User’s Guide
Build Number 1473

Anton Betten

November 7, 2022
Contents

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

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

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

4 Geometry 71
  4.1 Finite Projective Spaces .................................... 71
  4.2 Indexing Points and Lines .................................. 78
  4.3 Finite Desarguesian Projective Planes ......................... 83
  4.4 The Grassmannian .......................................... 87
  4.5 Algebraic Sets ............................................. 90
  4.6 The Klein Quadric and the Plücker Map ....................... 93
  4.7 Orthogonal Spaces ......................................... 95
  4.8 Hermitian Varieties ....................................... 104
  4.9 Advanced Topics ......................................... 106
  4.10 Geometric Objects ....................................... 113
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Group Theory</td>
<td>119</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>185</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>213</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>255</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>273</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>287</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>
</tbody>
</table>
18 Orbiter on Windows 497
  18.1 Using Windows Subsystem Linux ........................................ 497

19 The Makefile 507
  19.1 The Makefile ............................................................. 507
  References ............................................................................. 905
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 environ-
ment. However, it comes at a small performance cost when running Orbiter commands that
are computationally heavy. Orbiter has already been precompiled (by the Orbiter developer) into an image, which is a completely self-sustained copy of a unix-environment that can run
by the user under the docker front-end. The image is stored on a docker server under the
name abetten/orbiter. Docker will receive the name of the image from the command line,
pull a local copy of the image, and run the image in an encapsulated environment called a container. A copy of the image is stored locally, so that subsequent calls to Orbiter can be satisfied using the local copy, which increases turnaround speed. For instance, the following bare-bones makefile sets up Orbiter for use through Docker:

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

test:
   $(ORBITER_PATH)orbiter.out

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

make test

into a terminal window, Docker starts up and pulls a copy of Orbiter to the local machine,
which is then executed. Orbiter will start up, produce a few messages and then shut down.
Interestingly, this will work on a Windows machine also (using supershell as terminal). The
make command is passed through to the container, which contains the unix-like software
Orbiter comes with a version numbering system called a build number. The build number should match the commit number on the github tree, shown in Figure 2.1. When Orbiter starts up, the build number is displayed. In order to update to a more recent version of Orbiter, Docker needs to be instructed to discard the local image. To do so, the command

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

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

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

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

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

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

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

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

(c) Type

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

This will produce an output similar to the following:

sh-3.2$ docker run -it --volume $PWD:/mnt -w /mnt abetten/orbiter orbiter.out
Unable to find image 'abetten/orbiter:latest' locally
latest: Pulling from abetten/orbiter
004f1e38d7f: Pull complete
5d6f1a81727: Pull complete
48c2f66afe: Pull complete
234b700d479d: Pull complete
6fa0a0e2e0: Pull complete
9187bd98e24: Pull complete
ae87b7ef5b0: Pull complete
260a2765fa9: Pull complete
27d6ff93a5b: Pull complete
7a09ec54d4b8: Pull complete
1336494f7e: Pull complete
Digest: sha256:099d7e0b0a9ee168b7cb261d2da8ff64bd7d861c357e1caec59580d629ee9
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/~abetten/orbiter/users_guide.pdf
The sources are available here:
https://github.com/abetten/orbiter
An example makefile with many commands from the user’s guide is here:
https://github.com/abetten/orbiter/tree/master/examples/users_guide/makefile
SYSTEMUNIX is defined
sizeof(int)=4
sizeof(long int)=8
Orbiter session finished.
User time: 0:00

The first part is Docker downloading Orbiter as a container. This can take a while, depending
on the Internet speed. The second part (Welcome to Orbiter!) is the actual Orbiter session.
No specific commands were given, so Orbiter simply starts up and quits. The first part is
done only once. Once it has been downloaded, Docker will recycle the copy of orbiter and
a download is no longer required. However, once Orbiter updates, Docker will update the
local copy of Orbiter as well.

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

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

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

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

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

(c) Issue the following commands to compile Orbiter:

```
make
make install
```

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

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 poly.</td>
</tr>
<tr>
<td>-orbiter_path</td>
<td>p</td>
<td>Set the orbiter path to $p$. This is useful in case the Orbiter session has to clone or fork new Orbiter sessions. In most cases, the orbiter path will end with a forward slash “/.”</td>
</tr>
<tr>
<td>-magma_path</td>
<td>p</td>
<td>Set the magma path to $p$. This is useful in case the Orbiter session has to create a magma process.</td>
</tr>
<tr>
<td>-fork</td>
<td>L M f t s</td>
<td>Fork new Orbiter sessions in parallel. The new sessions will be indexed by the values $i$ that result from a loop with start value $f$ and increment $s$ bounded from above by $t$, equivalent to a C-loop of type “for (i=f; i &lt; t; i+= s).” Every occurrence of the string $L$ in the argument list is replaced by the resulting value of the loop variable $i$. The forked process will write to a file whose name is described through the mask $M$. The actual file name results from using the printf command from the C-library for $M$ with the integer value of the loop variable. All of the command line arguments after the fork command are passed through to the new Orbiter session, with all arguments $L$ replaced by the integer value of the loop counter. The number of Orbiter sessions forked is $(t - f)/s$. The orbiter path from -orbiter_path is used when starting the forked sessions.</td>
</tr>
</tbody>
</table>

Table 2.1: Orbiter session commands
The build number is the version number of the Orbiter software, as defined by the number of submits to the Git repository. Higher numbers mean more recent versions. After this message, Orbiter will start parsing the command line arguments. Once this is done, the session will execute these commands. At the end of the session, a short message is given that specifies the processor time used up by the session.
## 2.3 Makefiles and Shell Scripts

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

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

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

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

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

and used anywhere later using the

\[ $(A) \]

syntax. Rules are defined using the following syntax

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

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

The purpose of this command is to produce a file called 

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

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

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

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

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

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

One difficulty in installing Orbiter is the path of installation. In the sample makefile, there is a makefile variable called \texttt{ORBITER\_PATH} which contains the path to the orbiter executable \texttt{orbiter.exe}. Depending on the local installation of orbiter, the makefile variable needs to be changed accordingly. The actual command to run the \texttt{F_16} example is as follows:
F_16:
$(ORBITER_PATH)orbiter.out -v 3 -define F -finite_field -q 16 -end \ 
-define F -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 -end is necessary to terminate the definition command. For more details on the objects that exist, see the appropriate section listed in the table. For instance, the command

```
object F_2:
$ $(ORBITER) -v 3 -define F -finite_field -q 2 -end
```

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

```
object PG_3_2:
$ $(ORBITER) \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 particular 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><code>-finite_field</code></td>
<td>A finite field $\mathbb{F}_q$. See Sections 3.2 and 3.3.</td>
</tr>
<tr>
<td><code>-polynomial_ring</code></td>
<td>A multivariate polynomial ring. See Section 8.2.</td>
</tr>
<tr>
<td><code>-linear_group</code></td>
<td>A linear group. See Section 5.2.</td>
</tr>
<tr>
<td><code>-permutation_group</code></td>
<td>A permutation group. See Section 5.1.</td>
</tr>
<tr>
<td><code>-projective_space</code></td>
<td>A projective space $\text{PG}(n, q)$. See Section 4.1.</td>
</tr>
<tr>
<td><code>-orthogonal_space</code></td>
<td>A non-degenerate orthogonal space $O^+(n, q)$. See Section 4.7.</td>
</tr>
<tr>
<td><code>-BLT_set_classify</code></td>
<td>An object to classify BLT-sets. See Section 12.4.</td>
</tr>
<tr>
<td><code>-spread_classify</code></td>
<td>An object to classify spreads. See Section 12.1.</td>
</tr>
<tr>
<td><code>-formula</code></td>
<td>An algebraic / symbolic expression. See Section 2.8.</td>
</tr>
<tr>
<td><code>-cubic_surface</code></td>
<td>A cubic surface. See Section 7.1.</td>
</tr>
<tr>
<td><code>-quartic_curve</code></td>
<td>A quartic curve. See Section 7.2.</td>
</tr>
<tr>
<td><code>-classification_of_cubic_surfaces_with_double_sixes</code></td>
<td>An object to classify cubic surfaces using double sixes. See Section 7.3.</td>
</tr>
<tr>
<td><code>-collection</code></td>
<td>A collection of objects.</td>
</tr>
<tr>
<td><code>-geometric_object</code></td>
<td>A geometric object. See Section 4.10.</td>
</tr>
<tr>
<td><code>-graph</code></td>
<td>A graph. See Section 13.1.</td>
</tr>
<tr>
<td><code>-code</code></td>
<td>A code. See Section 10.2.</td>
</tr>
<tr>
<td><code>-spread</code></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>-finite_field_activity</td>
<td>An activity for finite fields, see Sections 3.2 and 3.3.</td>
</tr>
<tr>
<td>-projective_space_activity</td>
<td>An activity for a projective space, see Section 4.1.</td>
</tr>
<tr>
<td>-orthogonal_space_activity</td>
<td>An activity for an orthogonal space, see Section 4.7.</td>
</tr>
<tr>
<td>-group_theoretic_activity</td>
<td>An activity for a group, see Section 5.6.</td>
</tr>
<tr>
<td>-cubic_surface_activity</td>
<td>An activity for a cubic surface, see Section 7.1.</td>
</tr>
<tr>
<td>-quartic_curve_activity</td>
<td>An activity for a quartic curve, see Section 7.2.</td>
</tr>
<tr>
<td>-combinatorial_object_activity</td>
<td>An activity for a combinatorial object, see Section 4.5.</td>
</tr>
<tr>
<td>-graph_theoretic_activity</td>
<td>An activity for a graph, see Section 13.1.</td>
</tr>
<tr>
<td>-classification_of_cubic_surfaces_with_double_sixes_activity</td>
<td>An activity for a cubic surface, see Section 7.1.</td>
</tr>
<tr>
<td>-spread_table_activity</td>
<td>An activity associated with a table of spreads, see Section 12.3.</td>
</tr>
<tr>
<td>-packing_with_symmetry_assumption_activity</td>
<td>An activity related to creating packings with assumed symmetry group, see Section 12.3.</td>
</tr>
<tr>
<td>-packing_fixed_points_activity</td>
<td>An activity related to creating packings with assumed symmetry group, see Section 12.3.</td>
</tr>
<tr>
<td>-graph_classification_activity</td>
<td>An activity for a classification of graphs problem, see Section 13.3.</td>
</tr>
<tr>
<td>-diophant_activity</td>
<td>An activity for a diophantine system, see Section 11.2.</td>
</tr>
<tr>
<td>-design_activity</td>
<td>An activity for a combinatorial design, see Section 11.5.</td>
</tr>
<tr>
<td>-large_set_with_symmetry_assumption_activity</td>
<td>An activity related to creating large sets of designs with assumed symmetry group, see Section 11.6.</td>
</tr>
</tbody>
</table>

Table 2.4: Orbiter Activities
BLT-sets
BLT sets of $Q(4, q)$ exist for all odd prime powers. The classification of BLT-sets of $Q(4, q)$ is known to Orbiter for all $q$ $\leq$ 73.

Cubic Surfaces
Cubic surfaces with 27 lines exist for all finite fields apart from $F_2$, $F_3$, $F_5$. Orbiter knows the classification of cubic surfaces with 27 lines for all fields $F_q$ of order $q$ $\leq$ 128.

Quartic curves
Orbiter knows the classification of smooth quartic curves with 28 bitangents in projective planes over all fields $F_q$ for $q$ = 9, 13, 19, 25, 27, 29, 31.

Spreads
A spread is a set of $q^k$ + 1 pairwise non-intersecting $k$-dimensional subspaces of $F_q^{2k}$. Spreads are related to translation planes of order $q^k$. Orbiter knows the classification of spreads for $(q, k) \in \{(2, 2), (3, 2), (2, 4), (4, 2), (5, 2), (3, 3)\}$.

Hyperovals
A hyperoval in $PG(2, 2^e)$ is a set of $2^e$ + 2 points, no three collinear. Orbiter knows the classification of hyperovals for $e$ = 3, 4, 5.

Dual hyperovals
A $k$-dimensional dual hyperoval in an ambient space $F_2^n$ is called a DH($k, n$). Orbiter knows the classification of dual hyperovals DH(4, 7) and DH(4, 8).

Packings
Orbiter knows the classification of packings of $PG(3, 3)$.

Table 2.5: Mathematical Data Available in Orbiter

<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 $F_2$, $F_3$, $F_5$. Orbiter knows the classification of cubic surfaces with 27 lines for all fields $F_q$ 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 $F_q$ 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 $F_q^{2k}$. 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 $PG(2, 2^e)$ 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 $F_2^n$ 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 $PG(3, 3)$.</td>
</tr>
</tbody>
</table>

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

The command

```
create_surface_4_0:
  $(ORBITER) -v 3 \
  > define F -finite_field -q 4 -end \
  > define P -projective_space -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
```

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

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

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

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

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

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

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

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

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

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

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

vector_example1:

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

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

vector_example2:

```
$(ORBITER) -v 2 \
  -define F -finite_field -q 5 -end \
  -define v -vector -field F -format 2 -dense "0,1,2,3,4,0" -end \
  -print_symbols
```
For larger matrices, we can use makefile variables. For instance, the following command creates the generator matrix of the Hamming code:

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

```
matrix_example1:
    $(ORBITER) -v 2 \
    -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="\n110111000100001010000\n111010111110100001011\n000000100000100010101\n111110111010001011110\n01010100000001001101\n000010000000100010101\n001000000000100010101\n00010001100001110111\n111010010101001001111\n000000000001100100101\n0000000000000100000101\n0000000000000101010101\n0000000000000100010111\n0000000000000100010001"
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 1

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

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

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

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

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

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

The command

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

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

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

\[
\text{formula\_evaluate:} \\
\text{\texttt{\$\{ORBITER\} -v 3 \ \}}
\]

Figure 2.4: The abstract syntax tree of the formula
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 `-primitive_root` command can be used. The algorithm is randomized. For instance,

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

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

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

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

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

computes

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

which is 487320.

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

\[ b^k \equiv a \pmod{p}. \]

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

```
35
```
<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 $g, u, v$ 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>$q n_{\text{min}} n_{\text{max}}$</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 `-inverse_mod` command. The command

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

computes the inverse of 1865025205 modulo 2147483647 which is 579785381.

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

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

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

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

The output is

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

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

In order to compute the modular power

$$a^e \mod n,$$

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

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

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

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

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

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

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

\( \sqrt{mod} \)

\[ \text{\$ORBITER) -v 2 -square\_root\_mod 33 41} \]

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

\[ 19^2 \equiv 33 \mod 41. \]

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

\texttt{order\_of\_2\_mod\_n:}

\[ \text{\$ORBITER) -v 3 -order\_of\_q\_mod\_n 2 3 151} \]

\[ \text{\$ORBITER) -v 1 -csv\_file\_latex 1 \} \]

\[ \text{\$ORBITER) -v order\_of\_q\_mod\_n\_q2\_3\_151.csv} \]

\[ \text{pdflatex order\_of\_q\_mod\_n\_q2\_3\_151.tex} \]

\[ \text{open order\_of\_q\_mod\_n\_q2\_3\_151.pdf} \]

produces the output shown in Table 3.2.

The command

38
Table 3.3: The values of the Eulerfunction

<table>
<thead>
<tr>
<th>N</th>
<th>PHI</th>
<th>N</th>
<th>PHI</th>
<th>N</th>
<th>PHI</th>
<th>N</th>
<th>PHI</th>
<th>N</th>
<th>PHI</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>26</td>
<td>12</td>
<td>51</td>
<td>32</td>
<td>76</td>
<td>36</td>
<td>101</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>27</td>
<td>18</td>
<td>52</td>
<td>24</td>
<td>77</td>
<td>60</td>
<td>102</td>
<td>32</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>28</td>
<td>12</td>
<td>53</td>
<td>52</td>
<td>78</td>
<td>24</td>
<td>103</td>
<td>102</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>29</td>
<td>28</td>
<td>54</td>
<td>18</td>
<td>79</td>
<td>78</td>
<td>104</td>
<td>48</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>30</td>
<td>8</td>
<td>55</td>
<td>40</td>
<td>80</td>
<td>32</td>
<td>105</td>
<td>48</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>31</td>
<td>30</td>
<td>56</td>
<td>24</td>
<td>81</td>
<td>54</td>
<td>106</td>
<td>52</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>32</td>
<td>16</td>
<td>57</td>
<td>36</td>
<td>82</td>
<td>40</td>
<td>107</td>
<td>106</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>33</td>
<td>20</td>
<td>58</td>
<td>28</td>
<td>83</td>
<td>82</td>
<td>108</td>
<td>36</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
<td>34</td>
<td>16</td>
<td>59</td>
<td>58</td>
<td>84</td>
<td>24</td>
<td>109</td>
<td>108</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>35</td>
<td>24</td>
<td>60</td>
<td>16</td>
<td>85</td>
<td>64</td>
<td>110</td>
<td>40</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>36</td>
<td>12</td>
<td>61</td>
<td>60</td>
<td>86</td>
<td>42</td>
<td>111</td>
<td>72</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>37</td>
<td>36</td>
<td>62</td>
<td>30</td>
<td>87</td>
<td>56</td>
<td>112</td>
<td>48</td>
</tr>
<tr>
<td>13</td>
<td>12</td>
<td>38</td>
<td>18</td>
<td>63</td>
<td>36</td>
<td>88</td>
<td>40</td>
<td>113</td>
<td>112</td>
</tr>
<tr>
<td>14</td>
<td>6</td>
<td>39</td>
<td>24</td>
<td>64</td>
<td>32</td>
<td>89</td>
<td>88</td>
<td>114</td>
<td>36</td>
</tr>
<tr>
<td>15</td>
<td>8</td>
<td>40</td>
<td>16</td>
<td>65</td>
<td>48</td>
<td>90</td>
<td>24</td>
<td>115</td>
<td>88</td>
</tr>
<tr>
<td>16</td>
<td>8</td>
<td>41</td>
<td>40</td>
<td>66</td>
<td>20</td>
<td>91</td>
<td>72</td>
<td>116</td>
<td>56</td>
</tr>
<tr>
<td>17</td>
<td>16</td>
<td>42</td>
<td>12</td>
<td>67</td>
<td>66</td>
<td>92</td>
<td>44</td>
<td>117</td>
<td>72</td>
</tr>
<tr>
<td>18</td>
<td>6</td>
<td>43</td>
<td>42</td>
<td>68</td>
<td>32</td>
<td>93</td>
<td>60</td>
<td>118</td>
<td>58</td>
</tr>
<tr>
<td>19</td>
<td>18</td>
<td>44</td>
<td>20</td>
<td>69</td>
<td>44</td>
<td>94</td>
<td>46</td>
<td>119</td>
<td>96</td>
</tr>
<tr>
<td>20</td>
<td>8</td>
<td>45</td>
<td>24</td>
<td>70</td>
<td>24</td>
<td>95</td>
<td>72</td>
<td>120</td>
<td>32</td>
</tr>
<tr>
<td>21</td>
<td>12</td>
<td>46</td>
<td>22</td>
<td>71</td>
<td>70</td>
<td>96</td>
<td>32</td>
<td>121</td>
<td>110</td>
</tr>
<tr>
<td>22</td>
<td>10</td>
<td>47</td>
<td>46</td>
<td>72</td>
<td>24</td>
<td>97</td>
<td>96</td>
<td>122</td>
<td>60</td>
</tr>
<tr>
<td>23</td>
<td>22</td>
<td>48</td>
<td>16</td>
<td>73</td>
<td>72</td>
<td>98</td>
<td>42</td>
<td>123</td>
<td>80</td>
</tr>
<tr>
<td>24</td>
<td>8</td>
<td>49</td>
<td>42</td>
<td>74</td>
<td>36</td>
<td>99</td>
<td>60</td>
<td>124</td>
<td>60</td>
</tr>
<tr>
<td>25</td>
<td>20</td>
<td>50</td>
<td>20</td>
<td>75</td>
<td>40</td>
<td>100</td>
<td>40</td>
<td>125</td>
<td>100</td>
</tr>
</tbody>
</table>

Eulerfunction_150:
▷ $(\text{ORBITER})$ -v 1 -eulerfunction_interval 1 150
▷ $(\text{ORBITER})$ -v 1 -csv_file latex 1 \table eulerfunction_1_150.csv
▷ pdflatex table eulerfunction_1_150.tex
▷ open table eulerfunction_1_150.pdf

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

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

power_function_2_mod_11:
▷ $(\text{ORBITER})$ -v 5 -power_function_mod_n 2 11
▷ $(\text{ORBITER})$ -v 1 -csv_file latent 1 power_function_k2_n11.csv
▷ pdflatex power_function_k2_n11.tex
▷ open power_function_k2_n11.pdf

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

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

Figure 3.1: Cycle of powers of $2$ modulo $13$

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

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

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

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

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

Chinese_remainders_A:

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

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

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

$$
\begin{align*}
x &\equiv 2 \text{ mod } 2147483647 \\
x &\equiv 3 \text{ mod } 5915587277
\end{align*}
$$

using the command

Chinese_remainders_C2:

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

The answer is

$$
\begin{align*}
x &\equiv 5684294357108828365 \mod 12703626939758759219.
\end{align*}
$$

A quick check with Maple shows that

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

as required.
### Table 3.5: Options for Creating Finite Fields

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

#### 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,

\[ F_2: \]

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

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

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

\[ F_7: \]

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

creates a cheat sheet for $\mathbb{F}_7$ as shown below. The element $\alpha$ 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</td>
<td>$v$</td>
<td>Compute the product of all field elements in the vector $v$.</td>
</tr>
<tr>
<td>-sum_of</td>
<td>$v$</td>
<td>Compute the sum of all field elements in the vector $v$.</td>
</tr>
<tr>
<td>-negate</td>
<td>$v$</td>
<td>Negate each field element in the vector $v$.</td>
</tr>
<tr>
<td>-inverse</td>
<td>$v$</td>
<td>Compute the multiplicative inverse of each field element in the vector $v$.</td>
</tr>
<tr>
<td>-power_map</td>
<td>$k \ v$</td>
<td>Compute the $k$-th power of each field element in the vector $v$.</td>
</tr>
</tbody>
</table>

Table 3.6: Finite Field Activities

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

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

$+$

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

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

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

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

\[
\begin{align*}
\ \\
\end{align*}
\]

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

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

The cheat sheet contains this list of field elements at the very end. In Figure 3.2, the addition and multiplication tables of $\mathbb{F}_7$ are shown with respect to the cyclic ordering of elements as

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

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

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

```
F_101_wo:
```

```
$\$(ORBITER) -v 3 \
```

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

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

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

Any such $E$ can be constructed as a polynomial factor ring 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 $X + 1$:

- $X \cdot X = X^2 \equiv X + 1 \pmod{X^2 + X + 1},$
- $X \cdot (X + 1) = X^2 + X \equiv 1 \pmod{X^2 + X + 1},$
- $(X + 1) \cdot X = X^2 + X \equiv 1 \pmod{X^2 + X + 1},$
- $(X + 1) \cdot (X + 1) = X^2 + 1 \equiv X \pmod{X^2 + X + 1},$

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

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

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

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

The isomorphism type of the resulting field only depends on the order $q$ of the field, and not on the choice of the polynomial. However, for practical computations, the choice of the polynomial matters. For instance, results can only be shared between different computer algebra
### Table 3.7: More Finite Field Activities

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

systems if the same polynomials are used. Orbiter has a large collection of polynomials built in. Besides these, a polynomial can be specified. The polynomials that Orbiter offers are in fact primitive, which means that the root $\alpha$ is a primitive element for the field $\mathbb{F}_q$. This just means that it is a generator of the multiplicative group. So, any non-zero element in $\mathbb{F}_q$ is a suitable power of $\alpha$.

