end \\
11136  -with F -do \\
11137  -finite_field_activity -RREF v -end \\
11138  pdflatex RREF_example_q11_8_10.tex \\
11139  #gs -sDEVICE=png16 -dFIXEDMEDIA -dDEVICEWIDTHPOINTS=500 -dDEVICEHEIGHTPOINTS=450 \\
11140  # -r240 -oRREF_example_q11_8_10.png \\
11141  open RREF_example_q11_8_10.pdf
11142
11143  Code_RS_11_weight Enumerator:
11144  $(ORBITER) -v 2 \\
11145  -define F -finite_field -q 11 -end \\
11146  -define v -vector -format 8 -field F \\
11147  -compact $(CODE_RS_11_RREF) \\
11148  -end \\
11149  -define C -code -field F \\
11150  -linear_code_through_generator_matrix v \\
11151  -end \\
11152  -with C -do \\
11153  -coding_theoretic_activity \\
11154  -weight Enumerator \\
11155  -end \\
11156
11157  #1*y^10 + 1200*x^3*y^7 + 16800*x^4*y^6 + 209160*x^5*y^5 + 1734600*x^6*y^4 + 9918000*x^7*y^3 + 37189800*x^8*y^2 + 82644700*x^9*y + 82644620*x^(10) \\
11158
11159  RREF_RS_8_weight Enumerator:
11160  $(ORBITER) -v 2 \\
11161  -define F -finite_field -q 8 -end \\
11162  -define v -vector -format 5 -field F \\
11163  -compact $(CODE_RS_8) \\
11164  -end \\
11165  -define C -code -field F \\
11166  -linear_code_through_generator_matrix v \\
11167  -end \\
11168  -with C -do \\
11169
\end{verbatim}
# the group cannot be computed

**RS\textsubscript{8} field reduction:**

```
$($\texttt{ORBITER}) \texttt{-v 2 \}
$\texttt{-define F -finite_field -q 8 -end \}
$\texttt{-with F -do \}
$\texttt{-finite_field_activity \}
$\texttt{-field_reduction \"RS\textsubscript{8}_red\_2\" \}
$\texttt{2 5 7 \$(\texttt{CODE}\_RS\_8)} \}
$\texttt{-end \}
$\texttt{\$(\texttt{ORBITER}) \texttt{-draw_matrix -input_csv_file RS\textsubscript{8}_red\_2.csv \}}
$\texttt{-box_width 40 -bit_depth 24 \}
$\texttt{3,3,3,3,3" 3,3,3,3,3,3" -end \}
$\texttt{pdflatex field_reduction_Q8_q2\_5\_7.tex}
$\texttt{open field_reduction_Q8_q2\_5\_7.pdf}
```

**R\textsubscript{REF}RS\textsubscript{8} reduced weight enumerator:**

```
$\texttt{\$(\texttt{ORBITER}) \texttt{-v 2 \}}$
$\texttt{-define F -finite_field -q 2 -end \}
$\texttt{-define v -vector -format 15 -field F \}
$\texttt{-compact \$(RS\textsubscript{8} reduced) \}
$\texttt{-end \}
$\texttt{-define C -code -field F \}
$\texttt{-linear_code_through_generator_matrix v \}
$\texttt{-end \}
$\texttt{-with C -do \}
$\texttt{-coding_theoretic_activity \}
$\texttt{-weight enumerator \}
$\texttt{-end}
```

**CODE\textsubscript{21\_15\_4} store:**

```
$\texttt{\$(\texttt{ORBITER}) \texttt{-v 2 \}}$
$\texttt{-store_as_csv_file \"code\_21\_15\_4.csv\" \}
$\texttt{15 21 \$(\texttt{CODE}\_21\_15\_4) \}
$\texttt{\$(\texttt{ORBITER}) \texttt{-draw_matrix \}}$
$\texttt{-input_csv_file code\_21\_15\_4.csv \}
$\texttt{-box_width 40 -bit_depth 24 \}
$\texttt{-partition 4 \"15" \"21\" \}
```

750
CODE_21_15_4.weight Enumerator:

```bash
$ (ORBITER) -v 2 \ndefine F -finite_field -q 2 -end \ndefine v -vector -format 15 -field F \ndelete $(CODE_21_15_4) \ndelete -end \ndelete -define C -code -field F \ndelete -linear_code_through_generator_matrix v \ndelete -end \ndelete -define F -finite_field -q 2 -end \ndelete -define v -vector -format 15 -field F \ndelete -compact $(CODE_21_15_4) \ndelete -end \ndelete -with C -do \ndelete -coding_theoretic_activity \ndelete -weight Enumerator \ndelete -end
```

CODE_21_15_4.minimum distance:

```bash
$ (ORBITER) -v 2 \ndelete -define F -finite_field -q 2 -end \ndelete -define v -vector -format 15 -field F \ndelete -compact $(CODE_21_15_4) \ndelete -end \ndelete -with F -do \ndelete -coding_theoretic_activity \ndelete -minimum_distance v \ndelete -end
```

```bash
# d=4
```

Reed_soman_F8_work:

```bash
$ (ORBITER) -v 3 -define F -finite_field -q 8 -end \ndelete -with F -do -finite_field_activity \ndelete -parse_and_evaluate "test" "" "(t-a)*(t-a*a)" "a=2" -end
```

```
# Section 10.8: Coding Theory - Bounds
```

```
SECTION CODING THEORY BOUNDS:
```

```bash
bounds_for_d_given n6_k4_q7:
```

```bash
$ (ORBITER) -v 2 \ndelete -make_bounds_for_d_given_n_and_k_and_q 6 4 7
```

```bash
bounds_for_d_given n15_k6_q2:
```

```bash
$ (ORBITER) -v 2 \n```
coding_theory_bounds_q2:
$\text{(ORBITER)} -v 2 -table_of_bounds 20 2$

# produces table_of_bounds_n20_q2.csv

coding_theory_bounds_q8:
$\text{(ORBITER)} -v 2 -table_of_bounds 20 8$

gv_n15_k6_d5:
$\text{(ORBITER)} -v 2$

# [15,6] code created

bounds_for_d_given_n12_k4_q13:
$\text{(ORBITER)} -v 2$

GV_n15_k6_d5_weight Enumerator:
$\text{(ORBITER)} -v 2$

752
11315 \> \> -with C -do \ 
11316 \> \> -coding.theoretic_activity \ 
11317 \> \> \> -weight enumerator \ 
11318 \> \> -end \ 
11319 \ 
11320 \> #y^{15} + 27x^6y^9 + 24x^8y^7 + 9x^{10}y^5 + 3x^{12}y^3 \ 
11321 \> # surprise: d = 6 \ 
11322 \ 
11323 \ 
11324 code_n15_k6_d6_a_we: \ 
11325 \> $(ORBITER) -v 2 \ 
11326 \> \> -define F -finite_field -q 2 -end \ 
11327 \> \> -define v -vector -format 6 -field F \ 
11328 \> \> \> -compact $(CODE_15_6_6.A) \ 
11329 \> \> -end \ 
11330 \> \> -define C -code -field F \ 
11331 \> \> \> -linear_code_through_generator_matrix v \ 
11332 \> \> -end \ 
11333 \> \> -with C -do \ 
11334 \> \> -coding.theoretic_activity \ 
11335 \> \> \> -weight.enumerator \ 
11336 \> \> -end \ 
11337 \ 
11338 \> #y^{15} + 27x^6y^9 + 24x^8y^7 + 9x^{10}y^5 + 3x^{12}y^3 \ 
11339 \ 
11340 \ 
11341 \ 
11342 \> # weight enumerator \ 
11343 \> #y^{15} + 28x^6y^9 + 21x^8y^7 + 12x^{10}y^5 + 2x^{12}y^3 \ 
11344 \ 
11345 \ 
11346 code_n15_k6_d6_RREF: \ 
11347 \> $(ORBITER) -v 2 \ 
11348 \> \> -define F -finite_field -q 2 -end \ 
11349 \> \> -define v -vector -format 6 -field F \ 
11350 \> \> \> -compact $(CODE_GV_N15_K6) \ 
11351 \> \> -end \ 
11352 \> \> -with F -do -finite_field_activity \ 
11353 \> \> -RREF v -normalize_from_the_right \ 
11354 \> \> -end \ 
11355 \> pdflatex RREF.example.q2.6.15.tex \ 
11356 \> open RREF_example.q2.6.15.pdf \ 
11357 \ 
11358 code_n15_k6_d6_check_RREF: \ 
11359 \> $(ORBITER) -v 2 \ 
11360 \> \> -define F -finite_field -q 2 -end \ 
11361 \> \> -define v -vector -format 9 -field F \ 

753
11362 \> \> \> -compact $(\text{CODE}\_\text{GV}\_N15\_K6\_CHECK}) \ \\
11363 \> \> \> -end \ \\
11364 \> \> -with F -do \text{finite}\_\text{field}\_\text{activity} \ \\
11365 \> \> -\text{RREF} v \text{-normalize}\_\text{from}\_\text{the}\_\text{right} \ \\
11366 \> \> -end \ \\
11367 \> pdflatex \text{RREF}\_\text{example}\_q2\_9\_15.tex \ \\
11368 \> open \text{RREF}\_\text{example}\_q2\_9\_15.pdf \ \\
11369 \ \\
11370 \ \\
11371 \ \\
11372 \ \\
11373 \ \\
11374 \# Section 10.9: Coding Theory - Classification \ \\
11375 \ \\
11377 SECTION\_CODING\_THEORY\_CLASSIFICATION: \ \\
11378 \ \\
11379 \ \\
11380 \ \\
11381 \# code classification: \ \\
11382 \ \\
11383 codes_8_4_4: \ \\
11384 \> $(\text{ORBITER}) -v 6 \ \\
11385 \> \> -\text{orbiter}\_\text{path} $(\text{ORBITER}\_\text{PATH}) \ \\
11386 \> \> -\text{define} G \ \\
11387 \> \> -\text{linear}\_\text{group} -\text{PGL}\ 4\ 2 -\text{end} \ \\
11388 \> \> -with G -do \ \\
11389 \> \> -\text{group}\_\text{theoretic}\_\text{activity} \ \\
11390 \> \> -\text{poset}\_\text{classification}\_\text{control} \ \\
11391 \> \> \> -\text{problem}\_\text{label} codes_8_4_4 \ \\
11392 \> \> \> -\text{draw}\_\text{poset} \ \\
11393 \> \> \> -\text{draw}\_\text{options} -\text{embedded} -\text{radius} 250 \ \\
11394 \> \> \> -\text{line}\_\text{width} 1.0 -\text{spanning}\_\text{tree} -\text{end} \ \\
11395 \> \> \> -\text{report} -\text{end} \ \\
11396 \> \> -end \ \\
11397 \> \> -\text{linear}\_\text{codes} 3\ 8 \ \\
11398 \> \> -end \ \\
11399 \> pdflatex codes_8_4_4\_\text{poset}\_\text{lvl}\_8.tex \ \\
11400 \> open codes_8_4_4\_\text{poset}\_\text{lvl}\_8.pdf \ \\
11401 \> pdflatex codes_8_4_4\_\text{poset}.tex \ \\
11402 \> open codes_8_4_4\_\text{poset}.pdf \ \\
11403 \ \\
11404 codes_8_4_4\_\text{draw}: \ \\
11405 \> $(\text{ORBITER}) -v 3 \ \\
11406 \> \> -\text{draw}\_\text{layered}\_\text{graph} \ \\
11407 \> \> \> codes_8_4_4\_\text{poset}\_\text{lvl}\_8\_\text{layered}\_\text{graph} \ \\
11408 \> \> \> -\text{radius} 250 -\text{embedded} -\text{line}\_\text{width} 1.0 \ \\

754
codes_14_4_9_3:

```latex
$\text{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 
-linear_codes 9 14 
-end
```
```
$\text{pdflatex} codes_14_4_9_3.poset_lvl_13.tex$
```
```
open codes_14_4_9_3.poset_lvl_13.pdf
```

```
codes_15_6_6_2:
```
define G
-linear_group -PGL 9 2 -end  
-with G -do
-group_theoretic_activity 
-poset_classification_control 
-problem_label codes_15_6_6_2 
-draw_poset 
-draw_options 
-embedded -radius 250 
-end 
-end 
-linear_codes 6 15 
-end
```
```
$\text{pdflatex} codes_15_6_6_2.poset_lvl_15.tex$
```
```
open codes_15_6_6_2.poset_lvl_15.pdf
```

# ToDo
codes_16_5_9_3:

$ (ORBITER) -v 6 \n$ (ORBITER) -c -codes_classify -n 16 -k 5 -q 3 -d 9 -W -lex \n$ (ORBITER) -draw_poset \n$ (ORBITER) -end

# 5/31/2020: 28 min 22 sec on Mac

#0 : 1 orbits
#1 : 1 orbits
#2 : 1 orbits
#3 : 1 orbits
#4 : 1 orbits
#5 : 1 orbits
#6 : 1 orbits
#7 : 1 orbits
#8 : 1 orbits
#9 : 2 orbits
#10 : 3 orbits
#11 : 4 orbits
#12 : 5 orbits
#13 : 5 orbits
#14 : 4 orbits
#15 : 3 orbits
#16 : 1 orbits

#total: 36

codes_d4:

$ (ORBITER) -v 3 \n$ (ORBITER) -d -define G -linear_group -PGL 4 2 -end \n$ (ORBITER) -g -with G -do \n$ (ORBITER) -t -group_theoretic_activity \n$ (ORBITER) -p -poset_classification_control -W \n$ (ORBITER) -l -problem_label codes_r4_d4 -draw_poset \n$ (ORBITER) -e -embedded -end -linear_codes 4 100 \n$ (ORBITER) -e -end

codes_24_12_8:

$ (ORBITER) -v 6 \n$ (ORBTER) -p -orbiter_path $(ORBITER_PATH) \n$ (ORBITER) -d -define G \n$ (ORBITER) -l -linear_group -PGL 12 2 -end \n
756
with G -do \
-group_theoretic_activity \\
-poset_classification_control \\
-problem_label codes_24_12_8 \\
-draw_poset \\
draw_options -embedded -radius 250 \\
-line_width 1.0 -spanning_tree -end \\
-report -end \\
-end \\
-linear_codes 8 24 \\
-end \\
pdflatex codes_24_12_8_poset.tex \\
open codes_24_12_8_poset.pdf 
#codes_24_12_8_posetlvl24.layered_graph 

codes_24_12_8_draw: 
$(ORBITER) -v 3 \\
draw_layered_graph \\
codes_24_12_8_posetlvl24.layered_graph \\
-radius 100 -spanning_tree -embedded \\
-line_width 0.5 -x_stretch 1.4 \\
-scale 0.25 -nodes_empty \\
-end \\
pdflatex codes_24_12_8_posetlvl24_draw.tex \\
open codes_24_12_8_posetlvl24_draw.pdf 

glynn_arc: 
$(ORBITER) -v 5 \\
-orbiter_path $(ORBITER_PATH) \\
define G \\
-linear_group -PGGL 5 9 -end \\
-with G -do \\
group_theoretic_activity \\
poset_classification_control \\
-problem_label glynn_arc \\
draw_options -embedded -radius 250 \\
-line_width 1.0 -spanning_tree -end \\
draw_poset \\
-report -end \\
-end \\
-linear_codes 6 10 \\
-end \\
pdflatex glynn_arc_poset.tex \\
open glynn_arc_poset.pdf 

757
five_points_in_general:
$(ORBITER) -v 5 \\
- orbiter_path $(ORBITER_PATH) \\
-define G \\
-linear_group -PGL 4 2 -end \\
-with G -do \\
-group_theoretic_activity \\
poset_classification_control \\
- problem_label five_points_in_general \\
-draw_options -embedded -radius 250 \\
-line_width 1.0 -spanning_tree -end \\
-draw_poset \\
-report -end \\
-end \\
-linear_codes 4 5 \\
-end \\
pdflatex five_points_in_general.poset.tex \\
open five_points_in_general_poset.pdf

codes_q13_12_4:
$(ORBITER) -v 6 \\
- orbiter_path $(ORBITER_PATH) \\
-define G \\
-linear_group -PGL 4 13 -end \\
-with G -do \\
-group_theoretic_activity \\
poset_classification_control \\
- problem_label codes_q13 \\
-report -end \\
-end \\
-linear_codes 6 12 \\
-end \\
pdflatex codes_q13_poset.tex \\
open codes_q13_poset.pdf

# Chapter 11 - Combinatorics

###############################################################################
# Section 11.1: Combinatorics

SECTION COMBINATORICS:

Sym\_4 \_conj\_classes:

```
$\text{(ORBITER)} -v 2 -conjugacy\_classes\_Sym\_n 4
```

Sym\_10 \_conj\_classes:

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

```
open\_classes\_Sym\_10.csv
```

Sym\_15 \_conj\_classes:

```
$\text{(ORBITER)} -v 2 -conjugacy\_classes\_Sym\_n 15
```

Char\_Sym\_4:

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

Char\_Sym\_5:

```
$\text{(ORBITER)} -v 2 -character\_table\_symmetric\_group 5
```

Char\_Sym\_6:

```
$\text{(ORBITER)} -v 2 -character\_table\_symmetric\_group 6
```

all\_subsets\_10\_3:

```
$\text{(ORBITER)} -v 2 -tree\_of\_all\_k\_subsets 10 3
```

random\_k\_subsets:

```
$\text{(ORBITER)} -v 4 \$
```

```
$\text{create\_random\_k\_subsets 10 5 20}
```

rank\_k\_subsets\_test:

```
$\text{(ORBITER)} -v 2 \$
```

```
$\text{rank\_k\_subset 10 3 0,1,2,0,3,4,1,3,5,2,4,5,3,6,7,1,6,8,0,6,9}
```

Walsh\_matrix\_4:

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

```
$\text{define F -finite\_field -q 2 -end}$
```

```
$\text{with F -do -finite\_field\_activity}$
```

```
$\text{-Walsh\_matrix 4 -end}$
```
$(\text{ORBITER}) -v 2 -draw\_matrix \ \ $
$(\text{ORBITER}) -v 2 -input\_csv\_file Walsh\_01\_4.csv \ $
$(\text{ORBITER}) -box\_width 10 -bit\_depth 24 -partition 3 16 16 -end
#pdflatex GF_2.tex
#open GF_2.pdf

Dedekind_{10,10}:
$(\text{ORBITER}) -v 3 -Dedekind\_numbers 2 10 2 10

Dedekind_{30,2}:
$(\text{ORBITER}) -v 3 -Dedekind\_numbers 2 30 2 2

Dedekind_{100,2}:
$(\text{ORBITER}) -v 3 -Dedekind\_numbers 2 100 2 2

elementary\_symmetric\_functions_{4}:
$(\text{ORBITER}) -make\_elementary\_symmetric\_functions 4 4

elementary\_symmetric\_functions_{8}:
$(\text{ORBITER}) -make\_elementary\_symmetric\_functions 8 8

LARGE\_SET S0="0,1,2,3,4,5,6,7,8,9,10,11,12"
# identity

LARGE\_SET S1="6,8,9,2,7,10,1,11,0,3,5,4,12"
#(0, 2, 1) (3, 6, 11, 9, 8, 7, 5, 4),\$

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),\$

760
11691 LARGE_SET_S4="5,8,10,3,11,0,2,1,12,4,6,7,9"
11692 #(0, 5)(1, 8, 9, 4, 11, 7)(2, 10, 6),
11693 \n11694 LARGE_SET_S5="10,11,0,7,12,2,3,1,4,5,8,6,9"
11695 #(0, 10, 8, 4, 12, 9, 5, 2)(1, 11, 6, 3, 7),
11696 \n11697 LARGE_SET_S6="3,4,1,9,5,6,8,2,7,11,12,10"
11698 #(0, 3, 9, 11, 10, 12)(1, 4, 5, 6, 8, 7, 2),
11699 \n11700 LARGE_SET_S7="9,11,0,6,1,3,5,4,2,12,8,7,10"
11701 #(0, 9, 12, 10, 8, 2)(1, 11, 7, 4)(3, 6, 5),
11702 \n11703 LARGE_SET_S8="10,2,12,8,0,3,4,1,5,6,9,7,11"
11704 #(0, 10, 9, 6, 4)(1, 2, 12, 11, 7)(3, 8, 5),
11705 \n11706 LARGE_SET_S9="1,3,4,10,5,6,9,7,8,11,0,12,2"
11707 #(0, 1, 3, 10)(2, 4, 5, 6, 9, 11, 12),
11708 \n11709 LARGE_SET_S10="7,12,1,6,0,4,5,2,3,10,9,8,11"
11710 #(0, 7, 2, 1, 12, 11, 8, 3, 6, 5, 4)(9, 10).
11711 \n11712 file:S:
11713 \n11714 \n11715 \n11716 \n11717 \n11718 \n11719 \n11720 file:S:
11721 \n11722 \n11723 Large_set_H5:
11724 \n11725 \n11726 \n11727 \n11728 \n11729 \n11730 \n11731 \n11732 \n11733 \n11734 \n761
Large set \text{C13}:

\begin{verbatim}
$\text{ORBITER} -v 10 \\
\text{-define G -permutation_group -symmetric_group 13} \\
\text{-subgroup_by_generators C13 13 1 $(GENERATORS_C13)} -end \\
\text{-with G -do} \\
\text{-group_theoretic_activity} \\
\text{-export_orbiter} \\
\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 "C13_elts.csv"} \\
\text{-end} \\
\end{verbatim}

Large set mult \text{C13xS}:

\begin{verbatim}
$\text{ORBITER} -v 10 \\
\text{-define G -permutation_group -symmetric_group 13} -end \\
\text{-with G -do} \\
\text{-group_theoretic_activity} \\
\text{-multiply_elements_csv_column_major_ordering} \\
\text{-C13_elts.csv S.csv C13xS.csv} \\
\text{-end} \\
\end{verbatim}
11782 Large_set_mult_C13xSxH5:
11783 ▷ $(ORBITER) -v 10 
11784 ▷ ▷ -define G -permutation_group -symmetric_group 13 -end 
11785 ▷ ▷ -with G -do 
11786 ▷ ▷ -group_theoretic_activity 
11787 ▷ ▷ ▷ -multiply_elements_csv_column_major_ordering 
11788 ▷ ▷ ▷ - C13xS.csv H5_elts.csv C13xSxH5.csv 
11789 ▷ ▷ -end 
11790 Large_set_mult_C13xSxH5_apply:
11791 ▷ $(ORBITER) -v 10 
11792 ▷ ▷ -define G -permutation_group -symmetric_group 13 -end 
11793 ▷ ▷ -with G -do 
11794 ▷ ▷ -group_theoretic_activity 
11795 ▷ ▷ ▷ -apply_elements_csv_to_set 
11796 ▷ ▷ ▷ ▷ C13xSxH5.csv C13xSxH5_images.csv "0,1,2,3" 
11797 ▷ ▷ ▷ -end 
11798 ▷ ▷ -end 
11799
11800
11801
11802 domino_portrait:
11803 ▷ $(ORBITER) -v 3 -domino_portrait 7 4 anton_28x32 -end 
11804 domino_portrait_input:
11805 ▷ $(ORBITER) -v 2 
11806 ▷ ▷ -define all_one_r -vector -repeat 1 28 -end 
11807 ▷ ▷ -define all_one_c -vector -repeat 1 32 -end 
11808 ▷ ▷ -draw_matrix 
11809 ▷ ▷ ▷ -grayscale 
11810 ▷ ▷ ▷ -invert_colors 
11811 ▷ ▷ ▷ -input_csv_file anton_28x32.m.csv 
11812 ▷ ▷ ▷ -box_width 20 -bit_depth 8 
11813 ▷ ▷ ▷ -partition 3 
11814 ▷ ▷ ▷ ▷ all_one_c all_one_r 
11815 ▷ ▷ ▷ -end 
11816 ▷ ▷ -open anton_28x32.m_draw.bmp 
11817 ▷ -end 
11818
11819
11820 ###############################################################################
11821 # Section 11.2: Diophantine Systems
11822 SECTION_DIOPHANT:
11823
11824
11825
11826
11827 part10:
11828 ▷ $(ORBITER) -v 4 

763
-define A -vector -dense "10,9,8,7,6,5,4,3,2,1" -end \ 
-define D -diophant \ 
-define A -vector -dense "1,1,1,1" -end \ 
-define D -diophant \ 
-label part10 \ 
-coefficient_matrix A \ 
-RHS "10,10,1" \ 
-x_min_global 0 -x_max_global 10 \ 
-end \ 
-with D -do \ 
-diohant_activity -solve_mckay \ 
-end \ 
# Finds 42 solutions with 67 backtrack steps

octic_monomials:
-define A -vector -dense "1,1,1,1" -end \ 
-define D -diophant \ 
-label octic_monomials \ 
-coefficient_matrix A \ 
-RHS "8,8,1" \ 
-x_min_global 0 -x_max_global 8 \ 
-end \ 
-with D -do \ 
-diohant_activity -solve_mckay \ 
-end \ 
-sort -r octic_monomials.sol >octic_monomials_sorted.txt

# Found 165 solutions with 210 backtrack steps

# 165=binomial(11,3)

solve_test_system:
-define A -vector -format 7 -dense $(TEST_SYSTEM) -end \ 
-define D -diophant \ 
-label test_system \ 
-coefficient_matrix A \ 
-RHS $(TEST_RHS) \ 
-x_min_global 0 -x_max_global 1 \ 
-end

McKay_test:
$(\text{ORBITER}) -v 4 \$

-define A -vector -format 7 -dense $(\text{TEST\_SYSTEM}) -end 

-define D -diophant 

-label test_system 

-coefficient\_matrix A 

-RHS $(\text{TEST\_RHS}) 

-x\_min\_global 0 -x\_max\_global 1 

-end 

-with D -do 

-diophant\_activity -solve.mckay 

-end 

DLX_test: 

-define A -vector -format 7 -dense $(\text{TEST\_SYSTEM}) -end 

-define D 

-diophant -label linsp6 

-coefficient\_matrix A 

-RHS "$15,10,6,3,1" 

-x\_min\_global 0 -x\_max\_global 15 

-end 

-with D -do 

-diophant\_activity -solve.DLX 

-end 

# 1 solution in 6 backtrack steps 

SECTION COMBINATORIAL LINEAR SPACES: 

SECTION\_COMBINATORIAL\_LINEAR\_SPACES: 

linsp6: 

-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 

# 1 solution in 6 backtrack steps 

# Section 11.3: Combinatorial Linear Spaces 

Denotes specific configuration and parameters used in the Orbiter tool for solving the linear space problem, along with the solution process and the final result.
```plaintext
# Found 15 solutions with 22 backtrack steps

linsp7:

linsp30:

linsp30.pt_types:

linsp30.pt_distribution:

```

766
-define D -diophant \n-label linsp30.pt_distribution \n-coefficient_matrix A \n-RHS "30,30,1,7,7,1,135,135,1,96,96,1,0,351,2,0,276,2" \n-x_min_global 0 -x_max_global 30 \n-end \n-define Test_lines -set -loop 1 7 1 -end \n-define Geo -geometry_builder \n-V 6 -B 4 -TDO 2 -fuse 1 \n-fname_GEO pasch \n-test Test_lines \n-end \n
# Section 11.4: Combinatorial Linear Spaces

####

# geometry builder:

####

geo_pasch:

$ (ORBITER) -v 8 \n-define Test_lines -set -loop 1 7 1 -end \n-define Geo -geometry_builder \n-V 6 -B 4 -TDO 2 -fuse 1 \n-fname_GEO pasch \n-test Test_lines \n-end

geo_petersen:

$ (ORBITER) -v 8 \n-define Test_lines -set -loop 3 11 1 -end \n-define Geo -geometry_builder \n-V 10 -B 15 -TDO 3 -fuse 1 \n-fname_GEO petersen -girth 5 \n-search_tree \n-test Test_lines \n-end

geo_7_3:

$ (ORBITER) -v 2 \n-define Test_lines -set -loop 3 8 1 -end \n-define Geo -geometry_builder \n
geo_7_3_no_square.test:
$(ORBITER) -v 2 \
-define Test_lines -set -loop 3 8 1 -end \
-define Geo -geometry_builder \
-V 7 -B 7 -TDO 3 \n-fuse 1 -fname.geo 7_3 \
-test Test_lines \
-no_square_test \
-end

geo_7_3_no_square.test_draw:
$(ORBITER) -v 10 \
-draw.incidence_structure.description \
-width 60 -with 10 6 -end \
-define C -combinatorial_objects \
-file_of_incidence.geometries 7_3_nst.inc 7 7 21 \
-end \
-with C -do \
-combinatorial.object_activity \
-draw.incidence.matrices \
-7_3_nst \
-end

pdflatex 7_3_nst.incma.tex
open 7_3_nst.incma.pdf

geo_7_3_orderly:
$(ORBITER) -v 200 \
-define Test_lines -set -loop 3 8 1 -end \
-define Geo -geometry_builder \
-V 7 -B 7 -TDO 3 \n-fuse 1 -fname.geo 7_3 \
-test Test_lines \
-search_tree \
-orderly \
-end

geo_7_3_orderly_draw:
$(ORBITER) -v 20 \
-draw.options -embedded -radius 50 \
-xin 10000 -yin 10000 \