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

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

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

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

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

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

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

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

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

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

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

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

Subfields:

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

<table>
<thead>
<tr>
<th>$\gamma_i$</th>
<th>$-\gamma_i$</th>
<th>$\gamma_i^{-1}$</th>
<th>$\log_\gamma(\gamma_i)$</th>
<th>$\alpha'$</th>
<th>$Z_i$</th>
<th>$\phi(\gamma_i)$</th>
<th>$T(\gamma_i)$</th>
<th>$N(\gamma_i)$</th>
<th>$T_2(\gamma_i)$</th>
<th>$N_2(\gamma_i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 = 0</td>
<td>0</td>
<td>DNE</td>
<td>DNE</td>
<td>1</td>
<td>DNE</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1 = 1</td>
<td>1</td>
<td>1</td>
<td>15</td>
<td>2</td>
<td>12</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>$\alpha = \delta$</td>
<td>2</td>
<td>12</td>
<td>1</td>
<td>4</td>
<td>9</td>
<td>4</td>
<td>1</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td>$\alpha + 1 = \delta^{12}$</td>
<td>3</td>
<td>8</td>
<td>12</td>
<td>8</td>
<td>4</td>
<td>5</td>
<td>1</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>$\alpha^2 = \delta^2$</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>9</td>
<td>3</td>
<td>9</td>
<td>1</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>$\alpha^2 + 1 = \delta^9$</td>
<td>5</td>
<td>15</td>
<td>9</td>
<td>11</td>
<td>10</td>
<td>8</td>
<td>1</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>$\alpha^2 + \alpha = \delta^{13}$</td>
<td>6</td>
<td>4</td>
<td>13</td>
<td>15</td>
<td>8</td>
<td>13</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>$\alpha^2 + \alpha + 1 = \delta^7$</td>
<td>7</td>
<td>14</td>
<td>7</td>
<td>7</td>
<td>13</td>
<td>12</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>$\alpha^3 = \delta^3$</td>
<td>8</td>
<td>3</td>
<td>3</td>
<td>14</td>
<td>6</td>
<td>15</td>
<td>1</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>9</td>
<td>$\alpha^3 + 1 = \delta^4$</td>
<td>9</td>
<td>13</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>14</td>
<td>1</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>10</td>
<td>$\alpha^3 + \alpha = \delta^{10}$</td>
<td>10</td>
<td>11</td>
<td>10</td>
<td>10</td>
<td>5</td>
<td>11</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>$\alpha^3 + \alpha + 1 = \delta^5$</td>
<td>11</td>
<td>10</td>
<td>5</td>
<td>13</td>
<td>14</td>
<td>10</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>$\alpha^3 + \alpha^2 = \delta^{14}$</td>
<td>12</td>
<td>2</td>
<td>14</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>$\alpha^3 + \alpha^2 + 1 = \delta^{11}$</td>
<td>13</td>
<td>9</td>
<td>11</td>
<td>6</td>
<td>7</td>
<td>7</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>$\alpha^3 + \alpha^2 + \alpha = \delta^8$</td>
<td>14</td>
<td>7</td>
<td>8</td>
<td>12</td>
<td>11</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>15</td>
<td>$\alpha^3 + \alpha^2 + \alpha + 1 = \delta^6$</td>
<td>15</td>
<td>5</td>
<td>6</td>
<td>1</td>
<td>DNE</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>10</td>
</tr>
</tbody>
</table>
Table 3.10: The subfields of $\mathbb{F}_{64}$

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

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

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

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

Orbiter uses primitive polynomials for creating extension fields. Because of this, the element $\alpha$ is always primitive. Since the numerical rank of $\alpha$ is $p$, this means that the rank $p$ always represents a primitive element in an extension field. For the addition and multiplication tables of $\mathbb{F}_9$ arranged with respect to powers of a primitive element, see Figure 3.5.
Figure 3.5: Addition and multiplication table of $\mathbb{F}_9$ using the cyclic ordering of elements
Table 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

\[
\text{RREF:}\n\]

\[
\begin{align*}
\text{\texttt{\$ (ORBITER) -v 2 \}} \\
\text{\texttt{\> -define F -finite_field -q 2 -end \}} \}
\text{\texttt{\> -define v -vector -field F -format 2 \}} \}
\text{\texttt{\> -dense "1,1,1,0,1,1,0,0,1" \}} \}
\text{\texttt{\> -end \}} \\
\text{\texttt{\> -with F -do -finite_field_activity \}} \}
\text{\texttt{\> -RREF v -normalize_from_the_right \}} \}
\text{\texttt{\> -end \}}
\end{align*}
\]

computes the RREF form of the matrix

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

over \( \mathbb{F}_2 \). The output is the matrix
The -RREF command produces a latex log of the steps. This can be used to follow the algorithm along. For a somewhat longer example, consider the Vandermonde matrix over the field $\mathbb{F}_7$. Suppose we want to compute the inverse matrix directly. We can use the following command to do so. Notice how we first create the matrix and an identity matrix next to it. After that we apply the -RREF command:

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

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

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

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

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

After making pivot 1:

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

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

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

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

\begin{verbatim}
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"
\end{verbatim}

computes all eigenvectors and eigenvalues of the matrix

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

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

Orbiter can produce a list of all conjugacy classes of endomorphisms of \(\mathbb{F}_q^d\) by means of their rational normal forms. For instance
produces a list of all conjugacy classes of $GL(3, 2)$. There are 6 of them. The report includes
the order of the centralizer and the order of the conjugacy class. The order of the centralizer
is computed using Kung’s formula [41]. This command relies on the Orbiter catalogue of
irreducible polynomials. For an introduction to the rational normal form of endomorphisms,
see [47].

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

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

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

Class 0 / 6

3, 1, 0

centralizer order 7
class size 24

Class 1 / 6

2, 1, 0

centralizer order 7
class size 24
Class 2 / 6
0, 1, 0; 1, 1, 0
\[
\begin{bmatrix}
0 & 1 & 0 \\
1 & 1 & 0 \\
0 & 0 & 1 \\
\end{bmatrix}
\]
centralizer order 3
class size 56

Class 3 / 6
0, 3, 0
\[
\begin{bmatrix}
1 & 0 & 0 \\
1 & 1 & 0 \\
0 & 1 & 1 \\
\end{bmatrix}
\]
centralizer order 4
class size 42

Class 4 / 6
0, 3, 1
\[
\begin{bmatrix}
1 & 0 & 0 \\
1 & 1 & 0 \\
0 & 0 & 1 \\
\end{bmatrix}
\]
centralizer order 8
class size 21

Class 5 / 6
0, 3, 2
\[
\begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
\end{bmatrix}
\]
centralizer order 168
class size 1

59
3.5 Advanced Topics in Finite Fields

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

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

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

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

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

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

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

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

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

$$
b_1^\Phi = (1 + X + X^2)^2 = 1 + X^2 + X^4 = 1 + X^2 + X^3 + X = 1 + X + X^2 + X^2 + 1 = X = b_2,$n_2^\Phi = X^2 = b_3,$n_3^\Phi = X^4 = X^3 + X = X^2 + X + 1 = b_1.
$$

Thus,

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

under $\Phi$, as required.

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

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

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

Table 3.12: Finite Field Activities (Part I)
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-Vandermonde_matrix</td>
<td>$n$</td>
<td>Creates the Vandermonde matrix of order $q \times q$. The entry $(i, j)$ is $x_j^i$ where $w_0, \ldots, x_{q-1}$ is the list of field elements in ordered according to the Orbiter ranks.</td>
</tr>
<tr>
<td>-transversal</td>
<td>$L1 \ L2 \ P$</td>
<td>Computes the unique transversal to the lines $L1$ and $L2$ 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>$L1 \ L2$</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 $F_{64}$ has many subfields. Figure 3.7 shows the field reduction from $F_{64}$ to $F_4$ (left) and from $F_{64}$ to $F_2$ (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 $F_q$ can be computed using the `-nth_roots` command, which is a finite field activity. The activity is applied to the field $F_q$ over which the $n$-th roots are defined. The command constructs the field extension $F_{q^m}$ 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 $F_{q^m}$ 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 \n▷ ▷ -define F -finite_field -q 8 -override_polynomial 11 -end \n▷ ▷ -with F -do -coding_theoretic_activity \n▷ ▷ ▷ -nth_roots 21 \n▷ ▷ -end
▷ pdflatex Nth_roots_q8_n21.tex
▷ open Nth_roots_q8_n21.pdf
```

The output is:

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

$\beta^0 = 100000 = 1$
$\beta^1 = 000100 = \alpha^3$
$\beta^2 = 100001 = \alpha^5 + 1$
$\beta^3 = 111101 = \alpha^5 + \alpha^3 + \alpha^2 + \alpha + 1$
Let $\gamma$ be a primitive 7-th root in GF(64), so $\gamma = \alpha^9$.

\begin{align*}
\gamma^0 &= 100000 = 1 \\
\gamma^1 &= 111011 = \alpha^5 + \alpha^3 + \alpha^2 + \alpha + 1 \\
\gamma^2 &= 110100 = \alpha^3 + \alpha + 1 \\
\gamma^3 &= 011001 = \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
\end{align*}

The $q$-cyclotomic set for $q = 8$ are:

\begin{align*}
\{ 0 \} \\
\{ 1, 8 \} \\
\{ 2, 16 \} \\
\{ 3 \} \\
\{ 4, 11 \} \\
\{ 5, 19 \} \\
\{ 6 \} \\
\{ 7, 14 \} \\
\{ 9 \} \\
\{ 10, 17 \} \\
\{ 12 \} \\
\{ 13, 20 \} \\
\{ 15 \} \\
\{ 18 \}
\end{align*}
Subfield basis, a basis for GF(8) inside GF(64):

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

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

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

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

```
F_8.vandermonde:
▷ $(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 $\mathbb{F}_{1024}$:

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

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

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

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

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

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

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

For more on rings, see Chapter 8.
Chapter 4

Geometry

4.1 Finite Projective Spaces

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

```
PG_3_2_easy:
  ▷ $(ORBITER) -v 2 \
  ▷ ▷ -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:

```
PG_2_4:
  ▷ $(ORBITER) -v 2 \
  ▷ ▷ -define F -finite_field -q 4 -end \
  ▷ ▷ -define P -projective_space -n 2 -field F -v 0 -end \
```
Figure 4.1: The plane $\text{PG}(2, 4)$

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 $\text{PG}(2, 2)$. After that, the points are listed again, but with left-normalized vectors. Finally, the lines are shown.

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

Baer subgeometry:

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

There are 7 elements in the Baer subgeometry.

Normalized from the left:

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

The Lines of PG(2, 4). PG(2, 4) has 21 1-subspaces:
| $L_0 = \begin{bmatrix} 100 \\ 010 \end{bmatrix}$ | $L_7 = \begin{bmatrix} 101 \\ 012 \end{bmatrix}$ | $L_{14} = \begin{bmatrix} 120 \\ 001 \end{bmatrix}$ |
| $L_1 = \begin{bmatrix} 100 \\ 011 \end{bmatrix}$ | $L_8 = \begin{bmatrix} 101 \\ 013 \end{bmatrix}$ | $L_{15} = \begin{bmatrix} 103 \\ 010 \end{bmatrix}$ |
| $L_2 = \begin{bmatrix} 100 \\ 012 \end{bmatrix}$ | $L_9 = \begin{bmatrix} 110 \\ 001 \end{bmatrix}$ | $L_{16} = \begin{bmatrix} 103 \\ 011 \end{bmatrix}$ |
| $L_3 = \begin{bmatrix} 100 \\ 013 \end{bmatrix}$ | $L_{10} = \begin{bmatrix} 102 \\ 010 \end{bmatrix}$ | $L_{17} = \begin{bmatrix} 103 \\ 012 \end{bmatrix}$ |
| $L_4 = \begin{bmatrix} 100 \\ 001 \end{bmatrix}$ | $L_{11} = \begin{bmatrix} 102 \\ 011 \end{bmatrix}$ | $L_{18} = \begin{bmatrix} 103 \\ 013 \end{bmatrix}$ |
| $L_5 = \begin{bmatrix} 101 \\ 010 \end{bmatrix}$ | $L_{12} = \begin{bmatrix} 102 \\ 012 \end{bmatrix}$ | $L_{19} = \begin{bmatrix} 130 \\ 001 \end{bmatrix}$ |
| $L_6 = \begin{bmatrix} 101 \\ 011 \end{bmatrix}$ | $L_{13} = \begin{bmatrix} 102 \\ 013 \end{bmatrix}$ | $L_{20} = \begin{bmatrix} 010 \\ 001 \end{bmatrix}$ |

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

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

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

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

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

PG(3, 2) has 15 points:

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

Normalized from the left:

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

The lines of PG(3, 2)

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

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

Lines sorted by Pluecker coordinates

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

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

\[
2 = \text{Pl}(0, 0, 1, 0, 0, 0) = L_{4} = \begin{bmatrix} 1000 \\ 0010 \end{bmatrix}
\]

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

\[
4 = \text{Pl}(0, 0, 0, 0, 1, 0) = L_{6} = \begin{bmatrix} 1000 \\ 0001 \end{bmatrix}
\]

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

\[
\vdots
\]

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

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

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

\[
L_{4} = \begin{bmatrix} 1000 \\ 0010 \end{bmatrix} = \text{Pl}(0, 0, 1, 0, 0, 0)
\]

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

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

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

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

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

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

$: \quad \begin{bmatrix} 0100 \\ 0010 \\ 0001 \end{bmatrix}$

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

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>$(0, 0, 1, 0)$</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 \(x \in \mathbb{F}_q^n\), 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 \(\text{PG}(2,4)\):

\[
\text{PG}_2.4\_\text{rank\_point}: \quad \text{\$ (ORBITER) -v 2 \}
\]
Algorithm 1 Rank

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

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

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

1: procedure Unrank(int : a, field : $\mathbb{F}_q$, int : n)
2:   assert $a \in \mathbb{Z}_N$ where $N = \theta_{n-1}(q)$.
3:   if $a < n$ then
4:     return $e_a$
5:   $a \leftarrow a - n$
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

```
PG_2.4_unrank_point:
▷ $(ORBITER) -v 2 \n▷ ▷ -define v -vector -dense "19,20" -end \n▷ ▷ -define F -finite_field -q 4 -end \n▷ ▷ -with F -do -finite_field_activity \n▷ ▷ ▷ -unrank_point_in_PG 2 v -end
```

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

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

```
PG_2.8_incidence_matrix:
▷ $(ORBITER) -v 2 \n▷ ▷ -define F -finite_field -q 8 -end \n▷ ▷ -define P -projective_space -n 2 -field F -v 0 -end \n▷ ▷ -with P -do -projective_space_activity \n▷ ▷ ▷ -export_point_line_incidence_matrix \n▷ ▷ ▷ -end
▷ $(ORBITER) -v 2 \n▷ ▷ -define all_one -vector -repeat 1 73 -end \n▷ ▷ -draw_matrix \n▷ ▷ ▷ -input_csv_file PG_n2_q8_incidence_matrix.csv \n▷ ▷ ▷ -box_width 20 -bit_depth 8 \n▷ ▷ ▷ -partition 3 \n▷ ▷ ▷ ▷ all_one all_one \n▷ ▷ ▷ ▷ -end
▷ open PG_n2_q8_incidence_matrix_draw.bmp
```

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

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

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

The command

\[
\text{PG}_2\_16: \\
\text{\$ (ORBITER) -v 2 } \\
\text{\texttt{-draw_options -xin 20000 -yin 20000 } } \\
\text{\texttt{-radius 200 -line_width 0.3 -nodes_empty -end } } \\
\text{\texttt{-define F -finite_field -q 16 -end } } \\
\text{\texttt{-define P -projective_space -n 2 -field F -v 0 -end } } \\
\text{\texttt{-with P -do -projective_space_activity } } \\
\text{\texttt{-cheat_sheet } } \\
\text{\texttt{-end } } \\
pdflatex \text{PG}_2\_16\.tex \\
\text{open PG}_2\_16\.pdf
\]

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

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

\[
\text{PG}_2\_4\_with\_decomposition: \\
\text{\$ (ORBITER) -v 2 } \\
\text{\texttt{-define F -finite_field -q 4 -end } } \\
\text{\texttt{-define P -projective_space -n 2 -field F -v 0 -end } } \\
\text{\texttt{-with P -do -projective_space_activity } } \\
\text{\texttt{-cheat_sheet_for_decomposition_by_element_PG } } \\
\text{\texttt{1 "0,1,0, 0,0,1, 2,1,1, 0" } } \\
\text{\texttt{"PG}_2\_4\_singer" } \\
\text{\texttt{-end } } \\
pdflatex \text{PG}_2\_4\_singer\.tex \\
\text{open PG}_2\_4\_singer\.pdf
\]

83
Considering the cyclic group generated by

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

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:

<table>
<thead>
<tr>
<th>→</th>
<th>21</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>5</td>
</tr>
</tbody>
</table>

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

```bash
PG_2.4_incma_cyclic:
  $ (ORBITER) -v 2 \n  -list_arguments \n  -define R -vector -repeat 1 21 -end \n  -define C -vector -repeat 1 21 -end \n  -draw_matrix \n  -input_csv_file PG_2.4_singer_incma_cyclic.csv \n  -box_width 40 -bit_depth 24 \n  -partition 3 R C \n  -end
open PG_2.4_singer_incma_cyclic_draw.bmp
```

The cyclic incidence matrix is shown in Figure 4.4.
Figure 4.5: Tactical decomposition of the incidence matrix of PG(2, 4)

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

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

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

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

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

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 $N = \left[\begin{array}{c} n \\ k \end{array}\right]_q$. In order to do so, Orbiter picks a basis for each subspace. By writing the elements of the basis in the rows of a matrix, a $k \times n$ matrix is obtained. In order to make the matrix unique, we assume it to be in RREF. In coding theory, such a matrix is called a generator matrix.

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

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

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

$$L_0 = \begin{bmatrix} 100 \\ 010 \end{bmatrix}, \quad L_3 = \begin{bmatrix} 101 \\ 010 \end{bmatrix}, \quad L_6 = \begin{bmatrix} 010 \\ 001 \end{bmatrix}$$

$$L_1 = \begin{bmatrix} 100 \\ 011 \end{bmatrix}, \quad L_4 = \begin{bmatrix} 101 \\ 011 \end{bmatrix}, \quad L_5 = \begin{bmatrix} 110 \\ 001 \end{bmatrix}$$

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

```
```

87
PG_3_3_rank_lines:
▷ $(ORBITER) -v 2
▷ ▷ -define v1 -vector -format 3
▷ ▷ -dense "1,0,2,2,0,1,1,2, 1,0,2,0,0,1,1,2, 1,0,2,2,0,1,2,1"
▷ ▷ -end
▷ ▷ -define v2 -vector -format 3
▷ ▷ -dense "1,0,0,0,1,0,0,1,0,0,0,0,0,1,0,1,0,0,0,2,1"
▷ ▷ -end
▷ ▷ -define F -finite_field -q 3 -end
▷ ▷ -define P -projective_space -n 3 -field F -v 0 -end
▷ ▷ -with P -do
▷ ▷ -projective_space_activity
▷ ▷ ▷ -rank_lines_in_PG v1
▷ ▷ ▷ -end
▷ ▷ -with P -do
▷ ▷ -projective_space_activity
▷ ▷ ▷ -rank_lines_in_PG v2
▷ ▷ -end

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

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

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

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

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

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

\begin{verbatim}
EC_11_EQUATION="1,0,3,0,0,0,10,1,0,0"
\end{verbatim}

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

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

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

Hirschfeld_surface_q4.txt:

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

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

The command

\begin{verbatim}
GR_4.2.2: $(ORBITER) -v 2 \\
  -define F -finite_field -q 2 -end \\
  -with F -do -finite_field_activity \\
  -cheat_sheet_Gr 4 2 -end \\
  pdflatex Gr_4.2.2.tex \\
  open Gr_4.2.2.pdf
\end{verbatim}

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

There are 35 lines:

\begin{align*}
L_0 &= \begin{bmatrix} 1000 \\ 0100 \end{bmatrix} = \text{Pl}(1, 0, 0, 0, 0, 0) \\
L_1 &= \begin{bmatrix} 1000 \\ 0110 \end{bmatrix} = \text{Pl}(1, 0, 1, 0, 0, 0) \\
L_2 &= \begin{bmatrix} 1000 \\ 0101 \end{bmatrix} = \text{Pl}(1, 0, 0, 0, 1, 0) \\
&\vdots \\
L_{34} &= \begin{bmatrix} 0010 \\ 0001 \end{bmatrix} = \text{Pl}(0, 1, 0, 0, 0, 0)
\end{align*}

The Plücker coordinates satisfy

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

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

The Orbiter labeling of points of the $Q^+(5, q)$ quadric (see Section 4.7) can then be used to enumerate the lines of $\text{PG}(3, q)$ in a second, different way. In the example of $\text{PG}(3, 2)$, this yields the following list (output shortened):
\[ 0 = \text{Pl}(1, 0, 0, 0, 0, 0) = L_0 = \begin{bmatrix} 1000 \\ 0100 \end{bmatrix} \]
\[ 1 = \text{Pl}(0, 1, 0, 0, 0, 0) = L_{34} = \begin{bmatrix} 0010 \\ 0001 \end{bmatrix} \]
\[ 2 = \text{Pl}(0, 0, 1, 0, 0, 0) = L_4 = \begin{bmatrix} 1000 \\ 0010 \end{bmatrix} \]
\[ \vdots \]
\[ 34 = \text{Pl}(0, 1, 1, 1, 1, 1) = L_{26} = \begin{bmatrix} 1101 \\ 0011 \end{bmatrix} \]
Table 4.2: Nondegenerate Quadrics in PG($n, q$) and the canonical form adopted in Orbiter

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

4.7 Orthogonal Spaces

Orbiter can create and work with orthogonal spaces and their groups. An orthogonal space is created by a quadratic form. We assume that the form is nondegenerate. There are three types of nondegenerate quadratic forms in PG($n, q$). Two when $n$ is odd (hyperbolic and elliptic) and one if $n$ is even (parabolic). Basic information about these quadrics and their representative quadratic forms in Orbiter is given in Table 4.2. Here, $p(X, Y) = c_1X^2 + c_2XY + c_3Y^2 \in \mathbb{F}_q[X, Y]$ is irreducible over $\mathbb{F}_q$. To create an orthogonal space, the command can be used. Here, $d = n + 1$, $q$ is the order of the finite field, and

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

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

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

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

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

```plaintext
(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

\texttt{Op\_6\_2:}
\begin{itemize}
  \item \texttt{\$\{ORBITER\} -v 2 \}\textbackslash
  \item \quad \texttt{-define F -finite_field -q 2 -end \textbackslash}
  \item \quad \texttt{-define O -orthogonal_space 1 6 F -without_group -end \textbackslash}
  \item \quad \texttt{-with O -do -orthogonal_space_activity \textbackslash}
  \item \quad \texttt{-cheat_sheet orthogonal -end}
\end{itemize}
\texttt{pdflatex O\_1\_6\_2\_report.tex}
\texttt{open O\_1\_6\_2\_report.pdf}

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

\begin{table}
\centering
\begin{tabular}{lrrrrrr}
\hline
  & 9 & 36 & 18 & 18 & 6 & 9 & 9 \\
\hline
 6 & 3 & 6 & 0 & 0 & 0 & 0 & 0 \\
9 & 0 & 4 & 4 & 0 & 0 & 1 & 0 \\
9 & 0 & 4 & 0 & 4 & 0 & 0 & 1 \\
9 & 1 & 0 & 2 & 2 & 1 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 9 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 9 & 0 \\
\hline
\end{tabular}
\begin{tabular}{lrrrrrr}
\hline
  & 9 & 36 & 18 & 18 & 6 & 9 & 9 \\
\hline
 6 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\
9 & 0 & 1 & 2 & 0 & 0 & 1 & 0 \\
9 & 0 & 1 & 0 & 2 & 0 & 0 & 1 \\
9 & 1 & 0 & 1 & 1 & 3 & 1 & 1 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\hline
\end{tabular}
\end{table}

The number of points is 35 points:
\begin{itemize}
  \item $P_0 = (1, 0, 0, 0, 0, 0)$
  \item $P_1 = (0, 1, 0, 0, 0, 0)$
  \item $P_2 = (0, 0, 1, 0, 0, 0)$
  \item $P_3 = (1, 0, 1, 0, 0, 0)$
  \item $P_4 = (0, 1, 1, 0, 0, 0)$
  \item $P_5 = (0, 0, 0, 1, 0, 0)$
  \item $P_6 = (1, 0, 0, 1, 0, 0)$
  \item $P_7 = (0, 1, 0, 1, 0, 0)$
\end{itemize}
Orbiter has enumerators for points and lines in orthogonal spaces. For small spaces, the cheat sheet lists points and lines in the Orbiter ordering. Creating the groups can be expensive. For large spaces, it may be necessary to disable the group using the -without_group option. The command

```
Op_6_64_line_rank:
```

```
$($(ORBITER) -v 4 \n`$define F -finite_field -q 64 -end \n```

```
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 = \[
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 1 & 0
\end{bmatrix}
\] \{P_0, P_{32}, P_{33}\}
L_1 = \[
\begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 1 & 0
\end{bmatrix}
\] \{P_1, P_{32}, P_{34}\}
L_{104} = \[
\begin{bmatrix}
1 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0
\end{bmatrix}
\] \{P_8, P_9, P_{18}\}
```
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 $Q(5, 64)$:

```bash
Op_6.64_report:
> $(ORBITER) -v 4 \\
> -define F -finite_field -q 64 -end \\
> -define O -orthogonal_space 1 6 F -without_group -end \\
> -with 0 -do -orthogonal_space_activity \\
> -unrank_line_through_two_points 15447347 15225451 \\
> -end
```

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

The group is not available.
The quadratic form is:

$X_0X_1 + X_2X_3 + X_4X_5 = 0$

<table>
<thead>
<tr>
<th></th>
<th>16769025</th>
<th>1090252800</th>
<th>532350</th>
<th>532350</th>
<th>130</th>
<th>4225</th>
<th>4225</th>
</tr>
</thead>
<tbody>
<tr>
<td>16511040</td>
<td>65</td>
<td>4160</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>266175</td>
<td>0</td>
<td>4096</td>
<td>128</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>266175</td>
<td>0</td>
<td>4096</td>
<td>0</td>
<td>128</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4225</td>
<td>3969</td>
<td>0</td>
<td>126</td>
<td>126</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>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>
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 $\text{PG}(5, 5)$, the subspace generated by the rows of the matrix

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

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

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

The elliptic quadric has 26 points.
The coding of points and line in orthogonal spaces is different from the coding of points in projective spaces. We will create and print a set called BLT set (after [4]). This is a set of \( q + 1 \) points on the \( Q(4, q) \) quadric satisfying a special geometric property. According to Table 4.2, 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:**

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

```bash
$ (ORBITER) -v 2 \n  \triangleright -define F -finite_field -q 7 -end \n  \triangleright -define O -orthogonal_space 0 5 F -without_group -end \n  \triangleright -define S -set -file_orbiter_format BLT_7_1.txt -end \n  \triangleright -with O -do -orthogonal_space_activity \n  \triangleright -print_points S -end
```

```
pdflatex S_set_report.tex
open S_set_report.pdf
```

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

<table>
<thead>
<tr>
<th>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.

\[
\text{Doily}_W.2:
\text{\$ ( ORBITER ) - v 2 } \text{ \( \text{-define F - finite_field - q 2 - end } \) }
\text{\$ ( ORBITER ) - v 2 } \text{ \( \text{-define O - orthogonal_space 0 5 F - without_group - end } \) }
\text{\$ ( ORBITER ) - v 2 } \text{ \( \text{-define W2_points - set - loop 0 15 1 - end } \) }
\text{\$ ( ORBITER ) - v 2 } \text{ \( \text{-define W2_lines - set - loop 0 15 1 - end } \) }
\text{\$ ( ORBITER ) - v 2 } \text{ \( \text{- with 0 - do } \) }
\text{\$ ( ORBITER ) - v 2 } \text{ \( \text{- orthogonal_space_activity } \) }
\text{\$ ( ORBITER ) - v 2 } \text{ \( \text{- print_points W2_points } \) }
\text{\$ ( ORBITER ) - v 2 } \text{ \( \text{- end } \) }
\text{\$ ( ORBITER ) - v 2 } \text{ \( \text{- with 0 - do } \) }
\text{\$ ( ORBITER ) - v 2 } \text{ \( \text{- orthogonal_space_activity } \) }
\text{\$ ( ORBITER ) - v 2 } \text{ \( \text{- print_lines W2_lines } \) }
\text{\$ ( ORBITER ) - v 2 } \text{ \( \text{- end } \) }
\text{pdflatex W2_points_set_report.tex }
\text{open W2_points_set_report.pdf }
\text{pdflatex W2_lines_set_of_lines_report.tex }
\text{open W2_lines_set_of_lines_report.pdf }
\]
4.8 Hermitian Varieties

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

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

The command

$H_{2.4}$:

$\text{\textbackslash $(ORBITER) -v 2 \textbackslash$}\$  
$\text{\textbackslash$define F -finite_field -q 4 -end \textbackslash$}\$  
$\text{\textbackslash$with F -do -finite_field_activity \textbackslash$}\$  
$\text{\textbackslash$cheat_sheet_hermitian 2 -end \textbackslash$}\$  
$pdflatex H_{2.4}.tex$  
$open H_{2.4}.pdf$

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

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

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

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

The command

$H_{3.4}$:

$\text{\textbackslash $(ORBITER) -v 2 \textbackslash$}\$  
$\text{\textbackslash$define F -finite_field -q 4 -end \textbackslash$}\$  
$\text{\textbackslash$with F -do -finite_field_activity \textbackslash$}\$  
$\text{\textbackslash$cheat_sheet_hermitian 3 -end \textbackslash$}\$  
$pdflatex H_{3.4}.tex$  
$open H_{3.4}.pdf$

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

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

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

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 $\mathbb{F}_4$.  

105
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

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

\[
\texttt{\$ORBTER) -v 2 \}
\]

\[
\texttt{\$define F -finite\_field -q 4 -end \}
\]

\[
\texttt{\$define P -projective\_space -n 3 -field F -v 0 -end \}
\]

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

\[
\texttt{\$projective\_space\_activity \}
\]

\[
\texttt{\$cheat\_sheet\_for\_decomposition\_by\_element\_PG 1 \}
\]

\[
\texttt{\$fix\_structure\_2A \}
\]

\[
\texttt{\$-end \}
\]

\[
\texttt{pdflatex fix\_structure\_2A.tex}
\]

\[
\texttt{open fix\_structure\_2A.pdf}
\]

can be used.

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

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

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

The command
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--export_point_line_incidence_matrix</code></td>
<td></td>
<td>Create a csv file of the point line incidence matrix.</td>
</tr>
<tr>
<td><code>--table_of_cubic_surfaces_compute_properties</code></td>
<td><code>fname q0 col-offset</code></td>
<td>See Section 7.5.</td>
</tr>
<tr>
<td><code>--cubic_surface_properties_analyze</code></td>
<td><code>fname q0</code></td>
<td>See Section 7.5.</td>
</tr>
<tr>
<td><code>--canonical_form_of_code</code></td>
<td><code>label m n matrix</code></td>
<td>Compute the automorphism group of a linear code using Nauty. See Section 10.</td>
</tr>
<tr>
<td><code>--map</code></td>
<td><code>label parameters</code></td>
<td>evaluate a formula using the given parameters</td>
</tr>
<tr>
<td><code>--analyze_del_Pezzo_surface</code></td>
<td><code>label parameters</code></td>
<td></td>
</tr>
<tr>
<td><code>--cheat_sheet_for_decomposition_by_element_PG</code></td>
<td><code>power elt fname</code></td>
<td>Analyzes the orbit structure of the cyclic group generated by the given element in the action on PG(n, q).</td>
</tr>
<tr>
<td><code>--cheat_sheet_for_decomposition_by_subgroup</code></td>
<td><code>label descr</code></td>
<td>Analyzes the orbit structure of the subgroup H in the action on PG(n, q). The subgroup must be a linear group, and the description of H must come from the commands from Section 5.2.</td>
</tr>
<tr>
<td><code>--table_of_quartic_curves</code></td>
<td></td>
<td>Export the classification of quartic curves to a csv file.</td>
</tr>
<tr>
<td><code>--table_of_cubic_surfaces</code></td>
<td></td>
<td>Export the classification of cubic surfaces to a csv file.</td>
</tr>
<tr>
<td><code>--classify_surfaces_with_double_sixes</code></td>
<td><code>label control</code></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-descr</td>
<td></td>
</tr>
<tr>
<td>-sweep_4_27</td>
<td>fname surface-descr</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>-trihedra1_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$_{\text{min}}$ 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$_{\text{min}}$ s t T</td>
<td>Finds arcs with the two given sets as s-lines and t-lines, respectively.</td>
</tr>
<tr>
<td>-arc_with_three_given_sets_of_lines_after_dualizing</td>
<td>sz d d$_{\text{min}}$ s t T u U</td>
<td>Finds arcs with the three given sets as s-lines and t-lines and u-lines, respectively.</td>
</tr>
<tr>
<td>-dualize_hyperplanes_to_points</td>
<td></td>
<td>Turns ranks of hyperplanes into ranks of points.</td>
</tr>
<tr>
<td>-dualize_points_to_hyperplanes</td>
<td></td>
<td>Turns ranks of points into ranks of hyperplanes.</td>
</tr>
<tr>
<td>-dualize_rank_k_subspaces</td>
<td>k</td>
<td>Turns ranks of $k$-subspaces into ranks of $n - k$ subspaces.</td>
</tr>
<tr>
<td>-classify_arcs</td>
<td>descr</td>
<td>Classify arcs.</td>
</tr>
<tr>
<td>-classify_cubic_curves</td>
<td></td>
<td>Classify cubic curves.</td>
</tr>
</tbody>
</table>

Table 4.7: Projective Space Activities (Part 3)
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-latex_homogeneous_equation</td>
<td>$d$ 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>

Table 4.8: Projective Space Activities (Part 4)

"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
mogeneous in the managed variables. The other variables can be used to represent scalar parameters, for instance. Here is an example. Suppose we want to study the del Pezzo surfaces

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

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

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

for \( f_3 \) and

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

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

del_Pezzo_F13ab_report:

\[
\begin{verbatim}
$ORBITER) -v 3 \n  -define F -finite_field -q 13 -end \n  -define P -projective_space -n 3 -field F -v 0 -end \n  -define f3 -formula "del_Pezzo_F13a" "del_Pezzo_F13a" "x,y,z" \n  -define f4 -formula "x*x*x*x+y*y*y*y+z*z*z*z-x*x*y*y" \n  -define del_Pezzo13 -collection "f3,f4" \n  -with P -do \n  -projective_space_activity \n  -analyze_del_Pezzo_surface del_Pezzo13 "" \n  -end
pdflatex del_Pezzo_F13b_report.tex
open del_Pezzo_F13b_report.pdf
\end{verbatim}
\]

The third argument after the \(-formula\) command specifies the managed variables, which are \( x,y,z \). The command \(-collection\) is used to group objects together. In this case, both surfaces are grouped together under the new name. That way, we can issue the \(-analyze_del_Pezzo_surface\) once, and it applies to both surfaces.
4.10 Geometric Objects

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

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

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

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

```
ovoide_ST_q8:
  $(ORBITER) -v 2 \n    -define F -finite_field -q 8 -end \n    -define P -projective_space -n 3 -field F -v 0 -end \n    -define O -geometric_object P \n    -ovoide_ST \n    -end \n  -with O -do -combinatorial_object_activity -save \n  -end
```

The Edge curve is given by the equation

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

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

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

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, . . .)</td>
</tr>
<tr>
<td>-elliptic_quadric_</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>ε</td>
<td>Create the $Q^e(n,q)$ 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 $XX^q + YZ^q + ZY^q = 0$</td>
</tr>
<tr>
<td>-desarguesian_line_spread_in_PG_3_q</td>
<td></td>
<td>Create the desarguesian line spread in PG(3, q) as a set of 2-subspaces</td>
</tr>
<tr>
<td>-Buekenhout_Metz</td>
<td></td>
<td>Create the Buekenhout Metz unital</td>
</tr>
<tr>
<td>-Uab</td>
<td>$a \ b$</td>
<td>Create the Buekenhout Metz unital in the form of Barwick and Ebert [5]</td>
</tr>
<tr>
<td>-whole_space</td>
<td></td>
<td>Create the whole space</td>
</tr>
<tr>
<td>-hyperplane</td>
<td>$pt$</td>
<td>Create the hyperplane given by dual coordinates associated with the given point</td>
</tr>
<tr>
<td>-segre_variety</td>
<td>$a \ b$</td>
<td>Create the Segre variety</td>
</tr>
<tr>
<td>-Maruta_Hamada_arc</td>
<td></td>
<td>Create the Maruta Hamada arc</td>
</tr>
<tr>
<td>-projective_variety lab_ascii lab_tex d\ coeffs</td>
<td></td>
<td>Create a projective variety of degree $d$ from an equation. By default, the coefficients of the equation are listed in the partition ordering. A different ordering can be specified. A label for the variety in ascii and in tex is required. See Section 4.5.</td>
</tr>
<tr>
<td>-intersection_of_zariski_open_sets l d n C1 ... Cn</td>
<td></td>
<td>Create the intersection of the Zariski open sets given by equations $C_1, \ldots, C_n$ of degree $d$ with label $l$, see Section 4.5.</td>
</tr>
<tr>
<td>-projective_curve l r d C</td>
<td></td>
<td>Create the projective curve of degree $d$ with label $l$, with coefficient vector $C$ in $r$ variables</td>
</tr>
</tbody>
</table>

Table 4.11: 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 PG($n, q$) instead of orthogonal point ranks</td>
</tr>
<tr>
<td>-monomial_type_LEX</td>
<td></td>
<td>Select lexicographic ordering of coefficients in an algebraic equation.</td>
</tr>
<tr>
<td>-monomial_type_PART</td>
<td></td>
<td>Select partition ordering of coefficients in an algebraic equation (default).</td>
</tr>
</tbody>
</table>

Table 4.12: Orbiter Objects: Modifiers

115
FILE_Q17="orbit,curve,pts_on_curve,bitangents,go\n0,"$(EDGE_CURVE_Q17_EQUATION)","$(EDGE_CURVE_Q17_AS_POINTS)","",-1\n\nEND"

Edge_curve_17:
> $(ORBITER) -v 2 \n> -define F -finite_field -q 17 -end \n> -define R -polynomial_ring -field F \n> -number_of_variables 3 \n> -homogeneous_of_degree 4 \n> -end \n> -define P -projective_space -n 2 -field F -v 0 -end \n> -define C -geometric_object P \n> -projective_variety R \n> "Edge_q17" "Edge\_q17" \n> $(EDGE_CURVE_Q17_EQUATION) \n> -end \n> -with C -do -combinatorial_object_activity -save \n> -end

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

Edge_curve_17\_line\_type:
> echo $(FILE_Q17) >edge\_q17.csv
> $(ORBITER) -v 2 \n> -define F -finite_field -q 17 -end \n> -define R -polynomial_ring -field F \n> -number_of_variables 3 \n> -homogeneous_of_degree 4 \n> -end \n> -define P -projective_space -n 2 -field F -v 0 -end \n> -define C -geometric_object P \n> -projective_variety R \n> "Edge_q17" "Edge\_q17" \n> $(EDGE_CURVE_Q17_EQUATION) \n> -end \n> -with C -do \n> -combinatorial_object_activity \n> -line_type \n> -end \n> -print_symbols

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

Group Theory

5.1 Permutation Groups

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

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

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

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

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

Cyclic.6:
\[
\texttt{\$ (ORBITER) -v 3 \ }
\texttt{\textbackslash
}\texttt{\textbackslash
}\texttt{\textbackslash
}\texttt{\textbackslash
}\texttt{\define G -permutation_group -cyclic_group 6 -end \ }
\texttt{\textbackslash
}\texttt{\textbackslash
}\texttt{\textbackslash
}\texttt{\textbackslash
}\texttt{\with G -do \ }
\texttt{\textbackslash
}\texttt{\textbackslash
}\texttt{\textbackslash
}\texttt{\textbackslash
}\texttt{\-group_theoretic_activity \ }
\texttt{\textbackslash
}\texttt{\textbackslash
}\texttt{\textbackslash
}\texttt{\textbackslash
}\texttt{\-report \ }
\texttt{\textbackslash
}\texttt{\textbackslash
}\texttt{\textbackslash
}\texttt{\textbackslash
}\texttt{\-end}
\texttt{\textbackslash
}\texttt{\textbackslash
}\texttt{\textbackslash
}\texttt{\textbackslash
}\texttt{\pdflatex Perm6_report.tex}
\texttt{\textbackslash
}\texttt{\textbackslash
}\texttt{\textbackslash
}\texttt{\textbackslash
}\texttt{\open Perm6_report.pdf}
\]

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

Cyclic.6_group_table:
\[
\texttt{\$ (ORBITER) -v 3 \ }
\texttt{\textbackslash
}\texttt{\textbackslash
}\texttt{\textbackslash
}\texttt{\textbackslash
}\texttt{\define G -permutation_group -cyclic_group 6 -end \ }
\texttt{\textbackslash
}\texttt{\textbackslash
}\texttt{\textbackslash
}\texttt{\textbackslash
}\texttt{\with G -do \ }
\texttt{\textbackslash
}\texttt{\textbackslash
}\texttt{\textbackslash
}\texttt{\textbackslash
}\texttt{\-group_theoretic_activity \ }
\texttt{\textbackslash
}\texttt{\textbackslash
}\texttt{\textbackslash
}\texttt{\textbackslash
}\texttt{\-export_group_table \ }
\texttt{\textbackslash
}\texttt{\textbackslash
}\texttt{\textbackslash
}\texttt{\textbackslash
}\texttt{\-end}
\texttt{\$ (ORBITER) -v 2 \ }
\texttt{\textbackslash
}\texttt{\textbackslash
}\texttt{\textbackslash
}\texttt{\textbackslash
}\texttt{\define all_one_r -vector -repeat 1 6 -end \ }
\]

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

Symmetric_3:

```bash
$ ( ORBITER ) - v 3 \
```

```bash
- define G - permutation_group - symmetric_group 3 - end \
```

```bash
- with G - do \n```

```bash
- group_theoretic_activity \n```

```bash
- report \n```

```bash
- end \
```

```bash
pdflatex Perm3_report.tex \
```

```bash
open Perm3_report.pdf \
```

The report is shown below:

### Stabilizer chain

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

### Basic Orbit 0

![Diagram of Stabilizer chain]
Basic orbit 0 has size 3
0, 1, 2

Basic Orbit 1

1

Basic orbit 1 has size 2
1, 2

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

Symmetric.3_group_table:
▷ $(ORBITER) -v 3 \n▷ ▷ -define G -permutation_group -symmetric_group 3 -end \n▷ ▷ -with G -do \n▷ ▷ -group_theoretic_activity \n
Figure 5.2: The group table of Sym(3)
Figure 5.3: The elements of Sym(3)

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

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

Symmetric3_elements:
```
$ (ORBITER) -v 3
  -define G -permutation_group -symmetric_group 3 -end
  -with G -do
  -group_theoretic_activity
  -save_elements_csv "Symmetric3_elts.csv"
  -end
$ (ORBITER) -v 2
  -define Sym3_elts -vector -load_csv_data_column
  -Symmetric3_elts.csv 1 -end
  -save_matrix_csv Sym3_elts
$ (ORBITER) -v 2
  -define all_one_r -vector -repeat 1 6 -end
```
define all_one_c -vector -repeat 1 3 -end \
-draw_matrix \n\n-input_csv_file Sym3_elts_matrix.csv \n-box_width 50 -bit_depth 8 \n-partition 3 \n\n-all_one_r all_one_c \n-end

open Sym3_elts_matrix_draw.bmp
5.2 Linear Groups

Orbiter provides support for matrix groups and their various permutation representations. For background information about the classical groups of matrices over finite fields, see cf. [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
\[
M_k \cdot N_l = \left(M \cdot N^{\Phi^{-k}}\right)_{k+l}, \quad (5.2)
\]
\[
(M_k)^{-1} = \left((M^{-1})^{\Phi} \right)^{-k}. \quad (5.3)
\]

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

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

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

A correlation is a one-to-one mapping between the set of points and the set of hyperplanes which reverses incidence. So, if \( \rho \) is a correlation and \( P \) is a point and \( \ell \) is a hyperplane then \( P^\rho \) is a hyperplane and \( \ell^\rho \) is a point and
\[
\ell^\rho \in P^\rho \iff P \in \ell.
\]

A correlation of order two is called polarity. The standard polarity is the map
\[
\rho : \mathcal{P} \leftrightarrow \mathcal{L}, \; P(x) \leftrightarrow [x].
\]

A group \( G \) can act on \( V \) in one of the types listed in Table 5.1. One can create a matrix group over a finite field \( \mathbb{F}_q \) is created as described in in two steps. In the first step, the field \( \mathbb{F}_q \) is created as described in Sections 3.2 and 3.3. The field is stored in the symbol table. Then, the group is created using the symbolic label for the field. The basic types of matrix groups in Orbiter are listed in Table 5.2.

For instance,
<table>
<thead>
<tr>
<th>Type</th>
<th>Perm. Domain</th>
<th>Degree</th>
</tr>
</thead>
<tbody>
<tr>
<td>General linear GL((n, q))</td>
<td>all vectors of (V)</td>
<td>(q^n)</td>
</tr>
<tr>
<td>Affine AGL((n, q))</td>
<td>all vectors of (V)</td>
<td>(q^n)</td>
</tr>
<tr>
<td>Projective PGL((n, q))</td>
<td>(\mathfrak{S}r_1(V))</td>
<td>(q^n - 1)</td>
</tr>
<tr>
<td>Wreath product GL((d, q) \wr \text{Sym}(n))</td>
<td>(\mathfrak{S}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\(_4.2\):
▷ \$(\text{ORBITER}) \ -v \ 2 \$
▷▷ -define F -finite_field -q 2 -end
▷▷ -define G -linear_group -PGL 4 F -end
▷▷ -with G -do
▷▷ -group_theoretic_activity
▷▷▷ -report
▷▷ -end
▷ pdflatex PGL_4.2_report.tex
▷ open PGL_4.2_report.pdf

creates the group PGL\((4, 2)\) acting on the 15 elements of \(\mathfrak{S}r_1(\mathbb{F}_2^4)\). At first, the field \(\mathbb{F}_2\) is created. Secondly, the group \(G = \text{PGL}(3, 2)\) is created using the previously created field \(\mathbb{F}_2\), and a report is generated. The report gives information about the permutation group action, including the underlying field and the projective geometry.

The Group PGL\((4, 2)\)

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

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

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

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

The Action

Group action $\text{PGL}(4,2)$ 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 $\text{PG}(3,2)$

$q = 2$
$p = 2$
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>

\[ + \]

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

\[ \cdot \]

<table>
<thead>
<tr>
<th>( \cdot )</th>
<th>( 1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ 1^0 \equiv 1 \]
\[ 1^1 \equiv 1 \]

Base and Stabilizer Chain

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

\[
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1
\end{pmatrix},
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix},
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix},
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix},
\begin{pmatrix}
1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1,
\end{pmatrix}
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:
\[
G := \mathrm{Group}( [(4, 10)(5, 15)(11, 12)(13, 14),
(4, 11)(5, 14)(10, 12)(13, 15),
(4, 13)(5, 12)(10, 14)(11, 15),
(3, 4)(7, 10)(8, 11)(9, 12),
(2, 3)(6, 7)(11, 13)(12, 14),
(1, 2)(7, 8)(10, 11)(14, 15)] );
\]

Magma export:

\[
G := \mathrm{GeneralLinearGroup}(4, \mathrm{GF}(2));
H := \mathrm{sub}< G | [1,0,0,0, 0,1,0,0, 0,0,1,0, 1,0,0,1],
[1,0,0,0, 0,1,0,0, 0,0,1,0, 0,1,0,1],
[1,0,0,0, 0,1,0,0, 0,0,1,0, 0,0,1,1],
[1,0,0,0, 0,1,0,0, 0,0,0,1, 0,0,1,0],
[1,0,0,0, 0,0,1,0, 0,1,0,0, 0,0,0,1],
[0,1,0,0, 1,0,0,0, 0,0,1,0, 0,0,0,1] >;
\]

Compact form:

Generators in compact permutation form are:
\[
6 15
0 1 2 9 14 5 6 7 8 3 11 10 13 12 4
0 1 2 10 13 5 6 7 8 11 3 9 14 4 12
0 1 2 12 11 5 6 7 8 13 14 4 3 9 10
0 1 3 2 4 5 9 10 11 6 7 8 12 13 14
\]
The base has length 4
The basic orbits are:
Basic orbit 0 is orbit of 0 of length 15
Basic orbit 1 is orbit of 1 of length 14
Basic orbit 2 is orbit of 2 of length 12
Basic orbit 3 is orbit of 3 of length 8

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

PGL.4.2.export:
▷ $(ORBITER) -v 2 \\
▷ ▷ -define F -finite_field -q 2 -end \\
▷ ▷ -define G -linear_group -PGL 4 F -end \\
▷ ▷ -with G -do \\
▷ ▷ -group_theoretic_activity \\
▷ ▷ ▷ -report \\
▷ ▷ ▷ -end \\
▷ ▷ -with G -do \\
▷ ▷ -group_theoretic_activity \\
▷ ▷ ▷ -export_orbiter \\
▷ ▷ ▷ -end \\
▷ pdflatex PGL.4.2_report.tex \\
▷ open PGL.4.2_report.pdf

The file PGL_4_2.makefile is created:

PGL.4.2.generated:
▷ $(ORBITER) -v 2 \\
▷ ▷ -define gens -vector -file PGL.4.2.gens.csv -end \\
▷ ▷ -define G -permutation_group \\
▷ ▷ -bsgs PGL.4.2 "{\rm PGL}(4,2)" 15 20160 "0,1,2,3" 6 gens -end \\

This command can be used to recreate the group as permutation group directly. This group will be considered again in Section 5.2 below. The permutation representation itself is stored in the file PGL.4.2.gens.csv:

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

L\_5\_3:

\>$\text{L5}_3$:
  \>$\text{((ORBITER) -v 2 \text{ \backslash})}$
  \>$\text{define F \text{-finite_field -q 3 -end \text{ \backslash})}$
  \>$\text{define G \text{-linear_group -PSL 5 F -end \text{ \backslash})}$
  \>$\text{with G \text{-do \text{ \backslash})}$
  \>$\text{-group_theoretic_activity \text{ \backslash})}$
  \>$\text{\text{-report \text{ \backslash})}$
  \>$\text{\text{-end \text{ \backslash})}$
  \>$\text{pdflatex PSL5_3_report.tex}$
  \>$\text{open PSL5_3_report.pdf}$

creates PSL(5, 3) of order 237783237120.