768
```bash
12064 ▷ ▷ ▷ -xout 1000000 -yout 1000000 \
12065 ▷ ▷ ▷ -nodes_empty \
12066 ▷ ▷ ▷ -scale 0.5 -line_width 0.3 \
12067 ▷ ▷ ▷ -end \
12068 ▷ ▷ ▷ -tree_draw -file 7.3_tree.txt -end 
12069 ▷ ▷ ▷ pdflatex 7.3_tree_draw.tex 
12070 ▷ ▷ ▷ open 7.3_tree_draw.pdf
12071
12072 geo_7.3_orderly_mem_debug:
12073 ▷ $(ORBITER) -v 20 \
12074 ▷ ▷ -memory_debug 2 \
12075 ▷ ▷ ▷ -define Test_lines -set -loop 3 8 1 -end \
12076 ▷ ▷ ▷ -define Geo -geometry_builder \
12077 ▷ ▷ ▷ ▷ -V 7 -B 7 -TDO 3 \
12078 ▷ ▷ ▷ ▷ -fuse 1 -fname_GEO 7.3 \
12079 ▷ ▷ ▷ ▷ -test Test_lines \n12080 ▷ ▷ ▷ ▷ -search_tree \
12081 ▷ ▷ ▷ ▷ -orderly \n12082 ▷ ▷ ▷ ▷ -end 
12083
12084 geo_8.3:
12085 ▷ $(ORBITER) -v 2 \
12086 ▷ ▷ -define Test_lines -set -loop 3 9 1 -end \
12087 ▷ ▷ ▷ -define Geo -geometry_builder \
12088 ▷ ▷ ▷ ▷ -V 8 -B 8 -TDO 3 \
12089 ▷ ▷ ▷ ▷ -fuse 1 -fname_GEO 8.3 \n12090 ▷ ▷ ▷ ▷ -test Test_lines \n12091 ▷ ▷ ▷ ▷ -end 
12092
12093 #-print_at_line 4 
12094 # 1 geo: 0 11 18 29 30 38 44 54 
12095 # ago=48 
12096 
12097 
12098
12099
12100
12101 geo_9.3:
12102 ▷ $(ORBITER) -v 2 \
12103 ▷ ▷ -define Test_lines -set -loop 3 10 1 -end \
12104 ▷ ▷ ▷ -define Geo -geometry_builder \
12105 ▷ ▷ ▷ ▷ -V 9 -B 9 -TDO 3 \n12106 ▷ ▷ ▷ ▷ -fuse 1 -fname_GEO 9.3 \n12107 ▷ ▷ ▷ ▷ -test Test_lines \n12108 ▷ ▷ ▷ ▷ -end 
12109 
12110
769```
12111 geo_10.3:
12112 $(ORBITER) -v 2 \n12113 -define Test_lines -set -loop 4 11 1 -end \n12114 -define Geo -geometry_builder \n12115 -V 10 -B 10 -TDO 3 -fuse 1 \n12116 -fname_GEO 10.3 \n12117 -test Test_lines \n12118 -output_to_sage_file \n12119 -end
12120
12121
12122
12123 # 10 geos
12124 # 8/26/2021: 0 sec on Mac
12125
12126
12127 geo_10.3_inc_draw:
12128 $(ORBITER) -v 10 \n12129 -draw_incidence_structure_description \n12130 -width 60 -with_10 6 -end \n12131 -define C -combinatorial_objects \n12132 -file_of_incidence_geometries \n12133 -10.3.inc 10 10 30 \n12134 -end \n12135 -with C -do \n12136 -combinatorial_object_activity \n12137 -draw_incidence_matrices \n12138 -10.3_inc \n12139 -end
12140 pdflatex 10.3_inc_incma.tex
12141 open 10.3_inc_incma.pdf
12142
12143
12144 geo_10.3_orderly:
12145 $(ORBITER) -v 20 \n12146 -define Test_lines -set -loop 4 11 1 -end \n12147 -define Geo -geometry_builder \n12148 -V 10 -B 10 -TDO 3 -fuse 1 \n12149 -fname_GEO 10.3 \n12150 -test Test_lines \n12151 -orderly \n12152 -end
12153
12154 geo_10.3_orderly_mem_debug:
12155 $(ORBITER) -v 2 \n12156 -memory_debug 2 \n12157 -define Test_lines -set -loop 4 11 1 -end \n
770
```
12158  ▶ ▶ -define Geo -geometry_builder \n12159  ▶ ▶ ▶ -V 10 -B 10 -TDO 3 -fuse 1 \n12160  ▶ ▶ ▶ -fname_GEO 10_3 \n12161  ▶ ▶ ▶ -test Test_lines \n12162  ▶ ▶ ▶ -orderly \n12163  ▶ ▶ -end
12164
12165
12166 geo_10_3_tree:
12167  ▶ $(ORBITER) -v 20 \n12168  ▶ ▶ -define Test_lines -set -loop 0 11 1 -end \n12169  ▶ ▶ -define GEO -geometry_builder \n12170  ▶ ▶ ▶ -V 10 -B 10 -TDO 3 -fuse 1 \n12171  ▶ ▶ ▶ -fname_GEO 10_3 \n12172  ▶ ▶ ▶ -search_tree \n12173  ▶ ▶ ▶ -test Test_lines \n12174  ▶ ▶ -end
12175  ▶ $(ORBITER) -v 20 \n12176  ▶ ▶ -draw_options -embedded -radius 40 \n12177  ▶ ▶ ▶ -paperheight 220 \n12178  ▶ ▶ ▶ -paperwidth 330 \n12179  ▶ ▶ ▶ -x1n 10000 -y1n 10000 \n12180  ▶ ▶ ▶ -x1out 10000000 -y1out 500000 \n12181  ▶ ▶ ▶ -scale 2 -line_width 0.3 \n12182  ▶ ▶ ▶ -nodes_empty \n12183  ▶ ▶ -end \n12184  ▶ ▶ -tree_draw \n12185  ▶ ▶ ▶ -file 10_3_tree.txt \n12186  ▶ ▶ -end
12187  pdflatex 10_3_tree_draw.tex
12188  open 10_3_tree_draw.pdf
12189
12190
12191
12192
12193 geo_10_3_tree_path:
12194  ▶ $(ORBITER) -v 20 \n12195  ▶ ▶ -define Test_lines -set -loop 0 11 1 -end \n12196  ▶ ▶ -define GEO -geometry_builder \n12197  ▶ ▶ ▶ -V 10 -B 10 -TDO 3 -fuse 1 \n12198  ▶ ▶ ▶ -fname_GEO 10_3 \n12199  ▶ ▶ ▶ -search_tree \n12200  ▶ ▶ ▶ -test Test_lines \n12201  ▶ ▶ -end
12202  ▶ $(ORBITER) -v 20 \n12203  ▶ ▶ -draw_options -embedded -radius 20 \n12204  ▶ ▶ ▶ -paperheight 220 
```
Desargues_path_lex_least_draw:

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

-with C -do

-combinatorial\_object\_activity

draw\_incidence\_matrices

Desargues_path_lex_least

-end

pdflatex Desargues_path_lex_least_incma.tex

open Desargues_path_lex_least_incma.pdf

Desargues\_PATH\_CANONICAL\_ANCESTOR="10 10 3\n0\n1 12 119\n3 89 112 119\n4 118 89 82\n5 106 114 69 107 111\n6 85 105 112 99 83\n7 94 105 113 85 83\n8 6\n9 26 119 55 105 92 79 74 48\n10 0 119 93 106 1 5 26 79 55 73 47\n-1"

Desargues_path_can_anc_draw:

echo $(DESARGUES\_PATH\_CANONICAL\_ANCESTOR) >Desargues_path_can_anc.inc

$(ORBITER) -v 10

draw\_incidence\_structure\_description

-width 60 -with_10 6 -end

-define C -combinatorial\_objects

-file_of\_incidence\_geometries\_by\_row\_ranks Desargues_path_can_anc.inc 10 10 3
12248  >  >  -end \n12249  >  >  -with C -do \n12250  >  >  -combinatorial_object_activity \n12251  >  >  >  -draw_incidence_matrices \n12252  >  >  >  Desargues_path_can_anc \n12253  >  >  >  -end \n12254  >  >  pdflatex Desargues_path_can_anc_incma.tex \n12255  >  open Desargues_path_can_anc_incma.pdf \n12256 \n12257 \n12258 \n12259 geo_11_3: \n12260  >  >  $(ORBITER) -v 2 \n12261  >  >  >  -define Test_lines -set -loop 4 12 1 -end \n12262  >  >  >  -define Geo -geometry_builder \n12263  >  >  >  >  -V 11 -B 11 -TDO 3 \n12264  >  >  >  >  -fuse 1 -fname_GEO 11_3 \n12265  >  >  >  >  -test Test_lines \n12266  >  >  >  -end \n12267 \n12268  # 31 geos \n12269  # 8/26/2021: 0 sec on Mac \n12270 \n12271 geo_12_3: \n12272  >  >  $(ORBITER) -v 2 \n12273  >  >  >  -define Test_lines -set -loop 4 13 1 -end \n12274  >  >  >  -define Geo -geometry_builder \n12275  >  >  >  >  -V 12 -B 12 -TDO 3 \n12276  >  >  >  >  -fuse 1 -fname_GEO 12_3 \n12277  >  >  >  >  -test Test_lines \n12278  >  >  >  -output_to_sage_file \n12279  >  >  >  -f_orderly \n12280 \n12281  # 229 geos \n12282  #User time: 0.45 of a second, dt=45 tps = 100 \n12283  #nb_calls_to_denselyauty=24586 \n12284 \n12285 \n12286 geo_12_3_orderly: \n12287  >  >  $(ORBITER) -v 2 \n12288  >  >  >  -define Test_lines -set -loop 4 13 1 -end \n12289  >  >  >  -define Geo -geometry_builder \n12290  >  >  >  >  -V 12 -B 12 -TDO 3 \n12291  >  >  >  >  -fuse 1 -fname_GEO 12_3 \n12292  >  >  >  >  -test Test_lines \n12293  >  >  >  -f_orderly \n12294  >  >  >  -end \n
773
geo_13_3:
  $(ORBITER) -v 2 \
  -define Test_lines -set -loop 4 14 1 -end \
  -define Geo -geometry_builder \
  -V 13 -B 13 -TDO 3 \
  -fuse 1 -fname_GEO 13_3 \
  -test Test_lines \
  -end

# 2036 geos, 96, 39, 13, 12^4, 8^3, 6^16, 4^30, 3^20, 2^190, 1^1770
#User time: 0:4
#nb_calls_to_densenauty=216777

geo_13_3_orderly:
  $(ORBITER) -v 2 \
  -define Test_lines -set -loop 4 14 1 -end \
  -define Geo -geometry_builder \
  -V 13 -B 13 -TDO 3 \
  -fuse 1 -fname_GEO 13_3 \
  -test Test_lines \
  -f_orderly \
  -end

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 \

774
12341 ▶ ▶ ◀ -V 14 -B 14 -TDO 3 \n12342 ▶ ▶ ◀ -fuse 1 -fname_GEO 14_3 \n12343 ▶ ▶ ◀ -test Test_lines \n12344 ▶ ▶ ◀ -f_orderly \n12345 ▶ ◀ -end
12346
12347 #User time: 0:50
12348
12349
12350 15_3.inc:
12351 ▶ $(ORBITER) -v 2 \n12352 ▶ ▶ -define Test_lines -set -loop 4 16 1 -end \n12353 ▶ ▶ -define Geo -geometry_builder \n12354 ▶ ▶ ▶ -V 15 -B 15 -TDO 3 \n12355 ▶ ▶ ▶ -fuse 1 -fname_GEO 15_3 \n12356 ▶ ▶ ▶ -test Test_lines \n12357 ▶ ◀ -end
12358
12359 # 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
12360 # 8 min 11 sec on Mac
12361 # running out of memory
12362
12363
12364 geo_15_3_g4:
12365 ▶ $(ORBITER) -v 2 \n12366 ▶ ▶ -define Test_lines -set -loop 4 16 1 -end \n12367 ▶ ▶ -define Geo -geometry_builder \n12368 ▶ ▶ ▶ -V 15 -B 15 -TDO 3 \n12369 ▶ ▶ ▶ -fuse 1 -fname_GEO 15_3_g4 \n12370 ▶ ▶ ▶ -girth 4 \n12371 ▶ ▶ ▶ -search_tree \n12372 ▶ ▶ ▶ -test Test_lines \n12373 ▶ ◀ -end
12374 ▶ $(ORBITER) -v 2 \n12375 ▶ ▶ -draw_options -embedded -radius 50 \n12376 ▶ ▶ ▶ -nodes_empty \n12377 ▶ ▶ ▶ -scale 0.5 -line_width 0.3 -end \n12378 ▶ ▶ -tree_draw -file 15_3_g4_tree.txt -end
12379 ▶ pdflatex 15_3_g4_tree DRAW.tex
12380 ▶ open 15_3_g4_tree draw.pdf
12381
12382 # The unique Cremona Richmond configuration with group of order 720
12383 #User time: 0 of a second, dt=0 tps = 100
12384 #nb_calls_to_densenauty=23
12385
12386 #-sideways
geo_17_3_g4_orderly:

```
# 1 sol
```

```
geo_18_3_g4:

# 4 sol, 1 sec
```

```
geo_19_3_g4:

# 14 sol, 10 sec on Mac
```

```
geo_20_3_g4:
```

```
```

(define Geo -geometry builder \  
-V 20 -B 20 -TDO 3 \  
-fuse 1 -fname_GEO 20_3_g4 \  
-girth 4 \  
-test Test_lines \  
-end  

# 162 sol, User time: 2:5 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  

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 \  
-search_tree \  
-test Test_lines \  
-end  

draw_options -embedded -radius 50 \  
-nodes_empty \  
-scale 0.5 -line_width 0.3 -end \  
-tree_draw -file 15_4_tree.draw.txt -end \  
pdflatex 15_4_tree_draw.tex \  
open 15_4_tree_draw.pdf  

# 4 objects  
# ago=360, 30, 24, 15  
#User time: 0.15 of a second, dt=15 tps = 100  
#nb_calls.to_densenauty=6767  

geo_16_4_g4:  
$(ORBITER) -v 2 \  
(define Test_lines -set -loop 4 17 1 -end \  

777
12481 \> \> \> -define Geo -geometry_builder \\
12482 \> \> \> \> -V 16 -B 16 -TDO 4 \\
12483 \> \> \> \> -fuse 1 -fname_GEO 16_4_g4 \\
12484 \> \> \> \> -girth 4 \\
12485 \> \> \> \> -test Test_lines \\
12486 \> \> \> -end \\
12487 12488 # none \\
12489 12490 12491 12492 12493 12494 geo_LSQ6: \\
12495 \> \> $(ORBITER) -v 2 \\
12496 \> \> \> -define Test_lines -set -loop 1 19 1 -end \\
12497 \> \> \> -define Geo -geometry_builder \\
12498 \> \> \> \> -V 6,6,6 -B 1,1,36 -TDO \\
12499 \> \> \> \> "1,0,0,6, 0,1,0,6, 0,0,1,6" \\
12500 \> \> \> \> -fuse 3 -fname_GEO LSQ6 \\
12501 \> \> \> \> -test Test_lines \\
12502 \> \> \> -end \\
12503 12504 geo_16: \\
12505 \> \> $(ORBITER) -v 2 \\
12506 \> \> \> -define Test_lines -set -loop 3 17 1 -end \\
12507 \> \> \> -define Geo -geometry_builder \\
12508 \> \> \> \> -V 16 -B 20 -TDO 5 \\
12509 \> \> \> \> -fuse 1 -fname_GEO geo_16 \\
12510 \> \> \> \> -test Test_lines \\
12511 \> \> \> -end \\
12512 12513 12514 40_4_g4.inc: \\
12515 \> \> $(ORBITER) -v 5 \\
12516 \> \> \> -define Test_lines -set -loop 0 41 1 -end \\
12517 \> \> \> -define Geo -geometry_builder \\
12518 \> \> \> \> -V 40 -B 40 -TDO 4 \\
12519 \> \> \> \> -fuse 1 \\
12520 \> \> \> \> -fname_GEO 40_4_g4 \\
12521 \> \> \> \> -girth 4 \\
12522 \> \> \> \> -search_tree \\
12523 \> \> \> \> -special_test_not_orderly \\
12524 \> \> \> \> -test Test_lines \\
12525 \> \> \> \> -output_to_sage_file \\
12526 \> \> \> \> -output_to_inc_file \\
12527 \> \> \> -end
$(ORBITER) -v 2 \
-define Test -set -loop 4 64 1 -end \
-define Geo -geometry_builder \
-V 63 -B 63 -TDO 3 \
-fuse 1 -fname GEO 63_3_g6 \
-girth 6 \
-test Test_lines \
-special_test_not_orderly \
-output_to_sage_file \
-output_to_inc_file \
-search_tree \
-end \

# special_test_not_orderly is important for speed purposes

2 geos, ago=51840^2

40.4_g4.inc

g6.inc

g6: 63 3 g6:

$(ORBITER) -v 8 \
-define F -finite_field -q 3 -end \
-define D -design -field F -family PG_2.q -end \
-with D -do \
-design_activity \
-export_inc \

design_PG_2_4:

```
$\text{(ORBITER)} -v 8 \n-def F -finite_field -q 4 -end \n-def D -design -field F -family PG_2.q -end \n-with D -do \n-design_activity \n-export_inc \n-end
```

design_PG_2_3_table_create:

```
$\text{(ORBITER)} -v 2 \n-def F -finite_field -q 3 -end \n-def D -design -field F -family PG_2.q -end \n-def Sym13 -permutation_group -symmetric_group 13 -end \n-def T -design_table D "PG_2.13" Sym13 -end
```

# written file PG_2.13_design_table.csv

# 1108800 designs

#User time: 7:30

design_PG_2_3_group_5:

```
$\text{(ORBITER)} -v 2 \n-def F -finite_field -q 3 -end \n-def D -design -field F -family PG_2.q -end \n-def T -design_table D "PG_2.13" Sym13 -end \n-def LSW -large_set_with_symmetry_assumption T \n-H "5" $(GENERATORS_H5) \n-N "1200" $(GENERATORS_N5) \n-prefix "H5" \n-selected_orbit_length 5 \n-end \n-with LSW -do \n-large_set_with_symmetry_assumption_activity \n-normalizer_on_orbits_of_a_given_length 5 \n-end
```

#H5_N_orbit_reps.csv

# 678 orbits

#User time: 2:39

780
design_PG_2.3_group_5_sol_0:
$(ORBITER) -v 2 \ndefine F -finite_field -q 3 -end \ndefine D -design -field F -family PG_2,q -end \ndefine T -design_table D "PG_2.13" Sym13 -end \ndefine LSW -large_set_with_symmetry_assumption T \ndefine H "5" $(GENERATORS_H5) \ndefine N "1200" $(GENERATORS_N5) \n-prefix "H5" \n-selected_orbit_length 5 \n-end \n-with LSW -do \n-large_set_with_symmetry_assumption_activity \n-read_solution_file 5 case_0.sol.txt \n-end \n
wreath_product_designs_n4_k2_inc.txt:
$(ORBITER) -v 8 \ndefine D -design -wreath_product_designs 4 2 -end \n-with D -do \n-design_activity \n-export_inc \n-end

wreath_product_designs_n8_k6_inc.txt:
$(ORBITER) -v 8 \ndefine D -design -wreath_product_designs 8 6 -end \n-with D -do \n-design_activity \n-export_inc \n-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 \ndefine gens -vector -dense "1,2,3,4,5,6,0" -end \ndefine G -permutation_group -symmetric_group 7 \n-subgroup_by_generators "C7" 7 1 gens \n-end \n(define Control -poset_classification_control \n-problem_label C7 -W -depth 3 \n
# 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 \ndefine gens -vector -dense "1,2,3,4,5,6,0" -end \ndefine G -permutation_group -symmetric_group 7 \n-subgroup_by_generators "C7" 7 1 gens \n-end \n(define Control -poset_classification_control \n-problem_label C7 -W -depth 3 \n
# 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 \ndefine gens -vector -dense "1,2,3,4,5,6,0" -end \ndefine G -permutation_group -symmetric_group 7 \n-subgroup_by_generators "C7" 7 1 gens \n-end \n(define Control -poset_classification_control \n-problem_label C7 -W -depth 3 \n
# wreath_product_designs_n8_k6_inc.txt
The design with have 16 points and 3920 blocks of size 6.
12669 \[\triangleright\triangleright\triangleright -\text{Kramer} \_\text{Mesner} \_\text{matrix} \ 2 \ 3 \ \backslash \]
12670 \[\triangleright\triangleright\triangleright -\text{draw} \_\text{poset} \ \backslash \]
12671 \[\triangleright\triangleright\triangleright -\text{draw} \_\text{options} \ -\text{embedded} \ -\text{sideways} \ -\text{radius} \ 50 \ \backslash \]
12672 \[\triangleright\triangleright\triangleright -\text{scale} \ 0.5 \ -\text{line} \_\text{width} \ 0.3 \ -\text{end} \ \backslash \]
12673 \[\triangleright\triangleright\triangleright -\text{end} \ \backslash \]
12674 \[\triangleright\triangleright\triangleright -\text{define} \ \text{Orb} \ -\text{orbits} \ -\text{group} \ \text{G} \ \backslash \]
12675 \[\triangleright\triangleright\triangleright -\text{on} \_\text{subsets} \ 3 \ \text{Control} \ \backslash \]
12676 \[\triangleright\triangleright\triangleright -\text{end} \]
12677 \[\text{$\langle ORBITER \rangle -v \ 4 \rangle \}
12678 \[\triangleright\triangleright\triangleright -\text{define} \ \text{A} \ -\text{vector} \ -\text{file} \ \text{C7} \_\text{KM} \_2.3. \text{csv} \ -\text{end} \]
12679 \[\triangleright\triangleright\triangleright -\text{define} \ \text{D} \ -\text{diophant} \]
12680 \[\triangleright\triangleright\triangleright -\text{label} \ "\text{C7} \_\text{KM} \_2.3. \text{system}" \]
12681 \[\triangleright\triangleright\triangleright -\text{coefficient} \_\text{matrix} \ \text{A} \]
12682 \[\triangleright\triangleright\triangleright -\text{RHS} \_\text{constant} \ "1,1,1" \]
12683 \[\triangleright\triangleright\triangleright -\text{x} \_\text{min} \_\text{global} \ 0 \ -\text{x} \_\text{max} \_\text{global} \ 1 \]
12684 \[\triangleright\triangleright\triangleright -\text{end} \]
12685 \[\triangleright\triangleright\triangleright -\text{with} \ \text{D} \ -\text{do} \]
12686 \[\triangleright\triangleright\triangleright -\text{diophant} \_\text{activity} \ -\text{solve} \_\text{mckay} \]
12687 \[\triangleright\triangleright\triangleright -\text{end} \]
12688
12689
12690 # to create simple 7-designs on 33 points with block size 8 and lambda = 10 invariant under \text{PGGL}(2,32):
12691
12692 \text{KM} \_\text{PGGL} \_2.32:\n12693 \[\text{$\langle ORBITER \rangle -v \ 3 \rangle \]
12694 \[\triangleright\triangleright\triangleright -\text{define} \ \text{Control} \ -\text{poset} \_\text{classification} \_\text{control} \]
12695 \[\triangleright\triangleright\triangleright -\text{problem} \_\text{label} \ \text{KM} \_\text{PGGL} \_2.32 \ -\W \ -\text{depth} \ 8 \]
12696 \[\triangleright\triangleright\triangleright -\text{Kramer} \_\text{Mesner} \_\text{matrix} \ 7 \ 8 \]
12697 \[\triangleright\triangleright\triangleright -\text{draw} \_\text{poset} \]
12698 \[\triangleright\triangleright\triangleright -\text{draw} \_\text{options} \ -\text{embedded} \ -\text{sideways} \ -\text{radius} \ 50 \]
12699 \[\triangleright\triangleright\triangleright -\text{scale} \ 0.5 \ -\text{line} \_\text{width} \ 0.3 \ -\text{end} \]
12700 \[\triangleright\triangleright\triangleright -\text{end} \]
12701 \[\triangleright\triangleright\triangleright -\text{define} \ \text{G} \ -\text{linear} \_\text{group} \ -\text{PGGL} \ 2 \ 32 \ -\text{end} \]
12702 \[\triangleright\triangleright\triangleright -\text{define} \ \text{Orb} \ -\text{orbits} \ -\text{group} \ \text{G} \]
12703 \[\triangleright\triangleright\triangleright -\text{on} \_\text{subsets} \ 8 \ \text{Control} \]
12704 \[\triangleright\triangleright\triangleright -\text{end} \]
12705 \[\text{$\langle ORBITER \rangle -v \ 2 \rangle -\text{draw} \_\text{matrix} \]
12706 \[\triangleright\triangleright\triangleright -\text{input} \_\text{csv} \_\text{file} \ \text{KM} \_\text{PGGL} \_2.32. \text{KM} \_7.8. \text{csv} \]
12707 \[\triangleright\triangleright\triangleright -\text{box} \_\text{width} \ 20 \ -\text{bit} \_\text{depth} \ 24 \]
12708 \[\triangleright\triangleright\triangleright -\text{partition} \ 3 \ 32 \ 97 \ -\text{end} \]
12709 \[\text{pdflatex} \ \text{KM} \_\text{PGGL} \_2.32. \text{poset} \_\text{lvl} \_8. \text{tex} \]
12710 \[\text{open} \ \text{KM} \_\text{PGGL} \_2.32. \text{poset} \_\text{lvl} \_8. \text{pdf} \]
12711 \[\text{open} \ \text{KM} \_\text{PGGL} \_2.32. \text{KM} \_7.8. \text{draw} . \text{bmp} \]
12712 \[\text{$\langle ORBITER \rangle -v \ 4 \rangle \]
12713 \[\triangleright\triangleright\triangleright -\text{define} \ \text{A} \ -\text{vector} \ -\text{file} \ \text{KM} \_\text{PGGL} \_2.32. \text{KM} \_7.8. \text{csv} \ -\text{end} \]
12714 \[\triangleright\triangleright\triangleright -\text{define} \ \text{D} \ -\text{diophant} \]

782
12715 -label "KM_PGGL_2_32_KM_7_8_system" \
12716 -coefficient_matrix A \
12717 -RHS_constant "10,10,1" \
12718 -x_min_global 0 -x_max_global 1 \
12719 -end \
12720 -with D -do \
12721 -diophant_activity -solve_mckay \
12722 -end 
12723 
12724 
12725 
12726 
12727 KM_PSL_3_5: 
12728 $(ORBITER) -v 3 \
12729 -define Control -poset_classification_control \
12730 -problem_label KM_PSL_3_5 -W -depth 10 \
12731 -Kramer_Mesner_matrix 8 10 \
12732 -draw_poset \
12733 -draw_options -embedded -sideways \
12734 -radius 50 -scale 0.5 -line_width 0.3 -end \
12735 -end \
12736 -define G -linear_group -PSL 3 5 -end \
12737 -define Orb -orbits -group G \
12738 -on_subsets 10 Control \
12739 -end 
12740 $(ORBITER) -v 2 -draw_matrix \
12741 -input_csv_file KM_PSL_3_5_KM_8_10.csv \
12742 -box_width 10 -bit_depth 8 -partition 3 42 174 -end 
12743 $(ORBITER) -v 4 \
12744 -define A -vector -file KM_PSL_3_5_KM_8_10.csv -end \
12745 -define D -diophant \
12746 -label "KM_PSL_3_5_KM_8_10_system" \
12747 -coefficient_matrix A \
12748 -RHS_constant "93,93,1" \
12749 -x_min_global 0 -x_max_global 1 \
12750 -end \
12751 -with D -do \
12752 -diophant_activity -solve_mckay \
12753 -end 
12754 
12755 
12756 
12757 
12758 ###################################################################################################
12759 # Section 11.6: Design Theory – Large Sets
12760
12761 SECTION_DESIGN_THEORY_LARGE_SETS:
AG_2.3.inc:

```
$ORBITER -v 2 \ndefine Geo -geometry builder \n  -V 9 -B 12 \n  -TD 4 -fuse 1 \n  -fname Geo AG_2.3 \n  -test 3,4,5,6,7,8,9 \n$end
```

```
#9 12 3
#0 13 22 27 35 41 47 53 55 59 71 76
#-1 1
#432
```

```
$ORBITER -v 5 \ndefine B -vector -dense $(AG_2.3.BLOCKS) -end \ndefine D -design -list_of_blocks 9 3 B -end \ndefine Sym9 -permutation_group -symmetric_group 9 -end \ndefine T -design_table D "AG_2.3" Sym9 -end
```