The command

PSP\_4\_4:

\>$\text{PS4}_4$:
  \>$\text{((ORBITER) -v 2 \text{ \backslash})}$
  \>$\text{define F \text{-finite_field -q 4 -end \text{ \backslash})}$
  \>$\text{define G \text{-linear_group -PGL 4 F \text{ \backslash})}$
  \>$\text{symplectic_group \text{ \backslash})}$
  \>$\text{-end \text{ \backslash})}$
  \>$\text{with G \text{-do \text{ \backslash})}$
  \>$\text{-group_theoretic_activity \text{ \backslash})}$
  \>$\text{-report \text{ \backslash})}$
  \>$\text{-end \text{ \backslash})}$
  \>$\text{pdflatex PGL4_4_Sp4_4_report.tex}$
  \>$\text{open PGL4_4_Sp4_4_report.pdf}$

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

The command

PGO\_5\_2:

\>$\text{PG2}_5$:
  \>$\text{((ORBITER) -v 2 \text{ \backslash})}$
creates the group PGO(5, 2) acting on the 15 points of the \(Q(4, 2)\) quadric. The following latex report is produced:

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

The order of the group PGO(5, 2) is 720
The group acts on a set of size 15
Strong generators for a group of order 720:

\[
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 1
\end{bmatrix},
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 \\
1 & 0 & 0 & 1 & 1
\end{bmatrix},
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 1 \\
1 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 1
\end{bmatrix},
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 0
\end{bmatrix},
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 0
\end{bmatrix},
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1
\end{bmatrix},
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 1 \\
0 & 1 & 1 & 1 & 0
\end{bmatrix},
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 1
\end{bmatrix},
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 1
\end{bmatrix},
\begin{bmatrix}
1,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,1,0,1,0,0,1,1, \\
1,0,0,0,0,1,0,0,0,0,0,1,0,0,1,1,1,1,1,1,1,1,0,1, \\
1,0,0,0,0,0,1,0,0,0,1,1,1,1,1,1,1,1,1,1,0,0,0,1, \\
1,0,0,0,0,0,1,0,0,0,0,1,0,1,1,1,1,1,1,1,1,1,1,0,1,0, \\
1,0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,1, \\
1,0,0,0,0,0,0,0,0,1,0,1,0,1,0,0,0,0,0,1,0,0,0,0,0,1,
The Action

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

0 = (0, 1, 0, 0, 0) 8 = (0, 1, 1, 1, 1)
1 = (0, 0, 1, 0, 0) 9 = (1, 1, 1, 0, 0)
2 = (0, 0, 0, 1, 0) 10 = (1, 1, 1, 1, 0)
3 = (0, 1, 0, 1, 0) 11 = (1, 1, 1, 0, 1)
4 = (0, 0, 1, 1, 0) 12 = (1, 0, 0, 1, 1)
5 = (0, 0, 0, 0, 1) 13 = (1, 1, 0, 1, 1)
6 = (0, 1, 0, 0, 1) 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 $\mathbb{F}_2$

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

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

+ 0 1
0 0 1
1 1 0

$\cdot$ 1
1 1

137
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:

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

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:
\[ \begin{align*}
6 & 15 \\
0 & 1 2 3 4 12 13 14 11 9 10 8 5 6 7 \\
0 & 1 12 13 14 5 6 7 10 9 8 11 2 3 4 \\
0 & 11 13 12 4 5 6 9 8 7 10 1 3 2 14 \\
0 & 7 13 12 10 3 2 8 9 11 4 14 5 6 1 \\
9 & 1 2 10 4 5 11 7 13 0 3 6 12 8 14 \\
6 & 1 4 8 2 5 0 7 3 11 13 9 14 10 12 \\
-1
\end{align*} \]

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 \( \text{PSp}(6, 2) \) can be created using the following command:

\[
PSP\_6\_2: \begin{align*}
\texttt{\$((ORBITER) -v 2 \ \} \\
\texttt{\& \& -define F -finite_field -q 2 -end \} \\
\texttt{\& -define G -linear_group -PGL 6 F \} \\
\texttt{\& -define G -symplectic_group \} \\
\texttt{\& -end \} \\
\texttt{\& -with G -do \} \\
\texttt{\& -group_theoretic_activity \} }
\end{align*}
\]
The group $\text{PGO}(7,2)$, isomorphic to $\text{PSp}(6,2)$, can be created using the following command:

```
PGO_7_2:
  ▶ $(\text{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
```
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)
```

**Table 5.3: Commands for creating subgroups**

<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-Janko1</td>
<td></td>
<td>first Janko group, needs PGL(7,11)</td>
</tr>
<tr>
<td>-monomial</td>
<td></td>
<td>subgroup of monomial matrices</td>
</tr>
<tr>
<td>-diagonal</td>
<td></td>
<td>subgroup of diagonal matrices</td>
</tr>
<tr>
<td>-null_polarity_group</td>
<td></td>
<td>null polarity group</td>
</tr>
<tr>
<td>-symplectic_group</td>
<td></td>
<td>symplectic group</td>
</tr>
<tr>
<td>-singer</td>
<td>k</td>
<td>subgroup of index $k$ in the Singer cycle</td>
</tr>
<tr>
<td>-singer_and_frobenius</td>
<td>k</td>
<td>subgroup of index $k$ in the Singer cycle, extended by the Frobenius automorphism of $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$</td>
</tr>
<tr>
<td>-borel_upper</td>
<td></td>
<td>Borel subgroup of upper triangular matrices</td>
</tr>
<tr>
<td>-borel_lower</td>
<td></td>
<td>Borel subgroup of lower triangular matrices</td>
</tr>
<tr>
<td>-identity_group</td>
<td></td>
<td>identity subgroup</td>
</tr>
<tr>
<td>-subgroup_from_file</td>
<td>$f\ l$</td>
<td>read subgroup from file $f$ and give it the label $l$</td>
</tr>
<tr>
<td>-orthogonal</td>
<td>$\epsilon$</td>
<td>orthogonal group $O^\epsilon(n,q)$, with $\epsilon \in {\pm 1}$ when $n$ is even</td>
</tr>
<tr>
<td>-subgroup_by_generators</td>
<td>$l \ o \ n \ s_1 \ldots s_n$</td>
<td>Generate a subgroup from generators. The label “$l$” is used to denote the subgroup; $o$ is the order of the subgroup; $n$ is the number of generators and $s_1, \ldots, s_n$ are the generators for the subgroup in vector form.</td>
</tr>
</tbody>
</table>

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

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:

\[
\begin{verbatim}
$(ORBITER) -v 2 \\
-define gens -vector -dense $(GEN_C13) -end \\
-define G -permutation_group \\
-bsgs C13 C_{13} 13 13 0 1 \\
gens \\
-end \\
-with G -do \\
-group_theoretic_activity \\
-export_orbiter \\
-end \\
-with G -do \\
-group_theoretic_activity \\
-export_group_table \\
-end \\
-with G -do \\
-group_theoretic_activity \\
-report \\
-end \\
-with G -do \\
-group_theoretic_activity \\
-save_elements_csv "C13_elts.csv" \\
-end
\end{verbatim}
\]
The activity -report produces a report for the cyclic group, shown below:

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:

<table>
<thead>
<tr>
<th>Row</th>
<th>Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>&quot;0,1,2,3,4,5,6,7,8,9,10,11,12&quot;</td>
</tr>
<tr>
<td>1</td>
<td>&quot;1,2,3,4,5,6,7,8,9,10,11,12,0&quot;</td>
</tr>
<tr>
<td>2</td>
<td>&quot;2,3,4,5,6,7,8,9,10,11,12,0,1&quot;</td>
</tr>
<tr>
<td>3</td>
<td>&quot;3,4,5,6,7,8,9,10,11,12,0,1,2&quot;</td>
</tr>
</tbody>
</table>
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 $\text{Sym}(13)$.

\begin{verbatim}
C13_as_subgroup:
  $(ORBITER) -v 2 \\
  -define G -permutation_group -symmetric_group 13 \\
  -subgroup_by_generators C13 13 1 $(GEN_C13) -end \\
  -with G -do \\
  -group_theoretic_activity \\
  -export_orbiter \\
  -with G -do \\
  -group_theoretic_activity \\
  -report \\
  -end \\
  -with G -do \\
  -group_theoretic_activity \\
  -save_elements_csv "C13_elts.csv" \\
  -end \\
#pdflatex Perm13_Subgroup_C13_13_report.tex 
#open Perm13_Subgroup_C13_13_report.pdf
\end{verbatim}

The \texttt{subgroup\_by\_generators} command will be discussed in more detail in Section 5.3.

For instance, the command

\begin{verbatim}
J1:
  $(ORBITER) -v 2 \\
  -define G -linear_group -PGL 7 11 -Janko1 -end \\
  -with G -do \\
  -group_theoretic_activity \\
  -report \\
\end{verbatim}
The command

PGL\_3\_11\_singer:
\begin{verbatim}
$\text{(ORBITER)} -v 2 \ \\
$define G -linear_group -PGL 3 11 -singer 19 -end \ \\
$-with G -do \ \\
$-group_theoretic_activity \ \\
$report \ \\
-end
\end{verbatim}

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:
\begin{verbatim}
$\text{(ORBITER)} -v 2 \ \\
$define G -linear_group -PGL 3 11 -singer_and_frobenius 19 -end \ \\
-with G -do \ \\
-group_theoretic_activity \ \\
report \ \\
-end
\end{verbatim}

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 \(\mathbb{F}_{3^{11}}\) 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
quatertion:
▷ $\text{(ORBITER) -v 2} \$
▷ ▷ -define G -linear_group -SL 2 3 \$
▷ ▷ -subgroup_by_generators "quatertion" "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
▷ pdslatex GL_2.3_Subgroup_quatertion_8_elements.tex
▷ open GL_2.3_Subgroup_quatertion_8_elements.pdf
▷ pdslatex GL_2.3_Subgroup_quatertion_8_report.tex
▷ open GL_2.3_Subgroup_quatertion_8_report.pdf

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

<table>
<thead>
<tr>
<th>Element 0 / 8 of order 1:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\begin{bmatrix} 1 &amp; 0 \ 0 &amp; 1 \end{bmatrix}$</td>
</tr>
<tr>
<td>$(0)(1)(2)(3)(4)(5)(6)(7)(8)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Element 1 / 8 of order 4:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\begin{bmatrix} 2 &amp; 1 \ 1 &amp; 1 \end{bmatrix}$</td>
</tr>
<tr>
<td>$(0)(1,5,2,7)(3,4,6,8)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Element 2 / 8 of order 2:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\begin{bmatrix} 2 &amp; 0 \ 0 &amp; 2 \end{bmatrix}$</td>
</tr>
<tr>
<td>$(0)(1,2)(3,6)(4,8)(5,7)$</td>
</tr>
</tbody>
</table>
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:

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

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

The tetrahedral subgroup can be created as well:

tetra_group:

- $(ORBITER) -v 3 \$
- -define G -linear_group -GL 3 3 \
- -subgroup_by_generators "tetra" "12" 2 \
  "0,1,0,0,0,1,1,0,0, 0,0,1,2,0,0,0,2,0" \
- -end \
- -with G -do \
  -group_theoretic_activity \
  -print_elements.tex \
  -report \
- -end

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

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

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

```
pdflatex PGGL 3 4 Subgroup_Hesse_432_report.tex
open PGGL_3_4_Subgroup_Hesse_432_report.pdf
```

The group has order 432.

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

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

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


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

test_subgroup:

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

```
coset_reps:
▷ $(ORBITER) -v 2 
▷ ▷ -define F -finite_field -q 2 -end 
▷ ▷ -define G1 -linear_group -PGOp 6 F -end 
▷ ▷ -define G2 -linear_group -PGL 6 F 
▷ ▷ ▷ -symplectic_group 
▷ ▷ -end 
▷ ▷ -with G1 -and G2 -do 
▷ ▷ -group_theoretic_activity 
▷ ▷ ▷ -coset_reps 
▷ ▷ -end
▷ pdflatex PGOp_6_2.coset_reps.tex
▷ open PGOp_6_2.coset_reps.pdf
```

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

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

```
::
```
```
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.

```
coset_reps_read:
```

153
$(\text{ORBITER}) -v 2 \\
$\text{define F -finite_field -q 2 -end} \\
$\text{define G1 -linear_group -PGOp 6 F -end} \\
$\text{define G2 -linear_group -PGL 6 F} \\
$\text{define symplectic_group} \\
$\text{define CR -vector_ge -action G2} \\
$\text{read_csv} \\
$\text{PGOp6_2_coset_reps.csv Element} \\
$\text{end}$
5.4 Linear Groups, Advanced Topics

It is sometimes necessary to control the finite field that is used in the construction of a matrix group. For prime fields, this is not an issue. For extension fields, the choice of polynomial does matter, as the generators depend on specific choices made for the finite field. Magma and GAP use Conway polynomials, which are difficult to compute. Orbiter has a built-in table of primitive polynomials. As explained in Section 3.3, Orbiter allows to specify the polynomial that should be used to create the finite field. The next example shows an instance where choosing the polynomial is important. We are recreating a group from the electronic Atlas on finite simple groups [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
U_3.3:

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

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

Group theoretic activities will be discussed in Section 5.6.

As an example of a large group, consider the Conway group Co3. Following [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 `-subgroup_by_generators` command. Finally, we create a report for the group.

```
CONWAY_GEN1="\n  110111000100001010000\n  1111010111110100001011\n  000001000000100010101\n  111110111010001001110\n  0101010000000010011101\n  00000000000010010101\n  001000000000010010101\n  0001000000000000010000101\n  001100100101100010011\n  00000000000000001100101\n  00000000000000000100101\n  00000000010000000100101\n  00000000010000000100101\n  00000000000001000000101\n  00000000000001000000001\n  00000000000000000100111\n  0000000000000001001001\n  0000000000000000010001\n  0000000000000000010001\n  0000000000000000010001\n  0000000000000000010001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001\n  00000000000000000100001
```

156
CONWAY_GEN2="\
0101000010111010111111\n0110010100011110110000\n0011010000111111010111\n0001101110010110100111\n101001000100010111110\n1101000000010101000111\n110010101001111010101\n1000110100110101010101\n01001100101000000111\n1100000101001011001010\n0101110110011000000101\n0101111101010011110101\n1100010101010101010001\n000101000011100100111\n0011010010111011001111\n0100110010110011111010\n1101011001111101100011\n010010010010001000111\n0000001101111000101110\n11011010101110000101"

Co3:
▷ $(ORBITER) -v 2 \
▷   -define F -finite_field -q 2 -end \
▷   -define g1 -vector -field F -format 22 -compact $(CONWAY_GEN1) -end \
▷   -define g2 -vector -field F -format 22 -compact $(CONWAY_GEN2) -end \
▷   -define gens -vector -concatenate g1 -concatenate g2 -end \
▷   -define G -linear_group -PGL 22 2 \
▷     -subgroup_by_generators "Co3" "4957666656000" 2 gens \
▷     -end \
▷     -with G -do \
▷     -group_theoretic_activity \
▷     -report \
▷     -end 
▷ pdflatex PGL_22_2_Subgroup_Co3_4957666656000_report.tex
▷ open PGL_22_2_Subgroup_Co3_4957666656000_report.pdf
The next example creates the Ree group in 7 dimensions over the field $\mathbb{F}_{27}$. Again, we use makefile variables to specify the two generators as $7 \times 7$ matrices over $\mathbb{F}_{27}$ and concatenate them, before passing them to the `-subgroup_by_generators` command.

```plaintext
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, \n21,21,6,18,20,2,5"

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

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

158
<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 $\text{PGL}(2,q)$ on the conic in the plane $\text{PG}(2,q)$</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 $\text{GL}(d,q) \wr \text{Sym}(n)$ on the tensor space</td>
</tr>
<tr>
<td>-on_rank_one_tensors</td>
<td></td>
<td>induced action of $\text{GL}(d,q) \wr \text{Sym}(n)$ 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:
$>$ $(\text{ORBITER})$ -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 $\text{GL}(2,2) \wr \text{Sym}(3)$ acting on the 255 elements of $\text{PG}(7,2)$ which are identified with the tensors of type $(2,2,2)$ over $\mathbb{F}_2$. Elements of this group are denoted in the notation of the semidirect product. A vector of elements in the linear group is followed by a permutation of the components.
The Group GL(2, 2) o Sym(3)
The order of the group GL(2, 2) o Sym(3) is 1296
The group acts on a set of size 255

The Action
Group action GL(2, 2) o Sym(3)res255 of degree 255

Base and Stabilizer Chain
Group order 1296
tl=3, 2, 1, 3, 2, 3, 2, 3, 2,
Strong generators for a group of order 1296.
"
#"
#"
#
! "
#"
#"
#
! "
#"
#"
#
!
10
10
10
10
10
01
10
10
10
; id ,
; id ,
; id ,
01
01
11
01
01
10
01
11
01
"

10
01

#"

01
10

#"

#
! "
#"
#"
#
! "
#"
#"
#
!
10
10
10
10
01
10
10
; id ,
; id ,
; id ,
01
11
01
01
10
01
01
"
#"
#"
#
! "
#"
#"
#
!
10
10
10
10
10
10
; (1, 2) ,
; (0, 1)
01
01
01
01
01
01

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

160


Stabilizer chain

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

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

T3r1:

```
$\text{(ORBITER)} -v 4 \n
-define G \n
-linear_group -GL_d_q_wr_Sym_n 2 2 3 \n
-on_rank_one_tensors -end \n
-with G -do \n
-group_theoretic_activity \n
-do -report \n
-end \n
pdflatex GL_2_2_wreath_Sym3_report.tex \n
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 $\text{GL}(2,2) \wr \text{Sym}(3)_{\res 27}$ of degree 27
We act on the following set:

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

Base and Stabilizer Chain

Group order 1296
tl=3,2,1,3,2,3,2,3,2,

Strong generators for a group of order 1296:

\[
\begin{align*}
&\left( \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} ; id \right), \\
&\left( \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} ; id \right), \\
&\left( \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} ; id \right), \\
&\left( \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} ; id \right), \\
&\left( \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} ; (1,2) \right), \\
&\left( \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} ; (0,1) \right)
\end{align*}
\]
The group of a conic is isomorphic to the group of the projective line. This isomorphism from the group of the projective line to the group of the conic can be realized using the command `PGL2OnConic`. The action is changed using the induced action on the Veronese variety. The group elements are still represented as $2 \times 2$ matrices. Here is an example. We create the collineation group $\Gamma L(2,8)$ of $\Gamma G(1,8)$ and act on $\Gamma G(2,8)$:

```
PGL2.8_on_conic:
\$\$(ORBITER) \ -v \ 4 \$
\$\$ \ -define \ G \$
\$\$ \ -linear_group \ -PGGL \ 2 \ \ 8 \ -PGL2OnConic \ -end \$
\$\$ \ -with \ G \ -do \$
\$\$ \ -group_theoretic_activity \$
\$\$ \ -report \$
\$\$ \ -end
\$\$ pdflatex \ PGL2.8_on_conic
\$\$ open \ PGL2.8_on_conic.pdf
```

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

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

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

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

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

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

\[
\begin{align*}
0 &= (1, 0, 0) \\
1 &= (0, 1, 0) \\
2 &= (0, 0, 1) \\
3 &= (1, 1, 1) \\
4 &= (1, 1, 0) \\
5 &= (2, 1, 0) \\
\vdots \\
72 &= (7, 7, 1)
\end{align*}
\]

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

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

polynomial: \( X^3 + X^2 + 1 = 13 \)

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

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

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

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

**Base and Stabilizer Chain**

Group order 1512

tl=9, 8, 7, 3,

**Stabilizer chain**

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

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

Basic Orbit 1

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

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

```
SURFACE_q13_STAB="1,0,0,0,12,0,0,0,12,0,0,0,1,\n1,0,0,0,12,0,0,0,0,1,0,0,0,12,\n1,0,0,0,0,12,0,0,12,0,0,0,0,1,\n0,1,0,0,1,0,0,0,0,1,0,0,0,0,1"
```

```
surface_q13_stab_on_tritangents_orbits:
  > $(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 S -cubic_surface -space P -arc_lifting "0,1,2,3,43,113" -end \n  >   -with S -do \n  >     -cubic_surface_activity \n  >     -report_with_group \n  >   -end \n  >   -with S -do \n  >     -cubic_surface_activity \n  >     -export_tritangent_planes \n  >   -end
  > $(ORBITER) -v 2 \n  >   -orbiter_path $(ORBITER_PATH) \n  >   -define TriP -set -file \n  >     family_Eckardt_q13_a2_b1_tritangent_planes.csv \n  >   -end \n  >   -define G -linear_group -PGL 4 13 \n  >     -subgroup_by_generators "SURF_STAB" \n  >   -24 4 $(SURFACE_q13_STAB) \n  >   -end \n  >   -define G_on_planes -modified_group -from G \n  >     -on_k_subspaces 3 \n  >   -end \n  >   -define Gr -modified_group -from G_on_planes \n  >     -restricted_action TriP \n  >   -end \n  >   -with Gr -do \n  >     -group_theoretic_activity \n  >     -report \n  >   -end \n  >   -define Orb -orbits -group Gr \n```

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
```

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:

```
170
```
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-apply</td>
<td>a s</td>
<td>Applies the group element given by the coded vector s to the element a.</td>
</tr>
<tr>
<td>-multiply</td>
<td>s₁ s₂</td>
<td>Multiplies group elements s₁ and s₂, assuming the elements are given in coded form. Produces a latex report.</td>
</tr>
<tr>
<td>-inverse</td>
<td>s</td>
<td>Computes the inverse of s, which is given in coded form. Produces a latex report.</td>
</tr>
<tr>
<td>-consecutive_powers</td>
<td>s n</td>
<td>Computes all powers sᵢ for i = 1, ..., n. s is given in coded form. Produces a latex report.</td>
</tr>
<tr>
<td>-raise_to_the_power</td>
<td>s n</td>
<td>Computes the n-th power of of s, which is given in coded form. Produces a latex report.</td>
</tr>
<tr>
<td>-export_orbiter</td>
<td></td>
<td>Exports the group to Orbiter.</td>
</tr>
<tr>
<td>-export_gap</td>
<td></td>
<td>Exports the group to GAP [29].</td>
</tr>
<tr>
<td>-export_magma</td>
<td></td>
<td>Exports the group to Magma [14].</td>
</tr>
<tr>
<td>-search_element_of_order</td>
<td>i</td>
<td>Finds all elements of order i in the group (i ∈ 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>g₁...gₙ</td>
<td>Creates a table of the orders of all products gᵢgⱼ, 1 ≤ i, j ≤ n.</td>
</tr>
</tbody>
</table>

Table 5.5: Group theoretic activities (Part 1)
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-classify_arcs</td>
<td>description</td>
<td>Classify arcs in geometries. See Section 6.6.</td>
</tr>
<tr>
<td>-linear_codes</td>
<td>$d \ n_{\text{max}}$</td>
<td>Classify linear codes with prescribed minimum distance $d$. Assumes that the group is $\text{PGL}(r,q)$ or $\text{PGL}(r,q)$. 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 $\text{PGO}^+(6,q)$ as $6 \times 6$ matrixes, compute the inverse image of the generators in $\text{PGL}(4,q)$ (if possible).</td>
</tr>
<tr>
<td>-classify_cubic_curves</td>
<td>descr</td>
<td>Classifies cubic curves. Expects an arc description options as in Table 6.4.</td>
</tr>
</tbody>
</table>

Table 5.6: Group theoretic activities (Part 2)

Figure 5.4: The elements of $\text{Sym}(3)$ in lex-order
Cycle_{12\_power}:
\[
\text{	exttt{\$ (ORBITER) -v 5 \textbackslash
\texttt{ -define G -permutation\_group -symmetric\_group 12 -end \textbackslash
\texttt{ -with G -do \textbackslash
\texttt{ -group\_theoretic\_activity \textbackslash
\texttt{ -consecutive\_powers \textbackslash
\texttt{ "1,2,3,4,5,6,7,8,9,10,11,0" 12 \textbackslash
\texttt{ -end\textbackslash
\texttt{ pdflatex Perm12\_all\_powers.tex \textbackslash
\texttt{ open Perm12\_all\_powers.pdf}}
\]

We create the 12 powers of the cycle

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

The output is

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

The command
\[
PGL_{3.4\_singer}:
\]
\[
\text{	exttt{\$ (ORBITER) -v 5 \textbackslash
\texttt{}}}
\]

173
finds all Singer cycles in PGL(3, 4) whose matrix is the companion matrix of a polynomial. The first one found is

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

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

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

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

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:
```
\$ (ORBITER) -v 5 \n\$ \$define G \-linear_group \-GL 2 7 \-end \n\$ \$with G \-do \n\$ \$group_theoretic_activity \n\$ \$ \$-multiply "0,1,2,3" "4,5,6,0" \n\$ \$-end \n\$ pdflatex GL\_2\_7\_mult.tex
\$ open GL\_2\_7\_mult.pdf
```

The output is
```
\[
\begin{bmatrix}
0 & 1 \\
2 & 3
\end{bmatrix}
\cdot
\begin{bmatrix}
4 & 5 \\
6 & 0
\end{bmatrix}
= 
\begin{bmatrix}
6 & 0 \\
5 & 3
\end{bmatrix}
\]
0,1,2,3,
4,5,6,0,
6,0,5,3,
```

We can compute the inverse of a group element:

GL\_2\_7\_inv:
```
\$ (ORBITER) -v 5 \n\$ \$define G \-linear_group \-GL 2 7 \-end \n\$ \$with G \-do \n\$ \$group_theoretic_activity \n\$ \$ \$-inverse "0,1,2,3" \n\$ \$-end \n\$ 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:

\textbf{GL}_2.7\textunderscore\textit{power}:
\begin{itemize}
  \item \$\text{(ORBITER)} \-v 5 \$
  \item \$\text{-define G \text{-linear\_group }-GL 2 7 \-end }\$
  \item \$\text{-with G \-do }\$
  \item \$\text{-group\_theoretic\_activity }\$
  \item \$\text{-raise\_to\_the\_power }"0,1,2,3" \ 2 \$
  \item \$\text{-end }\$
\end{itemize}

\texttt{pdflatex GL\textunderscore2.7\textunderscore power.tex}
\texttt{open GL\textunderscore2.7\textunderscore 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.

\textbf{on\_planes}:
\begin{itemize}
  \item \$\text{(ORBITER)} \-v 2 \$
  \item \$\text{-define F \text{-finite\_field }-q 8 \-end }\$
  \item \$\text{-define P \text{-projective\_space }-n 3 \text{-field F }-v 0 \-end }\$
  \item \$\text{-define G \text{-linear\_group }-PGL 4 F \-end }\$
  \item \$\text{-define G\_on\_planes \text{-modified\_group }-\text{-from G }\$
  \item \$\text{-on_k\_subspaces 3 }\$
  \item \$\text{-end }\$
  \item \$\text{-with G\_on\_planes \-do }\$
  \item \$\text{-group\_theoretic\_activity }\$
  \item \$\text{-apply }"0,8,1,6,4,3,7,2,5" \$
  \item \$\text{"1,0,0,0, 0,1,0,0, 0,0,0,2, 0,0,1,1" }\$
  \item \$\text{-end }\$
\end{itemize}

\texttt{pdflatex PGL\_4.8\_Gr\_4.3\_apply.tex}
\texttt{open PGL\_4.8\_Gr\_4.3\_apply.pdf}

The output is

176
\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & \gamma \\
0 & 0 & 1 & 1
\end{bmatrix} = \begin{bmatrix}
1000 \\
0100 \\
0002 \\
0011
\end{bmatrix}
\]

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

178
computes the classes of elements in $\text{PGL}(2, 4)$ using Orbiter-Magma-Orbiter. The first Orbiter command produces the file `PGGL_2_4_classes.magma`. The magma command reads this file and produces the file `PGGL_2_4_classes_out.txt`. The second Orbiter command reads the file `PGGL_2_4_classes_out.txt` and produces the latex report `PGGL_2_4_classes_out.tex`.

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

```
Order of element = 2
Class size = 10
Centralizer order = 12
Normalizer order = 12
Representing element is
   $c_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

of order 2 and with 3 fixed points. 0, 1, 1, 0, 1,
The normalizer is generated by:
Strong generators for a group of order 12:
   $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $\begin{bmatrix} \omega^2 & 0 \\ 0 & 1 \end{bmatrix}$, $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
1,0,0,1,1,
1,0,0,2,1,
0,1,1,0,1,
```

The command sequence

```
PGGL_2_4_cent_2A:
  $\text{$(ORBITER)$ -v 3 \ $
  \text{\indent -define G \ $
  \text{\indent -linear_group -PGGL 2 4 -end \ $
  \text{\indent -with G -do \ $
  \text{\indent -group_theoretic_activity \ $
```

179
computes the centralizer of the Baer involution
\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}.
\]
The centralizer is a group of order 40320, isomorphic to \( \text{PGL}(4,2).Z_2 \). Orbiter produces a list of strong generators, shown below:

Strong generators for a group of order 40320:

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

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

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 \  
    -define G -permutation_group -symmetric_group 13 \  
    -subgroup_by_generators H5 5 1 \  
    $(GENERATORS_H5) -end \  
    -with G -do \  
    -group_theoretic_activity \  
    -normalizer \  
    -end  
```

computes the normalizer of $H$ in 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;
```

181
The Orbiter command has to be run one more time after that. The following report is produced:

The group \( Perm_{13} \) Subgroup \( H \) of order 5 is:
Strong generators for a group of order 5:

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

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

Strong generators for a group of order 1200:

\[(11, 12),\]

\[(10, 11),\]

\[(5, 9, 8, 7, 6),\]

\[(1, 2, 4, 3)(6, 7, 9, 8),\]

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

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

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

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

\begin{verbatim}
Normalizer_of_Z22_in_PGL_2_9:
  $ (ORBITER) -v 2 \n  -define G -linear_group -PGL 2 9 \n  -subgroup_by_generators Z22 4 2 \n    -"2,0,0,1, 0,1,1,0" -end \
    -with G -do \n  -group_theoretic_activity \n  -normalizer \
\end{verbatim}
It produces a report showing that the normalizer is a group of order 24 (it is isomorphic to \( \text{Sym}(4) \), though the report does not tell us this fact directly):

The group \( \text{PGL}(2,9) \), subgroup \( \mathbb{Z}_{22} \), 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 \
>   -define G -linear_group -PGL 4 2 -end \
>   -define Orb -orbits -group G \
>   -on_points \
> -end
> $(ORBITER) -v 3 \
>   -draw_layered_graph \
>   -PGL_4_2_0.layered_graph \
>   -radius 500 -spanning_tree -embedded \
>   -line_width 1.1 -x_stretch 1.4 -scale 0.25 \
> -end
> pdflatex PGL_4_2_0.draw.tex
> open PGL_4_2_0.draw.pdf
> pdflatex PGL_4_2.orbits_report.tex
> open PGL_4_2.orbits_report.pdf
```
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-orbits_on_subsets</td>
<td>$k$</td>
<td>Compute orbits on $k$-subsets.</td>
</tr>
<tr>
<td>-orbits_on_points</td>
<td></td>
<td>Compute orbits in the action that was created.</td>
</tr>
<tr>
<td>-orbits_of</td>
<td>$i$</td>
<td>Compute orbit of point $i$ in the given action.</td>
</tr>
<tr>
<td>-stabilizer</td>
<td></td>
<td>Compute the stabilizer of the orbit representative (needs -orbits_on_points).</td>
</tr>
<tr>
<td>-orbits_on_set_system_from_file</td>
<td>$fname$ $f$ $l$</td>
<td>Reads the csv file “fname” and extract sets from columns $[f, ..., f + l - 1]$.</td>
</tr>
<tr>
<td>-orbit_of_set_from_file</td>
<td>$fname$</td>
<td>Reads a set from the text file “fname” and computes orbits on the elements of the set.</td>
</tr>
<tr>
<td>-orbits_on_polynomials</td>
<td>$d$</td>
<td>Computes the orbits of the matrix group on homogeneous polynomials of degree $d$. The number of variables is determined by the degree of the matrix group.</td>
</tr>
<tr>
<td>-conjugacy_class_of</td>
<td>label $s$</td>
<td>Compute the conjugacy class of the group element encoded as $s$ using the given label for file-names. Write a file containing the ranks for all elements in the class. Writes a second file containing the transporter elements for each element in the class. A transporter element maps the class representative to the given element under conjugation.</td>
</tr>
<tr>
<td>-orbits_on_group_elements_under_conjugation</td>
<td>$fname$-C $fname$-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 $fname$-C and $fname$-T are the files containing the elements of the class and the transporter, respectively.</td>
</tr>
</tbody>
</table>

Table 6.1: Basic Orbit algorithms
The Schreier tree is shown in Figure 6.1.

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

T3r1_orbits:

```bash
gt $(ORBITER) -v 4 \\
gt -define G \\
gt -linear_group -GL_d_q_wr_Sym_n 2 2 3 \\
gt -on_rank_one_tensors -end \\
gt -define Orb -orbits -group G \\
gt -on_points \\
gt -end \\
pdflatex GL_2_2_wreath_Sym3_orbits_report.tex \\
open GL_2_2_wreath_Sym3_orbits_report.pdf
```

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

orbits_cubic_curves_q2:

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

Figure 6.1: The Schreier tree for the action of PGL(4, 2) on points of PG(3, 2)
This command computes the orbits of on all cubic forms in 4 variables, confirming the work of Dickson [24] and an enumerative result of Cooley [20].

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

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

\begin{verbatim}
#pdflatex PGGL_2_8OnConic_2_8_orbits_report.tex
#open PGGL_2_8OnConic_2_8_orbits_report.pdf
\end{verbatim}
The output shown below. First, the orbits are listed. Then for each orbit, the stabilizer is shown, together with the generators in the action on the plane. For the sake of space, some of the output has been shortened. The three orbits correspond to the conic, the nucleus and the remaining points of the plane.

### Group Orbits

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

Strong generators for a group of order 1512:

\[
\begin{align*}
1 & 0 \\
0 & 1
\end{align*}
\]

\[
\begin{align*}
\gamma & 0 \\
0 & 1
\end{align*}
\]

\[
\begin{align*}
1 & 0 \\
1 & 1
\end{align*}
\]

\[
\begin{align*}
1 & 0 \\
\gamma & 1
\end{align*}
\]

\[
\begin{align*}
1 & 0 \\
\gamma^2 & 1
\end{align*}
\]

\[
\begin{align*}
0 & 1
\end{align*}
\]

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

\[(1, 9, 63)\]

<table>
<thead>
<tr>
<th>$i$</th>
<th>orbit length</th>
<th>number of orbits</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>63</td>
<td>1</td>
</tr>
</tbody>
</table>

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

Orbits of length 1:
Orbit 1: (1)

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

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

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:

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


Generator 2 / 4 is:

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


Generator 3 / 4 is:

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


Orbits of length 9:

Orbit 0: (0, 2, 3, 29, 48, 38, 55, 60, 67)

Stabilizer of orbit representative 0:

Strong generators for a group of order 168:

\[ \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} \gamma^6 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} \gamma^4 & 0 \\ 0 & 1 \end{bmatrix} \]

1, 0, 0, 1, 1,
1, 0, 0, 2, 0,
1, 0, 3, 5, 0,

Generator 0 / 3 is:

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

Generator 1 / 3 is:
\[
\gamma^6 \begin{bmatrix} 0 \\ 0 \end{bmatrix}
\]

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

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

\[
\begin{bmatrix}
1 & 0 \\
\gamma^5 & 1
\end{bmatrix}_{2}
\]
as permutation: 

Generator 2 / 4 is:

\[
\begin{bmatrix}
1 & 0 \\
\gamma^3 & 1
\end{bmatrix}_{0}
\]
as permutation: 

Generator 3 / 4 is:

\[
\begin{bmatrix}
1 & 0 \\
\gamma & 1
\end{bmatrix}_{0}
\]
as permutation: 
6.2 Poset Classification

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

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

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

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

Orbiter uses the algorithm of Schmalz [62] to perform poset classification. Two versions are available: one for subset-type posets and one for subspace-type posets. Figure 6.4 shows the subspace lattice of $V(3, 2) = \mathbb{F}_2^3$. The basis elements are listed, using the enumerator for elements on the projective geometry $PG(2, 2)$ explained in Section 4.1.
The commands shown in Tables 6.2-6.3 can be used to control the poset classification algorithm. By default, Orbiter will choose the lexicographically least orbit representatives. It is possible to direct Orbiter to choose different orbit representatives. To this end, the nodes in the orbit tree are labeled. The node number is the zero-based number of a given node in the tree, using the breadth first ordering.

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

\[-\text{preferred\_choice}\ n\ a\ b\]

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

Figure 6.4: Subspace lattice of $V(3, 2)$
<table>
<thead>
<tr>
<th>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 Asup and Ainf.</td>
</tr>
<tr>
<td>-print_data_structure</td>
<td></td>
<td>Print the data structure.</td>
</tr>
<tr>
<td>-list</td>
<td></td>
<td>List orbits at level d.</td>
</tr>
</tbody>
</table>

Table 6.2: Options to control the poset classification algorithm (Part 1)
<table>
<thead>
<tr>
<th>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 ( L )</td>
<td></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 ( n a b )</td>
<td></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:

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

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

```
PGL_3_2_singer:
  > $(ORBITER) -v 3 \n  > -orbiter_path $(ORBITER_PATH) \n  > -define Control -poset_classification_control \n  > -define 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_1_poset.tex
  > open PGL_3_2_singer_1_poset.pdf
```

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

```
PG_3_2_subsets:
  > $(ORBITER) -v 3 \n  > -orbiter_path $(ORBITER_PATH) \n  > -define Control -poset_classification_control \n  > -define G -linear_group -PGL 4 2 -singer 1 -end \n  > -define Orb -orbits -group G \n  > -on_subsets 7 Control \n  > -end
  > pdflatex PGL_3_2_singer_1_poset.tex
  > open PGL_3_2_singer_1_poset.pdf
```
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 \( \text{PG}(2, 2) \).

\( \text{PGL}_{3, 2} \text{ on lines:} \)

\( \text{pdflatex PGL}_{3, 2}\text{.poset.tex} \)

\( \text{open PGL}_{3, 2}\text{.poset.pdf} \)
The following example computes the orbits of $\text{PGO}(5, 2)$ on the power set lattice of points of $Q(4, 2)$:

\begin{verbatim}
PGO_5_2_on_subsets:
  $(ORBITER) -v 3 \$
  $\text{-orbiter_path } (ORBITER_PATH) \$
  $\text{-define Control -poset_classification_control} \$
  $\text{-define G -linear_group -PGO 5 F} \$
  $\text{-define Orb -orbits -group G} \$
  $\text{-on_subsets 15 Control} \$
  \text{pdflatex PGO_5_2_poset.tex}
  \text{open PGO_5_2_poset.pdf}
\end{verbatim}

The poset of orbits is shown in Figure 6.6.
Figure 6.6: Orbits of $\text{PGO}(5, 2)$ on the poset of subsets of $Q(4, 2)$
Figure 6.7: The hyperbolic quadric in affine space $\mathbb{R}^3$

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 $\text{PG}(3, 2)$ under the action of the orthogonal group. In $\text{PG}(3, 2)$ 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. $\text{PG}(3, 2)$ has 15 points:

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

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

$$P_0, P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_{10}.$$ 

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

```
subspaces.Op_4.2:
▷ $(\text{ORBITER})$ -v 5 \\
▷ ▷ -orbiter_path $(\text{ORBITER\_PATH})$ \\
▷ ▷ -define Control -poset_classification_control \\
▷ ▷ ▷ -node_label_is_element \\
```
Figure 6.8: Hasse-diagram for the orbits of the orthogonal group PGO\(^+(4, 2)\) on subspaces of PG(3, 2).

- `draw_poset` - `draw_options` - `radius 200` - `end`
- `problem_label Op.4.2` - `W` - `depth 4`
- `report` - `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 PG(3, 2) under 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 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

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

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

The orbit count gives

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

The command

\begin{verbatim}
PGL_2_17_on_partition:
   $(ORBITER) -v 5 \
   -define Control -poset_classification_control \
   -problem_label PGL_2_17 \
   -depth 6 \
   -W \
   -end \
   -define G -linear_group -PGL 2 17 -end \ 
   -define Orb -orbits -group G \
   -on_partition 6 Control \
   -end
\end{verbatim}

204
computes the orbits of the group PGL(2, 17) on set_partitions of type 6 + 6 + 6. The number of set_partitions is
\[ \binom{18}{6} \cdot \binom{12}{6} \cdot 3! = 2858856 \]

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

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

### Orbits at Level 18

There are 2 orbits at level 18.

#### Orbit 0 / 2 at Level 18

Node number: 4212

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

\[
\begin{align*}
0 : 0 &= (1, 0, 0) \\
1 : 1 &= (0, 1, 0) \\
2 : 2 &= (0, 0, 1) \\
3 : 3 &= (1, 1, 1) \\
4 : 52 &= (3, 2, 1) \\
5 : 67 &= (2, 3, 1) \\
6 : 89 &= (8, 4, 1) \\
7 : 106 &= (9, 5, 1) \\
8 : 126 &= (13, 6, 1) \\
9 : 141 &= (12, 7, 1)
\end{align*}
\]
Strong generators for a group of order 144:

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

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 : 217 = ( 8, 12, 1 )
15 : 229 = ( 4, 13, 1 )
16 : 256 = ( 15, 14, 1 )
17 : 264 = ( 7, 15, 1 )

Strong generators for a group of order 16320:

\[
\begin{bmatrix}
\delta^6 & 0 & 0 \\
0 & \delta^3 & 0 \\
0 & 0 & 1
\end{bmatrix},
\begin{bmatrix}
\delta^9 & 0 & 0 \\
0 & \delta^7 & 0 \\
0 & 0 & 1
\end{bmatrix},
\begin{bmatrix}
\delta^2 & 0 & 0 \\
0 & \delta^{11} & 0 \\
\delta^4 & \delta^7 & 1
\end{bmatrix},
\begin{bmatrix}
\delta^{10} & 0 & 0 \\
0 & \delta^3 & 0 \\
\delta & \delta^{11} & 1
\end{bmatrix},
\begin{bmatrix}
\delta & 0 & 0 \\
\delta^{12} & \delta^2 & \delta^5 \\
\delta^{14} & \delta^{10} & 1
\end{bmatrix},
\begin{bmatrix}
\delta^5 & 0 & 0 \\
\delta^{15} & \delta^3 & \delta^6 \\
\delta^6 & \delta^6 & 1
\end{bmatrix},
\begin{bmatrix}
\delta^{12} & 1 & \delta^2 \\
\delta^4 & \delta^3 & \delta^7 \\
\delta^6 & \delta^3 & 1
\end{bmatrix},
\begin{bmatrix}
\delta^5 & \delta^3 & \delta^6 \\
\delta^{11} & \delta^6 & \delta^{10} \\
\delta^{10} & \delta^6 & 1
\end{bmatrix}
\]

208
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:

```plaintext
cnc_arcs_16:
\$ \texttt{(ORBITER) -v 4 \$
\> \> \> -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\> \> \> \> -classify_arcs \n\> \> \> \> \> -poset_classification_control \n\> \> \> \> \> \> -problem_label nc_arcs_q16_d2 \n\> \> \> \> \> \> -W -depth 6 \n\> \> \> \> \> \> -report -end \n\> \> \> \> \> -end \n\> \> \> \> -target_size 6 \n\> \> \> \> -d 2 \n\> \> \> -conic_test \n\> \> \> -end \n\> \> -end
\> pdflatex nc_arcs_q16_d2_poset.tex
\> open nc_arcs_q16_d2_poset.pdf
```

Cubic surfaces are associated with arcs of size 6 (in a many-to-one relationship when considering isomorphism classes). The number of Eckardt points of the surface can be recovered from properties of the arc. For this reason, it is interesting to classify arcs so that the associated cubic surface has a fixed number of Eckardt points. The following command shows how to create all arcs associated with cubic surfaces with 13 Eckardt points over the field $\mathbb{F}_{32}$:

```plaintext
nc_arcs_32_E13:
```

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

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

Here is an example. The command

cubic_curves_PG_2_8:

```bash
$(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 \n  -poset_classification_control \
  -problem_classification_label cc_8 -W -depth 9 \n  -draw_options -radius 200 -embedded -end \
  -recognize "0,1,2,3,35,28" \n  -recognize "1,2,3,51,28,61,46,71,40" \n  -draw_poset \n  -Kramer_Mesner_matrix 6 9 \n  -end \
  -end
```

```bash
$(ORBITER) -v 2 -draw_matrix \
  -input_csv_file cc_8_KM_6_9.csv \
  -box_width 50 -bit_depth 8 -end
```

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

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

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

Cubic Surfaces

7.1 Creation

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

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

Tables 7.1-7.3 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.

```plaintext
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" \
```

213
<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 $\mathbb{P} = \text{PG}(3,q)$.</td>
</tr>
<tr>
<td>-label_txt</td>
<td>label</td>
<td>Override the ascii label of the curve.</td>
</tr>
<tr>
<td>-label_tex</td>
<td>label</td>
<td>Override the latex label of the curve.</td>
</tr>
<tr>
<td>-label_for_summary</td>
<td>label</td>
<td>Override the ascii label of the curve, to be used in summary commands.</td>
</tr>
<tr>
<td>-catalogue</td>
<td>$i$</td>
<td>Create the $i$-th surface in the Orbiter catalogue. Here, $i$ is an index variable used to index all surfaces in $\text{PG}(3,q)$. The index $i$ is zero-based. The automorphism group is created as well.</td>
</tr>
<tr>
<td>-by_coefficients</td>
<td>list-of-coeff-pairs</td>
<td>Create a surface from a list of coefficient-monomial pairs. The automorphism group is not created.</td>
</tr>
<tr>
<td>-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 $\text{PG}(19,q_0)$ and use the Orbiter point rank.</td>
</tr>
<tr>
<td>-family_Eckardt</td>
<td>$a\ b$</td>
<td>Create the Eckardt surface with parameters $(a, b)$ as in see [12] (where it is called the Hilbert, Cohn-Vossen surface). The equation is $X_3^3 - b^2(X_0^2 + X_1^2 + X_2^2)X_3 + \frac{b^3}{a}(a + 1)X_0X_1X_2 = 0$. The automorphism group is created as well.</td>
</tr>
</tbody>
</table>

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

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

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

```
pdflatex surface_family_Eckardt_q13_a4_b12_with_group.tex
open surface_family_Eckardt_q13_a4_b12_with_group.pdf
```

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:

```
\begin{verbatim}
\text{F}_{abcd}\text{eqn}=-}\frac{\text{-(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 \} \\
\end{verbatim}
```
The following command parses the formula and creates the surface with \((a, b, c, d) = (4, 2, 2, 4)\):

```
surface_F_abcd:
   \((\text{ORBITER}) -v 3 \)
   \(-\text{define } F \ -\text{finite} \text{ field} \ -q 7 \ -\text{end} \)
   \(-\text{with } F \ -\text{do} \)
   \(-\text{finite} \text{ field} \text{ activity} \)
   \(-\text{parse} \text{ and} \text{ evaluate } "F_{abcd}" "X0,X1,X2,X3" \)
   \(-\text{end} \)
```

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:
   \((\text{ORBITER}) -v 3 \)
   \(-\text{define } F \ -\text{finite} \text{ field} \ -q 7 \ -\text{end} \)
   \(-\text{define } P \ -\text{projective} \text{ space} \ -n 3 \ -\text{field} F \ -v 0 \ -\text{end} \)
   \(-\text{define } F_{2345} \ -\text{cubic} \text{ surface} \ -\text{space} P \)
   \(-\text{by_equation } "F_{\alpha\beta\gamma\delta}\" \)
   \("\text{DF.}\{\alpha,\beta,\gamma,\delta}\text{\}D} "x0,x1,x2,x3" \)
   \(-\text{end} \)
   \(-\text{override} \text{ group} 6 2 \)
   \(-\text{end} \)
   \(-\text{with } F_{2345} \ -\text{do} \)
   \(-\text{cubic} \text{ surface} \text{ activity} \)
   \(-\text{report} \)
   \(-\text{report_with_group} \)
   \(-\text{end} \)
```

```
\text{pdflatex surface\_equation\_F\_alpha\_beta\_gamma\_delta\_q7\_report.tex}
\text{open surface\_equation\_F\_alpha\_beta\_gamma\_delta\_q7\_report.pdf}
```
pdflatex surface_equation_F_alpha_beta_gamma_delta_q7_with_group.tex
open surface_equation_F_alpha_beta_gamma_delta_q7_with_group.pdf
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-space</td>
<td>P</td>
<td>Specify the 2-dimensional projective space $P = PG(2, q)$.</td>
</tr>
<tr>
<td>-label_txt</td>
<td>label</td>
<td>Override the ascii label of the curve.</td>
</tr>
<tr>
<td>-label_tex</td>
<td>label</td>
<td>Override the latex label of the curve.</td>
</tr>
<tr>
<td>-label_for_summary</td>
<td>label</td>
<td>Override the ascii label of the curve, to be used in summary commands.</td>
</tr>
<tr>
<td>-catalogue</td>
<td>OCN</td>
<td>Create the quartic curve from the Orbiter catalogue with the given Orbiter catalogue number.</td>
</tr>
<tr>
<td>-by_coefficients</td>
<td>coeffs</td>
<td>Create a quartic curve given the coefficients of the equation.</td>
</tr>
<tr>
<td>-by_equation</td>
<td>eqn</td>
<td>Create a quartic curve from an equation.</td>
</tr>
<tr>
<td>-override_group</td>
<td>descr</td>
<td>Override the automorphism group of the curve by the given group.</td>
</tr>
<tr>
<td>-transform</td>
<td>elt</td>
<td>Apply the transformation given by the group element.</td>
</tr>
<tr>
<td>-transform_inverse</td>
<td>elt</td>
<td>Apply the inverse transformation given by the group element.</td>
</tr>
</tbody>
</table>

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 a 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:

```bash
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 \
```
<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