```
# creates AG_2.3.design_table.csv
# and AG_2.3.makefile
```

```
#0,0,13,22,27,35,41,47,53,55,59,71,76
# is the first design in AG_2.3.design_table.csv
```

```
poset.orbit_node::init_root_node storing strong generators for a group of order 362880
# stabilizer order 432
# 840 designs
```

```
User time: 0.13 of a second, dt=13 tps = 100
```

```
$ORBITER -v 2 \ndefine gens -vector -file AG_2.3.gens.csv -end \n```

-define `G' permutation_group \
-bsgs AG_2_3 "AG_2_3" 840 362880 "0,1,2,3,4,5,6,7" 8 gens -end \
-defn `Orb' orbits -group G \
 >> -on_points \
-end \
-with `Orb' -do -orbits_activity \
>> -stabilizer 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 
-defn `G' permutation_group \
-bsgs AG_2_3_stab_orb_0 "AG_2_3_stab_orb_0" \
>> 840 432 "0,1,2,3,4,5,6,7,8" 5 gens \
-end \
-defn `Gr' modified_group -from G \
-with `Gr' -do \
-group_theoretic_activity \
>> -export_orbiter \
-end 

AG_2_3_stab_orb_0.Perm840_res192: 
-define gens -vector -file Perm840_res192.gens.csv -end 
-defn `G' permutation_group \
-bsgs Perm840_res192 "Perm840 {\rm res192}" \
>> 192 432 "0,1,2,3,4,5,6,7,8" 4 gens \
-end \

785
12855 ▷ ▷ -with G -do \
12856 ▷ ▷ -group_theoretic_activity \
12857 ▷ ▷ ▷ -report \
12858 ▷ ▷ -end
12859 ▷ pdflatex Perm840_res192_report.tex
12860 ▷ open Perm840_res192_report.pdf
12861
12862
12863
12864 LS_AG_2_3.disjoint_sets_graph_and_cliques:
12865 ▷ $(ORBITER) -v 2 \
12866 ▷ ▷ -define Gamma -graph \
12867 ▷ ▷ ▷ -disjoint_sets_graph \
12868 ▷ ▷ ▷ AG.2.3.design_table.csv \
12869 ▷ ▷ -end \
12870 ▷ ▷ -with Gamma -do \
12871 ▷ ▷ -graph_theoretic_activity \
12872 ▷ ▷ ▷ -save \
12873 ▷ ▷ -end \
12874 ▷ ▷ -with Gamma -do \
12875 ▷ ▷ -graph_theoretic_activity \
12876 ▷ ▷ ▷ -find_cliques -target_size 7 -end \
12877 ▷ ▷ -end \
12878 ▷ ▷ -print_symbols
12879
12880
12881 #AG_2_3.design_table.disjoint_sets.colored_graph
12882 #User time: 0.66 of a second, dt=66 tps = 100
12883 #AG_2_3.design_table.disjoint_sets_sol.txt
12884 #AG_2_3.design_table.disjoint_sets_sol.csv
12885
12886 #15360 solutions
12887
12888 LS_AG_2_3.disjoint_sets_split:
12889 ▷ $(ORBITER) -v 4 \
12890 ▷ ▷ -define Gamma -graph -load \
12891 ▷ ▷ ▷ AG.2.3.design_table.disjoint_sets.colored_graph \
12892 ▷ ▷ -end \
12893 ▷ ▷ -with Gamma -do \
12894 ▷ ▷ -graph_theoretic_activity \
12895 ▷ ▷ ▷ -split_by_clique "0" "0" \
12896 ▷ ▷ -end
12897
12898
12899 #AG_2_3.design_table.disjoint_sets.0.graph
12900 #AG_2_3.design_table.disjoint_sets.0_subset.txt
12901
786
LS_AG_2_3.export_solutions:

12902 12903

12904 12905 $(ORBITER) -v 20 \n
12906 12907 12908 12909

12910 12911 12912 12913 12914 12915

12916 12917 12918 12919

12920 12921 12922 12923

12924 12925

12926 12927 12928

12929 12930 12931

12932 12933 12934 12935

12936 12937 12938 12939

12940 12941 12942 12943 12944

12945 12946 12947 12948

#User time: 0.39 of a second, dt=39 tps = 100

# solutions.csv

SECTION DESIGN THEORY DELANDTSHEER DOYEN:

DD_PP4:

DD_PP4.system:
12949
12950 DD_CC:
12951 ▶ $(ORBITER) -v 6 \\
12952 ▶ ▶ ▶ -Delandtsheer_Doyen -search_wrt_subgroup \\
12953 ▶ ▶ ▶ ▶ $(DELANDTSHEER_DOYEN_PROBLEM_COLBOURN_COLBOURN_7_13) \\
12954 ▶ ▶ ▶ ▶ $(DELANDTSHEER_DOYEN_PROBLEM_COLBOURN_COLBOURN_7_13_GROUP1) \\
12955 ▶ ▶ ▶ ▶ $(DELANDTSHEER_DOYEN_PROBLEM_COLBOURN_COLBOURN_7_13_MASK1) \\
12956 ▶ ▶ ▶ -end
12957
12958 #target level: 6
12959 #k2: 15
12960 #number of k-orbits at target level: 1774964
12961
12962 # creates DD_CC.7.13.pair_covering.csv
12963
12964
12965 DD_CC_system:
12966 ▶ $(ORBITER) -v 4 \\
12967 ▶ ▶ -define D -diophant -label DD_CC.7.13 \\
12968 ▶ ▶ -problem_of_Steiner_type 45 DD_CC.7.13.pair_covering.csv \\
12969 ▶ ▶ -has_sum 3 \\
12970 ▶ ▶ -end \\
12971 ▶ ▶ -with D -do \\
12972 ▶ ▶ -diophant_activity -solve_mckay \\
12973 ▶ ▶ -end
12974 ▶
12975
12976
12977 # no solution
12978
12979
12980
12981 # 18603 = 27 * 53 * 13
12982
12983 DD_M1_G1:
12984 ▶ $(ORBITER) -v 4 \\
12985 ▶ ▶ -Delandtsheer_Doyen \\
12986 ▶ ▶ ▶ $(DELANDTSHEER_DOYEN_PROBLEM_27_53) \\
12987 ▶ ▶ ▶ $(DELANDTSHEER_DOYEN_PROBLEM_27_53_GROUP1) \\
12988 ▶ ▶ ▶ $(DELANDTSHEER_DOYEN_PROBLEM_27_53_MASK1) \\
12989 ▶ ▶ ▶ -end
12990
12991 DD_M1_G1_S:
12992 ▶ $(ORBITER) -v 4 \\
12993 ▶ ▶ -Delandtsheer_Doyen \\
12994 ▶ ▶ ▶ $(DELANDTSHEER_DOYEN_PROBLEM_27_53) \\
12995 ▶ ▶ ▶ $(DELANDTSHEER_DOYEN_PROBLEM_27_53_GROUP1) \\

788
Section 11.8: Tactical Decompositions

max_arc_16_4_start:
$(ORBITER) -v 4 -maximal_arc_parameters 16 4

max_arc_16_4_convert_stack.tdo:
$(ORBITER) -v 4 -convert_stack.to.tdo max.arc.q16.r4.stack

max_arc_16_4_refine:
$(ORBITER) -v 4 -tdo_refinement \
-input_file max.arc.q16.r4.tdo -dual_is_linear_space -end

max.arc_16_4_print:
# Chapter 12 - Finite Geometry

# Section 12.1: Spreads

create_spread_9a:
$(ORBITER) -v 3 \
  -define F -finite_field -q 3 -end \n  -define G -linear_group -PGL 4 F -end \n  -define S -spread -kernel_field F \n  -group G -k 2 -catalogue 0 \n  -end

# desarguesian spread, ago = 5760

create_spread_9b:
$(ORBITER) -v 3 \
  -define F -finite_field -q 3 -end \n  -define G -linear_group -PGL 4 F -end \n  -define S -spread -kernel_field F \n  -group G -k 2 -catalogue 1 \n  -end

# Hall spread, ago = 1920

create_spread_25_7:
$(ORBITER) -v 3 \
  -define F -finite_field -q 5 -end \n  -define G -linear_group -PGL 4 F -end \n  -define S -spread -kernel_field F \n  -group G -k 2 -catalogue 7 \n  -end
create_spread_Rao_Rao_27:

\$\text{ORBITER} \ -v \ 3 \ \$

-define F -finite_field -q 3 -end \\n
-define SS -vector -dense \$(\text{SPREAD\ SET\ 27\ RAO})\ -end \\

-define G -linear_group -PGL 6 F -end \\

-define S -spread -kernel_field F \\

-group G -k 3 -spread_set SS \\

-end

SPREAD.S27.RAO.RAO="\n
0, 33879, 5418, 13103, 30556, 22107, 27225, 4045, 24924, 31961, \\
3196, 30100, 28081, 25862, 1339, 6696, 8242, 11747, 14000, 14705, \\
9784, 17843, 20772, 9271, 19413, 18678, 16109, 23924"

desarguesian_spread_in_PG_3_2:
desarguesian\_spread in PG\_5\_2:
\$\text{ORBITER} -v 3 \$
\$\text{define FQ -finite\_field -q 8 -end} \$
\$\text{define Fq -finite\_field -q 2 -end} \$
\$\text{with FQ -and Fq -do -finite\_field\_activity} \$
\$\text{cheat\_sheet\_desarguesian\_spread 2 -end} \$
pdflatex Desarguesian\_Spread\_5\_2.tex
open Desarguesian\_Spread\_5\_2.pdf

desarguesian\_spread in PG\_3\_4:
\$\text{ORBITER} -v 3 \$
\$\text{define FQ -finite\_field -q 16 -end} \$
\$\text{define Fq -finite\_field -q 4 -end} \$
\$\text{with FQ -and Fq -do -finite\_field\_activity} \$
\$\text{cheat\_sheet\_desarguesian\_spread 2 -end} \$
pdflatex Desarguesian\_Spread\_3\_4.tex
open Desarguesian\_Spread\_3\_4.pdf

desarguesian\_spread in PG\_3\_5:
\$\text{ORBITER} -v 3 \$
\$\text{define FQ -finite\_field -q 25 -end} \$
\$\text{define Fq -finite\_field -q 5 -end} \$
\$\text{with FQ -and Fq -do -finite\_field\_activity} \$
\$\text{cheat\_sheet\_desarguesian\_spread 2 -end} \$
pdflatex Desarguesian\_Spread\_3\_5.tex
open Desarguesian\_Spread\_3\_5.pdf

classify\_spreads\_4:
\$\text{ORBITER} -v 3 \$
\$\text{define F -finite\_field -q 2 -end} \$
\$\text{define P -projective\_space -n 3 -field F -v 0 -end} \$
\$\text{define C -spread\_classifier} \$
\$\text{projective\_space P} \$
\$\text{-k 2} \$
\$\text{-starter\_size 5} \$
\$\text{-poset\_classification\_control} \$
\$\text{-draw\_options} \$
\$\text{-embedded} \$

792
classify_spreads_16_4:

\$\texttt{(ORBITER)} -v 4 \$

\$\texttt{-define F -finite_field -q 4 -end} \$

\$\texttt{-define P -projective_space -n 3 -field F -v 0 -end} \$

\$\texttt{-define C -spread_classifier} \$

\$\texttt{-projective_space P} \$

\$\texttt{-k 2} \$

\$\texttt{-starter_size 17} \$

\$\texttt{-poset_classification_control} \$

\$\texttt{-draw_options} \$

\$\texttt{-radius 20} \$

\$\texttt{-nodes_empty} \$

\$\texttt{-line_width 0.2} \$

\$\texttt{-embedded} \$

\$\texttt{-end} \$

\$\texttt{-draw_poset} \$

\$\texttt{-problem_label spreads_2_2} \$

\$\texttt{-end} \$

\$\texttt{-output_prefix "."} \$

\$\texttt{-with C -do -spread_classify_activity} \$

\$\texttt{-compute_starter} \$

\$\texttt{-problem_label spreads_2_2} \$

\$\texttt{-W -depth 5} \$

\$\texttt{-report -end} \$

\$\texttt{-end} \$

\$\texttt{pdflatex spreads_2_2.poset.lvl.5.tex} \$

\$\texttt{open spreads_2_2.poset.lvl.5.pdf} \$

\$\texttt{pdflatex spreads_16_4.poset.lvl.17.tex} \$

\$\texttt{open spreads_16_4.poset.lvl.17.pdf} \$
classify_spreads_25_starter_lift_case_0:

$(ORBITER) -v 3 \n-define F -finite_field -q 5 -end \n-define P -projective_space -n 3 -field F -v 0 -end \n-define C -spread_classifier \n-projective_space P \n-starter_size 5 \n-recoordinatize \n-poset_classification_control \n-draw_options \n-radius 20 \n-nodes_empty \n-line_width 0.2 \n-embedded \n-end \n-\n-draw_poset \n-problem_label spreads_25 \n-end \n-output_prefix "" \n-end \n-with C -do -spread_classify_activity \n-compute_starter \n-problem_label spreads_25 \n-W -depth 5 \n-report -end \n-end \n-with C -do -spread_classify_activity \n-prepare_lifting_single_case 0 \n-end

#save_colored_graph fname=spreads_25_graph_0.bin
#save_colored_graph nb_vertices=225
#save_colored_graph nb_colors=21
#save_colored_graph nb_colors_per_vertex=1
#save_colored_graph done
#colored_graph::save done
#Written file spreads_25_graph_0.bin of size 5914
spreads_25_starter_0_cliques:

```bash
$ (ORBITER) -v 2 \n
define G -graph -load spreads_25_graph_0.bin -end \nwith G -do \n
graph_theoretic_activity \n
-find_cliques -rainbow -target_size 21 -end \nend \n
#graph_theoretic_activity::perform_activity Gr->label=spreads_25_graph_0 nb_sol = 7680

classify_spreads_25_starter_lift_all_cases:

$ (ORBITER) -v 3 \n
define F -finite_field -q 5 -end \n
define P -projective_space -n 3 -field F -v 0 -end \n
define C -spread_classifier \n-projective_space P \n-k 2 \n-starter_size 5 \n-recoordinatize \n-poset_classification_control \n-draw_options \n
-radius 20 \n-nodes_empty \n-line_width 0.2 \n-embedded \n-end \n-W \n-draw_poset \n-problem_label_spreads_25 \n-end \n-output_prefix "" \n-end \n-with C -do -spread_classify_activity \n-compute_starter \n-problem_label_spreads_25 \n-W -depth 5 \n-report -end \n-end \n-end \n-with C -do -spread_classify_activity \n
prepare_lifting_all_cases \n-end
```

spreads_25_starter_cliques:

```bash
$ (ORBITER) -v 2 \n```
classify_spreads_25.isomorph:

classify_spreads_25.isomorph:

classify_spreads_25.isomorph:
We found 21 isomorphism types

```bash
# We found 21 isomorphism types
```

```
classify_spreads_27_3_starter:
```
\begin{verbatim}
13418 \textbf{\textgreater} \textbf{\textgreater} \texttt{-define P -projective_space -n 5 -field F -v 0 -end} \\
13419 \textbf{\textgreater} \textbf{\textgreater} \texttt{-define C -spread_classifier} \\
13420 \textbf{\textgreater} \textbf{\textgreater} \texttt{-projective_space P} \\
13421 \textbf{\textgreater} \textbf{\textgreater} \texttt{-k 3} \\
13422 \textbf{\textgreater} \textbf{\textgreater} \texttt{-starter_size 5} \\
13423 \textbf{\textgreater} \textbf{\textgreater} \texttt{-recoordinatize} \\
13424 \textbf{\textgreater} \textbf{\textgreater} \texttt{-poset_classification_control} \\
13425 \textbf{\textgreater} \textbf{\textgreater} \texttt{-draw_options} \\
13426 \textbf{\textgreater} \textbf{\textgreater} \texttt{-radius 20} \\
13427 \textbf{\textgreater} \textbf{\textgreater} \texttt{-nodes_empty} \\
13428 \textbf{\textgreater} \textbf{\textgreater} \texttt{-line_width 0.2} \\
13429 \textbf{\textgreater} \textbf{\textgreater} \texttt{-embedded} \\
13430 \textbf{\textgreater} \textbf{\textgreater} \texttt{-end} \\
13431 \textbf{\textgreater} \textbf{\textgreater} \texttt{-draw_poset} \\
13432 \textbf{\textgreater} \textbf{\textgreater} \texttt{-problem_label spreads_27_3} \\
13433 \textbf{\textgreater} \textbf{\textgreater} \texttt{-end} \\
13434 \textbf{\textgreater} \textbf{\textgreater} \texttt{-output_prefix "."} \\
13435 \textbf{\textgreater} \textbf{\textgreater} \texttt{-end} \\
13436 \textbf{\textgreater} \textbf{\textgreater} \texttt{-with C -do -spread_classify_activity} \\
13437 \textbf{\textgreater} \textbf{\textgreater} \texttt{-compute_starter} \\
13438 \textbf{\textgreater} \textbf{\textgreater} \texttt{-problem_label spreads_27_3} \\
13439 \textbf{\textgreater} \textbf{\textgreater} \texttt{-W -depth 5} \\
13440 \textbf{\textgreater} \textbf{\textgreater} \texttt{-report -end} \\
13441 \textbf{\textgreater} \textbf{\textgreater} \texttt{-end} \\
13442 \textbf{\textgreater} \textbf{\textgreater} \texttt{-end} \\
13443 \textbf{\textgreater} \textbf{\textgreater} \texttt{-with C -do -spread_classify_activity} \\
13444 \textbf{\textgreater} \textbf{\textgreater} \texttt{-prepare_lifting_single_case 0} \\
13445 \textbf{\textgreater} \textbf{\textgreater} \texttt{-end} \\
13446 \textbf{\textgreater} \textbf{\textgreater} \texttt{pdflatex spreads_27_3.poset_detailed_lvl_5.tex} \\
13447 \textbf{\textgreater} \textbf{\textgreater} \texttt{open spreads_27_3.poset_detailed_lvl_5.pdf} \\
13448
13449 # 50 orbits at level 5: \\
13450 #5 : 50 orbits \\
13451 #total: 60 \\
13452 #((3^9, 2, 6^2, 10, 3^9, 2^9, 1^{24}) average is 4 + 26 / 50 \\
13453 \\
13454 # time 4:31 \\
13455 \\
13456 \\
13457 classify_spreads_27_starter_lift_all_cases: \\
13458 \textbf{\textgreater} \$(ORBITER) -v 3 \\
13459 \textbf{\textgreater} \texttt{-define F -finite_field -q 3 -end} \\
13460 \textbf{\textgreater} \texttt{-define P -projective_space -n 5 -field F -v 0 -end} \\
13461 \textbf{\textgreater} \texttt{-define C -spread_classifier} \\
13462 \textbf{\textgreater} \texttt{-projective_space P} \\
13463 \textbf{\textgreater} \texttt{-k 3} \\
13464 \textbf{\textgreater} \textbf{\textgreater} \texttt{-projective_space P} \\
\end{verbatim}
13465 ▶ ▶ ▶ -starter_size 5 \n13466 ▶ ▶ ▶ -recoordinatize \n13467 ▶ ▶ ▶ -poset_classification_control \n13468 ▶ ▶ ▶ ▶ -draw_options \n13469 ▶ ▶ ▶ ▶ ▶ -radius 20 \n13470 ▶ ▶ ▶ ▶ ▶ -nodes_empty \n13471 ▶ ▶ ▶ ▶ ▶ -line_width 0.2 \n13472 ▶ ▶ ▶ ▶ ▶ -embedded \n13473 ▶ ▶ ▶ ▶ ▶ -end \n13474 ▶ ▶ ▶ ▶ ▶ -W \n13475 ▶ ▶ ▶ ▶ ▶ -draw_poset \n13476 ▶ ▶ ▶ ▶ ▶ ▶ -problem_label spreads_27 \n13477 ▶ ▶ ▶ ▶ ▶ -end \n13478 ▶ ▶ ▶ ▶ ▶ -output_prefix "" \n13479 ▶ ▶ ▶ -end \n13480 ▶ ▶ ▶ -with C -do -spread_classify_activity \n13481 ▶ ▶ ▶ -compute_starter \n13482 ▶ ▶ ▶ ▶ -problem_label spreads_27 \n13483 ▶ ▶ ▶ ▶ ▶ -W -depth 5 \n13484 ▶ ▶ ▶ ▶ ▶ -report -end \n13485 ▶ ▶ ▶ ▶ ▶ -end \n13486 ▶ ▶ ▶ ▶ ▶ -end \n13487 ▶ ▶ ▶ ▶ ▶ -with C -do -spread_classify_activity \n13488 ▶ ▶ ▶ ▶ ▶ -prepare_lifting_all_cases \n13489 ▶ ▶ ▶ ▶ ▶ -end

13490
13491
13492 # 50 graphs
13493
13494
13495
13496 spreads_27_starter_cliques:
13497 ▶ $(ORBITER) -v 2 \n13498 ▶ ▶ -loop L 0 50 1 \n13499 ▶ ▶ ▶ -define G -graph -load spreads_27_graph_%L.bin -end \n13500 ▶ ▶ ▶ -with G -do \n13501 ▶ ▶ ▶ -graph_theoretic_activity \n13502 ▶ ▶ ▶ ▶ -find_cliques -rainbow -target_size 23 -end \n13503 ▶ ▶ ▶ ▶ -end \n13504 ▶ ▶ ▶ -end_loop
13505
13506 #4:41
13507
13508
13509
13510 classify_spreads_27_isomorph_and_recognize:
13511 ▶ $(ORBITER) -v 3 \n
799
-define F -finite_field -q 3 -end \
-define P -projective_space -n 5 -field F -v 0 -end \
-define C -spread_classifier \
  -projective_space P \
-define k 3 \
-define starter_size 5 \
-recordinatize \
-poset_classification_control \
-draw_options \
-radius 20 \
-nodes_empty \
-line_width 0.2 \
-embedded \
-end \
-W \
-draw_poset \
-problem_label spreads_27 \
-end \
-output_prefix "" \
-end \
-with C -do -spread_classify_activity \
-compute_starter \
-problem_label spreads_27 \
-W -depth 5 \
-report -end \
-end \
-with C -do -spread_classify_activity \
-isomorph \
-prefix_iso "/spreads_27" \
-use_database_for_starter \
-build_db \
-solution_prefix "" \
-base_fname "" \
-end \
-end \
-with C -do -spread_classify_activity \
-isomorph \
-prefix_iso "/spreads_27" \
-use_database_for_starter \
-read_solutions \
-solution_prefix "" \
-base_fname "spreads_27_graph" \
-end \
-end \
-with C -do -spread_classify_activity \
-isomorph \
-prefix_iso "/spreads_27" \
-use_database_for_starter \
-read_solutions \
-solution_prefix "" \
-base_fname "spreads_27_graph" \
-end \
-end \
-with C -do -spread_classify_activity \
-isomorph \
-prefix_iso "/spreads_27" \
-use_database_for_starter \
-read_solutions \
-solution_prefix "" \
-base_fname "spreads_27_graph" \
-end \
-end \
-with C -do -spread_classify_activity \
-isomorph \
-prefix_iso "/spreads_27" \
-use_database_for_starter \
-read_solutions \
-solution_prefix "" 

13559 ⊲ △ △ -prefix_iso "./spreads_27" \n13560 ⊲ △ △ -use_database_for_starter \n13561 ⊲ △ △ -compute_orbits \n13562 ⊲ △ △ -solution_prefix "" \n13563 ⊲ △ △ -base_fname "spreads_27_graph" \n13564 ⊲ △ -end \n13565 ⊲ △ -end \n13566 ⊲ △ -with C -do -spread_classify_activity \n13567 ⊲ △ -isomorph \n13568 ⊲ △ △ -prefix_iso "./spreads_27" \n13569 ⊲ △ △ -use_database_for_starter \n13570 ⊲ △ △ -isomorph_testing \n13571 ⊲ △ △ -solution_prefix "" \n13572 ⊲ △ △ -base_fname "spreads_27_graph" \n13573 ⊲ △ -end \n13574 ⊲ △ -end \n13575 ⊲ △ -with C -do -spread_classify_activity \n13576 ⊲ △ △ -isomorph \n13577 ⊲ △ △ -prefix_iso "./spreads_27" \n13578 ⊲ △ △ -use_database_for_starter \n13579 ⊲ △ △ -isomorph_report \n13580 ⊲ △ △ -solution_prefix "" \n13581 ⊲ △ △ -base_fname "spreads_27_graph" \n13582 ⊲ △ △ -end \n13583 ⊲ △ △ -end \n13584 ⊲ △ -with C -do -spread_classify_activity \n13585 ⊲ △ △ -isomorph \n13586 ⊲ △ △ △ -prefix_iso "./spreads_27" \n13587 ⊲ △ △ △ -use_database_for_starter \n13588 ⊲ △ △ △ -recognize $(SPREAD_S27_RAO_RAO) \n13589 ⊲ △ △ △ -solution_prefix "" \n13590 ⊲ △ △ △ -base_fname "spreads_27_graph" \n13591 ⊲ △ △ △ -end \n13592 ⊲ △ △ -end

13593 #pdflatex spreads_27_isomorphism_types.tex
13594 #open spreads_27_isomorphism_types.pdf
13595 #pdflatex spreads_27_aut_group.tex
13596 #open spreads_27_aut_group.pdf
13597
13598
13599 # SPREAD_S27_RAO_RAO is isomorphic to spread 0 in the list
13600 # (which is different from the ordering of the Orbiter catalogue)
13601 # the stabilizer of the spread has order 84.
13602
13603
13604 #substructure_lifting_data::write_hash_and_datref_file id_to_hash tallied:
13605 #( 1^6076, 2^289, 3^35, 4^5 )
# using 64 bit hash values, based on a modified version of Paul Hsieh's SuperFast Hash

We found 7 isomorphism types

#0:36

generators for the stabilizer of the Rao/Rao spread:

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

create spread 27.0:

```bash
$ (ORBITER) -v 3 \n-define F -finite_field -q 3 -end \n-define G -linear_group -PGL 6 F -end \n-define S -spread -kernel_field F \n-group G -k 3 -catalogue 0 \n-end \n-with S -do -spread_activity \n-report \n-end
```

create spread 27.1:

```bash
$ (ORBITER) -v 3 \n-define F -finite_field -q 3 -end \n-define G -linear_group -PGL 6 F -end \n-define S -spread -kernel_field F \n-group G -k 3 -catalogue 1 \n-end \n-with S -do -spread_activity \n-report \n-end
```

create spread 27.2:

```bash
$ (ORBITER) -v 3 \n-define F -finite_field -q 3 -end \n-define G -linear_group -PGL 6 F -end \n-define S -spread -kernel_field F \n-group G -k 3 -catalogue 2 \n-end \n-with S -do -spread_activity \n-report \n```
create_spread_27.3:
$\text{ORBITER} -v 3 \$
\
\begin{verbatim}
define F finite_field -q 3 -end 
define G linear_group -PGL 6 F -end 
define S spread -kernel_field F 
group G -k 3 -catalogue 3 
with S -do -spread_activity 
report 
end
\end{verbatim}

pdflatex catalogue_q3_k3_3_report.tex
open catalogue_q3_k3_3_report.pdf

create_spread_27.4:
$\text{ORBITER} -v 3 \$
\
\begin{verbatim}
define F finite_field -q 3 -end 
define G linear_group -PGL 6 F -end 
define S spread -kernel_field F 
group G -k 3 -catalogue 4 
with S -do -spread_activity 
report 
end
\end{verbatim}

pdflatex catalogue_q3_k3_4_report.tex
open catalogue_q3_k3_4_report.pdf

create_spread_27.5:
$\text{ORBITER} -v 3 \$
\
\begin{verbatim}
define F finite_field -q 3 -end 
define G linear_group -PGL 6 F -end 
define S spread -kernel_field F 
group G -k 3 -catalogue 5 
with S -do -spread_activity 
report 
end
\end{verbatim}

pdflatex catalogue_q3_k3_5_report.tex
open catalogue_q3_k3_5_report.pdf

create_spread_27.6:
$\text{ORBITER} -v 3 \$
\
\begin{verbatim}
define F finite_field -q 3 -end 
define G linear_group -PGL 6 F -end 
\end{verbatim}