```bash
▷▷ -with C -do \
▷▷▷ -quartic_curve_activity \
▷▷▷▷ -report \
▷▷▷▷ -end \
▷ pdflatex quartic_curve_catalogue_q9_iso0_report.tex \
▷ open quartic_curve_catalogue_q9_iso0_report.pdf
```

The report contains the following information:

**The equation**

The equation of the quartic curve is:

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

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

**The gradient**

The gradient of the quartic curve is:

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

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

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

$$(8,0,1,0,0,0,0,0,0,0,0)$$
\[ \alpha^4 X_0^3 + \alpha^6 X_1^3 \]
\[(2, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0)\]

**General information**

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

**All points on the curve**

The surface has 28 points:

The points on the quartic curve are:

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

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

The Kovalevski points are:

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

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

The points off the curve are:

<table>
<thead>
<tr>
<th>0 : ( P_6 = (3, 1, 0) )</th>
<th>22 : ( P_{35} = (7, 2, 1) )</th>
<th>44 : ( P_{66} = (2, 6, 1) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 : ( P_7 = (4, 1, 0) )</td>
<td>23 : ( P_{36} = (8, 2, 1) )</td>
<td>45 : ( P_{67} = (3, 6, 1) )</td>
</tr>
<tr>
<td>2 : ( P_8 = (5, 1, 0) )</td>
<td>24 : ( P_{37} = (0, 3, 1) )</td>
<td>46 : ( P_{68} = (4, 6, 1) )</td>
</tr>
<tr>
<td>3 : ( P_9 = (6, 1, 0) )</td>
<td>25 : ( P_{39} = (2, 3, 1) )</td>
<td>47 : ( P_{69} = (5, 6, 1) )</td>
</tr>
<tr>
<td>4 : ( P_{10} = (7, 1, 0) )</td>
<td>26 : ( P_{40} = (3, 3, 1) )</td>
<td>48 : ( P_{70} = (6, 6, 1) )</td>
</tr>
<tr>
<td>5 : ( P_{11} = (8, 1, 0) )</td>
<td>27 : ( P_{42} = (5, 3, 1) )</td>
<td>49 : ( P_{71} = (7, 6, 1) )</td>
</tr>
<tr>
<td>6 : ( P_{12} = (1, 0, 1) )</td>
<td>28 : ( P_{43} = (6, 3, 1) )</td>
<td>50 : ( P_{72} = (8, 6, 1) )</td>
</tr>
<tr>
<td>7 : ( P_{13} = (2, 0, 1) )</td>
<td>29 : ( P_{45} = (8, 3, 1) )</td>
<td>51 : ( P_{73} = (0, 7, 1) )</td>
</tr>
<tr>
<td>8 : ( P_{15} = (4, 0, 1) )</td>
<td>30 : ( P_{37} = (1, 4, 1) )</td>
<td>52 : ( P_{74} = (1, 7, 1) )</td>
</tr>
<tr>
<td>9 : ( P_{16} = (5, 0, 1) )</td>
<td>31 : ( P_{48} = (2, 4, 1) )</td>
<td>53 : ( P_{75} = (2, 7, 1) )</td>
</tr>
<tr>
<td>10 : ( P_{18} = (7, 0, 1) )</td>
<td>32 : ( P_{49} = (3, 4, 1) )</td>
<td>54 : ( P_{79} = (6, 7, 1) )</td>
</tr>
<tr>
<td>11 : ( P_{19} = (8, 0, 1) )</td>
<td>33 : ( P_{50} = (4, 4, 1) )</td>
<td>55 : ( P_{80} = (7, 7, 1) )</td>
</tr>
<tr>
<td>12 : ( P_{20} = (0, 1, 1) )</td>
<td>34 : ( P_{52} = (6, 4, 1) )</td>
<td>56 : ( P_{81} = (8, 7, 1) )</td>
</tr>
<tr>
<td>13 : ( P_{21} = (2, 1, 1) )</td>
<td>35 : ( P_{54} = (8, 4, 1) )</td>
<td>57 : ( P_{85} = (3, 8, 1) )</td>
</tr>
<tr>
<td>14 : ( P_{22} = (3, 1, 1) )</td>
<td>36 : ( P_{55} = (0, 5, 1) )</td>
<td>58 : ( P_{86} = (4, 8, 1) )</td>
</tr>
<tr>
<td>15 : ( P_{23} = (4, 1, 1) )</td>
<td>37 : ( P_{56} = (1, 5, 1) )</td>
<td>59 : ( P_{87} = (5, 8, 1) )</td>
</tr>
<tr>
<td>16 : ( P_{26} = (7, 1, 1) )</td>
<td>38 : ( P_{59} = (4, 5, 1) )</td>
<td>60 : ( P_{88} = (6, 8, 1) )</td>
</tr>
<tr>
<td>17 : ( P_{27} = (8, 1, 1) )</td>
<td>39 : ( P_{60} = (5, 5, 1) )</td>
<td>61 : ( P_{89} = (7, 8, 1) )</td>
</tr>
<tr>
<td>18 : ( P_{28} = (0, 2, 1) )</td>
<td>40 : ( P_{61} = (6, 5, 1) )</td>
<td>62 : ( P_{90} = (8, 8, 1) )</td>
</tr>
<tr>
<td>19 : ( P_{29} = (1, 2, 1) )</td>
<td>41 : ( P_{63} = (8, 5, 1) )</td>
<td></td>
</tr>
<tr>
<td>20 : ( P_{31} = (3, 2, 1) )</td>
<td>42 : ( P_{64} = (0, 6, 1) )</td>
<td></td>
</tr>
<tr>
<td>21 : ( P_{33} = (5, 2, 1) )</td>
<td>43 : ( P_{65} = (1, 6, 1) )</td>
<td></td>
</tr>
</tbody>
</table>
The lines and their points of contact are:

\[
\begin{align*}
a_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & \alpha^3 \end{bmatrix} & = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 8 \end{bmatrix}, \quad P_0 = P(1,0,0)\ 
4\times \\
a_2 = \begin{bmatrix} 1 & 0 & \alpha^6 \\ 0 & 1 & 1 \end{bmatrix} & = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 1 \end{bmatrix}, \quad P_{83} = P(1,8,1)\ 
4\times \\
a_3 = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & \alpha^6 \end{bmatrix} & = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 5 \end{bmatrix}, \quad P_{57} = P(2,5,1)\ 
4\times \\
a_4 = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & \alpha^2 \end{bmatrix} & = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 7 \end{bmatrix}, \quad P_{53} = P(7,4,1)\ 
4\times \\
a_5 = \begin{bmatrix} 1 & 0 & \alpha^2 \\ 0 & 1 & \alpha^7 \end{bmatrix} & = \begin{bmatrix} 1 & 0 & 7 \\ 0 & 1 & 4 \end{bmatrix}, \quad P_{30} = P(2,2,1)\ 
4\times \\
a_6 = \begin{bmatrix} 1 & 0 & \alpha^2 \\ 0 & 1 & \alpha^2 \end{bmatrix} & = \begin{bmatrix} 1 & 0 & 7 \\ 0 & 1 & 7 \end{bmatrix}, \quad P_5 = P(2,1,0)\ 
4\times \\
b_1 = \begin{bmatrix} 1 & 0 & \alpha^6 \\ 0 & 1 & \alpha^7 \end{bmatrix} & = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 4 \end{bmatrix}, \quad P_{58} = P(3,5,1)\ 
4\times \\
b_2 = \begin{bmatrix} 1 & 0 & \alpha^7 \\ 0 & 1 & \alpha^6 \end{bmatrix} & = \begin{bmatrix} 1 & 0 & 4 \\ 0 & 1 & 5 \end{bmatrix}, \quad P_{14} = P(3,0,1)\ 
4\times \\
b_3 = \begin{bmatrix} 1 & 0 & \alpha \\ 0 & 1 & 1 \end{bmatrix} & = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 1 \end{bmatrix}, \quad P_{62} = P(7,5,1)\ 
4\times \\
b_4 = \begin{bmatrix} 1 & 0 & \alpha^5 \\ 0 & 1 & \alpha^2 \end{bmatrix} & = \begin{bmatrix} 1 & 0 & 6 \\ 0 & 1 & 7 \end{bmatrix}, \quad P_{77} = P(4,7,1)\ 
4\times \\
b_5 = \begin{bmatrix} 1 & 0 & \alpha^5 \\ 0 & 1 & \alpha^3 \end{bmatrix} & = \begin{bmatrix} 1 & 0 & 6 \\ 0 & 1 & 8 \end{bmatrix}, \quad P_{41} = P(4,3,1)\ 
4\times \\
b_6 = \begin{bmatrix} 1 & 0 & \alpha \\ 0 & 1 & \alpha^2 \end{bmatrix} & = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 7 \end{bmatrix}, \quad P_3 = P(1,1,1)\ 
4\times \\
c_{12} = \begin{bmatrix} 1 & 0 & \alpha^3 \\ 0 & 1 & \alpha^4 \end{bmatrix} & = \begin{bmatrix} 1 & 0 & 8 \\ 0 & 1 & 2 \end{bmatrix}, \quad P_{17} = P(6,0,1)\ 
4\times \\
c_{13} = \begin{bmatrix} 1 & 0 & \alpha \\ 0 & 1 & \alpha^4 \end{bmatrix} & = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 2 \end{bmatrix}, \quad P_{84} = P(2,8,1)\ 
4\times \\
c_{14} = \begin{bmatrix} 1 & 0 & \alpha^5 \\ 0 & 1 & 1 \end{bmatrix} & = \begin{bmatrix} 1 & 0 & 6 \\ 0 & 1 & 1 \end{bmatrix}, \quad P_{32} = P(4,2,1)\ 
4\times \\
c_{15} = \begin{bmatrix} 1 & 0 & \alpha^5 \\ 0 & 1 & 0 \end{bmatrix} & = \begin{bmatrix} 1 & 0 & 6 \\ 0 & 1 & 0 \end{bmatrix}, \quad P_1 = P(0,1,0)\ 
4\times
\end{align*}
\]
\( c_{16} = \begin{bmatrix} 1 & 0 & \alpha \\ 0 & 1 & \alpha^6 \end{bmatrix} \), \( c_{23} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & \alpha^5 \end{bmatrix} \), \( c_{24} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & \alpha^7 \end{bmatrix} \), \( c_{25} = \begin{bmatrix} 1 & 0 & \alpha^2 \\ 0 & 1 & \alpha^4 \end{bmatrix} \), \( c_{26} = \begin{bmatrix} 1 & 0 & \alpha^2 \\ 0 & 1 & \alpha^3 \end{bmatrix} \), \( c_{34} = \begin{bmatrix} 1 & 0 & \alpha^4 \\ 0 & 1 & \alpha^3 \end{bmatrix} \), \( c_{35} = \begin{bmatrix} 1 & 0 & \alpha^6 \\ 0 & 1 & \alpha^4 \end{bmatrix} \), \( c_{36} = \begin{bmatrix} 1 & 0 & \alpha^4 \\ 0 & 1 & \alpha^7 \end{bmatrix} \), \( c_{45} = \begin{bmatrix} 1 & 0 & \alpha^4 \\ 0 & 1 & \alpha^6 \end{bmatrix} \), \( c_{46} = \begin{bmatrix} 1 & 0 & \alpha^6 \\ 0 & 1 & \alpha \end{bmatrix} \), \( c_{56} = \begin{bmatrix} 1 & 0 & \alpha^4 \\ 0 & 1 & 1 \end{bmatrix} \).

\( d = \begin{bmatrix} 1 & 0 & \alpha^6 \\ 0 & 0 & 1 \end{bmatrix} \), \( P_5 = P(5, 1, 1) \times 4 \times 35 \)

\( P_8 = P(0, 8, 1) \times 16 \)

\( P_{25} = P(6, 1, 1) \times 14 \)

\( P_{26} = P(3, 7, 1) \times 72 \)

\( P_{35} = P(1, 3, 1) \times 78 \)

\( P_{34} = P(5, 1, 1) \times 52 \)

\( P_{38} = P(5, 7, 1) \times 52 \)

\( P_{24} = P(6, 2, 1) \times 25 \)

\( P_{36} = P(0, 4, 1) \times 53 \)

\( P_4 = P(1, 1, 0) \times 21 \)

\( P_2 = P(0, 0, 1) \times 59 \)

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

Line type: \(1^{28}\)

| 28 | 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 0 |

Point types: \(1^{28}\)
| point types for points off the curve: $4^{63}$ |  
|---|---|
| 63 4 | 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 0 |

Lines on points off the curve:
- Off point $0 = P_6 = (3, 1, 0)$ lies on 4 bisecants: \{ 4, 9, 16, 17 \}
- Off point $1 = P_7 = (4, 1, 0)$ lies on 4 bisecants: \{ 13, 14, 23, 25 \}
- Off point $2 = P_8 = (5, 1, 0)$ lies on 4 bisecants: \{ 1, 3, 19, 24 \}
- Off point $3 = P_9 = (6, 1, 0)$ lies on 4 bisecants: \{ 6, 11, 12, 20 \}
- Off point $4 = P_{10} = (7, 1, 0)$ lies on 4 bisecants: \{ 2, 10, 22, 27 \}
- Off point $5 = P_{11} = (8, 1, 0)$ lies on 4 bisecants: \{ 7, 8, 18, 21 \}
- Off point $6 = P_{12} = (1, 0, 1)$ lies on 4 bisecants: \{ 2, 3, 17, 18 \}
- Off point $7 = P_{13} = (2, 0, 1)$ lies on 4 bisecants: \{ 21, 23, 24, 26 \}
- Off point $8 = P_{15} = (4, 0, 1)$ lies on 4 bisecants: \{ 8, 11, 13, 16 \}
- Off point $9 = P_{16} = (5, 0, 1)$ lies on 4 bisecants: \{ 4, 5, 19, 20 \}
- Off point $10 = P_{18} = (7, 0, 1)$ lies on 4 bisecants: \{ 1, 6, 22, 25 \}
- Off point $11 = P_{19} = (8, 0, 1)$ lies on 4 bisecants: \{ 9, 10, 14, 15 \}
- Off point $12 = P_{20} = (0, 1, 1)$ lies on 4 bisecants: \{ 1, 8, 14, 26 \}
- Off point $13 = P_{21} = (2, 1, 1)$ lies on 4 bisecants: \{ 7, 9, 20, 25 \}
- Off point $14 = P_{22} = (3, 1, 1)$ lies on 4 bisecants: \{ 3, 10, 12, 23 \}
- Off point $15 = P_{23} = (4, 1, 1)$ lies on 4 bisecants: \{ 5, 6, 17, 24 \}
- Off point $16 = P_{26} = (7, 1, 1)$ lies on 4 bisecants: \{ 16, 19, 21, 27 \}
- Off point $17 = P_{27} = (8, 1, 1)$ lies on 4 bisecants: \{ 2, 4, 13, 15 \}
- Off point $18 = P_{28} = (0, 2, 1)$ lies on 4 bisecants: \{ 12, 13, 19, 22 \}
- Off point $19 = P_{29} = (1, 2, 1)$ lies on 4 bisecants: \{ 6, 10, 16, 26 \}
- Off point $20 = P_{31} = (3, 2, 1)$ lies on 4 bisecants: \{ 2, 5, 21, 25 \}
- Off point $21 = P_{33} = (5, 2, 1)$ lies on 4 bisecants: \{ 1, 9, 18, 27 \}
- Off point $22 = P_{35} = (7, 2, 1)$ lies on 4 bisecants: \{ 7, 11, 17, 23 \}
- Off point $23 = P_{36} = (8, 2, 1)$ lies on 4 bisecants: \{ 3, 8, 15, 20 \}
- Off point $24 = P_{37} = (0, 3, 1)$ lies on 4 bisecants: \{ 4, 6, 18, 23 \}
- Off point $25 = P_{39} = (2, 3, 1)$ lies on 4 bisecants: \{ 1, 5, 12, 16 \}
- Off point $26 = P_{40} = (3, 3, 1)$ lies on 4 bisecants: \{ 8, 9, 22, 24 \}
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

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

```
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  -with S_%L -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:

```
NB_CUBIC_SURFACES_Q13=4
```
Just like before, we create all quartic curves arising from projections:

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

We combine the output files into one:

```
quartic_curves_q13_combine:
  ▷ $(ORBITER) -v 3 \n  ▷ ▷ -csv_file_concatenate_from_mask $(NB_CUBIC_SURFACES_Q13) \n  ▷ ▷ ▷ surface_catalogue_q13_iso%ld_quartics.csv \n  ▷ ▷ ▷ 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 \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
```

231
We find exactly two isomorphism types. The data is exported to C++ source code. The file \texttt{quartic_curves_q13.cpp} is created. This file is now part of Orbiter’s knowledge base of geometric objects.
7.3 Classification

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

In \( \text{PG}(3, 4) \), there is only one type of cubic surfaces with 27 lines. It is a member of the Hirschfeld family, described in [33]. The following Orbiter command can be used to construct this surface and to prove its uniqueness for \( \mathbb{F}_4 \). The following command utilizes the algorithm of [12] to do so:

```latex
\text{surface}\_\text{classify}\_q4:\n\text{\texttt{\$}}(\text{ORBITER}) -v 5 \n\text{\texttt{\$}} -define \ F -finite\_field -q 4 -end \n\text{\texttt{\$}} -define \ P -projective\_space -n 3 -field F -v 0 -end \n\text{\texttt{\$}} -with \ P -do \n\text{\texttt{\$}} -projective\_space\_activity \n\text{\texttt{\$}} -classify\_surfaces\_with\_double\_sixes \, \text{Surf27} -W -end \n\text{\texttt{\$}} -end \n\text{\texttt{\$}} -with \ \text{Surf27} -do \n\text{\texttt{\$}} -classification\_of\_cubic\_surfaces\_with\_double\_sixes\_activity \n\text{\texttt{\$}} -report -end \n\text{\texttt{\$}} -end \n\text{\texttt{\$}} -print\_symbols \n\text{\texttt{\$}} \text{pdflatex} \, \text{Surfaces}\_q4.tex \n\text{\texttt{\$}} \text{open} \, \text{Surfaces}\_q4.pdf
```

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

---

The semilinear group

The Action

Group action \( \text{PGL}(4, 4) \) of degree 85
The group is a matrix group.

\[ q = 4 \]
The orthogonal group

The Action

Group action $P\Gamma L(4, 4)\text{On} \text{Wedge of degree } 1365$

The group is a matrix group.

The base action is on projective space $\text{PG}(3, 4)$

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

The group stabilizing the fixed line

The Action

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


Strong generators for a group of order 5529600: 

The classification of five-plus-ones

Poset classification up to depth 5
The Orbits

Number of Orbits By Level

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

Summary of Orbit Representatives

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

Table 7.7: 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></td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>1</td>
<td>{ 0, 3, 56, 76, 97 }</td>
<td>(96, 57600)</td>
<td></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></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>

236
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:

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

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

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

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

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

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
\omega & \omega & 0 & 0 \\
0 & \omega^2 & 0 & 0
\end{bmatrix}
\]
The overall number of objects is: 1370880

**Flag orbits for surfaces**

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

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

Strong generators for a group of order 1440:

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

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

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

\[x\]
Strong generators for a group of order 51840:

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

The overall number of objects is: 38080

**The Group** PGL(4, 4)

The order of the group is 1974067200

**Cubic Surfaces with 27 Lines in** PG(3, 4)

The order of the group is 1974067200

The group has 1 orbits:

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

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

The overall number of objects is: 38080

**Surface 4#0**

**The equation**

The equation of the surface is:

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

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

Number of points on the surface 45
The automorphism group of the surface has order 51840
The automorphism group is the following group

Strong generators for a group of order 51840:

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

\[1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, \]
\[1, 0, 0, 0, 0, 2, 0, 0, 0, 0, 2, 0, 0, 0, 0, 1, 0, 1, 0, \]
\[1, 0, 0, 0, 0, 3, 0, 0, 0, 0, 3, 0, 1, 0, 0, 0, 1, 0, 1, 0, \]
\[1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, \]
\[1, 0, 0, 0, 3, 2, 2, 0, 0, 0, 2, 0, 1, 0, 3, 1, 0, 1, 0, \]
\[1, 0, 0, 0, 1, 0, 2, 0, 2, 2, 0, 0, 2, 2, 1, 1, 0, 1, 0, \]
\[1, 3, 1, 2, 1, 0, 2, 0, 3, 2, 0, 2, 0, 0, 0, 0, 0, 0, \]
\[1, 1, 3, 3, 0, 3, 0, 1, 1, 2, 0, 1, 0, 3, 0, 0, 0, 0, \]

General information

Points on lines:

\[5^{27}\]

Lines on points:

\[3^{45}\]

The 27 Lines

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

\[\ell_1 = a_2 = \begin{bmatrix} 1 & 0 & \omega & 0 \\ 0 & 1 & 0 & \omega^2 \end{bmatrix}_{54} = \begin{bmatrix} 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 3 \end{bmatrix}_{54} = \mathbf{Pl}(2, 3, 0, 2, 1)_{238}\]
\[ \ell_2 = a_3 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}, \quad \ell_3 = a_4 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \ell_4 = a_5 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \quad \ell_5 = a_6 = \begin{bmatrix} 1 & 0 & \omega^2 & 1 \\ 0 & 1 & 0 & \omega \end{bmatrix}, \quad \ell_6 = b_1 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \ell_7 = b_2 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}, \quad \ell_8 = b_3 = \begin{bmatrix} 1 & \omega & 0 & 0 \\ 0 & 0 & 1 & \omega^2 \end{bmatrix}, \quad \ell_9 = b_4 = \begin{bmatrix} 1 & 0 & \omega^2 & 1 \\ 0 & 1 & 1 & \omega \end{bmatrix}, \quad \ell_{10} = b_5 = \begin{bmatrix} 1 & \omega^2 & 0 & 1 \\ 0 & 0 & 1 & \omega \end{bmatrix}, \quad \ell_{11} = b_6 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad \ell_{12} = c_{12} = \begin{bmatrix} 1 & 0 & \omega & 1 \\ 0 & 1 & 0 & \omega^2 \end{bmatrix}, \quad \ell_{13} = c_{13} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}, \quad \ell_{14} = c_{14} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \ell_{15} = c_{15} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} \]

\[ \text{Pl}(1, 1, 0, 0, 1)_{177} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}, \quad \text{Pl}(0, 1, 0, 0, 0)_{156} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \text{Pl}(1, 0, 0, 0, 0)_{0} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \quad \text{Pl}(1, 0, 0, 0, 0)_{0} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \quad \text{Pl}(3, 2, 0, 2, 3, 1)_{314} = \begin{bmatrix} 1 & 0 & 3 & 1 \\ 0 & 1 & 0 & 2 \end{bmatrix}, \quad \text{Pl}(0, 0, 0, 1, 0, 0)_{9} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \text{Pl}(0, 0, 1, 1, 1)_{198} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}, \quad \text{Pl}(0, 0, 2, 3, 2, 1)_{265} = \begin{bmatrix} 1 & 2 & 0 & 0 \\ 0 & 0 & 1 & 3 \end{bmatrix}, \quad \text{Pl}(3, 0, 3, 2, 3, 1)_{335} = \begin{bmatrix} 1 & 0 & 3 & 1 \\ 0 & 1 & 1 & 2 \end{bmatrix}, \quad \text{Pl}(0, 2, 3, 2, 3, 1)_{337} = \begin{bmatrix} 1 & 3 & 0 & 1 \\ 0 & 0 & 1 & 2 \end{bmatrix}, \quad \text{Pl}(0, 0, 1, 0, 0, 0)_{2} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad \text{Pl}(2, 3, 0, 3, 2, 1)_{256} = \begin{bmatrix} 1 & 0 & 2 & 1 \\ 0 & 1 & 0 & 3 \end{bmatrix}, \quad \text{Pl}(1, 1, 0, 1, 1)_{189} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}, \quad \text{Pl}(0, 1, 0, 1, 0, 0)_{13} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \text{Pl}(1, 0, 0, 1, 0, 0)_{10} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} \]

243
\[ \ell_{16} = c_{16} = \begin{bmatrix} 1 & 0 & \omega^2 & 0 \\ 0 & 1 & 0 & \omega \end{bmatrix}_{71} = \begin{bmatrix} 1 & 0 & 3 & 0 \\ 0 & 1 & 0 & 2 \end{bmatrix}_{71} = \text{Pl}(3, 2, 0, 0, 3, 1)_{299} \]

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

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

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

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

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

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

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

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

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

\[ \ell_{26} = c_{56} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}_1 = \text{Pl}(1, 0, 1, 0, 0, 0)_{3} \]

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 : $E_{52} = a_5 \cap b_2 \cap c_{25} = P_5 = P_5 = P(1,1,0,0) = P(1,1,0,0), T = 1$
6 : $E_{54} = a_5 \cap b_4 \cap c_{45} = P_6 = P_6 = P(\omega,1,0,0) = P(2,1,0,0), T = 2$
7 : $E_{33} = a_5 \cap b_3 \cap c_{35} = P_7 = P_7 = P(\omega^2,1,0,0) = P(3,1,0,0), T = 3$
8 : $E_{36} = a_3 \cap b_6 \cap c_{36} = P_8 = P_8 = P(1,0,1,0) = P(1,0,1,0), T = 5$
9 : $E_{16} = a_1 \cap b_6 \cap c_{16} = P_9 = P_9 = P(\omega,0,1,0) = P(2,0,1,0), T = 10$
10 : $E_{26} = a_2 \cap b_6 \cap c_{26} = P_{10} = P_{10} = P(\omega^2,0,1,0) = P(3,0,1,0), T = 15$
11 : $E_{14,23,56} = c_{14} \cap c_{23} \cap c_{56} = P_{11} = P_{11} = P(0,1,1,0) = P(0,1,1,0), T = 9$
12 : $E_{13,24,56} = c_{13} \cap c_{24} \cap c_{56} = P_{12} = P_{12} = P(1,1,1,0) = P(1,1,1,0), T = 6$
13 : $E_{65} = a_6 \cap b_5 \cap c_{56} = P_{13} = P_{13} = P(\omega,1,1,0) = P(2,1,1,0), T = 12$
14 : $E_{12,34,56} = c_{12} \cap c_{34} \cap c_{56} = P_{14} = P_{14} = P(\omega^2,1,1,0) = P(3,1,1,0), T = 18$
15 : $E_{15,23,46} = c_{15} \cap c_{23} \cap c_{46} = P_{15} = P_{23} = P(1,0,0,1) = P(1,0,0,1), T = 21$
16 : $E_{13} = a_3 \cap b_1 \cap c_{13} = P_{16} = P_{26} = P(0,1,0,1) = P(0,1,0,1), T = 25$
17 : $E_{15,24,36} = c_{15} \cap c_{24} \cap c_{36} = P_{17} = P_{27} = P(1,1,0,1) = P(1,1,0,1), T = 22$
18 : $E_{21} = a_2 \cap b_1 \cap c_{12} = P_{18} = P_{30} = P(0,0,1,1) = P(0,2,0,1), T = 46$
19 : $E_{15,26,34} = c_{15} \cap c_{26} \cap c_{34} = P_{19} = P_{31} = P(1,0,0,1) = P(1,2,0,1), T = 24$
20 : $E_{61} = a_6 \cap b_1 \cap c_{16} = P_{20} = P_{34} = P(0,0,1,0) = P(0,3,0,1), T = 67$
21 : $E_{15} = a_1 \cap b_5 \cap c_{15} = P_{21} = P_{35} = P(1,0,1,1) = P(1,3,0,1), T = 23$
22 : $E_{42} = a_4 \cap b_2 \cap c_{24} = P_{22} = P_{38} = P(0,0,1,1) = P(0,0,1,1), T = 41$
23 : $E_{13,25,46} = c_{13} \cap c_{25} \cap c_{46} = P_{23} = P_{39} = P(1,0,1,1) = P(1,0,1,1), T = 26$
24 : $E_{14,25,36} = c_{14} \cap c_{25} \cap c_{36} = P_{24} = P_{42} = P(0,1,1,1) = P(0,1,1,1), T = 30$
25 : $E_{62} = a_6 \cap b_2 \cap c_{26} = P_{25} = P_{47} = P(\omega,1,1) = P(2,2,1,1), T = 53$
26 : $E_{25} = a_2 \cap b_5 \cap c_{25} = P_{26} = P_{48} = P(\omega^2,0,1,1) = P(3,2,1,1), T = 80$
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 : $E_{43} = a_4 \cap b_3 \cap c_{34} = P_{29} = P_{53} = P(0,0,1,0) = P(0,0,2,1), T = 62$
30 : $E_{12,35,46} = c_{12} \cap c_{35} \cap c_{46} = P_{30} = P_{54} = P(1,0,0,1) = P(1,0,2,1), T = 36$
31 : $E_{35} = a_3 \cap b_5 \cap c_{35} = P_{31} = P_{59} = P(\omega,\omega,1) = P(2,1,2,1), T = 49$
32 : $E_{63} = a_6 \cap b_3 \cap c_{36} = P_{32} = P_{60} = P(\omega^2,1,\omega,1) = P(3,1,2,1), T = 76$
33 : $E_{14,26,35} = c_{14} \cap c_{26} \cap c_{35} = P_{33} = P_{61} = P(0,\omega,0,1) = P(0,2,2,1), T = 51$
34 : $E_{23} = a_2 \cap b_3 \cap c_{23} = P_{34} = P_{62} = P(1,\omega,\omega,1) = P(1,2,2,1), T = 39$
35 : $E_{13} = a_1 \cap b_3 \cap c_{13} = P_{35} = P_{67} = P(\omega,\omega^2,\omega,1) = P(2,3,2,1), T = 50$
36 : $E_{16,24,35} = c_{16} \cap c_{24} \cap c_{35} = P_{36} = P_{68} = P(\omega^2,\omega^2,\omega,1) = P(3,3,2,1), T = 74$
37 : $E_{45} = a_4 \cap b_5 \cap c_{45} = P_{37} = P_{69} = P(0,0,\omega^2,1) = P(0,0,3,1), T = 83$
38 : $E_{64} = a_6 \cap b_4 \cap c_{46} = P_{38} = P_{70} = P(1,0,\omega^2,1) = P(1,0,3,1), T = 31$
39 : $E_{12,36,45} = c_{12} \cap c_{36} \cap c_{45} = P_{39} = P_{75} = P(\omega,1,\omega^2,1) = P(2,1,3,1), T = 59$
40 : $E_{34} = a_3 \cap b_4 \cap c_{34} = P_{40} = P_{76} = P(\omega^2,1,\omega^2,1) = P(3,1,3,1), T = 71$
41: \( E_{24} = a_2 \cap b_4 \cap c_{24} = P_{41} = P_{79} = P(\omega, \omega, \omega^2, 1) = P(2, 2, 3, 1), \ T = 58 \)
42: \( E_{13,26,45} = c_{13} \cap c_{26} \cap c_{45} = P_{42} = P_{80} = P(\omega^2, \omega, \omega^2, 1) = P(3, 2, 3, 1), \ T = 70 \)
43: \( E_{14} = a_1 \cap b_4 \cap c_{14} = P_{43} = P_{81} = P(0, \omega^2, \omega^2, 1) = P(0, 3, 3, 1), \ T = 72 \)
44: \( E_{16,23,45} = c_{16} \cap c_{23} \cap c_{45} = P_{44} = P_{82} = P(1, \omega^2, \omega^2, 1) = P(1, 3, 3, 1). \ T = 33 \)

Set of tangent planes: \( (0, 4, 20, 84, 27, 1, 2, 3, 5, 10, 15, 9, 6, 12, 18, 21, 25, 22, 46, 24, 67, 23, 41, 26, 30, 53, 80, 55, 79, 62, 36, 49, 76, 51, 39, 50, 74, 83, 31, 59, 71, 58, 70, 72, 33) \)
Line type of Eckardt points: \( 5^{27}, 3^{240}, 1^{90} \)
Plane type of Eckardt points: \( 13^{45}, 9^{40} \)

**Hesse planes**

Number of Hesse planes: 40
Set of Hesse planes: \( (7, 8, 11, 13, 14, 16, 17, 19, 28, 29, 32, 34, 35, 37, 38, 40, 42, 43, 44, 45, 47, 48, 52, 54, 56, 57, 60, 61, 63, 64, 65, 66, 68, 69, 73, 75, 77, 78, 81, 82) \)
subspace 0 / 40 is 7:

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

...:
subspace 39 / 40 is 82:

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

0 : 7 : \( E_{56}, E_{31}, E_{15,24,36}, E_{16,25,34}, E_{12}, E_{14,26,35}, E_{23}, E_{45}, E_{64} \)

...:
39 : 82 : \( E_{41}, E_{52}, E_{16}, E_{12,34,56}, E_{15,24,36}, E_{35}, E_{23}, E_{64}, E_{13,26,45} \)

**Axes**

Number of axes: 240
Axes:
0 : 0 = 0,0 = \( E_{23}, E_{31}, E_{12} \)

...:
239 : 239 = 119,1 = \( E_{12,36,45}, E_{14,26,35}, E_{13,25,46} \)
Tritangent planes

The 45 tritangent planes are:

\[
\pi_{12} = \pi_0 = 79 = \begin{bmatrix} 1 & 0 & 0 & \omega^2 \\ 0 & 1 & 0 & \omega^2 \\ 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 1 \end{bmatrix} = V(\omega^2X_0 + \omega^2X_1 + X_2 + X_3) = V(3X_0 + 3X_1 + X_2 + X_3)
\]
dual pt rank = 52 = (3, 3, 1, 1).

\[
\pi_{16,25,34} = \pi_{44} = 55 = \begin{bmatrix} 1 & 0 & 0 & \omega \\ 0 & 1 & 0 & \omega \\ 0 & 0 & 1 & \omega^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 3 \end{bmatrix} = V(\omega X_0 + \omega X_1 + \omega^2 X_2 + X_3) = V(2X_0 + 2X_1 + 3X_2 + X_3)
\]
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 \n    ▷ ▷ -define P -projective_space -n 3 -field F -v 0 -end \n    ▷ -with P -do \n    ▷ ▷ -projective_space_activity \n    ▷ ▷ ▷ -control_six_arcs -problem_label sixarcs_q4 -end \n    ▷ ▷ ▷ -classify_surfaces_through_arcs_and_two_lines \n    ▷ ▷ -end
    pdflatex surfaces_arc_lifting_4.tex
    open surfaces_arc_lifting_4.pdf
```

classifies all cubic surfaces with 27 lines over the field \(\mathbb{F}_4\) using the algorithm of Karaoglu. The result agrees with the previous algorithm. In PG(3, 4), the only surface with 27 lines is the Hirschfeld surface.
Table 7.8: Activities related to the classification of cubic surfaces

<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>

### 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,

```
$\text{surface\_recognize\_q7\_abcd\_2\_3\_3\_4:}$
```

```bash
$\text{\$\{ORBITER\} -v 3 \$
$\text{\quad -define F -finite\_field -q 7 -end $\}
$\text{\quad -define P -projective\_space -n 3 -field F -v 0 -end $\}
$\text{\quad -with P -do $\}
$\text{\quad -projective\_space\_activity $\}
$\text{\quad -classify\_surfaces\_with\_double\_sixes Surf -W -end $\}
$\text{\quad -end $\}
```

248
identifies the surface (cf. Table 8.5)

\[ X_0^2X_3 + X_1^2X_2 + X_1X_2^2 + X_0X_3^2 + X_1X_2X_3 = 0 \] (7.1)

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

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

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

\[
\begin{align*}
& X_0^2X_2 + X_1^2X_2 + X_1X_2^2 + X_0X_3^2 + X_1X_2X_3 + \delta^{13}X_1X_3^2 + \delta^{13}X_2X_3^2 + \\
& \delta^{11}X_0X_2X_3 + \delta^{12}X_1X_2X_3 = 0,
\end{align*}
\]

and

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

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

$$
\begin{bmatrix}
12 & 13 & 0 & 0 \\
8 & 13 & 0 & 0 \\
0 & 0 & 13 & 0 \\
12 & 13 & 11 & 1
\end{bmatrix}_0
$$

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

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

The command

```
$ORBITER) -v 3 \\
$define F -finite_field -q 8 -end \\
$define P -projective_space -n 3 -field F -v 0 -end \\
$with P -do \\
$projective_space_activity \\
$with Surf27 -do \\
-classify_surfaces_with_double_sixes Surf27 -W -end \\
-end \\
-with Surf27 -do \\
-classification_of_cubic_surfaces_with_double_sixes_activity \\
-recognize \\
-q 8 \\
-by_coefficients "1,6,1,8,1,11,13,1,19"
```
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 \
  -projective_space_activity \
  -classify_surfaces_with_double_sixes Surf27 -W -end \
  -end \
  -with Surf27 -do \
  -classification_of.cubic_surfaces_with_double_sixes_activity \
  -sweep_Cayley \
  -end \
  -print_symbols
```

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

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

In Section 6.1, cubic surfaces in $\text{PG}(3, 2)$ were classified using this Orbiter command:

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

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

\begin{verbatim}
poly_orbits_d3_n3_q2_F2.csv: poly_orbits_d3_n3_q2.csv
  ▶ $\text{(ORBITER)}$ \text{-v 4} \\
  ▶  ▶ -define F -finite_field -q 2 -end \\
  ▶  ▶ -define P -projective_space -n 3 -field F -v 0 -end \\
  ▶  ▶ -with P -do \\
  ▶  ▶ -projective_space_activity \\
  ▶  ▶ -table_of_cubic_surfaces_compute_properties \\
  ▶  ▶  ▶ poly_orbits_d3_n3_q2.csv 2 0 \\
  ▶  ▶ -end
\end{verbatim}

and

\begin{verbatim}
Dickson_q2_analyze: poly_orbits_d3_n3_q2_F2.csv
  ▶ $\text{(ORBITER)}$ \text{-v 4} \\
  ▶  ▶ -define F -finite_field -q 2 -end \\
  ▶  ▶ -define P -projective_space -n 3 -field F -v 0 -end \\
  ▶  ▶ -with P -do \\
  ▶  ▶ -projective_space_activity \\
  ▶  ▶ -cubic_surface_properties_analyze \\
  ▶  ▶  ▶ poly_orbits_d3_n3_q2_F2.csv 2 \\
  ▶  ▶ -end \\
  ▶ #pdflatex poly_orbits_d3_n3_q2_F2_report.tex \\
  ▶ #open poly_orbits_d3_n3_q2_F2_report.pdf
\end{verbatim}

252
To investigate the properties of these surfaces over the extension field $\mathbb{F}_4$, the following two commands can be used:

```
poly_orbits_d3n3q2F4.csv: poly_orbits_d3n3q2.csv
  $$(\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{with P -do} \ \\
  \ -\text{projective_space_activity} \ \\
  \ -\text{table_of_cubic_surfaces_compute_properties} \ \\
  \ poly_orbits_d3n3q2.csv \ 2 \ 0 \ \\
  \ -\text{end}
```

and

```
Dickson_q4.analyze: poly_orbits_d3n3q2F4.csv
  $$(\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{with P -do} \ \\
  \ -\text{projective_space_activity} \ \\
  \ -\text{cubic_surface_properties_analyze} \ \\
  \ poly_orbits_d3n3q2F4.csv \ 2 \ \\
  \ -\text{end}
```

```
pdflatex poly_orbits_d3n3q2F4_report.tex
  open poly_orbits_d3n3q2F4_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 \( q = 17 \):

\[
\text{MAKE_TABLE_OF_CUBIC_SURFACES}=-\text{define } \\
\quad P -\text{projective_space} -n 3 -\text{field F} -v 0 -\text{end} \\
\quad -\text{with P} -\text{do} \\
\quad \quad -\text{projective_space_activity} \\
\quad \quad -\text{table_of_cubic_surfaces} \\
\quad \quad -\text{end}
\]

\[
cubic_surfaces_tables_{17}:
\quad $(\text{ORBITER}) -v 3 \\
\quad \quad -\text{define F} -\text{finite_field} -q 17 -\text{end} \\
\quad \quad $(\text{MAKE_TABLE_OF_CUBIC_SURFACES})
\]

A file \text{table_of_cubic_surfaces_q17_info.csv} is created. The command

\[
cubic_surfaces_table_latex_{17}:
\quad $(\text{ORBITER}) -v 3 -\text{csv_file_latex} 1 \\
\quad \quad \text{table_of_cubic_surfaces_q17_info.csv}
\]

produces a latex table from the csv file.
Chapter 8

Ring Theory

8.1 Polynomials Over Finite Fields

For \( p \) prime, the finite field \( \mathbb{F}_p \) of order \( p \) can be constructed as factorring of the integers modulo \( p \). In this section, we will consider polynomials over \( \mathbb{F}_p \). The ring of polynomials in one variable with coefficients in \( \mathbb{F}_p \) is denoted as \( \mathbb{F}_p[X] \). Table 8.1 lists Orbiter activities for polynomials over finite fields. The activities are finite field activities. For instance, the command

\[
\text{poly \_division:}
\]
\[
\text{\texttt{\$(ORBITER) -v 2 \}}}
\]
\[
\quad \text{\texttt{\texttt{-define F -finite \_field -q 2 -end}}}
\]
\[
\quad \text{\texttt{\texttt{-with F -do}}}
\]
\[
\quad \text{\texttt{\texttt{-finite \_field \_activity}}}
\]
\[
\quad \text{\texttt{\texttt{-polynomial \_division "1,0,0,0,0,0,0,0,0,0,1" "1,0,1,1" -end}}}
\]

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.

\[
\text{poly \_division2:}
\]
\[
\text{\texttt{\texttt{$\{$ORBITER$ -v 2 \}}}}
\]
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-polynomial_division</td>
<td>( A(X) ) ( B(X) )</td>
<td>Polynomial division of ( A(X) ) by ( B(X) ) over ( \mathbb{F}_q ). ( A(X) ) and ( B(X) ) are given as coefficient list, starting from the lowest coefficient.</td>
</tr>
<tr>
<td>-extended_gcd_for_polynomials</td>
<td>( A(X) ) ( B(X) )</td>
<td>Extended gcd for polynomials ( A(X) ) and ( B(X) ) over ( \mathbb{F}_q ). ( A(X) ) and ( B(X) ) are given as coefficient list, starting from the lowest coefficient.</td>
</tr>
<tr>
<td>-polynomial_mult_mod</td>
<td>( A(X) ) ( B(X) ) ( M(X) )</td>
<td>Multiply the polynomials ( A(X) ) and ( B(X) ) modulo ( M(X) ) in ( \mathbb{F}_q[X] ).</td>
</tr>
<tr>
<td>-Berlekamp_matrix</td>
<td>( A(X) )</td>
<td>Computes the rank of the Berlekamp matrix associated to the polynomial ( A(X) ) over ( \mathbb{F}_q ). The polynomial ( A(X) ) is irreducible over ( \mathbb{F}_q ) if the Berlekamp matrix has rank ( d - 1 ) where ( d ) is the degree of ( A(X) ). The Berlekamp matrix is ( F - I ) where ( F ) is the Frobenius matrix and ( I ) is the identity matrix. The Frobenius matrix is the matrix of the Frobenius endomorphism with respect to the standard basis of the polynomial ring: ( 1, X, X^2, \ldots, X^{d-1} ).</td>
</tr>
<tr>
<td>-polynomial_find_roots</td>
<td>( A(X) )</td>
<td>Find the roots of ( A(X) ) over ( \mathbb{F}_q ).</td>
</tr>
<tr>
<td>-make_table_of_irreducible_polynomials</td>
<td>( d )</td>
<td>Produces a list of all irreducible polynomials of degree ( d ) over ( \mathbb{F}_q ).</td>
</tr>
</tbody>
</table>

Table 8.1: Finite Field Activities Related to Polynomials
The command `-extended_gcd_for_polynomials` takes two polynomials $A(X)$ and $B(X)$ and computes polynomials $U(X)$ and $V(X)$ and $G(X)$ such that $G(X)$ is the greatest common divisor of $A(X)$ and $B(X)$ and

$$G(X) = U(X) \cdot A(X) + V(X) \cdot B(X).$$

For instance,

```
poly_gcd:
```

```bash
$ (ORBITER) -v 2 \
  -define F -finite_field -q 2 -end \
  -with F -do \
  -finite_field_activity \
  -extended_gcd_for_polynomials "1,0,0,0,0,0,0,0,0,0,1" "1,0,1,1" -end
```

does not compute

$$U(X) = X + 1, \quad V(X) = X^8 + X^5 + X^4 + X^3 + X, \quad G(X) = 1.$$  

The next command computes

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

```
poly_mult_mod1:
```

```bash
$ (ORBITER) -v 2 \
  -define F -finite_field -q 7 -end \
  -with F -do \
  -finite_field_activity \
  -polynomial_mult_mod "1,2,3" "3,4,5" "6,0,0,1" -end
```

which has a result of

$$X^2 + 4X + 4.$$  

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

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

257
poly_mult_mod2:

```
$ (ORBITER) -v 2 \
  -define F -finite_field -q 7 -end \
  -with F -do \
  -finite_field_activity \
  -polynomial_mult_mod "3,1,2" "5,3,4" "6,0,0,1" -end
```

The result is

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

The finite field \( \mathbb{F}_4 \) can be defined by using polynomial arithmetic over \( \mathbb{F}_2 \) modulo \( X^2 + X + 1 \). Here is a command that computes the three non-trivial products of polynomials:

poly_mult_mod_F4:

```
$ (ORBITER) -v 2 \
  -define F -finite_field -q 2 -end \
  -with F -do \
  -finite_field_activity \
  -polynomial_mult_mod "1,1" "1,1" "1,1,1" -end
$ (ORBITER) -v 2 \
  -define F -finite_field -q 2 -end \
  -with F -do \
  -finite_field_activity \
  -polynomial_mult_mod "0,1" "1,1" "1,1,1" -end
$ (ORBITER) -v 2 \
  -define F -finite_field -q 2 -end \
  -with F -do \
  -finite_field_activity \
  -polynomial_mult_mod "0,1" "0,1" "1,1,1" -end
```

It is possible to use numerical values for polynomials, using the representation in radix \( q \). The following command computes the product of the polynomials 5 and 7 over \( \mathbb{F}_2 \):

mult_polynomials_2_5_7:

```
$ (ORBITER) -v 2 \
  -define F -finite_field -q 2 -end \
  -with F -do \
  -finite_field_activity -mult_polynomials 5 7 -end
pdflatex polynomial_mult_5_7.tex
open polynomial_mult_5_7.pdf
```

The next command performs polynomial long division based on numerical polynomials:
Here is a somewhat larger example for numerical arguments. We wish to multiply 999 by 997 modulo 1033. The first command performs multiplication:

```plaintext
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:

```plaintext
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:

```plaintext
mult_polynomials_1024.999.997_check:
  $(ORBITER) -v 3 \
  ▶ -define F -finite_field -q 1024 -end \
  ▶ -with F -do \\end{quote}
In this last command, the formula $a \times b$ is used and evaluated over $F_{1024}$, 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 $2^{32} - 2$:

```
$\begin{align*}
\text{CRC32\_SPARSE}="1,32,1,26,1,23,1,22,1,16,1,12,1,11, \\
1,10,1,8,1,7,1,5,1,4,1,2,1,1,1,0"
\end{align*}$
```

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

```
\begin{verbatim}
\text{power\_mod\_inverse:} \\
\text{\texttt{(ORBITER) -v 2 \}} \\
\text{\texttt{-define F -finite\_field -q 2 -end \}} \\
\text{\texttt{-define M -vector -field F -sparse 33 $(CRC32\_SPARSE) -end \}} \\
\text{\texttt{-define A -vector -field F -sparse 2 "1,1" -end \}} \\
\text{\texttt{-with F -do \}} \\
\text{\texttt{-finite\_field\_activity \}} \\
\text{\texttt{-polynomial\_power\_mod A $(TWO\_TO\_THE\_32\_MINUS\_2) M \}} \\
\text{\texttt{-end \}}
\end{verbatim}
```

This command produces the polynomial

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

In order to test that this polynomial really is the multiplicative inverse of $X$ modulo CRC32, we perform the following command:

```
\begin{verbatim}
\text{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"
\end{verbatim}
```

```
\begin{verbatim}
\text{mult\_mod\_to\_get\_one:} \\
\text{\texttt{(ORBITER) -v 2 \}} \\
\text{\texttt{-define F -finite\_field -q 2 -end \}} \\
\text{\texttt{-define M -vector -field F -sparse 33 $(CRC32\_SPARSE) -end \}}
\end{verbatim}
```

260
The product is indeed 1.

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

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

computes the Berlekamp matrix associated with the polynomial $X^3 + X + 1$ over $\mathbb{F}_2$. The matrix is

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

Since the matrix has rank 2, the polynomial is irreducible.