803
define S -spread -kernel_field F 
define F -finite_field -q 2 -end 
define P -projective_space -n 9 -field F -v 0 -end 
define C -spread_classifier 
projective_space P 
-k 5 
-starter_size 5 
-recoordinatize 
-poset_classification_control 
-draw_options 
-radius 20 
-nodes_empty 
-line_width 0.2 
-embedded 
-end 
-W -depth 5 
-draw_poset 
-problem_label spreads_32 
-end 
-output_prefix "" 
-end 
-with C -do -spread_classify_activity 
-compute_starter 
-problem_label spreads_32 
-W -depth 5 
-report -end 
-end 
-end 
-with C -do -spread_classify_activity 
-prepare_lifting_single_case 0 

classify_spreads_32_starter: 
$\$(ORBITER) -v 5 
-define F -finite_field -q 2 -end 
-define P -projective_space -n 9 -field F -v 0 -end 
-define C -spread_classifier 
-projective_space P 
-k 5 
-starter_size 5 
-recoordinatize 
-poset_classification_control 
-draw_options 
-radius 20 
-nodes_empty 
-line_width 0.2 
-embedded 
-end 
-W -depth 5 
-draw_poset 
-problem_label spreads_32 
-end 
-output_prefix "" 
-end 
-with C -do -spread_classify_activity 
-compute_starter 
-problem_label spreads_32 
-W -depth 5 
-report -end 
-end 
-end 
-with C -do -spread_classify_activity 
-prepare_lifting_single_case 0 

804
# we found 2887680 live points

spread_classify::init The stabilizer of the first three components has order 59996160

poset_with_group_action::init_subset_lattice degree of action = 109221651

# orbits at level in spreads_32_reps_lvl_4.csv

#0,"0,109221650,3515245,6933315",252,1454127528968048128661913600

#1,"0,109221650,3515245,7030463",124,2955162397580226842119372800

END

classify

-starter

-lift all cases:

-define F -finite_field -q 7 -end

-define P -projective_space -n 3 -field F -v 0 -end

-define C -spread_classifier

-projective space P

-k 2

-starter_size 5

-recoordinatize

-poset_classification_control

-draw_options

-radius 20

-nodes_empty

-line_width 0.2

-embedded

-end

-W

-draw_poset

-problem_label spreads_49

-end

-output_prefix ""

-end

-with C -do -spread_classify_activity

-compute_starter

-problem_label spreads_49

-W -depth 5

-report -end

-end

-end
# save_colored_graph fname=spreads_49_graph_125.bin

# 126 cases

# 126 cases

spreads_49_starter_cliques_loop:

$(ORBITER) -v 2 \
-define G -graph -load spreads_49_graph_0.bin -end \
-with G -do \
-graph_theoretic_activity \
-find_cliques -rainbow -target_size 45 -end \
-end \
-end_loop

spreads_49_starter_cliques_0:

$(ORBITER) -v 2 \
-define G -graph -load spreads_49_graph_0.bin -end \
-with G -do \
-graph_theoretic_activity \
-find_cliques -rainbow -target_size 45 -end \
-end 

# Section 12.2: Translation planes

SECTION_TRANSLATION_PLANES:

create_translation_plane_9b:

$(ORBITER) -v 3 \
-define F -finite_field -q 3 -end \
-define G -linear_group -PGL 4 F -end \
-define G1 -linear_group -PGL 5 F -end \
-define S -spread -kernel_field F \n-group G -k 2 -catalogue 1 \n-end \
-define T -translation_plane S G G1 -end \
-with T -do -translation_plane_activity \
-export_incma \n-end \

with T -do -translation_plane_activity \  
-define A -linear_group -import_group_of_plane T -end \  
-define Orb -orbits -group A \  
-define on_points \  
-end \  
-with Orb -do -orbits_activity \  
-report \  
-end \  
-with Orb -do -orbits_activity \  
-stabilizer 92 \  
-end \  
-with Orb -do -orbits_activity \  
-export_trees \  
-end \  
=$(ORBITER) -v 2 \  
-draw_matrix \  
-input_csv_file plane_catalogue_q3_k2_1_incma.csv \  
-box_width 6 -bit_depth 8 \  
-partition 2 91 91 \  
-end \  
=$(ORBITER) -v 3 \  
-draw_layered_graph \  
-orbit_PGL_5_3_on_andre_3.layered_graph \  
-radius 250 -spanning_tree -embedded -nodes_empty \  
-line_width 1.1 -x_stretch 2.4 -scale 0.15 \  
-end \  
pdflatex orbit_PGL_5_3_on_andre_3.draw.tex  
pdflatex group_of_plane_plane_catalogue_q3_k2_1_orbits_report.tex  
open group_of_plane_plane_catalogue_q3_k2_1_orbits_report.pdf  
pdflatex group_of_plane_plane_catalogue_q3_k2_1_stab_pt_92_report.tex  
open group_of_plane_plane_catalogue_q3_k2_1_stab_pt_92_report.pdf  
create_translation_plane_16_4_0: \  
=$(ORBITER) -v 3 \  
-define F -finite_field -q 4 -end \  
-define G -linear_group -PGGL 4 F -end \  
-define G1 -linear_group -PGGL 5 F -end \  
-define S -spread -kernel_field F \  
-group G -k 2 -catalogue 0 \  
-end \  
-define T -translation_plane S G G1 -end
$\texttt{(ORBITER)} -v 2 \ \$

$\texttt{\textasciitilde -draw\_matrix} \$

$\texttt{\textasciitilde -input\_csv\_file plane\_catalogue\_q4\_k2.0\_incma.csv} \$

$\texttt{\textasciitilde -box\_width 6 -bit\_depth 8} \$

$\texttt{\textasciitilde -partition 4 273 273} \$

$\texttt{-end} \$

$\texttt{open plane\_catalogue\_q4\_k2.0\_incma\_draw.bmp} \$

$\texttt{\textasciitilde}$

13894 #0 : "1200", // Hall spread
13895 #1 : "81600", // Desarguesian spread
13896 #2 : "576", // Semifield spread
13897
13898
13899 create\_translation\_plane\_16\_2\_0:
13900 $\texttt{(ORBITER)} -v 3 \$
13901 $\texttt{\textasciitilde -define F \textasciitilde \_finite\_field \_q 2 \_end} \$
13902 $\texttt{\textasciitilde -define G \textasciitilde \_linear\_group \_PGL 8 F \_end} \$
13903 $\texttt{\textasciitilde -define G1 \textasciitilde \_linear\_group \_PGL 9 F \_end} \$
13904 $\texttt{\textasciitilde -define S \textasciitilde \_spread \_kernel\_field F} \$
13905 $\texttt{\textasciitilde \_group G \_k 4 \_catalogue 0} \$
13906 $\texttt{\_end} \$
13907 $\texttt{\textasciitilde -define T \textasciitilde \_translation\_plane S G G1 \_end} \$
13908 $\texttt{(ORBITER)} -v 2 \$
13909 $\texttt{\textasciitilde -draw\_matrix} \$
13910 $\texttt{\textasciitilde -input\_csv\_file plane\_catalogue\_q2\_k4.0\_incma.csv} \$
13911 $\texttt{\textasciitilde -box\_width 6 -bit\_depth 8} \$
13912 $\texttt{\textasciitilde -partition 4 273 273} \$
13913 $\texttt{-end} \$
13914 $\texttt{open plane\_catalogue\_q2\_k4.0\_incma\_draw.bmp} \$
13915
13916 #0 : "1008",
13917 #1 : "1008",
13918 #2 : "1728",
13919 #3 : "216",
13920 #4 : "360",
13921 #5 : "288",
13922 #6 : "3600",
13923 #7 : "244800",
13924
13925 RREF\_plane\_16\_2\_0.\_rank\_of\_incma:
13926 $\texttt{(ORBITER)} -v 2 \$
13927 $\texttt{\textasciitilde -define F \textasciitilde \_finite\_field \_q 2 \_end} \$
13928 $\texttt{\textasciitilde -define v \textasciitilde \_vector \_field F} \$
13929 $\texttt{\textasciitilde -file plane\_catalogue\_q2\_k4.0\_incma.csv} \$
13930 $\texttt{-end} \$
13931 $\texttt{\_with F \_do \_finite\_field\_activity} \$

808
create_translation_plane_25_14_rank:

create_translation_plane_27_Rao_Rao:

# error message because the group of the spread is not available
RREF_Rao_Rao_plane_incma_rank:

```
$(ORBITER) -v 2 
-define F -finite_field -q 3 -end 
-define v -vector -field F 
-file plane_incma.csv 
-end 
-with F -finite_field_activity 
-RREF v -normalize_from_the_right 
-end
```

# 3-rank is 271, so the Rao / Rao plane is Moorhouse IV.

create_translation_plane_27.p_rank_of_incidence_matrix:

```
$(ORBITER) -v 3 
-define F -finite_field -q 3 -end 
-define G -linear_group -PGL 6 F -end 
-define G1 -linear_group -PGL 7 F -end 
-define S -spread -kernel_field F 
-group G -k 3 -catalogue 6 
-end 
-define T -translation_plane S G G1 -end 
-with T -do -translation_plane_activity 
-p_rank 3 
-end
```

# OCN : 3-rank : spread stab : pt-orb : line-orb : Moorhouse list
# 1 : 262 : 2106 : 1,27,729 : 1,27,729 : generalized twisted field = Moorhouse II
# 2 : 268 : 1014 : 2,26,729 : 1,54,702 : Andre = Moorhouse VII
# 4 : 274 : 1092 : 28,729 : 1,756 : Hering = Moorhouse III

# so, Rao / Rao is OCN=5

create_translation_plane_27.5.Rao_Rao:

```
$(ORBITER) -v 3 
-define F -finite_field -q 3 -end 
-define G -linear_group -PGL 6 F -end 
-define G1 -linear_group -PGL 7 F -end 
-define S -spread -kernel_field F 
```

810
define A -linear_group -import_group_of_plane T -end \\
> define Orb -orbits -group A \ 
> > on_points  
> end

create_translation_plane_27_4_block_stab:

create_translation_plane_27_5_block_stab:

$(ORBITER) -v 3
create_translation_plane_27_6_block_stab:

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

\[\text{(define F -finite_field -q 3 -end \} }\]
\[\text{(define G -linear_group -PGL 6 F -end \} }\]
\[\text{(define G1 -linear_group -PGL 7 F -end \} }\]
\[\text{(define S -spread -kernel_field F \} }\]
\[\text{(define T -translation_plane S G G1 -end \} }\]
\[\text{(with T -do -translation_plane_activity \} }\]
\[\text{(export incma \} }\]
\[\text{(end \} }\]
\[\text{(with T -do -translation_plane_activity \} }\]
\[\text{(report \} }\]
\[\text{(end \} }\]
\[\text{(define A -linear_group -import_group_of_plane T -end \} }\]
\[\text{(define Orb -orbits -group A \} }\]
\[\text{(on_points \} }\]
\[\text{(end \} }\]
\[\text{(end \} }\]

# Section 12.3: Packings
SECTION_PACKINGS:

spread_table_PG_3_4:

```
mkdir SPREAD_TABLES_4
$(ORBITER) -v 6 \n  -define F -finite_field -q 4 -end \n  -define P -projective_space -n 3 -field F -v 0 -end \n  -define T -spread_table P 2 "0,1,2" "SPREAD_TABLES_4/" \n
5096448 spreads
1020 self dual spreads
User time: 56:38 on Mac
```

spread_table_PG_3_5_regular:

```
mkdir SPREAD_TABLES_5_REG
$(ORBITER) -v 6 \n  -define F -finite_field -q 5 -end \n  -define P -projective_space -n 3 -field F -end \n  -define T -spread_table P 2 "12" "SPREAD_TABLES_5_REG/" \n  -print_symbols
```

# error.

# 21 isomorphism types of spreads in PG(3,5)
# 12 is the index of the regular spread in the classification of spreads
# 12/9/2020: 34 sec on Mac
# 155000 spreads

PG_3_5_element_of_order_31_GL_normalizer:

```
$(ORBITER) -v 6 -define G \n  -linear_group -GL 4 5 -end \n  -with G -do \n  -group_theoretic_activity \n  -normalize_of_cyclic_subgroup "124" \n  "2,0,0,0, 0,0,1,0, 0,0,0,1, 0,3,0,4" \n  -end
```

# mv normalizer_of_31_in_PGL_4_5.tex normalizer_of_31_AB_in_PGL_4_5.tex
# pdflatex normalizer_of_124_in_GL_4_5.tex
# open normalizer_of_124_in_GL_4_5.pdf

# needs magma

# the group has order 124.
the normalizer has order 1488
normalizer has order 1488 = 4 * 372 = 4 * 4 * 3 * 31
PG 3,5_element_of_order_31_ME_normalizer:
\$(\text{ORBITER}) -v 6 -define G \\n\$-\text{linear}\_\text{group} -\text{PGL} 4 5 -end \$
\$-\text{with}\ G -do \$
\$-\text{group}\_\text{theoretic}\_\text{activity} \$
\$-\text{normalizer}\_\text{of}\_\text{cyclic}\_\text{subgroup}\ "31" \$
\$-\"1,0,0,0, 0,3,4,3, 0,3,3,4, 0,3,2,3\" \$
\$-\text{end} \$
\$\text{mv}\ \text{normalizer}\_\text{of}\_31\ \text{in}\_\text{PGL}\_4 5\_\text{.tex}\ \text{normalizer}\_\text{of}\_31\_\text{ME}\_\text{in}\_\text{PGL}\_4 5\_\text{.tex}\$
\$\text{pdflatex}\ \text{normalizer}\_\text{of}\_31\_\text{ME}\_\text{in}\_\text{PGL}\_4 5\_\text{.tex}\$
\$\text{open}\ \text{normalizer}\_\text{of}\_31\_\text{ME}\_\text{in}\_\text{PGL}\_4 5\_\text{.pdf}\$
\$\text{# group has order 31}\$
\$\text{normalizer has order 372}\$
\$\text{PG 3,5_assume_31_graph:}\$
\$\$(\text{ORBITER}) -v 5 \$
\$\text{-define F -finite\_field -q 5 -end} \$
\$\text{-define P -projective\_space -n 3 -field F -end} \$
\$\text{-define T -spread\_table P 2 "12" "SPREAD\_TABLES.5\_REG/"} \$
\$\text{-define PW -packing\_with\_symmetry\_assumption T} \$
\$\text{-define PWF -packing\_choose\_fixed\_points PW 0 -end} \$
\$\text{-define L -packing\_long\_orbits PWF} \$
\$\text{-orbit\_length 31 -clique\_size 1 -create\_graphs -end} \$
\$\text{-print\_symbols} \$
\$\text{pdflatex H31\_reduced\_spread\_orbits\_orbits\_report.tex} \$
\$\text{open H31\_reduced\_spread\_orbits\_orbits\_report.pdf} \$
\$\text{pdflatex H31\_line\_orbits\_orbits\_report.tex} \$
\$\text{open H31\_line\_orbits\_orbits\_report.pdf} \$
\$\text{pdflatex H31\_line\_orbits\_orbits\_report.tex} \$
\$\text{open H31\_line\_orbits\_orbits\_report.pdf} \$
\$\text{pdflatex N31\_line\_orbits\_orbits\_report.tex} \$
\$\text{open N31\_line\_orbits\_orbits\_report.pdf} \$
\$\text{pdflatex H31\_point\_orbits\_orbits\_report.tex} \$
\$\text{open H31\_point\_orbits\_orbits\_report.pdf} \$
\$\text{pdflatex N31\_point\_orbits\_orbits\_report.tex} \$
\$\text{open N31\_point\_orbits\_orbits\_report.pdf} \$
\$814$
**Section 12.4: BLT-sets**

**BLT 5-1:**

```bash
$ (ORBITER) -v 2 \\
-define F -finite_field -q 5 -end \\
-define 0 -orthogonal_space 0 5 F -end \\
-with 0 -do -orthogonal_space_activity \\
-create_BLT_set -catalogue 1 -end \\
-end
```

```
pdflatex catalogue_q5_isol1.tex
open catalogue_q5_isol1.pdf
```

**BLT 5_Linear:**

```bash
$ (ORBITER) -v 2 \\
-define F -finite_field -q 5 -end \\
-define 0 -orthogonal_space 0 5 F -end \\
-with 0 -do -orthogonal_space_activity \\
-create_BLT_set -family "Linear" -end \\
-end
```

```
pdflatex BLT_Linear_q5.tex
open BLT_Linear_q5.pdf
```

**BLT 9_K1:**

```bash
$ (ORBITER) -v 2 \\
-define F -finite_field -q 9 -end \\
-define 0 -orthogonal_space 0 5 F -end \\
-with 0 -do -orthogonal_space_activity \\
-create_BLT_set -family "K1" -end \\
-end
```

```
pdflatex BLT_K1_q9.tex
open BLT_K1_q9.pdf
```
14263 BLT_11_0:
14264 \>(\text{ORBITER}) -v 2 \n14265 \> -define F -finite_field -q 11 -end \n14266 \> -define O -orthogonal_space 0 5 F -end \n14267 \> -with 0 -do -orthogonal_space_activity \n14268 \> \> -create_BLT_set -catalogue 0 -end \n14269 \> -end
14270 \> \#pdflatex 0_1_6_2_report.tex
14271 \> \#open 0_1_6_2_report.pdf
14272
14273
14274 BLT_11_Fisher:
14275 \>(\text{ORBITER}) -v 2 \n14276 \> -define F -finite_field -q 11 -end \n14277 \> -define O -orthogonal_space 0 5 F -end \n14278 \> -with 0 -do -orthogonal_space_activity \n14279 \> \> -create_BLT_set -family "Fisher" -end \n14280 \> -end
14281 \> pdflatex BLT_Fisher_q11.tex
14282 \> open BLT_Fisher_q11.pdf
14283
14284 BLT_11_Mondello:
14285 \>(\text{ORBITER}) -v 2 \n14286 \> -define F -finite_field -q 11 -end \n14287 \> -define O -orthogonal_space 0 5 F -end \n14288 \> -with 0 -do -orthogonal_space_activity \n14289 \> \> -create_BLT_set -family "Mondello" -end \n14290 \> -end
14291 \> pdflatex BLT_Mondello_q11.tex
14292 \> open BLT_Mondello_q11.pdf
14293
14294
14295 BLT_13_FTWKB:
14296 \>(\text{ORBITER}) -v 2 \n14297 \> -define F -finite_field -q 11 -end \n14298 \> -define O -orthogonal_space 0 5 F -end \n14299 \> -with 0 -do -orthogonal_space_activity \n14300 \> \> -create_BLT_set -family "FTWKB" -end \n14301 \> -end
14302 \> pdflatex BLT_FTWKB_q11.tex
14303 \> open BLT_FTWKB_q11.pdf
14304
14305
14306 \# for K2, q must be congruent to 2 or 3 mod 5
14307 BLT_13_K2:
14308 \>$\text{(ORBITER)} \ -v \ 2 \ \\
14309 \> \> \> -define F \ -finite_field \ -q \ 13 \ -end \ \\
14310 \> \> \> -define O \ -orthogonal_space \ 0 \ 5 \ F \ -end \ \\
14311 \> \> \> -with O \ -do \ -orthogonal_space_activity \ \\
14312 \> \> \> \> -create_BLT_set \ -family \ "Kantor2" \ -end \ \\
14313 \> \> \> -end
14314 \> \> pdflatex BLT_K2_q13.tex
14315 \> \> open BLT_K2_q13.pdf
14316
14317
14318
14319
14320
14321
14322 BLT_13_deep_14:
14323 \>$\text{(ORBITER)} \ -v \ 2 \ \\
14324 \> \> \> -define F \ -finite_field \ -q \ 13 \ -end \ \\
14325 \> \> \> -define O \ -orthogonal_space \ 0 \ 5 \ F \ -end \ \\
14326 \> \> \> -with O \ -do \ -orthogonal_space_activity \ \\
14327 \> \> \> \> -BLT_set_starter \ 14 \ \\
14328 \> \> \> \> \> -problem_label BLT_q13 \ -W \ -depth \ 14 \ -end \ \\
14329 \> \> \> \> -end
14330
14331 BLT_11_deep_search:
14332 \>$\text{(ORBITER)} \ -v \ 2 \ \\
14333 \> \> \> -define F \ -finite_field \ -q \ 11 \ -end \ \\
14334 \> \> \> -define O \ -orthogonal_space \ 0 \ 5 \ F \ -end \ \\
14335 \> \> \> -define C \ -BLT_set_classifier \ 0 \ -starter_size \ 12 \ -end \ \\
14336 \> \> \> -with C \ -do \ -BLT_set_classify_activity \ \\
14337 \> \> \> \> -compute_starter \ \\
14338 \> \> \> \> \> -problem_label BLT_q11 \ \\
14339 \> \> \> \> \> \> -W \ -depth \ 12 \ \\
14340 \> \> \> \> \> \> \> -report \ -end \ \\
14341 \> \> \> \> \> \> -end \ \\
14342 \> \> \> -end
14343 \> \> pdflatex BLT_q11_poset.tex
14344 \> \> open BLT_q11_poset.pdf
14345
14346
14347
14348
14349 BLT_13_deep_search:
14350 \>$\text{(ORBITER)} \ -v \ 2 \ \\
14351 \> \> \> -define F \ -finite_field \ -q \ 13 \ -end \ \\
14352 \> \> \> -define O \ -orthogonal_space \ 0 \ 5 \ F \ -end \ \\
14353 \> \> \> -define C \ -BLT_set_classifier \ 0 \ -starter_size \ 14 \ -end \ \\
817
14354 \> \> \> with C -do -BLT_set_classify_activity \\
14355 \> \> \> -compute_starter \\
14356 \> \> \> \> -problem_label BLT_q13 \\
14357 \> \> \> \> \> -W -depth 14 \\
14358 \> \> \> \> \> -report -end \\
14359 \> \> \> \> \> -end \\
14360 \> \> \> -end \\
14361 \> pdflatex BLT_q13_poset.tex \\
14362 \> open BLT_q13_poset.pdf \\
14363 \\
14364 \\
14365 \\
14366 \\
14367 BLT.13.classify_starter: \\
14368 \> $(ORBITER) -v 2 \\
14369 \> \> -define F -finite_field -q 13 -end \\
14370 \> \> -define O -orthogonal_space 0 5 F -end \\
14371 \> \> \> -define C -BLT_set_classifier O -starter size 5 -end \\
14372 \> \> \> \> -with C -do -BLT_set_classify_activity \\
14373 \> \> \> \> -compute_starter \\
14374 \> \> \> \> \> -problem_label BLT_q13 \\
14375 \> \> \> \> \> \> \> -W -depth 5 \\
14376 \> \> \> \> \> \> \> \> -end \\
14377 \> \> \> \> \> \> -end \\
14378 \> \> \> \> \> \> \> \> \> -with C -do -BLT_set_classify_activity \\
14379 \> \> \> \> \> \> \> \> \> \> -create_graphs \\
14380 \> \> \> \> \> \> \> \> \> \> -end \\
14381 \\
14382 \\
14383 \\
14384 \\
14385 BLT.13.clique: \\
14386 \> $(ORBITER) -v 2 \\
14387 \> \> -loop L 0 38 1 \\
14388 \> \> \> \> -define G -graph -load BLT_q13_graph.5_%L.bin -end \\
14389 \> \> \> \> \> -with G -do \\
14390 \> \> \> \> \> \> -graph_theoretic_activity \\
14391 \> \> \> \> \> \> \> \> \> -find_cliques -rainbow -target_size 9 -end \\
14392 \> \> \> \> \> \> \> \> \> \> -end \\
14393 \> \> \> \> \> \> \> \> \> \> \> -end_loop \\
14394 \\
14395 \\
14396 # 3 solutions: \\
14397 #BLT_q13_graph.5_0_sol.txt \\
14398 #BLT_q13_graph.5_0_sol.csv \\
14399 \\
14400
BLT_13_isomorph_read_DB:

$\text{ORBITER} -v 2 \$

-define F -finite_field -q 13 -end \\
-define O -orthogonal_space 0 5 F -end \\
-define C -BLT_set_classifier 0 -starter_size 5 -end \\
-with C -do -BLT_set_classify_activity \\
-compute_starter \\
-problem_label BLT_q13 \\
-W -depth 5 \\
-end \\
-with C -do -BLT_set_classify_activity \\
isomorph \\
-prefix ./BLT_q13 \\
-use_database_for_starter \\
-build_db \\
solution_prefix "" \\
-base_fname "" \\
-end \\
-end

BLT_13_isomorph_read_solutions:

$\text{ORBITER} -v 2 \$

-define F -finite_field -q 13 -end \\
-define O -orthogonal_space 0 5 F -end \\
-define C -BLT_set_classifier 0 -starter_size 5 -end \\
-with C -do -BLT_set_classify_activity \\
-compute_starter \\
-problem_label BLT_q13 \\
-W -depth 5 \\
-end \\
-with C -do -BLT_set_classify_activity \\
isomorph \\
-prefix ./BLT_q13 \\
-use_database_for_starter \\
-read_solutions \\
-list_of_cases BLT_q13.list_of_cases_5.0.1.csv \\
solution_prefix "" \\
-base_fname "BLT_q13_graph" \\
-end \\
-end
14448 BLT_13.isomorph_stabilizer_orbits:
14449 ⦿ $(ORBITER) -v 2 \ 
14450 ⦿ ⦿ -define F -finite_field -q 13 -end \ 
14451 ⦿ ⦿ -define O -orthogonal_space 0 5 F -end \ 
14452 ⦿ ⦿ -define C -BLT_set_classifier 0 -starter_size 5 -end \ 
14453 ⦿ ⦿ -with C -do -BLT_set_classify_activity \ 
14454 ⦿ ⦿ ⦿ -compute_starter \ 
14455 ⦿ ⦿ ⦿ ⦿ -problem_label BLT_q13 \ 
14456 ⦿ ⦿ ⦿ ⦿ -W -depth 5 \ 
14457 ⦿ ⦿ ⦿ ⦿ -end \ 
14458 ⦿ ⦿ ⦿ -end \ 
14459 ⦿ ⦿ ⦿ -with C -do -BLT_set_classify_activity \ 
14460 ⦿ ⦿ ⦿ ⦿ -isomorph \ 
14461 ⦿ ⦿ ⦿ ⦿ ⦿ -prefix_iso "./BLT_q13" \ 
14462 ⦿ ⦿ ⦿ ⦿ ⦿ -use_database_for_starter \ 
14463 ⦿ ⦿ ⦿ ⦿ ⦿ -compute_orbits \ 
14464 ⦿ ⦿ ⦿ ⦿ ⦿ -list_of_cases BLT_q13_list_of_cases_5_0_1.csv \ 
14465 ⦿ ⦿ ⦿ ⦿ ⦿ ⦿ -solution_prefix "" \ 
14466 ⦿ ⦿ ⦿ ⦿ ⦿ ⦿ -base_fname "BLT_q13_graph" \ 
14467 ⦿ ⦿ ⦿ ⦿ -end \ 
14468 ⦿ ⦿ ⦿ -end \ 
14469 ⦿ -end \ 
14470 BLT_13.isomorph_testing:
14471 ⦿ $(ORBITER) -v 4 \ 
14472 ⦿ ⦿ -define F -finite_field -q 13 -end \ 
14473 ⦿ ⦿ -define O -orthogonal_space 0 5 F -end \ 
14474 ⦿ ⦿ -define C -BLT_set_classifier 0 -starter_size 5 -end \ 
14475 ⦿ ⦿ -with C -do -BLT_set_classify_activity \ 
14476 ⦿ ⦿ ⦿ -compute_starter \ 
14477 ⦿ ⦿ ⦿ ⦿ -problem_label BLT_q13 \ 
14478 ⦿ ⦿ ⦿ ⦿ -W -depth 5 \ 
14479 ⦿ ⦿ ⦿ ⦿ -report -end \ 
14480 ⦿ ⦿ ⦿ -end \ 
14481 ⦿ ⦿ -end \ 
14482 ⦿ ⦿ -with C -do -BLT_set_classify_activity \ 
14483 ⦿ ⦿ ⦿ -isomorph \ 
14484 ⦿ ⦿ ⦿ ⦿ -prefix_iso "./BLT_q13" \ 
14485 ⦿ ⦿ ⦿ ⦿ -use_database_for_starter \ 
14486 ⦿ ⦿ ⦿ ⦿ -isomorph_testing \ 
14487 ⦿ ⦿ ⦿ ⦿ -solution_prefix "" \ 
14488 ⦿ ⦿ ⦿ ⦿ -base_fname "BLT_q13_graph" \ 
14489 ⦿ ⦿ ⦿ -end \ 
14490 ⦿ ⦿ -end \ 
14491 ⦿ ⦿ -with C -do -BLT_set_classify_activity \ 
14492 ⦿ ⦿ ⦿ -isomorph \ 
14493 ⦿ ⦿ ⦿ ⦿ -prefix_iso "./BLT_q13" \ 
14494 ⦿ ⦿ ⦿ ⦿ -use_database_for_starter \ 
14495 ⦿ ⦿ -end \
SECTION_CREATING_GRAPHS:

Cycle_graph_13:

$\text{ORBITER} -v 2 \\
\text{-define Gamma -graph} \\
\text{-cycle 13} \\
\text{-end}$

make_triangle_graph:

$\text{echo} $(TRIANGLE_GRAPH) >\text{triangle_graph.csv}$

$\text{ORBITER} -v 6 \\
\text{-define G -graph} \\
\text{-load_csv_no_border} \\
\text{-triangle_graph.csv} \\
\text{-end}$

Chain_232:

$\text{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 

trihedral_pair_graph:
$(ORBITER) -v 2 \\
-define Gamma \\
-graph -trihedral_pair_disjointness_graph \\
-end

small_graph:
$(ORBITER) -v 2 \\
-define G -graph -edges_as_pairs \\
5 "0,1,0,2,0,3,0,4,1,3,1,4,2,4" \\
-end

petersen:
$(ORBITER) -v 2 \\
-define G -graph -Johnson 5 2 0 -end 

Johnson_6.2.0:
$(ORBITER) -v 2 \\
-define G -graph -Johnson 6 2 0 -end 

Hamming_graph_3:
$(ORBITER) -v 2 \\
-define G -graph -Hamming 3 2 -end
Hamming graph 7:
$\$(ORBITER) -v 2 \$
$>$-define G -graph -Hamming 7 2 -end

# needs halljanko315.csv
# from https://www.win.tue.nl/~aeb/drg/graphs/HJ315.html

There is a unique distance-regular graph Gamma with intersection array \{10,8,8,2 ; 1,1,4,5\}. It was constructed in Cohen (1981), and uniqueness (given the intersection array) was proved in Cohen & Tits (1985).

HJ graph:
$\$(ORBITER) -v 6 \$
$>$-define G -graph \$
$>$ $>$ $>$ -load_csv_no_border \$
$>$ $>$ $>$ halljanko315.csv \$
$>$ $>$ $>$ -end \$

HJ315 orbital graph 3:
$\$(ORBITER) -v 2 \$
$>$ $>$ $>$ -define gens -vector -file \$
$>$ $>$ $>$ $>$ halljanko315 gens.csv -end \$
$>$ $>$ $>$ $>$ -define G -permutation_group \$
$>$ $>$ $>$ $>$ $>$ -bsgs halljanko315 "File\ halljanko315" \$
$>$ $>$ $>$ $>$ 315 1209600 "0,1,2" 6 gens \$
$>$ $>$ $>$ $>$ -end \$
$>$ $>$ $>$ -define Gamma -graph \$
$>$ $>$ $>$ $>$ -orbital_graph G 3 \$
$>$ $>$ $>$ -end \$

HJ_d2_graph:
$\$(ORBITER) -v 6 \$
$>$ $>$ -define G -graph \$
$>$ $>$ $>$ -load_csv_no_border \$
$>$ $>$ $>$ halljanko315.csv \$
$>$ $>$ $>$ -distance_2 \$
$>$ $>$ $>$ -end \$

Cayley Z11_1mod3:
$\$(ORBITER) -v 2 \$
$>$ $>$ -define F -finite_field -q 11 -end \$
$>$ $>$ $>$ -define S -vector -dense \$

823
Cayley_Sym4.coxeter:

$($ORBITER) -v 2 \
$define S -vector -dense "1,0,2,3, 0,2,1,3, 0,1,3,2" -end \n$define G -permutation_group -symmetric_group 4 \n$end \
$define Gamma -graph \n$Cayley_graph G S \n$end

Cayley_Sym4.star:

$($ORBITER) -v 2 \
$define S -vector -dense "1,0,2,3, 2,1,0,3, 3,1,2,0" -end \n$define G -permutation_group -symmetric_group 4 \n$end \
$define Gamma -graph \n$Cayley_graph G S \n$end

# Section 13.2: Graphs Theoretic Activities
SECTION_GRAPH_THEORETIC_ACTIVITIES:

triangle_graph_properties:

echo $(TRIANGLE_GRAPH) >triangle_graph.csv
$ORBITER -v 6 \
$define G -graph \n$load_csv_no_border \
$triangle_graph.csv \
$end \
$with G -do \
$graph_theoretic_activity -properties \

Cycle_13.draw:
$(ORBITER) -v 2 \ 
-define Gamma -graph -cycle 13 -end \n-with Gamma -do \n-graph_theoretic_activity -export.csv -end \n-with Gamma -do \n-graph_theoretic_activity -export_graphviz -end \n$(ORBITER) -v 2 -draw_matrix \n-input_csv_file Cycle_13.csv \n-box_width 20 -bit_depth 8 -partition 4 13 13 -end \n-dt -Tpng Cycle_13.gv >Cycle_13.png \n#twopi -Tpng Cycle_13.gv >Cycle_13.png \n#open Cycle_13_draw.bmp \n#pdflatex Cycle_13_report.tex \n#open Cycle_13_report.pdf
Cycle_9.eigenvalues:
$(ORBITER) -v 2 \n-define Gamma -graph \ncycle 9 \n-end \n-with Gamma -do \n-graph_theoretic_activity -eigenvalues -end \npdflatex Cycle_9_eigenvalues.tex \nopen Cycle_9_eigenvalues.pdf
Paley_13.draw:
$(ORBITER) -v 2 \n-define Gamma -graph -Paley 13 -end \n-with Gamma -do \n-graph_theoretic_activity -export.csv -end \n-with Gamma -do \n-graph_theoretic_activity -export_graphviz -end \n$(ORBITER) -v 2 -draw_matrix \n-input_csv_file Paley_13.csv \n-box_width 20 -bit_depth 8 -partition 4 13 13 -end \ndt -Tpng Paley_13.gv >Paley_13.png \nopen Paley_13_draw.bmp
Paley_13.eigenvalues:
$(ORBITER) -v 2 \n-define Gamma -graph \n
Substitute: $\gamma$ for $\lambda$.

Cayley graph AE: $G = \text{AGL}(1, q)$, $S = \langle a, b \rangle$, $a = (i, 0)$, $b = (i, a_{2}i + b_{2})$.
\begin{verbatim}
14775 \$\text{ORBITER} \-v 2 \$
14776 \$ -draw_options \ -xin 1000000 \ -yin 1000000 \$
14777 \$ -embedded \ -radius 10000 \ -nodes_empty \-end \$
14778 \$ -define S \ -vector \ -dense \$
14779 \$ -define G \ -permutation_group \ -symmetric_group 5 \$
14780 \$ -define Gamma \ -graph \$
14781 \$ -Cayley \graph G S \$
14782 \$ -end \$
14783 \$ -with Gamma \ -do \$
14784 \$ -graph_theoretic_activity \ -draw \ -end \$
14785 \$ \text{pdflatex} \text{Cayley}\_\text{graph}\_\text{Perm5}\_\text{draw}.\text{tex} \$
14786 \$ \text{open} \text{Cayley}\_\text{graph}\_\text{Perm5}\_\text{draw}.\text{pdf} \$
14787 \$ \text{Cayley}\_\text{Sym4}\_\text{star}\_\text{eigenvalues}\_\text{and}\_\text{draw}: \$
14788 \$ \$\text{ORBITER} \-v 2 \$
14789 \$ \$ -draw_options \ -xin 1000000 \ -yin 1000000 \ -embedded \ -end \$
14790 \$ \$ -define S \ -vector \ -dense \"1,0,2,3,4, 0,2,1,3,4, 0,1,3,2,4, 0,1,2,4,3\" \ -end \$
14791 \$ \$ -define G \ -permutation_group \ -symmetric_group 4 \$
14792 \$ \$ -define Gamma \ -graph \$
14793 \$ \$ -Cayley \graph G S \$
14794 \$ \$ -end \$
14795 \$ \$ -with Gamma \ -do \$
14796 \$ \$ -graph_theoretic_activity \ -eigenvalues \ -end \$
14797 \$ \$ -with Gamma \ -do \$
14798 \$ \$ -graph_theoretic_activity \ -draw \ -end \$
14799 \$ \$ \text{pdflatex} \text{Cayley}\_\text{graph}\_\text{Perm4}\_\text{draw}.\text{tex} \$
14800 \$ \$ \text{open} \text{Cayley}\_\text{graph}\_\text{Perm4}\_\text{draw}.\text{pdf} \$
14801 \$ \$ \text{pdflatex} \text{Cayley}\_\text{graph}\_\text{Perm4}\_\text{eigenvalues}.\text{tex} \$
14802 \$ \$ \text{open} \text{Cayley}\_\text{graph}\_\text{Perm4}\_\text{eigenvalues}.\text{pdf} \$
14803 \$ \$ \text{graph_v5}_e7.colored_\text{graph}: \$
14804 \$ \$ \text{ORBITER} \-v 2 \$
14805 \$ \$ -define G \ -graph \ -edges_as_pairs 5 \$
14806 \$ \$ -define G \ -graph \ -edges_as_pairs 5 \$
14807 \$ \$ -with G \ -do \$
14808 \$ \$ -graph_theoretic_activity \ -save \ -end \$
14809 \$ \$ \text{graph_v5}_e7.colored_\text{graph} \$
14810 \$ \$ \text{graph_v5}_e7.colored_\text{graph} \$
14811 \$ \$ \text{graph_v5}_e7.colored_\text{graph} \$
14812 \$ \$ \text{graph_v5}_e7.colored_\text{graph} \$
14813 \$ \$ \text{graph_v5}_e7.colored_\text{graph} \$
14814 \$ \$ \text{graph_v5}_e7.colored_\text{graph} \$
14815 \$ \$ \text{graph_v5}_e7.colored_\text{graph} \$
14816 \$ \$ \text{graph_v5}_e7.colored_\text{graph} \$
14817 \$ \$ \text{graph_v5}_e7.colored_\text{graph} \$
14818 \$ \$ \text{graph_v5}_e7.colored_\text{graph} \$
14819 \$ \$ \text{graph_v5}_e7.colored_\text{graph} \$
14820 \$ \$ \text{graph_v5}_e7.colored_\text{graph} \$
14821 \$ \$ \text{graph_v5}_e7.colored_\text{graph} \$
\end{verbatim}
14822 small_graph_draw:
14823 $ (ORBITER) -v 2 \\
14824 $define G -graph -edges_as_pairs 5 \\
14825 $ "0,1,0,2,0,3,0,4,1,3,1,4,2,4" \\
14826 $ -end \\
14827 $with G -do \\
14828 $graph_theoretic_activity -export_csv -end \\
14829 $with G -do \\
14830 $graph_theoretic_activity -export_graphviz -end \\
14831 $with G -do \\
14832 $graph_theoretic_activity -save -end \\
14833 $(ORBITER) -v 2 -draw_matrix \\
14834 $input_csv_file graph_v5_e7.csv \\
14835 $box_width 40 -bit_depth 24 \\
14836 $partition 4 "1,1,1,1" "1,1,1,1" -end \\
14837 dot -Tpng graph_v5_e7.gv > graph_v5_e7.png \\
14838 # creates graph_v5_e7.csv
14840 # creates graph_v5_e7.colored_graph
14841
14842 petersen_draw:
14843 $(ORBITER) -v 2 \\
14844 $define G -graph -Johnson 5 2 0 -end \\
14845 $with G -do \\
14846 $graph_theoretic_activity -export_csv -end \\
14847 $with G -do \\
14848 $graph_theoretic_activity -export_graphviz -end \\
14849 $with G -do \\
14850 $graph_theoretic_activity -save -end \\
14852 $(ORBITER) -v 2 -draw_matrix \\
14853 $input_csv_file Johnson_v5_2_0.csv \\
14854 $box_width 40 -bit_depth 24 -partition 4 "10" "10" -end \\
14855 dot -Tpng Johnson_v5_2_0.gv > Johnson_v5_2_0.png \\
14856
14857 Johnson_6_2_0_draw:
14859 $(ORBITER) -v 2 \\
14860 $define G -graph -Johnson 6 2 0 -end \\
14861 $with G -do \\
14862 $graph_theoretic_activity -export_csv -end \\
14863 $with G -do \\
14864 $graph_theoretic_activity -export_graphviz -end \\
14865 $with G -do \\
14866 $graph_theoretic_activity -save -end \\
14867 $(ORBITER) -v 2 -draw_matrix \\
14868 $input_csv_file Johnson_6_2_0.csv \\

828
Hamming_graph_3_draw:
$(ORBITER) -v 2 \n\ndefine G -graph -Hamming 3 2 -end \n\nwith G -do \n
-graph_theoretic_activity -export_csv -end \n
with G -do \n
-graph_theoretic_activity -export_graphviz -end \n
-graph_theoretic_activity -save -end

$(ORBITER) -v 2 -draw_matrix \n\n-input_csv_file Hamming_3_2.csv \n\n-box_width 40 -bit_depth 24 \n
-partition 4 "1,1,1,1,1,1,1,1" "1,1,1,1,1,1,1,1" -end

dot -Tpng Hamming_3_2.gv >Hamming_3_2.png

Hamming_graph_7_draw:
$(ORBITER) -v 2 \n\ndefine G -graph -Hamming 7 2 -end \n\nwith G -do \n
-graph_theoretic_activity -export_csv -end \n
with G -do \n
-graph_theoretic_activity -export_graphviz -end \n
-graph_theoretic_activity -save -end

$(ORBITER) -v 2 -draw_matrix \n\n-input_csv_file Hamming_7_2.csv \n\n-box_width 8 -bit_depth 24 -partition 4 128 128 -end

dot -Tpng Hamming_7_2.gv >Hamming_7_2.png

Chain_232.properties:
$(ORBITER) -v 2 \n\ndefine P1 -vector -dense 2,3,2 -end \n\ndefine P2 -vector -dense 2,3,2 -end \n\ndefine Gamma -graph \n\n-chain_graph P1 P2 \n\n-end \n
829
with Gamma -do \  
-graph_theoretic_activity -export_csv \  
-end \  
-with Gamma -do \  
-graph_theoretic_activity -properties \  
-end  

Chain_232_eigen:  
$(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 \  
eigenvalues \  
-end  
pdflatex chain_graph_eigenvalues.tex  
open chain_graph_eigenvalues.pdf  

# 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 \  

triangular_pair_graph_draw:
	$(ORBITER) -v 2 -define Gamma \\
	$ORBITER -v 2 -graph triangular_pair_disjointness_graph -end \\
	$ORBITER -v 2 -graph_theoretic_activity -export_csv -end

gamma_graph: O 5 2 incidence matrix.csv

# Degree type: (80^-315)
tournament_classify_4:

```
$(ORBITER) -v 2 \
-define GC -graph_classification \
-n 4 -tournament \
-poset_classification_control \
-problem_label tournament_4 \
-depth 6 -draw_poset \
-draw_options \
-radius 250 -embedded \
-end \
-end \
-with GC -do \
-graph_classification_activity \
-draw_options \
-radius 300 -nodes_empty \
-line_width 1.5 \
-scale 0.1 \
-end \
-draw_graphs_at_level 5 \
-end \
-print_symbols
```

15057 ▸ pdflatex tournament_4_level_6_reps.tex
15058 ▸ open tournament_4_level_6_reps.pdf
15059 ▸
15060
15061
15062
15063
15064 graph_classify_8_r3:
15065 ▸ $(ORBITER) -v 3 \\
15066 ▸ ▸ -define GC -graph_classification \n15067 ▸ ▸ ▸ -n 8 -regular 3 \n15068 ▸ ▸ ▸ -poset_classification_control \n15069 ▸ ▸ ▸ ▸ -problem_label graphs_v8_r3 \n15070 ▸ ▸ ▸ ▸ -depth 12 -draw_poset \n15071 ▸ ▸ ▸ ▸ -draw_options -radius 250 \n15072 ▸ ▸ ▸ ▸ ▸ -line_width 0.2 -embedded \n15073 ▸ ▸ ▸ ▸ ▸ ▸ -end \n15074 ▸ ▸ ▸ ▸ ▸ ▸ ▸ -end \n15075 ▸ ▸ ▸ ▸ ▸ ▸ ▸ -with GC -do \n15076 ▸ ▸ ▸ ▸ ▸ ▸ -graph_classification_activity \n15077 ▸ ▸ ▸ ▸ ▸ ▸ -draw_options \n15078 ▸ ▸ ▸ ▸ ▸ ▸ ▸ -radius 400 \n15079 ▸ ▸ ▸ ▸ ▸ ▸ ▸ ▸ -line_width 2 -scale 0.10 \n15080 ▸ ▸ ▸ ▸ ▸ ▸ ▸ ▸ ▸ -end \n15081 ▸ ▸ ▸ ▸ ▸ ▸ ▸ ▸ ▸ ▸ -end \n15082 ▸ ▸ ▸ ▸ ▸ ▸ ▸ ▸ ▸ ▸ ▸ -print_symbols
15083 ▸ ▸ ▸ ▸ ▸ ▸ ▸ ▸ ▸ ▸ ▸ ▸ #pdflatex graphs_v8_r3_poset_lvl_12.tex
15084 ▸ ▸ ▸ ▸ ▸ ▸ ▸ ▸ ▸ ▸ ▸ ▸ ▸ #open graphs_v8_r3_poset_lvl_12.pdf
15085
15086
15087
15088
15089
15090
15091 Symmetric_4_inversion_graph_recognize:
15092 ▸ $(ORBITER) -v 10 \\
15093 ▸ ▸ -define G -permutation_group -symmetric_group 4 -end \n15094 ▸ ▸ -with G -do \n15095 ▸ ▸ -group_theoretic_activity \n15096 ▸ ▸ ▸ -export_inversion_graphs "Symmetric4_inversion_graphs.csv" \n15097 ▸ ▸ ▸ -end
15098 ▸ ▸ ▸ ▸ $(ORBITER) -v 2 \\
15099 ▸ ▸ ▸ ▸ -define GC -graph_classification \n15100 ▸ ▸ ▸ ▸ ▸ -n 4 \n15101 ▸ ▸ ▸ ▸ ▸ ▸ -poset_classification_control \n15102 ▸ ▸ ▸ ▸ ▸ ▸ ▸ -problem_label graphs_v4 -depth 6 -draw_poset \n15103 ▸ ▸ ▸ ▸ ▸ ▸ ▸ ▸ -draw_options -radius 250 -embedded -end \n
833
SECTION GRAPH THEORY: CLIQUE FINDING:

Section 13.4: Graph Theory: Clique finding

Symmetric5 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 

-end 

-print_symbols

\#pdflatex graphs_v5 poset.tex

\#open graphs_v5 poset.pdf

# Section 13.4: Graph Theory: Clique finding
small_graph_cliques: graph_v5_e7.colored_graph
$($ORBITER) -v 2 \
define G -graph -load graph_v5_e7.colored_graph -end \
with G -do \
-graph_theoretic_activity \
-find_cliques -target_size 3 \
-end

# nb_sol = 3

BLT_q13_graph_5_0_cliques_bw:
$($ORBITER) -v 2 \
define G -graph -load BLT_q13_graph_5_0.bin -end \
with G -do \
-graph_theoretic_activity \
-find_cliques -target_size 9 \
-end

# all_cliques_black_and_white

BLT_q13_graph_5_0_cliques_rainbow:
$($ORBITER) -v 2 \
define G -graph -load BLT_q13_graph_5_0.bin -end \
with G -do \
-graph_theoretic_activity \
-find_cliques -rainbow -target_size 9 \
-end

# all_rainbow_cliques

small_graph_cliques_Sajeeb:
$($ORBITER) -v 2 \
define G -graph -load graph_v5_e7.colored_graph -end \
with G -do \
-graph_theoretic_activity \
-find_cliques -Sajeeb -target_size 3 \
-end

# nb_sol = 3

Paley_13_aut:
$($ORBITER) -v 2 \

835
define Gamma -graph -Paley 13 -end \
with Gamma -do \
-graph_theoretic_activity \
\[ \text{define Group} \text{-permutation_group} \] \
\[ \text{bsgs Paley 13 "Paley\_13" 13 78 "0,1" 3 gens} -end \] \
\[ \text{define Gamma -graph -Paley 13} \] \
\[ \text{define Orb -orbits -group G} \] \
\[ \text{on subsets 5 Control} \] \
\[ \text{end} \] \
\[ \text{define gens -vector -file Paley\_13\_gens.csv -end} \] \
\[ \text{define G -permutation_group} \] \
\[ \text{bsgs Paley 13 "Paley\_13" 13 78 "0,1" 3 gens} -end \] \
\[ \text{define Gamma -graph -Paley 13} \] \
\[ \text{define Orb -orbits -group G} \] \
\[ \text{on subsets 5 Control} \] \
\[ \text{end} \] \
\[ \text{define gens -vector -file Paley\_13\_gens.csv -end} \] \
\[ \text{define G -permutation_group} \] \
\[ \text{bsgs Paley 13 "Paley\_13" 13 78 "0,1" 3 gens} -end \] \
\[ \text{define Gamma -graph -Paley 13} \] \
\[ \text{define Orb -orbits -group G} \] \
\[ \text{on subsets 5 Control} \] \
\[ \text{end} \] \
\[ \text{define gens -vector -file Paley\_13\_gens.csv -end} \] \
\[ \text{define G -permutation_group} \] \
\[ \text{bsgs Paley 13 "Paley\_13" 13 78 "0,1" 3 gens} -end \] \
\[ \text{define Gamma -graph -Paley 13} \] \
\[ \text{define Orb -orbits -group G} \] \
\[ \text{on subsets 5 Control} \] \
\[ \text{end} \] \
\[ \text{define gens -vector -file Paley\_13\_gens.csv -end} \] \
\[ \text{define G -permutation_group} \] \
\[ \text{bsgs Paley 13 "Paley\_13" 13 78 "0,1" 3 gens} -end \] \
\[ \text{define Gamma -graph -Paley 13} \] \
\[ \text{define Orb -orbits -group G} \] \
\[ \text{on subsets 5 Control} \] \
\[ \text{end} \] \
\[ \text{define gens -vector -file Paley\_13\_gens.csv -end} \] \
\[ \text{define G -permutation_group} \] \
\[ \text{bsgs Paley 13 "Paley\_13" 13 78 "0,1" 3 gens} -end \] \
\[ \text{define Gamma -graph -Paley 13} \] \
\[ \text{define Orb -orbits -group G} \] \
\[ \text{on subsets 5 Control} \] \
\[ \text{end} \] 

# writes Paley\_13\_group.makefile

#User time: 0 of a second, dt=0 tps = 100

#nb_calls_to_densenauty=1

Paley\_13:

# User time: 0.01 of a second, dt=1 tps = 100
PGO_5_2_cliques: 0_5_2_incidence_matrix.csv

$\text{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

HJ_d2_c5:

$\text{ORBITER} -v 6 \$

-define G -graph 

-load_csv_no_border 

-halljanko315.csv 

-distance_2 

-end 

-with G -do 

-graph_theoretic_activity 

-find_cliques -target_size 5 -end 

-end

HJ64_cliques5:

$\text{ORBITER} -v 6 \$

-define Gamma -graph 

-load \n
-Group_Perm315_Orbital_3.colored_graph 

-end 

-with Gamma -do 

-graph_theoretic_activity 

-find_cliques -target_size 5 -end 

-end

Group_Perm315_Orbital_3.sol.csv

#graph_theoretic_activity::perform_activity Gr->label=halljanko315 nb_sol = 26208

#Group_Perm315_Orbital_3

HJ64_cliques5:

Group_Perm315_Orbital_3

sol.csv

Group_Perm315_Orbital_3_sol.csv
HJ64_cliques5_classify:

$\text{(ORBITER)} -v 6 $

-define Control -poset_classification_control \
-define -W \
-problem_label HJ64_cliques \
-clique_test Gamma \
-depth 5 \
-end \

-define Gamma -graph \
-end \

-define gens -vector \
-file halljanko315_gens.csv \
-end \

-define G -permutation_group \
-bsgs halljanko315 "File\haljanko315" \
-315 1209600 "0,1,42,95" 6 gens -end \

-define Orb -orbits -group G \
-on_subsets 5 Control \
-end

#HJ64_cliques_reps_lvl5.csv

# 1 orbit
#ROW,REP,AGO,OL
#0,"0,8,31,110,283",1200,1008
#END

###############################################################################
# Chapter 14 - Combinatorial Objects
###############################################################################

SECTION COMBINATORIAL OBJECTS:
Hirschfeld_q4_from_set:

```
define H -set -here
$(HIRSCHFELD_SURFACE_Q4_SET_OF_POINTS)
end
```

```
define C -combinatorial_objects
set_of_points H
end
```

```
hyperoval.16_create:
```

```
EC_read: elliptic_curve_b1_c3_q11.txt
```

```
PG_3.5_assume_31_read:
```

```
LS_AG_2.3_read:
```

```

```

```

```

839
geo_7_3_read:
$ (ORBITER) \text{-v 10} \\
\text{-draw_incidence_structure_description} \\
\text{-width 60 \text{-with_10 6 -end}} \\
\text{-define C \text{-combinatorial_objects}} \\
\text{-file_of_incidence_geometries} \\
\text{-7.3.inc 7 7 21} \\
\text{-end}

Desargues.path_lex.least_read:
\text{echo $(DESARGUES_PATH_LEX_LEAST) >Desargues.path_lex.least.inc}
$ (ORBITER) \text{-v 10} \\
\text{-draw_incidence_structure_description} \\
\text{-width 60 \text{-with_10 6 -end}} \\
\text{-define C \text{-combinatorial_objects}} \\
\text{-file_of_incidence_geometries_by_row_ranks} \\
\text{Desargues.path_lex.least.inc 10 10 3} \\
\text{-end}

passch_read:
$(ORBITER) \text{-v 10} \\
\text{-define C \text{-combinatorial_objects}} \\
\text{-file_of_incidence_geometries} \\
\text{pasch.inc 6 4 12} \\
\text{-end}

geo_pasch_given:
$(ORBITER) \text{-v 10} \\
\text{-define C \text{-combinatorial_objects}} \\
\text{-incidence_geometry} \\
\text{"0,1,4,6,8,11,13,14,17,19,22,23"} \\
\text{6 4 12} \\
\text{-end}
# Chapter 15 - Canonical Forms with Nauty

## Section 15.1: Overview of Canonical Forms

```latex
SECTION OVERVIEW_CANONICAL_FORMS:
```

## Section 15.2: Objects in projective Space

```latex
SECTION_OBJECTS_IN_PROJECTIVE_SPACE:
```

```latex
EC_canon: elliptic_curve_b1_c3_q11.txt
```

```latex
\$(ORBITER) -v 3 \ 
\$-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 -n 2 -field F -v 0 -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_incidence_matrices \ 
\$-export_group_GAP \ 
\$-end \ 
\$-end \ 
pdflatex EC_classification.tex
```

841
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 \\
-define C -combinatorial_objects \\
-file_of_points Hirschfeld_surface_q4.txt \\
-end \\
-define F -finite_field -q 4 -end \\
-define P -projective_space -n 3 -field F -v 0 -end \\
-with C -do \\
-combinatorial_object_activity \\
-canonical_form_PG P \\
-classification_prefix Hirschfeld_surface_q4 \\
-save_ago \\
-max_TDO_depth 10 \\
-end \\
-report \\
-prefix Hirschfeld_surface_q4 \\
-export_flag_orbits \\
-show_TDO \\
-show_TDA \\
dont_show_incidence_matrices \\
-export_group_GAP \\
-end \\
-report \\
pdflatex Hirschfeld_surface_q4_classification.tex

open Hirschfeld_surface_q4_classification.pdf

# group order is 51840

HIRSCHFELD_STAB_GENERATORS="1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1, 1,0,0,0,0,2,0,0,0,0,2,0,0,0,0,1,0, 0,2,0,0,0,1,0, 1,0,0,0,3,0,0,0,0,3,0,1,0,0,1,0,1,0,0,0,1,0,1,1,0,0,1,0 
,1,1, 1,0,0,0,0,0,1,0,1,0,0,0,0,0,1,0, 0,1,0,0,1,1,0,0,0,0,0,0,1,0,0,1,0,0,0"
-orbiter_path $(ORBITER_PATH) \\
-define G -linear_group -PGGL 4 4 \\
-subgroup_by_generators "Hirschfeld_Stab" \\
-define $G$ -linear_group -PGGL 4 4 \\
-subgroup_by_generators "Hirschfeld_Stab" \\
-end \\
-define Gsp -modified_group -from G \\
-create_special_subgroup \\
-end \\
-with Gsp -do \\
-group_theoretic_activity \\
-report \\
-end \\
-define Orb -orbits -group Gsp \\
-on_points \\
-end \\
-Hirschfeld_q4.set_c: \\
-define H -set -here \\
-define $H$ -set -here \\
-end \\
-define C -combinatorial_objects \\
-set_of_points H \\
-end \\
-define F -finite_field -q 4 -end \\
-define $F$ -finite_field -q 4 -end \\
-define P -projective_space -n 3 -field F -v 0 -end \\
-with C -do \\
-combinatorial_object_activity \\
-canonical_form_PG P \\
-classification_prefix Hirschfeld_surface_q4 \\
-save_ago \\
-end \\
pdflatex Hirschfeld_surface_q4.classification.tex \\
open Hirschfeld_surface_q4.classification.pdf \\
Dickson_sets_stabilizer: \\
-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"
\begin{align*}
\text{-set of points } & "0,1,2,3,5,6,7" \\
\text{-set of points } & "0,1,2,3,5,6,9" \\
\text{-set of points } & "0,1,2,3,5,6,10" \\
\text{-set of points } & "0,1,2,3,5,6,4" \\
\text{-set of points } & "0,1,2,3,8,11,13" \\
\text{-set of points } & "3,6,9,7,10,12,8,11,13,14,4" \\
\text{-set of points } & "3,5,6,9,7,10,12,11,13,14,4" \\
\text{-set of points } & "0,1,2,3,5,6,9,7,10,12,4" \\
\text{-define } F & -finite \text{ field } -q 2 -end \\
\text{-define } P & -projective \text{ space } -n 3 -field F -v 0 -end \\
\text{-with } C & -do \\
\text{-combinatorial_object_activity} & \\
\text{-canonical_form} & PG P \\
\text{-classification_prefix} & Dickson_sets \\
\text{-save}_\text{ago} & \\
\text{-end} & \\
\text{-report} & \\
\text{-end} & \\
\text{pdflatex Dickson_sets.classification.tex} \\
\text{open Dickson_sets.classification.pdf} \\
\text{Endrass_7c: Endrass_F7.txt} \\
\text{$(ORB\text{ITER}) -v 2$} \\
\text{-define } C & -combinatorial_objects \\
\text{-file_of_points} & Endrass_F7.txt \\
\text{-define } F & -finite \text{ field } -q 7 -end \\
\text{-define } P & -projective \text{ space } -n 3 -field F -v 0 -end \\
\text{-with } C & -do \\
\text{-combinatorial_object_activity} & \\
\text{-canonical_form} & PG P \\
\text{-classification_prefix} & Endrass_F7 \\
\text{-save}_\text{ago} & \\
\text{-end} & \\
\text{-report} & \\
\text{-end} & \\
\text{pdflatex Endrass_F7.classification.tex} \\
\text{open Endrass_F7.classification.pdf} \\
\text{# group order is 32} \\
\text{hyperoval 16 canonical form:}
\end{align*}
\$\texttt{(ORBITER) -v 2 \ 15616}\
\$\texttt{define C -combinatorial_objects \ 15617}\
\$\texttt{-set_of_points $(HYPEROVAL_{16,16320}) \ 15618}\
\$\texttt{-set_of_points $(HYPEROVAL_{16,144}) \ 15619}\
\$\texttt{-end \ 15620}\
\$\texttt{-define F -finite_field -q 16 -end \ 15621}\
\$\texttt{-define P -projective_space -n 2 -field F -v 0 -end \ 15622}\
\$\texttt{-with C -do \ 15623}\
\$\texttt{-combinatorial_object_activity \ 15624}\
\$\texttt{-canonical_form_PG P \ 15625}\
\$\texttt{-classification_prefix hyperoval_q16 \ 15626}\
\$\texttt{-label hyperoval_q16 \ 15627}\
\$\texttt{-save \ 15628}\
\$\texttt{-save_transversal \ 15629}\
\$\texttt{-max_TDO_depth 10 \ 15630}\
\$\texttt{-end \ 15631}\
\$\texttt{-report \ 15632}\
\$\texttt{-prefix hyperoval_q16 \ 15633}\
\$\texttt{-export_flag_orbits \ 15634}\
\$\texttt{-show_TDO \ 15635}\
\$\texttt{-show_TDA \ 15636}\
\$\texttt{-dont_show_incidence_matrices \ 15637}\
\$\texttt{-export_group_GAP \ 15638}\
\$\texttt{-end \ 15639}\
\$\texttt{-export \ 15640}\
\$\texttt{pdflatex hyperoval_q16_classification.tex \ 15641}\
\$\texttt{open hyperoval_q16_classification.pdf \ 15642}\
\$\texttt{(ORBITER) -v 2 -draw_matrix \ 15643}\
\$\texttt{-input_csv_file hyperoval_q16_object0_TDA_flag_orbits.csv \ 15644}\
\$\texttt{-secondary_input_csv_file hyperoval_q16_object0_TDA.csv \ 15645}\
\$\texttt{-box_width 4 -bit_depth 24 \ 15646}\
\$\texttt{-end \ 15647}\
\$\texttt{open hyperoval_q16_object0_TDA_flag_orbits_draw.bmp \ 15648}\
\$\texttt{(ORBITER) -v 2 -draw_matrix \ 15649}\
\$\texttt{-input_csv_file hyperoval_q16_object1_TDA_flag_orbits.csv \ 15650}\
\$\texttt{-secondary_input_csv_file hyperoval_q16_object1_TDA.csv \ 15651}\
\$\texttt{-box_width 4 -bit_depth 24 \ 15652}\
\$\texttt{-end \ 15653}\
\$\texttt{open hyperoval_q16_object1_TDA_flag_orbits_draw.bmp \ 15654}\
\$\texttt{cubic_curves_PG_2_8.canon: \ 15655}\
\$\texttt{(ORBITER) -v 6 \ 15656}\
\$\texttt{-define C -combinatorial_objects \ 15657}\
\$\texttt{845}
-set_of_points "2,3,28,46,51,61,40,71" \\
-define F -finite_field -q 8 -end \\
-define P -projective_space -n 2 -field F -v 0 -end \\
-with C -do \\
-combinatorial_object_activity \\
-canonical_form_PG P \\
-classification_prefix cc_8 \\
-save_ago \\
-max_TDO_depth 10 \\
-end \\
-report \\
-end \\
pdflatex cc_8.classification.tex \\
open cc_8.classification.pdf \\

F_alpha_beta_gamma_delta_classify_q7.nauty: F_alpha_beta_gamma_delta_q7_points.txt \\
$(ORBITER) -v 6 \\
-define C -combinatorial_objects \\
-file_of_points \\
F_alpha_beta_gamma_delta_q7_points.txt \\
-end \\
-define F -finite_field -q 7 -end \\
-define P -projective_space -n 3 -field F -v 0 -end \\
-with C -do \\
-combinatorial_object_activity \\
-canonical_form_PG P \\
-classification_prefix surface_15_lines_q7 \\
-save_ago \\
-save_transversal \\
-end \\
end \\
pdflatex surface_15_lines_q7.classification.tex \\
open surface_15_lines_q7.classification.pdf \\
#4:38 \\
#User time: 4:12 on Mac \\
6 orbits \\
ovoid_q8.canon: ovoid_q8.txt \\
$(ORBITER) -v 6 \\

846
-define C -combinatorial_objects \\n-define F -finite_field -q 8 -end \\n-define P -projective_space -n 3 -field F -v 0 -end \\n-with C -do \\n-combinatorial_object_activity \\n-canonical_form_PG P \\n-classification_prefix 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_GAP \\n-end \\n-end 

pdflatex ovoid_classification.tex
open ovoid_classification.pdf

#-report \\n#-prefix ovoid \\n#-export_flag_orbits \\n#-show_TDO \\n#-show_TDA \\n#-dont_show_incidence_matrices \\n#-export_group_GAP \\n#-end \\n-end 

ovoid_ST_q8.canon: ovoid_ST_q8.txt
$(ORBITER) -v 6 \\n-define C -combinatorial_objects \\n-define F -finite_field -q 8 -end \\n-define P -projective_space -n 3 -field F -v 0 -end \\n-with C -do \\n-combinatorial_object_activity \\n-canonical_form_PG P \\n-classification_prefix ovoid_ST \\n-label ovoid_ST
\texttt{pdflatex ovoid_ST\_classification.tex}
\texttt{open ovoid_ST\_classification.pdf}

\texttt{levenshtein 15770 # group order 87360 = 3 * 29120}
\texttt{levenshtein SUZUKI\_8\_GENERATORS="}
\texttt{levenshtein 1,0,0,0,1,0,0,0,0,1,0,0,0,1,1,}
\texttt{levenshtein 1,0,0,0,6,0,0,0,0,2,0,0,0,0,3,0,}
\texttt{levenshtein 1,0,0,0,1,1,1,0,0,0,1,0,1,0,1,0,}
\texttt{levenshtein 1,0,0,0,3,6,2,2,5,0,2,0,3,0,6,3,2,}
\texttt{levenshtein 0,1,0,0,1,0,0,0,0,0,1,0,0,1,0,2"}
\texttt{levenshtein Suzuki\_8:}
\texttt{levenshtein $(ORBITER) -v 6 \ 
\texttt{levenshtein define F -finite_field -q 8 -end \ 
\texttt{levenshtein define gens -vector -field F \ 
\texttt{levenshtein -compact $(SUZUKI\_8\_GENERATORS) -end \ 
\texttt{levenshtein define G -linear_group -PGGL 4 8 \ 
\texttt{levenshtein subgroup_by_generators "Sz8" "87360" 5 gens \ 
\texttt{levenshtein -end \ 
\texttt{levenshtein -with G -do \ 
\texttt{levenshtein -group_theoretic_activity \ 
\texttt{levenshtein -report \ 
\texttt{levenshtein -end \ 
\texttt{pdflatex PGGL\_4\_8\_Subgroup\_Sz8\_87360\_report.tex}
\texttt{open PGGL\_4\_8\_Subgroup\_Sz8\_87360\_report.pdf}

\begin{verbatim}
levenshtein 15794 # Section 15.3: Incidence Geometries
levenshtein 15796
levenshtein 15797
levenshtein 15798 SECTION\_INCIDENCE\_GEOMETRIES:
levenshtein 15799
levenshtein 15800
levenshtein 15801
levenshtein 15802
\end{verbatim}
geo_7_3.c:

$\texttt{(ORBITER) \ -v \ 10 \ \}
\texttt{-draw.incidence.structure.description \ }
\texttt{-width \ 60 \ -with_10 \ 6 \ -end \ }
\texttt{-define \ C \ -combinatorial.objects \ }
\texttt{-file.of.incidence.geometries \ 7_3.inc \ 7 \ 7 \ 21 \ }
\texttt{-end \ }
\texttt{-with \ C \ -do \ }
\texttt{-combinatorial.object.activity \ }
\texttt{-canonical.form \ }
\texttt{-classification.prefix \ 7_3 \ }
\texttt{-label \ 7_3 \ }
\texttt{-save.ago \ }
\texttt{-save_transversal \ }
\texttt{-end \ }
\texttt{-report \ }
\texttt{-prefix \ 7_3 \ }
\texttt{-export.flag.orbits \ }
\texttt{-show.incidence.matrices \ }
\texttt{-export.group.GAP \ }
\texttt{-end \ }
\texttt{-pdflatex \ 7_3 \ classification.tex \ }
\texttt{-open \ 7_3 \ classification.pdf \ }
$\texttt{(ORBITER) \ -v \ 2 \ -draw.matrix \ }
\texttt{-input.csv.file \ 7_3.object0.TDA.flag.orbits.csv \ }
\texttt{-secondary.input.csv.file \ 7_3.object0.TDA.csv \ }
\texttt{-box.width \ 32 \ -bit.depth \ 24 \ }
\texttt{-end \ }
\texttt{-open \ 7_3.object0.INP.flag.orbits.draw.bmp \ }

geo_10_3.c:

$\texttt{(ORBITER) \ -v \ 10 \ \}
\texttt{-draw.incidence.structure.description \ }
\texttt{-width \ 60 \ -with_10 \ 6 \ -end \ }
\texttt{-define \ C \ -combinatorial.objects \ }
\texttt{-file.of.incidence.geometries \ 10_3.inc \ 10 \ 10 \ 30 \ }
\texttt{-end \ }
\texttt{-with \ C \ -do \ }
- combinatorial_object_activity \
- canonical_form \
- classification_prefix 10_3 \
- label 10_3 \
- save Ago \
- save_transversal \
- report \
- prefix 10_3 \
- export_flag_orbits \
- show_incidence_matrices \
- export_group_GAP \
- end \
- end 

pdflatex 10_3_classification.tex
open 10_3_classification.pdf
$(ORBITER) -v 2 -draw_matrix 
$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 
- end
- 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
- end

geo_10_3_c_lex_least:
$ORBITER) -v 10 
- draw_incidence_structure_description 
- width 60 -with_10 6 -end 
- 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 
- end
- define C -combinatorial_objects 
- file_of_incidence_geometries 10_3.inc 10 10 30 
- end 
- with C -do 
- combinatorial_object_activity 
- canonical_form 

850
-classification_prefix 10.3 \n-label 10.3 \n-save_ago \n-save_transversal \n-end \n-report \n-prefix 10.3 \n-export_flag_orbits \n-show_incidence_matrices \n-export_group_GAP \n-show_TDO \n-show_TDA \n-lex_least Geo \n-end \n-pdflatex 10.3_classification.tex
-open 10.3_classification.pdf
-(ORBITER) -v 2 -draw_matrix
-input_csv_file 10.3_object7.TDA_flag_orbits.csv
-secondary_input_csv_file 10.3_object7.TDA.csv
-box_width 16 -bit_depth 24
-end
-(ORBITER) -v 2 -draw_matrix
-input_csv_file 10.3_object7.INP_flag_orbits.csv
-secondary_input_csv_file 10.3_object7.INP.csv
-box_width 16 -bit_depth 24
-end
#10.3_object7.TDA_flag_orbits.csv

geo.14.3.c:
$(ORBITER) -v 2
-draw_incidence_structure_description
-width 60 -with_10 6 -end
-define Test_lines -set -loop 4 15 1 -end
-define C -combinatorial_objects
-file_of_incidence_geometries 14.3.inc 14 14 42
-end
-with C -do
-combinatorial_object_activity
-canonical_form
-classification_prefix 14.3
-label 14.3
15944 \> \> \> \> -save_ago \\
15945 \> \> \> \> -save_transversal \\
15946 \> \> \> -end \\
15947 \> \> -end \\
15948 \\
15949 \\
15950 \#> \> \> -report \\
15951 \#> \> \> -prefix 14_3 \\
15952 \#> \> \> -export_flag_orbits \\
15953 \#> \> \> -show_incidence_matrices \\
15954 \#> \> \> -export_group_GAP \\
15955 \#> \> \> -end \\
15956 \\
15957 \\
15958 geo_15_3_c:
15959 \> $(ORBITER) -v 2 \\
15960 \> -draw_incidence_structure_description \\
15961 \> \> \> -width 50 -with_10 5 -end \\
15962 \> \> \> -define C -combinatorial_objects \\
15963 \> \> \> -file_of_incidence_geometries 15_3.inc 15 15 45 \\
15964 \> \> \> -end \\
15965 \> \> \> -with C -do \\
15966 \> \> \> \> -combinatorial_object_activity \\
15967 \> \> \> \> -canonical_form \\
15968 \> \> \> \> -classification_prefix 10_3 \\
15969 \> \> \> \> -label 10_3 \\
15970 \> \> \> \> -save_ago \\
15971 \> \> -end \\
15972 \> pdflatex 15_3_classification.tex \\
15973 \> open 15_3_classification.pdf \\
15974 \\
15975 TFC_24_3_c:
15976 \> echo $(FILE_24_3_TFC_INC) >24_3_TFC.inc \\
15977 \> $(ORBITER) -v 6 \\
15978 \> \> -define C -combinatorial_objects \\
15979 \> \> \> -file_of_incidence_geometries 24_3_TFC.inc 24 24 72 \\
15980 \> \> \> -end \\
15981 \> \> \> -with C -do \\
15982 \> \> \> \> -combinatorial_object_activity \\
15983 \> \> \> \> -canonical_form \\
15984 \> \> \> \> -classification_prefix 24_3.TFC \\
15985 \> \> \> \> -label 24_3_TFC \\
15986 \> \> \> \> -save_ago \\
15987 \> \> \> -end \\
15988 \> \> \> -report \\
15989 \> \> \> \> -prefix 24_3.TFC \\
15990 \> \> \> \> -export_flag_orbits \\

852
15991 ▷ ▷ ▷ ▷ -show_TDO \\
15992 ▷ ▷ ▷ ▷ -show_TDA \\
15993 ▷ ▷ ▷ ▷ -show_incidence_matrices \\
15994 ▷ ▷ ▷ -end \\
15995 ▷ -end
15996 ▷ pdflatex 24_3_TFC_classification.tex
15997 ▷ open 24_3_TFC_classification.pdf
15998 ▷ $(ORBITER) -v 2 -draw_matrix \\
15999 ▷ ▷ -input_csv_file 24_3_TFC_object2_TDA_flag_orbits.csv \\
16000 ▷ ▷ -secondary_input_csv_file 24_3_TFC_object2_TDA.csv \\
16001 ▷ ▷ -box_width 40 -bit_depth 24 \\
16002 ▷ ▷ -end
16003 ▷ open 24_3_TFC_object2_TDA_flag_orbits_draw.bmp
16004
16005
16006 geo_40_4_g4.c:
16007 ▷ $(ORBITER) -v 2 \\
16008 ▷ ▷ -draw_incidence_structure_description \\
16009 ▷ ▷ ▷ -width 50 -with_10 5 -end \\
16010 ▷ ▷ -define C -combinatorial_objects \\
16011 ▷ ▷ ▷ -file_of_incidence_geometries 40_4_g4.inc 40 40 160 \\
16012 ▷ ▷ ▷ -end \\
16013 ▷ ▷ -with C -do \\
16014 ▷ ▷ -combinatorial_object_activity \\
16015 ▷ ▷ ▷ -canonical_form \\
16016 ▷ ▷ ▷ ▷ -classification_prefix 40_4_g4 \\
16017 ▷ ▷ ▷ ▷ -label 40_4_g4 \\
16018 ▷ ▷ ▷ -save_ago \\
16019 ▷ ▷ ▷ -end \\
16020 ▷ ▷ -report \\
16021 ▷ ▷ ▷ -prefix 40_4_g4 \\
16022 ▷ ▷ ▷ -export_flag_orbits \\
16023 ▷ ▷ ▷ -show_TDO \\
16024 ▷ ▷ ▷ -show_TDA \\
16025 ▷ ▷ ▷ -show_incidence_matrices \\
16026 ▷ ▷ ▷ -end \\
16027 ▷ ▷ -end
16028 ▷ pdflatex 40_4_g4_classification.tex
16029 ▷ open 40_4_g4_classification.pdf
16030
16031 geo_17_3_g4.c:
16032 ▷ $(ORBITER) -v 2 \\
16033 ▷ ▷ -draw_incidence_structure_description \\
16034 ▷ ▷ ▷ -width 50 -with_10 5 -end \\
16035 ▷ ▷ -define C -combinatorial_objects \\
16036 ▷ ▷ ▷ -file_of_incidence_geometries 17_3_g4.inc 17 17 51 \\
16037 ▷ ▷ ▷ -end \\

853
-with C -do \
-combinatorial_object_activity \
-cannonical_form \
-classification_prefix 17_3.g4 \
-label 17_3.g4 \
-save_ago \
-end \
-report \
-prefix 17_3.g4 \
-export_flag_orbits \
-show_TDO \
-show_TDA \
-show_incidence_matrices \
-end \
-end

pdflatex 17_3.g4_classification.tex
open 17_3.g4_classification.pdf

AG_2.3.c: AG_2.3.inc
$(ORBITER) -v 2 \
-define C -combinatorial_objects \
-file_of_incidence_geometries \
AG_2.3.inc 9 12 36 \
-end \
-with C -do \
-combinatorial_object_activity \
-cannonical_form \
-classification_prefix AG_2.3 \
-label AG_2.3 \
-save_ago \
-max_TDO_depth 10 \
-end \
-report \
-prefix AG_2.3 \
-export_flag_orbits \
-show_TDO \
-show_TDA \
-show_incidence_matrices \
-end \
-end
pdflatex AG_2.3_classification.tex
open AG_2.3_classification.pdf

$(ORBITER) -v 2 -draw_matrix \
-input_csv_file AG_2.3_object0_INP_flag_orbits.csv \
-secondary_input_csv_file AG_2.3_object0_INP.csv \
-box_width 40 -bit_depth 24 \

854
16085 \triangleright \triangleright -end
16086 \triangleright open AG_2_3.object0_INP_flag_orbits_draw.bmp
16087
16088
16089
16090
16091 geo_LSQ6.c:
16092 \triangleright $(ORBITER) -v 10 \ 
16093 \triangleright \triangleright -draw_incidence_structure_description \ 
16094 \triangleright \triangleright \triangleright -width 60 -with_10 6 -end \ 
16095 \triangleright \triangleright -define C -combinatorial_objects \ 
16096 \triangleright \triangleright \triangleright -file_of_incidence_geometries \ 
16097 \triangleright \triangleright \triangleright -LSQ6.inc 18 39 126 \ 
16098 \triangleright \triangleright -end \ 
16099 \triangleright \triangleright -with C -do \ 
16100 \triangleright \triangleright -combinatorial_object_activity \ 
16101 \triangleright \triangleright \triangleright -canonical_form \ 
16102 \triangleright \triangleright \triangleright \triangleright -classification_prefix LSQ6 \ 
16103 \triangleright \triangleright \triangleright \triangleright -label LSQ6 \ 
16104 \triangleright \triangleright \triangleright \triangleright -save_agd \ 
16105 \triangleright \triangleright \triangleright \triangleright -save_transversal \ 
16106 \triangleright \triangleright \triangleright -end \ 
16107 \triangleright \triangleright \triangleright -report \ 
16108 \triangleright \triangleright \triangleright \triangleright -prefix LSQ6 \ 
16109 \triangleright \triangleright \triangleright \triangleright -export_flag_orbits \ 
16110 \triangleright \triangleright \triangleright \triangleright -show_incidence_matrices \ 
16111 \triangleright \triangleright \triangleright \triangleright -export_group_GAP \ 
16112 \triangleright \triangleright \triangleright -end \ 
16113 \triangleright \triangleright -end
16114 \triangleright pdflatex LSQ6_classification.tex
16115 \#open LSQ6_classification.pdf
16116 \triangleright $(ORBITER) -v 2 -draw_matrix \ 
16117 \triangleright \triangleright -input_csv_file LSQ6_object0_TDA_flag_orbits.csv \ 
16118 \triangleright \triangleright -secondary_input_csv_file LSQ6_object0_TDA_flag_orbits.csv \ 
16119 \triangleright \triangleright -box_width 32 -bit_depth 24 \ 
16120 \triangleright \triangleright -end
16121 \triangleright $(ORBITER) -v 2 -draw_matrix \ 
16122 \triangleright \triangleright -input_csv_file LSQ6_object0_INP_flag_orbits.csv \ 
16123 \triangleright \triangleright -secondary_input_csv_file LSQ6_object0_INP_flag_orbits.csv \ 
16124 \triangleright \triangleright -box_width 32 -bit_depth 24 \ 
16125 \triangleright \triangleright -end
16126 \triangleright open LSQ6_object0_INP_flag_orbits_draw.bmp
16127
16128
16129
16130
16131

855
quartic_curve_25.0.0.canonical:

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

$\text{\texttt{-define F -finite\_field -q 25 -end \}}$

$\text{\texttt{-define P -projective\_space -n 2 -field F -v 0 -end \}}$

$\text{\texttt{-projective\_space\_activity \}}$

$\text{\texttt{-canonical\_form\_PG \}}$

$\text{\texttt{-input \}}$

$\text{\texttt{-set\_of\_points "10,11,59,63,124,135,136,170,206,257,275,284,285,367,378,393,433,619,641,644" \}}$

$\text{\texttt{-set\_of\_points "9,24,62,67,77,84,87,89,125,130,158,172,197,219,266,271,325,356,391,392,400,429,454,458,470,503,531,553,605,625,627,646" \}}$


$\text{\texttt{-set\_of\_points "2,12,48,65,87,120,189,246,305,323,354,375,434,435,455,482,496,557,586,595" \}}$

$\text{\texttt{-classification\_prefix quartic_25.0.0 \}}$

$\text{\texttt{-report \}}$

$\text{\texttt{-end \}}$

$\text{\texttt{pdflatex quartic_25.0.0.classification.tex \}}$

$\text{\texttt{open quartic_25.0.0.classification.pdf \}}$

geo_16.c:

$\text{\texttt{(ORBITER) -v 10 \}}$

$\text{\texttt{-draw\_incidence\_structure\_description \}}$

$\text{\texttt{-width 60 -with\_10 6 -end \}}$

$\text{\texttt{-define C \_combinatorial\_objects \}}$

$\text{\texttt{-file\_of\_incidence\_geometries geo_16.inc 16 20 80 \}}$

$\text{\texttt{-end \}}$

$\text{\texttt{-with C \_do \}}$

$\text{\texttt{-combinatorial\_object\_activity \}}$

$\text{\texttt{-canonical\_form \}}$
16171 ▶ ▶ ▶ ▶ -classification_prefix 16 \  
16172 ▶ ▶ ▶ ▶ -label 16 \  
16173 ▶ ▶ ▶ ▶ -save \  
16174 ▶ ▶ ▶ ▶ -save_transversal \  
16175 ▶ ▶ ▶ ▶ -end \  
16176 ▶ ▶ ▶ ▶ -report \  
16177 ▶ ▶ ▶ ▶ -prefix 16 \  
16178 ▶ ▶ ▶ ▶ -export_flag_orbits \  
16179 ▶ ▶ ▶ ▶ -show_incidence_matrices \  
16180 ▶ ▶ ▶ ▶ -export_group_GAP \  
16181 ▶ ▶ ▶ ▶ -end \  
16182 ▶ ▶ ▶ ▶ -end \  
16183 ▶ pdflatex 16_classification.tex  
16184 ▶ open 16_classification.pdf  
16185 ▶ $(ORBITER) -v 2 -draw_matrix \  
16186 ▶ ▶ -input_csv_file 16_object0_TDA_flag_orbits.csv \  
16187 ▶ ▶ -secondary_input_csv_file 16_object0_TDA.csv \  
16188 ▶ ▶ -box_width 16 -bit_depth 24 \  
16189 ▶ ▶ -end \  
16190 ▶ $(ORBITER) -v 2 -draw_matrix \  
16191 ▶ ▶ -input_csv_file 16_object0_INP_flag_orbits.csv \  
16192 ▶ ▶ -secondary_input_csv_file 16_object0_INP.csv \  
16193 ▶ ▶ -box_width 16 -bit_depth 24 \  
16194 ▶ ▶ -end \  
16195  
16196  
16197 # Section 15.4: Objects from Design Theory  
16198  
16199  
16200  
16201 SECTION_OBJECTS_FROM_DESIGN_THEORY:  
16202  
16203  
16204  
16205 LS_AG_2_3_solutions_classify:  
16206 ▶ $(ORBITER) -v 2 \  
16207 ▶ ▶ -draw_incidence_structure_description \  
16208 ▶ ▶ ▶ -width 20 -width_10 2 -end \  
16209 ▶ ▶ -define C -combinatorial_objects \  
16210 ▶ ▶ ▶ -file_of_designs \  
16211 ▶ ▶ ▶ solutions.csv 9 84 3 12 \  
16212 ▶ ▶ ▶ -end \  
16213 ▶ ▶ -with C -do \  
16214 ▶ ▶ -combinatorial_object_activity \  
16215 ▶ ▶ ▶ -canonical_form \  
16216 ▶ ▶ ▶ ▶ -save \  
16217 ▶ ▶ ▶ ▶ -save_transversal \  

857
$\text{(ORBITER)} -v 2 -draw_matrix$

$\text{(ORBITER)} -v 2 -draw_matrix$

$\text{(ORBITER)} -v 2 -draw_matrix$

$\text{(ORBITER)} -v 2 -draw_matrix$

$\text{(ORBITER)} -v 2 -draw_matrix$

$\text{(ORBITER)} -v 4$
design_PG_2_3_canonical:

$(ORBITER) -v 3 \n-define F -finite_field -q 3 -end \n-define D -design -field F -family PG_2.q -end \n-with D -do \n-design_activity \n-define D -design -field F -family PG_2.q -end \n-with D -do \n-design_activity \n-end \n-define D -design -field F -family PG_2.q -end \n-with D -do \n-design_activity \n-end \n$(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-define C -combinatorial_objects \n-file_of_incidence_geometries PG_2_3_inc.txt 13 13 52 \n-end \n-define C -combinatorial_objects \n-file_of_incidence_geometries PG_2_3_inc.txt 13 13 52 \n-end \n-prex \n-label PG_2_3 \n-save_agamos \n-save_transversal \n-end \n-report \n-prefix PG_2_3 \n-export_flag_orbits \n-show_incidence_matrices \n-export_group_GAP \n-end \n-pdflatex PG_2_3_classification.tex

$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

wreath_product_designs_n4_k2_c: wreath_product_designs_n4_k2_inc.txt

$ORBITER -v 10 \n-draw_incidence_structure_description \n-width 60 -with_10 6 -end \n-define C -combinatorial_objects \n
wreath_product_designs_n4_k2_inc.txt
8 12 24
-wend
-with C -do
-combinatorial_object_activity
-canonical_form
-classification_prefix wreath_4_2
-label wreath_4_2
-save_agso
-save_transversal
-end
-report
-prefix wreath_4_2
-export_flag_orbits
-show_incidence_matrices
-export_group_GAP
-end
-pdflatex wreath_4_2_classification.tex
-open wreath_4_2_classification.pdf
-wreath_product_designs_n8_k6_c: wreath_product_designs_n8_k6_inc.txt
-$(ORBITER) -v 10
-draw_incidence_structure_description
-width 60 -with_10 6 -end
-define C -combinatorial_objects
-file_of_incidence_geometries
-wreath_product_designs_n8_k6_inc.txt
-16 3920 23520
-end
-with C -do
-combinatorial_object_activity
-canonical_form
-classification_prefix wreath_8_6
-label wreath_8_6
-save_agso
-save_transversal
-end
-report
-prefix wreath_8_6
-export_flag_orbits
-export_group_GAP
-end
-pdflatex wreath_8_6_classification.tex
# Section 15.5: Linear Codes

SECTION_CANONICAL_FORMS_OF_LINEAR_CODES:

code_3_2_aut:

```bash
$(ORBITER) -v 20 \n$define F -finite_field -q 2 -end \n$define genma -vector -field F -format 2 \n$define P -projective_space -n 1 -field F -v 0 -end \n$projective_space_activity \n$canonical_form_of_code \n"3_2" genma -save_ago -label "3_2" \n-classification_prefix "3_2" \n-end \n-pdflatex 3_2_classification.tex
```

open 3_2_classification.pdf

```bash
$(ORBITER) -v 2 -draw_matrix \n-input_csv_file 3_2_object0_TDA_flag_orbits.csv \n-secondary_input_csv_file 3_2_object0_TDA.csv \n-box_width 16 -bit_depth 24 \n-end
```

open 3_2_object0_TDA_flag_orbits_draw.bmp

```

code_6_3_aut:

```bash
$(ORBITER) -v 20 \n#define F -finite_field -q 2 -end \n#define genma -vector -field F -format 3 \n-compact $(CODE_N6_K3_Q2_GENMA) \n-end
```

open 6_3_classification.pdf
```
16406 \> \> -define P -projective_space -n 2 -field F -v 0 -end \\
16407 \> \> -with P -do \\
16408 \> \> -projective_space_activity \\
16409 \> \> \> -canonical_form_of_code \\
16410 \> \> \> \> "6.3" genma -save_ago -label "6.3" \\
16411 \> \> \> \> -classification_prefix "6.3" \\
16412 \> \> \> \> -end \\
16413 \> \> \> -end \\
16414 \> \> pdflatex 6.3_classification.tex 
16415 \> \> open 6.3_classification.pdf 
16416 \> \> $(ORBITER) -v 2 -draw_matrix \\
16417 \> \> \> -input_csv_file 6.3_object0_TDA_flag_orbits.csv \\
16418 \> \> \> -secondary_input_csv_file 6.3_object0_TDA.csv \\
16419 \> \> \> -box_width 16 -bit_depth 24 \\
16420 \> \> \> -end \\
16421 \> \> \> open 6.3_object0_TDA_flag_orbits_draw.bmp 
16422 
16423 \> # group of order 24 
16424 
16425 
16426 RM_3.1_group: 
16427 \> $(ORBITER) -v 2 \\
16428 \> \> -define F -finite_field -q 2 -end \\
16429 \> \> -define genma -vector -field F -format 4 \\
16430 \> \> \> -compact $(CODE_RM_3.1_GENMA) \\
16431 \> \> \> -end \\
16432 \> \> \> -define P -projective_space -n 3 -field F -v 0 -end \\
16433 \> \> \> -with P -do \\
16434 \> \> \> -projective_space_activity \\
16435 \> \> \> \> -canonical_form_of_code \\
16436 \> \> \> \> \> "RM_3.1" genma -save_ago -label "RM_3.1" \\
16437 \> \> \> \> \> -classification_prefix "RM_3.1" \\
16438 \> \> \> \> \> -end \\
16439 \> \> \> \> -end \\
16440 \> \> pdflatex RM_3.1_classification.tex 
16441 \> \> open RM_3.1_classification.pdf 
16442 
16443 \> # group order 1344 
16444 \> #RM_3.1_object0_INP_flag_orbits.csv 
16445 
16446 RM_3.1_group_and_diagram: 
16447 \> $(ORBITER) -v 2 \\
16448 \> \> -define F -finite_field -q 2 -end \\
16449 \> \> -define genma -vector -field F -format 4 \\
16450 \> \> \> -compact $(CODE_RM_3.1_GENMA) \\
16451 \> \> \> -end \\
16452 \> \> -define P -projective_space -n 3 -field F -v 0 -end 

862
\begin{verbatim}
16453 \> \> -with P -do \ 
16454 \> \> -projective_space_activity \ 
16455 \> \> \> -canonical_form_of_code \ 
16456 \> \> \> \> "RM_3_1" genma -save_ago -label "RM_3_1" \ 
16457 \> \> \> \> -classification_prefix "RM_3_1" \ 
16458 \> \> \> -end \ 
16459 \> \> -end \ 
16460 \> pdflatex RM_3_1_classification.tex \ 
16461 \> \> \> open RM_3_1_classification.pdf \ 
16462 \> $(\textsc{Orbiter}) -v 2 -\texttt{draw\_matrix} \ 
16463 \> \> -input_csv_file RM_3_1_object0_INP_flag_orbits.csv \ 
16464 \> \> -secondary_input_csv_file RM_3_1_object0_INP.csv \ 
16465 \> \> -box_width 16 -bit_depth 24 \ 
16466 \> \> -end \ 
16467 \> $(\textsc{Orbiter}) -v 2 -\texttt{draw\_matrix} \ 
16468 \> \> -input_csv_file RM_3_1_object0_TDA_flag_orbits.csv \ 
16469 \> \> -secondary_input_csv_file RM_3_1_object0_TDA.csv \ 
16470 \> \> -box_width 16 -bit_depth 24 \ 
16471 \> \> -end \ 
16472 \> \> open RM_3_1_object0_INP_flag_orbits_draw.bmp \ 
16473 \> \> open RM_3_1_object0_TDA_flag_orbits_draw.bmp \ 
16474 \ 
16475 \ 
16476 \> RM_4_1_group: \ 
16477 \> \> $(\textsc{Orbiter}) -v 2 \ 
16478 \> \> -\texttt{define}\ F -\texttt{finite}\_\texttt{field} -q 2 -\texttt{end} \ 
16479 \> \> -\texttt{define}\ genma -\texttt{vector}\ -\texttt{field}\ F -\texttt{format}\ 5 \ 
16480 \> \> -\texttt{compact} $(\text{CODE_R}\_4\_1\_\text{GENMA}) \ 
16481 \> \> -\texttt{end} \ 
16482 \> \> -\texttt{define}\ P -\texttt{projective}\_\texttt{space} -n 4 -\texttt{field}\ F -v 0 -\texttt{end} \ 
16483 \> \> -\texttt{with}\ P -\texttt{do} \ 
16484 \> \> -\texttt{projective}\_\texttt{space}\_\texttt{activity} \ 
16485 \> \> \> -\texttt{canonical}\_\texttt{form}\_\texttt{of}\_\texttt{code} \ 
16486 \> \> \> \> "RM_4_1" genma -save_ago -label "RM_4_1" \ 
16487 \> \> \> \> -classification_prefix "RM_4_1" \ 
16488 \> \> \> \> -end \ 
16489 \> \> \> -end \ 
16490 \> \> pdflatex RM_4_1_classification.tex \ 
16491 \> \> \> open RM_4_1_classification.pdf \ 
16492 \> $(\textsc{Orbiter}) -v 2 -\texttt{draw\_matrix} \ 
16493 \> \> -input_csv_file RM_4_1_object0_INP_flag_orbits.csv \ 
16494 \> \> -secondary_input_csv_file RM_4_1_object0_INP.csv \ 
16495 \> \> -box_width 16 -bit_depth 24 \ 
16496 \> \> -end \ 
16497 \> \> $(\textsc{Orbiter}) -v 2 -\texttt{draw\_matrix} \ 
16498 \> \> -input_csv_file RM_4_1_object0_TDA_flag_orbits.csv \ 
16499 \> \> -end \ 
\end{verbatim}

863
-secondary_input_csv_file RM_4_1_object0_TDA.csv
-box_width 16 -bit_depth 24
-end
open RM_4_1_object0_INP_flag_orbits_draw.bmp
open RM_4_1_object0_TDA_flag_orbits_draw.bmp

# group order 322560 = 24*30*28*16

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
-with P -do
-projective_space_activity
-cannotonal_form_of_code
"RS_6" genma -save_ago -label "RS_6"
-classification_prefix "RS_6"
-end
-end

GV_n15_k6_d5_group:
$(ORBITER) -v 20
-define F -finite_field -q 2 -end
-define genma -vector -field F -format 6
-compact $(CODE_GV_N15_K6)
-end
-with P -do
-projective_space_activity
-cannotonal_form_of_code
"GV_n15_k6_d5" genma -save_ago -label "GV_n15_k6_d5"
-classification_prefix "GV_n15_k6_d5"
-end
-end

pdflatex GV_n15_k6_d5_classification.tex
open GV_n15_k6_d5_classification.pdf
ago=12
code_n15_k6_d6_a_group:
$(ORBITER) -v 20 \
-define F -finite_field -q 2 -end \
-define genma -vector -field F -format 6 \
-compact $(CODE_15_6_6.A) \
-end \
-define P -projective_space -n 5 -field F -v 0 -end \
-with P -do \
-projective_space_activity \
-cannonical_form_of_code \
"n15_k6_d6_a" genma -save ago -label "n15_k6_d6_a" \
-classification_prefix "n15_k6_d6_a" \
-end \
-end 

code_n15_k6_d6_b_group:
$(ORBITER) -v 20 \
-define F -finite_field -q 2 -end \
-define genma -vector -field F -format 6 \
-compact $(CODE_15_6_6.B) \
-end \
-define P -projective_space -n 5 -field F -v 0 -end \
-with P -do \
-projective_space_activity \
-cannonical_form_of_code \
"n15_k6_d6_b" genma -save ago -label "n15_k6_d6_b" \
-classification_prefix "n15_k6_d6_b" \
-end \
-end 

# Section 15.6: General Codes

SECTION_CANONICAL_FORMS_OF_GENERAL_CODES:

Hamming_graph_7_with_Hamming_code:
$(ORBITER) -v 2 \$

-define G -graph -Hamming 7 2 \$

-define $G$ -graph $\text{Hamming code}$ \$

-define $G$ -graph $\text{Hamming code}$ \$

-graph_theoretic_activity -export_csv -end \$

-graph_theoretic_activity -export_graphviz -end \$

-graph_theoretic_activity -save -end \$

-graph_theoretic_activity -automorphism_group -end

-graph_theoretic_activity -automorphism_group -end

-pdflatex Hamming_7_2_Hamming_code_report.tex

-open Hamming_7_2_Hamming_code_report.pdf

# group of order 2688 = 16 * 168

SECTION_CANONICAL_FORMS_OF_GRAPHS:

Cycle_13_aut:

-inversion_graph:

-inversion_graph "$1,0,2,3" \$

-graph_theoretic_activity -properties \$

-graph_theoretic_activity -automorphism_group \$

-end
Chain_232.aut:

```bash
$ (ORBITER) -v 2 \
  -define P1 -vector -dense 2,3,2 -end \
  -define P2 -vector -dense 2,3,2 -end \
  -define Gamma -graph \
  -chain_graph P1 P2 \
  -end \
  -with Gamma -do \
  -graph_theoretic_activity -automorphism_group \
  -end
```

```
pdflatex chain_graph_report.tex
open chain_graph_report.pdf
```

JK_graph_pp16.1:

```bash
$ (ORBITER) -v 2 \
  -define Gamma -graph -load_dimacs \
  ../JUNTTILA_KASKI/benchmarks/pp/pp16-1 \
  -end \
  -with Gamma -do \
  -graph_theoretic_activity -save -end \
  -with Gamma -do \
  -graph_theoretic_activity -automorphism_group -end \
```

```
# go=34217164800
# nauty_interface_with_group::create_automorphism_group_of_graph_with_partition_and_labeling: nb_backtrack1 = 6
```

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 \
```

```
# does not finish
```

JK_graph_pp16.9:
JK_graph_grid_3_3:

JK_graph sts_13:

# Written file grid-w-3-3_group.makefile of size 579
# User time: 0 of a second, dt=0 tps = 100
# nb_calls_to_densenaauty=1

make ORBITER
PATH=$(ORBITER
PATH) -f sts-13_group.makefile sts-13

# nauty_interface_with_group::create_automorphism_group_of_graph_with_partition_and_labeling: nb_backtrack1 = 3
# nauty_interface_with_group::create_automorphism_group_of_graph_with_partition_and_labeling: nb_backtrack2 = 24
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

group and orbits:

⊿ $(ORBITER) -v 2 \ 
⊿ ⊿ -define Control -poset_classification_control \ 
⊿ ⊿ ⊿ -W \ 
⊿ ⊿ ⊿ -problem_label HJ_orbits \ 
⊿ ⊿ ⊿ -depth 2 \ 
⊿ ⊿ ⊿ -end \ 
⊿ ⊿ ⊿ -define gens -vector -file \ 
⊿ ⊿ ⊿ -halljanko315 gens.csv -end \ 
⊿ ⊿ ⊿ -define G -permutation_group \ 
⊿ ⊿ ⊿ -bsgs halljanko315 "File\halljanko315" \ 
⊿ ⊿ ⊿ -315 1209600 "0,1,2" 6 gens \ 
⊿ ⊿ ⊿ -end \ 
⊿ ⊿ ⊿ -define Orb -orbits -group G \ 
⊿ ⊿ ⊿ -on_subsets 2 Control \ 
⊿ ⊿ ⊿ -end

#ROW,REP,AGO,OL

#0,"0,1",96,12600
#1,"0,2",48,25200
#2,"0,4",768,1575
#3,"0,8",120,10080
#END

END

orbital graph 3:
# Group Perm315.Orbital_3.colored_graph

# Degree type: $64^{-315}$

PGO_5_2_graph_group: 0_5_2_incidence_matrix.csv

define Inc -vector -file 0_5_2_incidence_matrix.csv -end

define Gamma -graph -collinearity_graph Inc -end

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity

graph_theoretic_activity

with Gamma -do

define Gamma -graph -collinearity_activity
\begin{verbatim}
16823
16824 F_17.edge:
16825 \$\texttt{ORBITER} -v 3 \$
16826 \$\texttt{-define F -finite_field -q 17 -end} \$
16827 \$\texttt{-with F -do -finite_field_activity} \$
16828 \$\texttt{-cheat_sheet_GF -end} \$
16829 \$\texttt{pdflatex GF_17.tex} \$
16830 \$\texttt{open GF_17.pdf} \$
16831
16832
16833
16834
16835
16836
16837
16838
16839
16840 \texttt{Edge.curve_17.nauty:} 
16841 \$\texttt{ORBITER} -v 3 \$
16842 \$\texttt{-define C -combinatorial_objects} \$
16843 \$\texttt{-file_of_points Edge_q17.txt} \$
16844 \$\texttt{-end} \$
16845 \$\texttt{-define F -finite_field -q 17 -end} \$
16846 \$\texttt{-define P -projective_space -n 2 -field F -v 0 -end} \$
16847 \$\texttt{-with C -do} \$
16848 \$\texttt{-combinatorial_object_activity} \$
16849 \$\texttt{-canonical_form_PG P} \$
16850 \$\texttt{-classification_prefix Edge_curve_q17} \$
16851 \$\texttt{-label Edge_curve_q17} \$
16852 \$\texttt{-save_ago} \$
16853 \$\texttt{-save_transversal} \$
16854 \$\texttt{-max.TDO_depth 10} \$
16855 \$\texttt{-end} \$
16856 \$\texttt{-report} \$
16857 \$\texttt{-prefix Edge_curve_q17} \$
16858 \$\texttt{-export_flag_orbits} \$
16859 \$\texttt{-show_TDO} \$
16860 \$\texttt{-show_TDA} \$
16861 \$\texttt{-dont_show_incidence_matrices} \$
16862 \$\texttt{-export_group_GAP} \$
16863 \$\texttt{-end} \$
16864 \$\texttt{-end} \$
16865 \$\texttt{pdflatex Edge_curve_q17_classification.tex} \$
16866 \$\texttt{open Edge_curve_q17_classification.pdf} \$
16867 \$\texttt{ORBITER} -v 2 -draw_matrix \$
16868 \$\texttt{-input_csv_file Edge_curve_q17_object0_TDA_flag_orbits.csv} \$
16869 \$\texttt{-secondary_input_csv_file Edge_curve_q17_object0_TDA.csv} \$
\end{verbatim}
16870 \textgreater \textgreater -box_width 4 -bit_depth 24 \textbackslash
16871 \textgreater \textgreater -end
16872 open Edge_curve_q17_object0_TDA_flag_orbits_draw.bmp
16873
16874 \# 9 backtrack nodes total
16875
16876
16877 \# aut = 24
16878 \# User time: 0.04 of a second, dt=4 tps = 100
16879
16880
16881 \# generators for a group of order 24:
16882 \# 1,0,0,0,13,0,0,0,4, 
16883 \# 1,0,0,0,16,0,16,0, 
16884 \# 0,1,16,2,4,4,15,4,4, 
16885
16886
16887 Edge_curve_17_group:
16888 \textgreater \textgreater $(ORBITER) -v 3 \textbackslash
16889 \textgreater \textgreater -define G -linear_group -PGL 3 17 \textbackslash
16890 \textgreater \textgreater -subgroup_by_generators "Stab_Edge" "24" 3 \textbackslash
16891 \textgreater \textgreater \textgreater "1,0,0,0,13,0,0,0,4" \textbackslash
16892 \textgreater \textgreater \textgreater "1,0,0,0,16,0,16,0" \textbackslash
16893 \textgreater \textgreater \textgreater "0,1,16,2,4,4,15,4,4" \textbackslash
16894 \textgreater \textgreater \textgreater -end \textbackslash
16895 \textgreater \textgreater -with G -do \textbackslash
16896 \textgreater \textgreater -group_theoretic_activities \textbackslash
16897 \textgreater \textgreater -print_elements.tex \textbackslash
16898 \textgreater \textgreater -group_table \textbackslash
16899 \textgreater \textgreater -report \textbackslash
16900 \textgreater \textgreater -end
16901 pdflatex PGL_3_17_Subgroup_Stab_Edge_24_report.tex
16902 open PGL_3_17_Subgroup_Stab_Edge_24_report.pdf
16903
16904
16905
16906
16907
16908
16909 # Chapter 16 - Interfaces
16910
16911 # Section 16.1: Graphical Output
16912
16913
16914
16915
16916
872
SECTION_GRAPHICAL_OUTPUT:

F_7_tables:
$(ORBITER) -v 3 \
  -define F -finite_field -q 7 -end \
  -with F -do -finite_field_activity \
  -cheat_sheet_GF \
  -end

$(ORBITER) -v 2 \
  -draw_matrix \
  -input_csv_file GF_q7_addition_table.csv \
  -box_width 40 \
  -bit_depth 24 \
  -partition 3 7 7 \
  -end

open GF_q7_addition_table_draw.bmp

PG_2_4_cyclic_incma:
$(ORBITER) -v 2 \
  -define F -finite_field -q 4 -end \
  -define P -projective_space -n 2 -field F -v 0 -end \
  -with P -do -projective_space_activity \
  -cheat_sheet_for_decomposition_by_element_PG \
  1 "0,1,0, 0,0,1, 2,1,1, 0" "PG_2_4_singer" \
  -end

$(ORBITER) -v 4 \
  -list_arguments \
  -define R -vector -repeat 1 21 -end \
  -define C -vector -repeat 1 21 -end \
  -draw_matrix \
  -input_csv_file PG_2_4_singer_incma_cyclic.csv \
  -box_width 40 -bit_depth 24 \
  -partition 3 R C \
  -end

open PG_2_4_singer_incma_cyclic_draw.bmp

PGL_4_2_Wedge_4_0_graphical_output:
$(ORBITER) -v 4 \
  -define G -linear_group -PGL 4 2 \
  -wedge_detached \
  -end \
  -with G -do \

873
schreier_tree_graphical_output:

```bash
$(ORBITER) -v 4 \
  -draw_options \
  -yout 500000 \
  -radius 15 -nodes_empty \
  -line_width 0.5 -y_stretch 0.25 \
  -embedded \
  -end \
  -define G -linear_group -PGL 4 2 -end \
  -define Orb -orbits -group G \
  -on_polynomials 3 \
  -end \
  -with Orb -do -orbits_activity \
  -draw_tree 6 \
  -end
```

```bash
poly_orbits_d3_n3_q2_orbit_6_tree.tex
open poly_orbits_d3_n3_q2_orbit_6_tree.pdf
```

Queens_graph:

```bash
$(ORBITER) -v 2 \
  -define G -graph -non_attacking_queens_graph 8 -end \
  -with G -do \
  -graph_theoretic_activity -export.csv -end \
  -with G -do \
  -graph_theoretic_activity -export_graphviz -end \
  -with G -do \
  -graph_theoretic_activity -save -end \
  -with G -do \
  -graph_theoretic_activity -automorphism_group -end \
  -with G -do \
  -graph_theoretic_activity -find_cliques \
  -target_size 8 -output_file 8queens -end \
  -end
```

874
# Section 16.2: The Povray Interface

SECTION
POVRAY:

cube:

$(ORBITER) -v 2 -povray \
-round 0 -nb_frames_default 30 \n-output_mask cube_%d%03d.pov \n-video_options -W 1024 -H 768 \n-global_picture.scale 0.5 \n-default_angle 75 \n-clipping_radius 2.7 \n-end \n
-scene_objects \n-obj_file cube_centered.obj \n-edge "0, 1" \n-edge "0, 2" \n-edge "0, 4" \n-edge "1, 3" \n-edge "1, 5" \n-edge "2, 3" \n-edge "2, 6" \n-edge "3, 7" \n-edge "4, 5" \n-edge "4, 6" \n-edge "5, 7" \n-edge "6, 7" \n-group_of_things_as_interval 0 8 \n-spheres 0 0.3 $(POLISHED_CHROME_WHITE) \n-group_of_things_as_interval 0 6 \n-prisms 1 0.05 $(YELLOW_TRANSPARENT) \n-group_of_things_as_interval 0 12 \n-cylinders 2 0.15 $(COLOR_RED) \n-scene_objects_end \n
-povray_end

-math261_test:

$(ORBITER) -v 2 -povray \n
875
-round 0 -nb_frames_default 30
-output_mask math261_%d_%03d.pov
-video_options -W 1024 -H 768
-global_picture_scale 0.1
-default_angle 75
-clipping_radius 2.7
-end
-scene_objects
-point "0,0,0"
-point "5,0,0"
-point "0,5,0"
-point "0,0,5"
-point "1,2,3"
-point "4,5,6"
-point "5,7,9"
-edge "0,1"
-edge "0,2"
-edge "0,3"
-edge "0,4"
-edge "0,5"
-edge "4,6"
-edge "5,6"
-face "0,4,6,5"
-group_of_things_as_interval 0 7
-spheres 0 0.1 $(POLISHED_CHROME_WHITE)
-group_of_things_as_interval 0 7
-cylinders 1 0.05 $(COLOR_RED)
-prisms 2 0.05 $(YELLOW_TRANSPARENT)
-group_of_things_as_interval 0 1
-scene_objects_end
-povray_end

plane1:
$(ORBITER) -v 2 -povray
-round 0 -nb_frames_default 30
-output_mask plane1_%d_%03d.pov
-video_options -W 1024 -H 768
-global_picture_scale 0.40
-default_angle 75
-clipping_radius 5 -omit_bottom_plane
-camera 0 "0,0,1" "5,5,3" "0,0,0"
17105 ➢ ➢ -rotate_about_z_axis \ 
17106 ➢ ➢ -boundary_box \ 
17107 ➢ ➢ -end \ 
17108 ➢ ➢ -scene_objects \ 
17109 ➢ ➢ ➢ -line_through_two_points_recentered_from_csv_file coordinate_grid.csv \ 
17110 ➢ ➢ ➢ -plane_by_dual_coordinates "0,0,1,0" \ 
17111 ➢ ➢ ➢ -plane_by_dual_coordinates "0,1,0,0" \ 
17112 ➢ ➢ ➢ -plane_by_dual_coordinates "1,0,0,0" \ 
17113 ➢ ➢ ➢ -point "-2.25,0,0" \ 
17114 ➢ ➢ ➢ -point "0,-1.8,0" \ 
17115 ➢ ➢ ➢ -point "0,0,9" \ 
17116 ➢ ➢ ➢ -face "0,1,2,0" \ 
17117 ➢ ➢ ➢ -group_of_things "0" \ 
17118 ➢ ➢ ➢ -group_of_things "1" \ 
17119 ➢ ➢ ➢ -group_of_things "2" \ 
17120 ➢ ➢ ➢ -lines 0 0.15 $(COLOR_RED_SHINY) \ 
17121 ➢ ➢ ➢ -lines 1 0.15 $(COLOR_GREEN_SHINY) \ 
17122 ➢ ➢ ➢ -lines 2 0.15 $(COLOR_BLUE_SHINY) \ 
17123 ➢ ➢ ➢ -group_of_things_as_interval 3 39 \ 
17124 ➢ ➢ ➢ -lines 3 0.05 $(COLOR_BLACK_SHINY) \ 
17125 ➢ ➢ ➢ -group_of_things "0" \ 
17126 ➢ ➢ ➢ -planes 0 $(COLOR_BLUESEE_THROUGH) \ 
17127 ➢ ➢ ➢ -group_of_things "1" \ 
17128 ➢ ➢ ➢ -group_of_things "2" \ 
17129 ➢ ➢ ➢ -group_of_things "0" \ 
17130 ➢ ➢ ➢ -prisms 0 0.05 $(COLOR_YELLOW_THICK) \ 
17131 ➢ ➢ ➢ -scene_objects_end \ 
17132 ➢ ➢ -povray_end
17133 ➢ - rm -rf POV
17134 ➢ mkdir POV
17135 ➢ mv plane1_0_*.pov POV
17136 ➢ mv makefile_animation POV
17137
17138
17139
17140 plane2:
17141 ➢ $(ORBITER) -v 2 -povray \ 
17142 ➢ ➢ -round 0 -nb_frames_default 30 \ 
17143 ➢ ➢ -output_mask plane2_%d_%03d.pov \ 
17144 ➢ ➢ -video_options -W 2560 -H 1920 \ 
17145 ➢ ➢ -global_picture.scale 0.40 \ 
17146 ➢ ➢ -default_angle 75 \ 
17147 ➢ ➢ -clipping_radius 5 -omit_bottom_plane \ 
17148 ➢ ➢ ➢ -camera 0 "0,0,1" "6,6,2" "0,0,0" \ 
17149 ➢ ➢ ➢ -rotate_about_z_axis \ 
17150 ➢ ➢ ➢ -boundary_box \ 
17151 ➢ ➢ -end \ 

877
-scene_objects

- line_through_two_points_recentered_from_csv_file coordinate_grid.csv

- plane_by_dual_coordinates "0,0,1,0"
- plane_by_dual_coordinates "0,1,0,0"
- plane_by_dual_coordinates "1,0,0,0"
- plane_by_dual_coordinates "4,5,-1,9"

- group_of_things 0
- group_of_things 1
- group_of_things 2

- group_of_things_as_interval 3 39

- lines 0 0.15 $(COLOR_RED_SHINY)
- lines 1 0.15 $(COLOR_GREEN_SHINY)
- lines 2 0.15 $(COLOR_BLUE_SHINY)
- lines 3 0.05 $(COLOR_BLACK_SHINY)

- group_of_things 0
- planes 4 $(COLOR_BLUE_SEE_THROUGH)
- group_of_things 3

- scene_objects_end

-povray_end

- rm -rf POV

- mkdir POV

- mv plane2_0.*.pov POV

- mv makefile_animation POV

- analytic_geo_1:

- $(ORBITER) -v 2 -povray

- -round 0 -nb_frames_default 30

- -output_mask analytic_geo_1_%d_%03d.pov

- -video_options -W 2560 -H 1920

- -global_picture_scale 0.80

- -default_angle 75

- -clipping_radius 5 -omit_bottom_plane

- -camera 0 "0,0,1" "6,6,2" "0,0,0"

- -rotate_about_z_axis

- -boundary_box

- -end

- -scene_objects

- line_through_two_points_recentered_from_csv_file coordinate_grid.csv

- plane_by_dual_coordinates "0,0,1,0"

- plane_by_dual_coordinates "0,1,0,0"

- plane_by_dual_coordinates "1,0,0,0"

- group_of_things 0
-group_of_things "1"
-group_of_things "2"
-lines 0 0.05 $(COLOR_RED_SHINY)
-lines 1 0.05 $(COLOR_GREEN_SHINY)
-lines 2 0.05 $(COLOR_BLUE_SHINY)
-lines 3 0.04 $(COLOR_BLACK_SHINY)
-group_of_things "0"
-group_of_things "1"
-group_of_things "2"
-planes 4 $(COLOR_BLUESEE_THROUGH)
-planes 5 $(COLOR_GREENSEE_THROUGH)
-planes 6 $(COLOR_RESEE_THROUGH)
-point "0,0,0"
-point "1,0,0"
-point "1,2,0"
-point "1,2,3"
-edge "84,85"
-edge "85,86"
-edge "86,87"
-edge "84,87"
-group_of_things "84,85,86"
-spheres 7 0.1 $(POLISHED_CHROME_WHITE)
-group_of_things "87"
-spheres 8 0.10 $(COLOR_YELL_SHINY)
-group_of_things "0,1,2"
-cylinders 9 0.075 $(POLISHED_CHROME_WHITE)
-cylinders 10 0.075 $(COLOR_YELL_SHINY)
-scene_objects_end
-povray_end
-rm -rf POV
mkdir POV
mv analytic_geo_1_0.*.pov POV
mv makefile_animation POV

analytic_geo_1_video:
- rm -r FRAMES
-mkdir FRAMES
- rm analytic_geo_1.mp4
$(ORBITER)
-prepare_frames
-i 0 30 PNG/ANALYTIC_GEO_1/analytic_geo_1_0_%03d.png
-output_starts_at 0
-o FRAMES/frame%04d.png
-end
ffmpeg -r 5 -f image2 -i FRAMES/frame%04d.png \
$ (ORBITER) -v 2 -povray \n-round 0 -nb_frames_default 30 \n-output_mask monkey%d.%03d.pov \n-video_options -W 1024 -H 768 \n-global_picture_scale 0.8 \n-default_angle 75 \n-clipping_radius 0.8 \n-camera 0 "0,0,1" "1,1,0.5" "0,0,0" \n-rotate_about_z_axis \n-end \n-scene_objects \n-cubic_lex $(MONKEY_SADDLE_CUBIC) \n-plane_by_dual_coordinates "0,0,1,0" \n-group_of_things "0" \n-group_of_things "0" \n-cubics 0 $(COLOR_GOLD) \n-planes 1 $(COLOR_BLUE) \n-scene_objects_end \n-povray_end \n- rm -rf POV \nmkdir POV \nmv monkey_0.*.pov POV \nmv makefile_animation POV

Eckardt:
$(ORBITER) -v 2 -povray \n-round 0 -nb_frames_default 30 \n-output_mask Eckardt%d.%03d.pov \n-video_options -W 1024 -H 768 \n-global_picture_scale 0.9 \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-Hilbert_Cohn_Vossen_surface \n-group_of_things "0" \n-cubics 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_with_exceptions 12 15 \\
"14,19,23" \\
-lines 1 0.02 $(COLOR_RED_SHINY) \\
-lines 2 0.02 $(COLOR_BLUE_SHINY) \\
-lines 3 0.02 $(COLOR_YELLOW_SHINY) \\
-label 0 "a1" \\
-label 2 "a2" \\
-label 4 "a3" \\
-label 6 "a4" \\
-label 8 "a5" \\
-label 10 "a6" \\
-label 12 "b1" \\
-label 14 "b2" \\
-label 16 "b3" \\
-label 18 "b4" \\
-label 20 "b5" \\
-label 22 "b6" \\
-label 24 "c12" \\
-label 26 "c13" \\
-label 30 "c15" \\
-label 32 "c16" \\
-label 34 "c23" \\
-label 36 "c24" \\
-label 40 "c26" \\
-label 42 "c34" \\
-label 44 "c35" \\
-label 48 "c45" \\
-label 50 "c46" \\
-label 52 "c56" \\
-group_of_things_as_interval 0 6 \\
-texts 4 0.2 0.15 $(COLOR_BLACK_NO_SHADOW) \\
-group_of_things_as_interval 6 6 \\
-texts 5 0.2 0.15 $(COLOR_BLACK_NO_SHADOW) \\
-group_of_things_as_interval 12 12 \\
-texts 6 0.2 0.15 $(COLOR_BLACK_NO_SHADOW)
-scene_objects_end 
-povray_end
- rm -rf POV
mkdir POV
mv Eckardt_0*.pov POV
mv makefile.animation POV

#"-3,2.333,4" * 1.5 = "-4.5,3.5,6"
M := Matrix([[-4.5, 3.5, 6], [1, 1, 1]])
NullSpace(M)
=#0.186080731891197,-0.781539073943026,0.595458342051830

881
Eckardt deform:

```bash
$(ORBITER) -v 2 -povray \\
-round 0 -nb_frames_default 93 \\
-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 \\
-scene_objects \\
-Hilbert_Cohn_Vossen_surface \\
-group_of_things "0" \\
deformation_of_cubic_lex 93 1.107148718 1.570796327 0 \\
(ECKARDT_CUBIC_DEFORM1_LEX) \\
(ECKARDT_CUBIC_DEFORM2_LEX) \\
-group_of_things_as_interval 0 93 \\
-group_is_animated 1 \\
cubics 1 $(SURFACE_COLOR_SEETHROUGH) \\
-scene_objects_end \\
povray_end \\
rm -rf POV \\
mkdir POV \\
mv Eckardt_deform.0.*.pov POV \\
mv makefile_animation POV
```

Eckardt_deform_2:

```bash
$(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" 
-scene_objects 
- Hilbert.Cohn.Vossen_surface 
-group_of_things "0" 
deformation_of_cubic_lex 93 1.107148718 1.570796327 0 
$(ECKARDT_CUBIC_DEFORM1_LEX) 
$(ECKARDT_CUBIC_DEFORM2_LEX) 
--group_of_things_as_interval 0 93 
--group_is_animated 1 
-group_of_things "0" 
cubics 1 $(SURFACE_COLOR_SEETHROUGH) 
cubics 2 $(COLOR_RED) 
cubics 7 $(COLOR_BLUE) 
cubics 3 $(COLOR_BLUE) 
-scene_objects_end 
povray_end 
- rm -rf POV 
mkdir POV 
mv Eckardt_deform_0.*.pov POV 
mv makefile_animation POV 

Clebsch: 
$(ORBITER) -v 2 -povray 
-round 0 -nb_frames_default 30 
-output_mask Clebsch_%d%03d.pov 
-video_options -W 1024 -H 768 
-global_picture_scale 0.9 
default_angle 80 
-clipping_radius 2.4 
camera 0 "1,1,1" "-4.5,3.5,6" "0,0,0" 
-end 
-scene_objects 
-Clebsch_surface 
-group_of_things "0" 
cubics 0 $(SURFACE_COLOR) 
group_of_things_as_interval 0 6 
group_of_things_as_interval 6 6 
group_of_things_as_interval 12 15 
-lines 1 0.02 $(COLOR_RED_SHINY) 
-lines 2 0.02 $(COLOR_BLUE_SHINY) 
-lines 3 0.02 $(COLOR_YELLOW_SHINY) 
group_of_things_as_interval 0 12 
group_of_things_as_interval 0 12 
spheres 4 0.08 $(COLOR_TURQUOISE) 

17432    -scene_objects_end \
17433    -povray_end
17434    rm -rf POV
17435    mkdir POV
17436    mv Clebsch_0_*.pov POV
17437    mv makefile_animation POV
17438
17439
17440
17441  endrass8:
17442    $(ORBITER) -v 2 -povray \
17443    -round 0 -nb_frames_default 30 \
17444    -output_mask endrass_octic_%d_%03d.pov \
17445    -video_options -W 1024 -H 768 \
17446    -global_picture_scale 0.75 \
17447    -default_angle 75 \
17448    -clipping_radius 3.7 \
17449    -no_bottom_plane \
17450    -camera 0 "1,1,1" "6,6,3" "0,0,0" \
17451    -rotate_about_111 \
17452    -end \
17453    -scene_objects \
17454    -line_through_two_points_recentered_from_csv_file \
17455    -coordinate_grid.csv \
17456    -group_of_things "0" \
17457    -group_of_things "1" \
17458    -group_of_things "2" \
17459    -group_of_things_as_interval 3 39 \
17460    -lines 0 0.15 $(COLOR_RED_SHINY) \
17461    -lines 1 0.15 $(COLOR_GREEN_SHINY) \
17462    -lines 2 0.15 $(COLOR_BLUE_SHINY) \
17463    -lines 3 0.05 $(COLOR_BLACK_SHINY) \
17464    -octic_lex_165 $(ENDRASS_OCTIC_LEX_165) \
17465    -plane_by_dual_coordinates "0,0,1,0" \
17466    -group_of_things "0" \
17467    -group_of_things "0" \
17468    -octics 4 $(SURFACE_COLOR_SEETHROUGH) \
17469    -planes 5 "texture{ pigment{ color Blue transmit 0.5 }" \
17470    finish { diffuse 0.9 phong 1}"
17471    -scene_objects_end \
17472    -povray_end
17473    rm -rf POV
17474    mkdir POV
17475    mv endrass_octic_0_*.pov POV
17476    mv makefile_animation POV
17477
17478
884
# Section 16.3: Creating Animations

SECTION_ANIMATIONS:

```bash
# dode:
$(ORBITER) -v 2 -povray -round 0 -nb_frames_default 30 -output_mask dode_%d%03d.pov -video_options -W 1024 -H 768 -global_picture_scale 0.50 -default_angle 45 -clipping_radius 5 -camera 0 "1,1,1" "-2,2,4" "0,0,0" -rotate_about_111 -default_angle 45 -clipping_radius 5 -camera 0 "1,1,1" "-2,2,4" "0,0,0" -rotate_about_111

# scene_objects
```

```bash
$(ORBITER) -i 0 30 DODE/dode_%d%03d.png
```

mv dode.mp4 POV
mv makefile_animation POV

```bash
# dode_video:
$(ORBITER)
```

rm -rf FRAMES
```
```
```
```
ffmpeg -r 5 -f image2 -i FRAMES/frame%04d.png \
-f mp4 -q:v 0 -vcodec mpeg4 dode.mp4

- rm -r FRAMES
- mkdir FRAMES
- rm monkey.mp4
$(ORBITER) \
-prepare_frames \
- i 0 30 monkey_0.%03d.png \
-output_starts_at 0 \
-o FRAMES/frame%04d.png \
- end
ffmpeg -r 5 -f image2 -i FRAMES/frame%04d.png \
-f mp4 -q:v 0 -vcodec mpeg4 monkey.mp4

Eckardt_deform_video:
- rm -r FRAMES
- mkdir FRAMES
- rm Eckardt_deform.mp4
$(ORBITER) \
-prepare_frames \
- i 0 93 Eckardt_deform_0/Eckardt_deform_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 Eckardt_deform.mp4

Eckardt_surface:
$(ORBITER) -v 2 -povray \
-rounded 0 -nb_frames_default 30 \
-output_mask Eckardt.%d.%03d.pov \
-video_options -W 1024 -H 768 \
-global_picture_scale 0.9 \
-default_angle 75 \
-clipping_radius 2.4 \
-camera 0 "1,1,1" "-3,1,3" "0.12,0.12,0.12" \
-end \
-scene_objects \
-cubic_Goursat "6,3,-15" \
-group_of_things "0" \
-cubics 0 $(SURFACE_COLOR_SEETHROUGH) \
-scene_objects_end \

886
```
17573  ▶ ▶ -povray_end
17574  ▶ - rm -rf POV
17575  ▶ mkdir POV
17576  ▶ mv Eckardt_0_*_.pov POV
17577  ▶ mv makefile_animation POV
17578
17579
17580
17581  Kummer_surface:
17582  ▶ $(ORBITER) -v 2 -povray \\
17583  ▶ ▶ -round 0 -nb_frames_default 30 \\
17584  ▶ ▶ -output_mask Kummer_%d_%03d.pov \\
17585  ▶ ▶ -video_options -W 1024 -H 768 \\
17586  ▶ ▶ -global_picture.scale 0.9 \\
17587  ▶ ▶ -default_angle 75 \\
17588  ▶ ▶ -clipping_radius 2.4 \\
17589  ▶ ▶ -camera 0 "1,1,1" "-3,1,3" "0.12,0.12,0.12" \\
17590  ▶ ▶ -end \\
17591  ▶ ▶ -scene_objects \\
17592  ▶ ▶ ▶ -quartic_lex_35 $(KUMMER_QUARTIC_LEX_35) \\
17593  ▶ ▶ ▶ -group_of_things "0" \\
17594  ▶ ▶ ▶ -quartics 0 $(SURFACE_COLOR_SEETHROUGH) \\
17595  ▶ ▶ ▶ -scene_objects_end \\
17596  ▶ ▶ -povray_end
17597  ▶ - rm -rf POV
17598  ▶ mkdir POV
17599  ▶ mv Kummer_0_*_.pov POV
17600  ▶ mv makefile_animation POV
17601
17602
17603  # Maple:
17604  #Kummer := expand((x0^2 + x1^2 + x2^2 + x3^2)^2 - 3*(x0^4 + x1^4 + x2^4 + x3^4))
17605
17606
17607  Kummer_video:
17608  ▶ - rm -r FRAMES
17609  ▶ - mkdir FRAMES
17610  ▶ - rm Kummer.mp4
17611  ▶ $(ORBITER) \\
17612  ▶ ▶ -prepare_frames \\
17613  ▶ ▶ ▶ -i 0 30 KUMMER/Kummer_0_%03d.png \\
17614  ▶ ▶ ▶ -output_starts_at 0 \\
17615  ▶ ▶ ▶ -o FRAMES/frame%04d.png \\
17616  ▶ ▶ -end
17617  ▶ ffmpeg -r 5 -f image2 -i FRAMES/frame%04d.png \\
17618  ▶ ▶ -f mp4 -q:v 0 -vcodec mpeg4 Kummer.mp4
17619
```
Beauville_surface:

# 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.

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:

\begin{verbatim}
Matrix(4, 4, 
[-0.44721360215312733, 1.1708204000530853, 1.1708204000530853, -0.4472135957999158], 
[-1.1708204000530853, 0.4472136021531272, 1.4472136021531272, 0.4472135957999158], 
[4.2360680044124255, -4.2360680044124255, -4.2360680044124255, -4.2360680044124255], 
[1.6180340022062127, -2.6180340022062127, -1.6180340022062127, 0.])
\end{verbatim}

\begin{verbatim}
#-0.44721360215312733, 1.1708204000530853, 1.1708204000530853, -0.4472135957999158

#-1.1708204000530853, 0.4472136021531272, 1.4472136021531272, 0.4472135957999158

#4.2360680044124255, -4.2360680044124255, -4.2360680044124255, -4.2360680044124255, 0.

#1.6180340022062127, -2.6180340022062127, -1.6180340022062127, 0.

To0=-0.44721360215312733
To1=1.1708204000530853
To2=1.1708204000530853
To3=-0.4472135957999158
T0=-1.1708204000530853
T11=0.4472136021531272
T12=1.4472136021531272
T13=0.4472135957999158
T20=4.2360680044124255
T21=-4.2360680044124255
T22=-4.2360680044124255
T23=0.
T30=1.6180340022062127
T31=-2.6180340022062127
T32=-1.6180340022062127
T33=0.

CLEBSCH_CUBICS=\push b push b mult push d push c push m mult add mult \end{verbatim}
push a push d mult push m push b push c mult mult add mult \push a push c push d push m mult add push b mult \push m push d push c push 1 push m mult add mult mult add 
push a mult push b push c mult push d push 1 push m mult add mult add 
push m push a mult push c add mult mult 
store n022 \push m \push c push d push m mult add push b mult \push m push d push c push 1 push m mult add mult mult add 
push a mult push b push c mult push d push 1 push m mult add mult add 
push a push d mult push m push b push c mult mult add mult mult \store n122 

Clebsch_up_create_points:

$\text{(ORBITER)} -v 2 \$

-smooth_curve "Clebsch_map_of_circle_to_defininig_eqn_r2" \0.07 1000 5 0 $(TWO.PI) \-$const a $(CLEBSCH_A) b $(CLEBSCH_B) c $(CLEBSCH_C) d $(CLEBSCH_D) \t00 $(T00) t01 $(T01) t02 $(T02) t03 $(T03) \t10 $(T10) t11 $(T11) t12 $(T12) t13 $(T13) \t20 $(T20) t21 $(T21) t22 $(T22) t23 $(T23) \t30 $(T30) t31 $(T31) t32 $(T32) t33 $(T33) r 2 one 1 m -1 \-const_end 
-var t \c001 c002 c011 c012 \d001 d011 d012 d112 \m002 m012 m022 m122 \n002 n012 n112 n022 n122 \y0 y1 y2 \y001 y002 y011 y012 y022 y112 y122 \x0 x1 x2 x3 \-var_end 
-code \push t cos push r mult store y0 \push t sin push r mult store y1 \push one store y2 \push y0 push y0 push y1 mult mult store y001 \push y0 push y0 push y2 mult mult store y002 \push y0 push y1 push y1 mult mult store y011 \push y0 push y1 push y2 mult mult store y012 \push y0 push y2 push y2 mult mult store y022 \push y1 push y1 push y2 mult mult store y112 \892
17843 ▶ ▶ ▶ push y1 push y2 push y2 mult mult store y122 \ 17844 ▶ ▶ ▶ $(CLEBSCH_CUBICS) \ 17845 ▶ ▶ ▶ ▶ ▶ push c001 push y001 mult \ 17846 ▶ ▶ ▶ ▶ ▶ push c002 push y002 mult add \ 17847 ▶ ▶ ▶ ▶ ▶ push c011 push y011 mult add \ 17848 ▶ ▶ ▶ ▶ ▶ push c012 push y012 mult add \ 17849 ▶ ▶ ▶ ▶ ▶ store x0 \ 17850 ▶ ▶ ▶ ▶ ▶ push d001 push y001 mult \ 17851 ▶ ▶ ▶ ▶ ▶ push d011 push y011 mult add \ 17852 ▶ ▶ ▶ ▶ ▶ push d012 push y012 mult add \ 17853 ▶ ▶ ▶ ▶ ▶ push d112 push y112 mult add \ 17854 ▶ ▶ ▶ ▶ ▶ store x1 \ 17855 ▶ ▶ ▶ ▶ ▶ push m002 push y002 mult \ 17856 ▶ ▶ ▶ ▶ ▶ push m012 push y012 mult add \ 17857 ▶ ▶ ▶ ▶ ▶ push m022 push y022 mult add \ 17858 ▶ ▶ ▶ ▶ ▶ push m122 push y122 mult add \ 17859 ▶ ▶ ▶ ▶ ▶ store x2 \ 17860 ▶ ▶ ▶ ▶ ▶ push n002 push y002 mult \ 17861 ▶ ▶ ▶ ▶ ▶ push n012 push y012 mult add \ 17862 ▶ ▶ ▶ ▶ ▶ push n022 push y022 mult add \ 17863 ▶ ▶ ▶ ▶ ▶ push n112 push y112 mult add \ 17864 ▶ ▶ ▶ ▶ ▶ push n122 push y122 mult add \ 17865 ▶ ▶ ▶ ▶ ▶ store x3 \ 17866 ▶ ▶ ▶ ▶ ▶ push x0 push t00 mult \ 17867 ▶ ▶ ▶ ▶ ▶ push x1 push t10 mult add \ 17868 ▶ ▶ ▶ ▶ ▶ push x2 push t20 mult add \ 17869 ▶ ▶ ▶ ▶ ▶ push x3 push t30 mult add \ 17870 ▶ ▶ ▶ ▶ ▶ return \ 17871 ▶ ▶ ▶ ▶ ▶ push x0 push t01 mult \ 17872 ▶ ▶ ▶ ▶ ▶ push x1 push t11 mult add \ 17873 ▶ ▶ ▶ ▶ ▶ push x2 push t21 mult add \ 17874 ▶ ▶ ▶ ▶ ▶ push x3 push t31 mult add \ 17875 ▶ ▶ ▶ ▶ ▶ return \ 17876 ▶ ▶ ▶ ▶ ▶ push x0 push t02 mult \ 17877 ▶ ▶ ▶ ▶ ▶ push x1 push t12 mult add \ 17878 ▶ ▶ ▶ ▶ ▶ push x2 push t22 mult add \ 17879 ▶ ▶ ▶ ▶ ▶ push x3 push t32 mult add \ 17880 ▶ ▶ ▶ ▶ ▶ return \ 17881 ▶ ▶ ▶ ▶ ▶ push x0 push t03 mult \ 17882 ▶ ▶ ▶ ▶ ▶ push x1 push t13 mult add \ 17883 ▶ ▶ ▶ ▶ ▶ push x2 push t23 mult add \ 17884 ▶ ▶ ▶ ▶ ▶ push x3 push t33 mult add \ 17885 ▶ ▶ ▶ ▶ ▶ return \ 17886 ▶ ▶ ▶ ~code_end 17887 17888 17889 Clebsch_surface:
Clebsch_surface_defining_equation_and_curves:

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 "0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,2,2,2,2,2,2,2" \
-cubics 0 $(SURFACE_COLOR_SEETHROUGH) \
-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 } \
finish { 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 \n- default_angle 80 \n- clipping_radius 4.4 \n- omit_bottom_plane \n- camera 0 "1,1,1" "-4.5,3.5,6" "0,0,0" \n- end \n- scene_objects \n- 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" \n- group_of_things "0" \n- cubics 0 $(SURFACE_COLOR_SEETHROUGH) \n- line_through_point_with_direction "0, 0, 0, 1, 0, 0" \n- line_through_point_with_direction "0, 0, -1, 0, 1, 0" \n- line_through_point_with_direction "0, 0, 0, 0, 0, -1" \n- line_through_point_with_direction "1, 0, 0, 1, 1, 1" \n- line_through_point_with_direction "-1.414213562, 0, 0, 4.146264370, 1.732050808, 1.732050808" \n- line_through_point_with_direction "0, 1.732050808, -1, 2.414213562, -0.317837246, 2.414213562" \n- line_through_point_with_direction "-2.133352390, 0, -1, 1.674708020, 1, 0" \n- line_through_point_with_direction "-2.539058015, 0, 0, 2.211360755, 1, 0" \n- line_through_point_with_direction "0, 1.48188060, 0, 0, -0.94354406, 1" \n- line_through_point_with_direction "-0.9711971171, 0, 0, 1.162155272, 0, 1" \n- line_through_point_with_direction "2.096037870, 2.096037870, 0, -1.065851905, -1.065851905, 1" \n- line_through_point_with_direction "3.921555783, 2.921555781, 0, -1.722456585, -1.722456585, 1" \n- group_of_things_as_interval 0 12 \n- lines 1 0.04 $(COLOR_YELLOW) \n- scene_objects_end \n- povray_end

896
F7_video:
- rm -r FRAMES
- mkdir FRAMES
- rm fifteen_with_lines.mp4
$(ORBITER) \\n-prepare_frames \\n- i 0 30 F7b/F7_15_lines_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 fifteen_with_lines.mp4

McKean_povray:
$(ORBITER) -v 2 -povray \\n-round 0 -nb_frames_default 30 \\n-output_mask McKean%d%03d.pov \\n-video_options -W 1024 -H 768 \\n-global_picture_scale 1.5 \\n-default_angle 80 \\n-clipping_radius 4.4 \\n-omit_bottom_plane \\n-camera 0 "1,1,1" "-4.5,3.5,6" "0,0,0" \\n-end \\n-scene_objects \\n-cubic_lex "0, 0, 1, 0, 0, -1, -2, 1, 2, 0, 0, 1, 1,-1, -1, -1, 0, 0, 0, 0" \\n-group_of_things "0" \\n-cubics 0 $(SURFACE_COLOR_SEETHROUGH) \\n-scene_objects_end \\n-povray_end
- rm -rf POV
- mkdir POV
mv McKean_0_*.pov POV
mv makefile_animation POV

McKean_video:
- rm -r FRAMES
- mkdir FRAMES
- rm McKean.mp4
$(ORBITER) \\n-prepare_frames \\n- i 0 30 MCKEAN/McKean_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 McKean.mp4

# Section 16.4: Continuous Function Plotter

SECTION CONTINUOUS_FUNCTION_PLOTTER:

lissajous:

```bash
$ORBITER -v 2
  -smooth_curve "lissajous" 0.07 0 0 0 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 sin push r mult return
  push t push b sin push r mult return
  -code_end

#function lissajous_N2000_points.csv
```

lissajous_plot:

```bash
$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\n-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 \n-transmit 0.5 } }"

scene_objects_end \n-povray_end 
- rm -rf POV
mkdir POV
mv lissajous_0_* .pov POV
mv makefile_animation POV

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
lissajous_3d_plot:

$(ORBITER) -v 2 -povray \n-round 0 -nb_frames_default 30 \n-output_mask lissajous_3d_part%03d.pov \n-video_options -W 1024 -H 768 \n-global_picture_scale 0.40 \n-default_angle 45 \n-clipping_radius 5 
-omit_bottom_plane 
-camera 0 "0,0,1" "7,7,5" "0,0,1" 
-rotate_about_z_axis \n-end \n-scene_objects \n-line_through_two_points_recentered_from_csv_file 
-coordinate_grid.csv 
-group_of_things "0" 
-group_of_things "1" 
-group_of_things "2" 

18160 #>  -lines 0 0.09 "texture{ pigment{ color Yellow } }"
18161 #>  -lines 1 0.09 "texture{ pigment{ color Yellow } }"
18162 #>  -lines 2 0.09 "texture{ pigment{ color Yellow } }"
18163 #>  -group_of_things_as_interval 3 39
18164 #>  -lines 3 0.02 "texture{ pigment{ color Black } }"
18165 #>  -point_list_from_csv_file
18166 #>  -function_lissajous_3d_N2000_points.csv
18167 #>  -group_of_things_as_interval 0 6538
18168 #>  -spheres 4 0.1 "texture{ pigment{ color Red } }"
18169 #>  -plane_by_dual_coordinates "0,0,1,0"
18170 #>  -group_of_things "0"
18171 #>  -scene_objects_end
18172 #>  -povray_end
18173 - rm -rf POV
18174 - mkdir POV
18175 - mv lissajous_3d_0.*.pov POV
18176 - mv makefile_animation POV
18177
18178 #>  -planes 5 "texture{ pigment{ color Blue*0.5 transmit 0.5 } }"
18180
18181
18182
18183
18184
18185 # Chapter 17 - Miscellaneous
18186
18187
18188
18189
18190 # Section 17.1: Miscellaneous
18191
18192 SECTION_MISCELLANEOUS:
18193
18194
18195
18196
18197 misc_select:
18198 $ (ORBITER) -v 3
18199 $ (ORBITER) -v 3
18200 $ (ORBITER) -v 3
18201 $ (ORBITER) -v 3
18202 $ (ORBITER) -v 3
18203 $ (ORBITER) -v 3
18204 $ (ORBITER) -v 3
18205 $ (ORBITER) -v 3
18206 misc_join:
\$\text{ORBITER} -v 4 \$
\$\text{csv_file_join poly_orbits_d3_n3_q2_select_F2.csv Orbit_idx} \$
\$\text{csv_file_join poly_orbits_d3_n3_q2_select_F4.csv Orbit_idx} \$
\$\text{csv_file_join poly_orbits_d3_n3_q2_select_F8.csv Orbit_idx} \$
\$\text{csv_file_join poly_orbits_d3_n3_q2_select_F16.csv Orbit_idx} \$
\$\text{csv_file_join poly_orbits_d3_n3_q2_select_F32.csv Orbit_idx} \$
\tablemod{12}
\$\text{ORBITER} -v 2 \$
\$\text{define M -vector -load_csv_no_border clock_mult_excel.csv -end} \$
\$\text{define all_one_r -vector -repeat 1 12 -end} \$
\$\text{define all_one_c -vector -repeat 1 12 -end} \$
\$\text{draw_matrix} \$
\$\text{input_object M} \$
\$\text{box_width 50 -bit_depth 24} \$
\$\text{partition 3} \$
\$\text{draw_eigenvalue diag23} \$
\$\text{ORBITER} -v 3 \$
\$\text{extract_from_file makefile Cremona_map make_Cremona_map.txt} \$
\$\text{/bin/a2tex.out -numbers -text_width 80 <make_Cremona_map.txt >make_Cremona_map.tex} \$
\$\text{draw_eigenvalue_diag23} \$
\$\text{ORBITER} -v 2 \$

# Section 17.2: Limitations

SECTION LIMITATIONS:

####

# unclassified:

extract:

$\text{/bin/a2tex.out -numbers -text_width 80 <make_Cremona_map.txt >make_Cremona_map.tex}$
-draw_options \\
-radius 10 \\
-line_width 1.5 -embedded \\
end \\
-draw_mod_n -n 20 \\
-file ev_diag23 \\
eigenvalues 2 0 0 3 \\
end \\
pdflatex ev_diag23_draw.tex \\
open ev_diag23_draw.pdf
Bibliography


[38] Fatma Karaoglu and Anton Betten. The number of cubic surfaces with 27 lines over a finite field. To appear in Journal of Algebraic Combinatorics.


[61] L. Schläfli, 1858. An attempt to determine the twenty-seven lines upon a surface of the third order and to divide such surfaces into species in reference to the reality of the lines upon the surface, Quart. J. Math. 2 (1858), 55–110.