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

For instance, the command 

```
search_primitive_poly_2:
$ (ORBITER) \ -v \ 3 \ $
$ \ -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
<table>
<thead>
<tr>
<th>No</th>
<th>Page</th>
<th>Time</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>1002</td>
<td>66</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>1312</td>
<td>118</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>1011</td>
<td>69</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>1132</td>
<td>94</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>1201</td>
<td>97</td>
<td></td>
</tr>
<tr>
<td>Command</td>
<td>Arguments</td>
<td>Purpose</td>
<td></td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------</td>
<td>--------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>-field</td>
<td>label</td>
<td>Specify the field of coefficients.</td>
<td></td>
</tr>
<tr>
<td>-homogeneous_of_degree</td>
<td>$d$</td>
<td>Specify the degree $d$ of polynomials.</td>
<td></td>
</tr>
<tr>
<td>-number_of_variables</td>
<td>$n$</td>
<td>Specify the number $n$ of variables.</td>
<td></td>
</tr>
<tr>
<td>-monomial_ordering_partition</td>
<td></td>
<td>Set monomial ordering to partition ordering.</td>
<td></td>
</tr>
<tr>
<td>-monomial_ordering_lex</td>
<td></td>
<td>Set monomial ordering to lexicographic ordering.</td>
<td></td>
</tr>
<tr>
<td>-variables</td>
<td>label-txt label-tex</td>
<td>Specify variable labels in ascii and in latex.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-cheat_sheet</td>
<td></td>
<td>Create a cheat sheet.</td>
</tr>
<tr>
<td>-ideal</td>
<td>label-txt label-tex label-set</td>
<td>Compute the ideal of the set of points.</td>
</tr>
<tr>
<td>-apply_transformation</td>
<td>eqn elt</td>
<td>Apply the transformation given by the group element to the given equation using the contragredient action on equations.</td>
</tr>
</tbody>
</table>

Table 8.2: Commands to create a multivariate polynomial ring

Table 8.3: Activities for a multivariate polynomial ring

8.2 Multivariate Polynomial Rings

Orbiter can work with multivariate polynomial rings. Table 8.2 lists the commands for creating a multivariate polynomial ring. Table 8.3 lists the activities for a multivariate polynomial ring.

There are two orderings of the monomials which can be chosen:

1. The partition ordering is grouping terms according to the partition that results from the degrees of the variables first, and then applies the lexicographic ordering as a tie breaker.

2. The lexicographic ordering orders the monomials lexicographically.

By default, the partition ordering is used. Table 8.4 shows the monomials in the partition ordering for degrees 1, 2, 3 and 4 in a plane.
Table 8.4: 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*y2*y2*y2+y0*y1*y1*y1*y1*y2+
+6*y0*y1*y1*y2*y2*y2+9*y0*y2*y2*y2*y2*y2"

CREMONA
MAP
Y1="y0*y0*y0*y0*y0*y0*y1*y1*1*y1*y2+
+12*y0*y0*y0*y1*y2*y2*y2+3*y0*y1*y1*y1*y1*y1+
+5*y0*y1*y1*y2*y2*y2+y0*y1*y2*y2*y2*y2"

CREMONA
MAP
Y2="10*y0*y0*y0*y0*y0+y0*+11*y0*y0*y0*y0*y0*y1*y1+
+11*y0*y0*y0*y0*y2*y2+4*y0*y0*y0*y1*y1*y1*
+9*y0*y0*y1*y1*y2*y2+4*y0*y0*y2*y2*y2*y2"

CREMONA
MAP
Y3="0"
<table>
<thead>
<tr>
<th>$h$</th>
<th>mon</th>
<th>vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$X_0^2$</td>
<td>$(2, 0, 0, 0)$</td>
</tr>
<tr>
<td>1</td>
<td>$X_1^2$</td>
<td>$(0, 2, 0, 0)$</td>
</tr>
<tr>
<td>2</td>
<td>$X_2^2$</td>
<td>$(0, 0, 2, 0)$</td>
</tr>
<tr>
<td>3</td>
<td>$X_3^2$</td>
<td>$(0, 0, 0, 2)$</td>
</tr>
<tr>
<td>4</td>
<td>$X_0X_1$</td>
<td>$(1, 1, 0, 0)$</td>
</tr>
<tr>
<td>5</td>
<td>$X_0X_2$</td>
<td>$(1, 0, 1, 0)$</td>
</tr>
<tr>
<td>6</td>
<td>$X_0X_3$</td>
<td>$(1, 0, 0, 1)$</td>
</tr>
<tr>
<td>7</td>
<td>$X_1X_2$</td>
<td>$(0, 1, 1, 0)$</td>
</tr>
<tr>
<td>8</td>
<td>$X_1X_3$</td>
<td>$(0, 1, 0, 1)$</td>
</tr>
<tr>
<td>9</td>
<td>$X_2X_3$</td>
<td>$(0, 0, 1, 1)$</td>
</tr>
</tbody>
</table>

Table 8.5: The partition ordering of monomials of degree 1, 2 and 3 in PG(3, $q$)
Cremona_map:

```
$\text{(ORBITER)} -v 3 \\
$\text{define F -finite_field -q 13 -end} \\
$\text{define P -projective_space -n 2 -field F -v 0 -end} \\
$\text{define R -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 -formula} \\
$\text{"y0" "y_0" "y0,y1,y2"} \\
$\text{define Y1 -formula} \\
$\text{"y1" "y_1" "y0,y1,y2"} \\
$\text{define Y2 -formula} \\
$\text{"y2" "y_2" "y0,y1,y2"} \\
$\text{define Cremona -collection "Y0,Y1,Y2"} \\
$\text{with P -do} \\
$\text{projective_space_activity} \\
$\text{-map R Cremona "}" \\
$\text{-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):

```
arcs_5_2_q11: \\
$\text{(ORBITER)} -v 4 \\
$\text{define F -finite_field -q 11 -end} \\
$\text{define P -projective_space -n 2 -field F -v 0 -end} \\
$\text{with P -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} \\
$\text{-end} \\
$\text{-target_size 5} \\
$\text{-d 2} \\
$\text{-end} \\
```

267
It finds exactly two isomorphism types of arcs. The representative sets are

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

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

```
> arcs_5_2_q11_ideal:
> $(ORBITER) -v 2 \n> -define F -finite_field -q 11 -end \n> -define R -polynomial_ring \n> -field F \n> -number_of_variables 3 \n> -homogeneous_of_degree 2 \n> -monomial_ordering_lex \n> -variables "x0,x1,x2" "x_0,x_1,x_2" \n> -end \n> -define C -combinatorial_objects \n> -file_of_points arcs_5_2_q11_lvl_5 \n> -end \n> -with C -do \n> -combinatorial_object_activity \n> -ideal R \n> -end
```

The ideals are generated by

\[ 7x0*x1 + 5*x0*x2 + 10*x1*x2 \]

and

\[ 4*x0*x1 + 8*x0*x2 + 10*x1*x2, \]

respectively.

Let us consider a smooth cubic surface with 9 lines and 4 Eckardt points. Suppose we have the set of points and we wish to determine the equation of the object. To do so, we first define the object from the given set of points.

```
> PTS_OF_SURFACE_ORBIT211_Q3_L9_E4="\n> 0,1,2,5,7,8,10,14,9,12, \n> 15,3,16,37,31,34,20,19,17,32,36,33"
```

268
Then, we create a ring and compute the ideal:

```
surface_9lines_4E_ideal:
▷ $(ORBITER) -v 2 \\
▷ ▷ -define Pts -vector -dense \\
▷ ▷ ▷ $(PTS_OF_SURFACE_ORBIT211_Q3_L9_E4) \\
▷ ▷ ▷ -end \\
▷ ▷ -define F -finite_field -q 3 -end \\
▷ ▷ ▷ -define R -polynomial_ring \\
▷ ▷ ▷ -define F -finite_field -q 3 -end \\
▷ ▷ ▷ -define R -polynomial_ring \\
▷ ▷ ▷ -number_of_variables 4 \\
▷ ▷ ▷ -homogeneous_of_degree 3 \\
▷ ▷ ▷ -monomial_ordering_lex \\
▷ ▷ ▷ -variables "x0,x1,x2,x3" "x_0,x_1,x_2,x_3" \\
▷ ▷ ▷ -end \\
▷ ▷ -with R -do \\
▷ ▷ ▷ -ring_theoretic_activity \\
▷ ▷ ▷ -ideal "surf_eqn" "surf\_eqn" Pts \\
▷ ▷ ▷ -end
```

We find a two-dimensional ideal. Generators are:

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

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

```
SURFACE_F_9="x_0 \cdot x_0 \cdot x_1 - x_0 \cdot x_1 \cdot x_1 - x_0 \cdot x_1 \cdot x_3 - x_2 \cdot x_2 \cdot x_3 - x_2 \cdot x_3 \cdot x_3"
```

```
F_9_q7:
▷ $(ORBITER) -v 3 \\
▷ ▷ -define F -finite_field -q 7 -end \\
▷ ▷ -define P -projective_space -n 3 -field F -v 0 -end \\
▷ ▷ -define F_9 -cubic_surface -space P \\
▷ ▷ ▷ -by_equation "F_9" \\
▷ ▷ ▷ "\DF_9\D" "x_0,x_1,x_2,x_3" \\
▷ ▷ ▷ $(SURFACE_F_9) \\
▷ ▷ ▷ "\Dno parameters\D" \\
▷ ▷ ▷ -end \\
▷ ▷ -with F_9 -do \\
▷ ▷ -cubic_surface_activity \\
▷ ▷ ▷ -report \\
▷ ▷ ▷ -end
```

```
pdflatex surface_equation_F_9_q7_report.tex
open surface_equation_F_9_q7_report.pdf
```
In the next example, we wish to explore the relationship between conics and \((5, 2)\)-arcs. We consider the plane \(\text{PG}(2, 11)\). 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:

```
rkpg_{2.11}:
  \(\uparrow\) $(\text{ORBITER}) -v 4 \$
  \(\uparrow\) \(\uparrow\) -create_rkpg_{2.11} 133 5 20
```

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

```
line.type.in_PG_2.11:
  \(\uparrow\) $(\text{ORBITER}) -v 3 \$
  \(\uparrow\) \(\uparrow\) -orbit_path $(\text{ORBITER\_PATH}) \$
  \(\uparrow\) \(\uparrow\) -define F -finite_field -q 11 -end \$
  \(\uparrow\) \(\uparrow\) -define P -projective_space -n 2 -field F -v 0 -end \$
  \(\uparrow\) \(\uparrow\) -define C -combinatorial_objects \$
  \(\uparrow\) \(\uparrow\) \(\uparrow\) -file_of_points random_k_subsets_n133_k5_nb20.csv \$
  \(\uparrow\) \(\uparrow\) \(\uparrow\) -end \$
  \(\uparrow\) \(\uparrow\) \(\uparrow\) -with C -do \$
  \(\uparrow\) \(\uparrow\) \(\uparrow\) -combinatorial_object_activity \$
  \(\uparrow\) \(\uparrow\) \(\uparrow\) \(\uparrow\) -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:

```
rkpg_{2.11}_{3.33.40.83.102}:
  \(\uparrow\) $(\text{ORBITER}) -v 2 \$
  \(\uparrow\) \(\uparrow\) -define F -finite_field -q 11 -end \$
  \(\uparrow\) \(\uparrow\) -define R -polynomial_ring \$
  \(\uparrow\) \(\uparrow\) \(\uparrow\) -field F \$
  \(\uparrow\) \(\uparrow\) \(\uparrow\) \(\uparrow\) -number_of_variables 3 \$
  \(\uparrow\) \(\uparrow\) \(\uparrow\) \(\uparrow\) -homogeneous_of_degree 2 \$
  \(\uparrow\) \(\uparrow\) \(\uparrow\) \(\uparrow\) -monomial_ordering_lex \$
  \(\uparrow\) \(\uparrow\) \(\uparrow\) \(\uparrow\) -variables "x0,x1,x2" "x_0,x_1,x_2" \$
  \(\uparrow\) \(\uparrow\) \(\uparrow\) \(\uparrow\) -end \$
  \(\uparrow\) \(\uparrow\) \(\uparrow\) -define C -combinatorial_objects \$
  \(\uparrow\) \(\uparrow\) \(\uparrow\) \(\uparrow\) -set_of_points "3,33,40,83,102" \$
  \(\uparrow\) \(\uparrow\) \(\uparrow\) \(\uparrow\) -end \$
  \(\uparrow\) \(\uparrow\) \(\uparrow\) -with C -do \$
  \(\uparrow\) \(\uparrow\) \(\uparrow\) -combinatorial_object_activity \$
  \(\uparrow\) \(\uparrow\) \(\uparrow\) \(\uparrow\) -ideal R \$
  \(\uparrow\) \(\uparrow\) \(\uparrow\) \(\uparrow\) -end
```

270
The ideal is generated by

\[ 10 \times x_0 \times x_0 + 3 \times x_0 \times x_1 + 8 \times x_0 \times x_2 + 2 \times x_1 \times x_1 + 10 \times x_2 \times x_2. \]

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="\n 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, \n 2,68,6,70,3,77,2,79,6,83,6,120,2,123,5,125,3,130,1,132,3,134,3,141, \n 2,143,6,147,3,156"
```

```
Endrass_F7.txt:
\>$\text{ORBITER}$ -v 2 \n\>$\text{define F}$ -finite_field -q 7 -end \n\>$\text{define R}$ -polynomial_ring -field F \n\>$\text{define eqn}$ -vector -field F -sparse 165 \n\>$\text{define P}$ -projective_space -n 3 -field F -v 0 -end \n\>$\text{define Endrass_F7}$ -geometric_object P \n\>$\text{define Endrass}_F7$ -projective_variety R \n\>$\text{define eqn}$ \n\>$\text{with Endrass}_F7$ -do \n\>$\text{combinatorial_object_activity}$ -save \n\>$\text{end}$
```

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:

```
ocritic
octic_prepare:
\>$\text{ORBITER}$ -v 4 \n\>$\text{define A}$ -vector -format 1 -dense "1,1,1,1" -end \n```

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,

\[ \text{inverse\_mod\_a:} \]
\[ \text{\( \triangledown \)} (\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{\( \triangledown \)} (\text{ORBITER}) \ -v \ 5 \ -\text{jacobi} \ 2221 \ 7817 \]
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-jacobi</td>
<td>$a \ p$</td>
<td>Computes the Jacobi symbol $(\frac{a}{p})$</td>
</tr>
<tr>
<td>-sift_smooth</td>
<td>$a \ n$ primes</td>
<td>Computes all smooth numbers in the interval $[a, a + n - 1]$. Smooth means that they factor completely over the list of primes given.</td>
</tr>
<tr>
<td>-random</td>
<td>$n$ fname</td>
<td>Creates $n$ random numbers and writes them to the csv file $fname$</td>
</tr>
<tr>
<td>-random_last</td>
<td>$n$</td>
<td>Creates $n$ random numbers prints the last one</td>
</tr>
<tr>
<td>-affine_sequence</td>
<td>$a \ b \ p$</td>
<td>Splits the interval $[0, p - 1]$ into affine sequences of the form $x_{n+1} = ax_n + b \mod p$</td>
</tr>
</tbody>
</table>

### Table 9.1: Number Theoretic Commands

computes the Jacobi symbol

\[
\left(\frac{2221}{7817}\right).
\]

In the Jacobi symbol, the denominator $p$ has to be a positive odd integer. This command creates the file `jacobi_2221_7817.tex` which contains a detailed step-by-step description of the computation. The steps correspond to the basic rules for computing the Jacobi symbol and can be found in many textbooks. After reformatting, the description looks like this:

\[
\left(\frac{2221}{7817}\right) = \left(\frac{2221}{2221}\right) \cdot \left(\frac{7817}{2221}\right) \\
= \left(\frac{7817}{2221}\right) \cdot \left(-1\right)^{\frac{2221-1}{2} \cdot \frac{7817-1}{2}} \\
= \left(\frac{7817}{2221}\right) \\
= \left(\frac{1154}{2221}\right) \\
= \left(\frac{2}{2221}\right) \cdot \left(\frac{577}{2221}\right) \\
= \left(-1\right)^{\frac{2221^2-1}{2} \cdot \frac{577}{2221}} \\
= \left(-1\right) \cdot \left(\frac{577}{2221}\right) \\
= \left(-1\right) \cdot \left(\frac{577}{2221}\right) \\
= \left(-1\right) \cdot \left(\frac{2221}{577}\right) \\
= \left(-1\right) \cdot \left(\frac{490}{577}\right)
\]
\[
= (-1) \cdot \left( \frac{2}{577} \right) \cdot \left( \frac{245}{577} \right) \\
= (-1) \cdot (-1)^{\frac{577^2 - 1}{8}} \cdot \left( \frac{245}{577} \right) \\
= (-1) \cdot \left( \frac{245}{577} \right) \\
= (-1) \cdot \left( \frac{577}{245} \right) \cdot (-1)^{\frac{245 - 1}{2} \cdot \frac{577 - 1}{2}} \\
= (-1) \cdot \left( \frac{577}{245} \right) \\
= (-1) \cdot \left( \frac{87}{245} \right) \\
= (-1) \cdot \left( \frac{245}{87} \right) \cdot (-1)^{\frac{87 - 1}{2} \cdot \frac{245 - 1}{2}} \\
= (-1) \cdot \left( \frac{87}{245} \right) \\
= (-1) \cdot \left( \frac{71}{87} \right) \\
= (-1) \cdot \left( \frac{87}{71} \right) \cdot (-1)^{\frac{71 - 1}{2} \cdot \frac{87 - 1}{2}} \\
= \left( \frac{87}{71} \right) \\
= \left( \frac{16}{71} \right) \\
= \left( \frac{2}{71} \right)^4 \cdot \left( \frac{1}{71} \right) \\
= \left( (-1)^{\frac{71^2 - 1}{8}} \right)^4 \cdot \left( \frac{1}{71} \right) \\
= \left( \frac{1}{71} \right) \\
= 1
\]

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 \texttt{square_root_mod} command from Section 3.1 to compute a square root of 2221 modulo 7817 and verify this fact. The command

\texttt{sqrt_mod_7817:}
\texttt{> > $(ORBITER) -v 2 -square_root_mod 2221 7817}

yields that 7634 is a square root. Indeed,

\[7634^2 \equiv 2221 \mod 7817.\]
Command | Arguments | Purpose
--- | --- | ---
-orbits_on_polynomials | $d$ | 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.

Table 9.2: Representation Theory Commands

### 9.2 Representation Theory

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

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

creates $G = \text{PGL}(4,3)$ and computes the representation on polynomials of degree 3 in 4 variables. The representation has degree 20. The second command produces bitmap drawings for the representing matrices associated with a generating set of the group. Figure 9.1 shows the representing matrices for a generating set of size 9.
Figure 9.1: Representation of PGL(4, 3) on cubic polynomials
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-solovay_strassen</td>
<td>$a \ n$</td>
<td>Performs $n$ Solovay / Strassen tests on the number $a$</td>
</tr>
<tr>
<td>-miller_rabin</td>
<td>$a \ n$</td>
<td>Performs $n$ Miller / Rabin tests on the number $a$</td>
</tr>
<tr>
<td>-fermat</td>
<td>$a \ n$</td>
<td>Performs $n$ Fermat tests on the number $a$</td>
</tr>
<tr>
<td>-find_pseudoprime</td>
<td>$a \ n_1 \ n_2 \ n_3$</td>
<td>Computes a pseudoprime which survives $n_1$ Fermat tests, $n_2$ Miller Rabin tests, $n_3$ Solovay Strassen tests</td>
</tr>
<tr>
<td>-find_strong_pseudoprime</td>
<td>$a \ n_1 \ n_2$</td>
<td>Computes a pseudoprime which survives $n_1$ Fermat tests and $n_2$ Miller Rabin tests</td>
</tr>
<tr>
<td>-RSA_encrypt_text</td>
<td>$d \ n \ b \ \text{text}$</td>
<td>Using blocks of $b$ letters at a time, encrypt “text” using RSA with exponent $d$ modulo $n$</td>
</tr>
<tr>
<td>-RSA</td>
<td>$d \ n \ \text{list-of-integers}$</td>
<td>encrypt the given sequence of integers using RSA with exponent $d$ modulo $n$</td>
</tr>
</tbody>
</table>

Table 9.3: Cryptographic Commands

9.3 Cryptography

In Table 9.3, some global cryptographic commands are shown. Some cryptographic commands require a finite field and appear as a finite field activity, see Table 9.4. For instance,

```
EC_add:
▷ $(\text{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:
▷ $(\text{ORBITER}) -v 2 \$
▷ ▷ -define F -finite_field -q 11 -end \$
▷ ▷ -with F -do \$
▷ ▷ -finite_field_activity \$
▷ ▷ -EC_cyclic_subgroup 1 3 "1,4" -end
```

278
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-EC_add</td>
<td>a b i₁ i₂</td>
<td>On the elliptic curve ( y^2 \equiv x^3 + ax + b ) in ( \mathbb{F}_q ), add the points with indices ( i_1 ) and ( i_2 ), each given as a pair ( x, y )</td>
</tr>
<tr>
<td>-EC_points</td>
<td>a b</td>
<td>Computes all points of the elliptic curve ( y^2 \equiv x^3 + ax + b ) over ( \mathbb{F}_q )</td>
</tr>
<tr>
<td>-EC_multiple_of</td>
<td>a b pt n</td>
<td>Computes the ( n ) fold multiple of the given point ( pt ) on the elliptic curve ( y^2 \equiv x^3 + ax + b ) over ( \mathbb{F}_q )</td>
</tr>
<tr>
<td>-EC_cyclic_subgroup</td>
<td>a b pt</td>
<td>Computes the cyclic subgroup generated by the given point ( pt ) on the elliptic curve ( y^2 \equiv x^3 + ax + b ) over ( \mathbb{F}_q )</td>
</tr>
<tr>
<td>-EC_Koblitz_encoding</td>
<td>a b s pt plain</td>
<td>Computes the Koblitz encoding of “plain” (all caps) on the elliptic curve ( y^2 \equiv x^3 + ax + b ) over ( \mathbb{F}_q ) using the base point ( pt ) and the secret exponent ( s )</td>
</tr>
<tr>
<td>-EC_bsgs</td>
<td>a b pt n cipher</td>
<td>Prepare the baby-step giant-step tables for the ciphertext “cipher” on the elliptic curve ( y^2 \equiv x^3 + ax + b ) over ( \mathbb{F}_q ) using the base point ( pt ) of order ( n )</td>
</tr>
<tr>
<td>-EC_bsgs_decode</td>
<td>a b pt n cipher</td>
<td>Decodes the ciphertext “cipher” on the elliptic curve ( y^2 \equiv x^3 + ax + b ) over ( \mathbb{F}_q ) using the base point ( pt ) of order ( n ) and the round keys “keys”</td>
</tr>
<tr>
<td>-EC_discrete_log</td>
<td>a b pt base-pt</td>
<td>Computes the elliptic curve discrete log analogue of ( pt ) with respect to ( base-pt ) on the elliptic curve ( y^2 \equiv x^3 + ax + b ) over ( \mathbb{F}_q )</td>
</tr>
<tr>
<td>-NTRU_encrypt</td>
<td>N p H R M</td>
<td>NTRU encryption for the message ( M(X) ) using the public key ( H(X) ) and one-time-key ( R(X) )</td>
</tr>
<tr>
<td>-polynomial_center_lift</td>
<td>A(X)</td>
<td>Compute the center lift mod ( q ) for the coefficients of ( A )</td>
</tr>
<tr>
<td>-polynomial_reduce_mod_p</td>
<td>p A(X)</td>
<td>Reduce the coefficients of the polynomial ( A ) modulo ( p )</td>
</tr>
</tbody>
</table>

Table 9.4: Finite Field Activities related to Cryptography
Figure 9.2: The elliptic curve $y^2 = x^3 + 5x + 7 \mod 199$

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

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

computes all points on the curve $y^2 = x^3 + 5x + 7 \mod 199$ and produces a bitmap drawing of the points in the affine plane shown in Figure 9.2. Both the $x$-axis and the $y$-axis are indexed by the field elements from 0 to 198.

The command
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

EC̶bsgs̶:  

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

EC̶bsgs̶_decode̶:  

281
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 convention as for field elements in \(\mathbb{F}_q\), Orbiter allows negative coefficients as well. The assumption is that \(q\) is prime and negative coefficients are considered modulo \(q\). In the example, Alice picks the private polynomials \(f(x) = x^6 - x^4 + x^3 + x^2 - 1\) (with \(d + 1\) coefficients equal to plus one and \(d\) coefficients equal to minus one) and \(g(x) = x^6 + x^4 - x^2 - x\) with \(d\) coefficients plus one and \(d\) coefficients minus one. We also need the polynomial \(x^N - 1\). The makefile commands

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

\[
\text{ALICE} \_PRIVATE \_F="-1,0,1,-1,0,1" \\
\text{ALICE} \_PRIVATE \_G="0,-1,-1,0,1,0,1"
\]

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

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

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

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

**NTRU_Alice1:**
```makefile
$(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:**
```makefile
$(ORBITER) -v 2 \\
  -define F -finite_field -q $(NTRU_P) -end \\
  -with F -do \\
  -finite_field_activity \\
  -extended_gcd_for_polynomials \\
  $(NTRUE_XN1) $(ALICE_PRIVATE_F) \\
  -end
```

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

**ALICE_PRIVATE_FP**="1,1,1,1,0,2,1"

There is a chance that the polynomial \( f(x) \) does not have an inverse in either \( \mathbb{F}_p[x] \) or in \( \mathbb{F}_q[x] \). In that case, Alice simply chooses a different polynomial \( f(x) \) and tries again. Alice can now compute her public key:

**NTRU_Alice_public_key:**
```makefile
$(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
```

283
ALICE_PUBLIC_KEY="30,26,8,38,2,40,20"

The public key is assigned to the makefile variable ALICE_PUBLIC_KEY. Now, Bob chooses his message to Alice and his one-time-key. The message must be the center lift of a polynomial in $\mathbb{F}_p[x]$. The round-key must have exactly $d$ coefficients one and $d$ coefficients $-1$ (rest zeroes).

BOB_MESSAGE="1,-1,1,1,0,-1"

BOB_ONE_TIME_KEY="-1,1,0,0,0,-1,1"

The encryption proceeds using the NTRU_encrypt command, and the result is stored in the makefile variable BOB_ENCRYPT:

```
NTRU_encrypt:
  ᵁ $(ORBITER) -v 2 \n  ᵁ ᵁ -define F -finite_field -q $(NTRU_Q) -end \n  ᵁ ᵁ -with F -do \n  ᵁ ᵁ -finite_field_activity \n  ᵁ ᵁ -NTRU_encrypt $(NTRU_N) $(NTRU_P) $(ALICE_PUBLIC_KEY) \n  ᵁ ᵁ $BOB_ONE_TIME_KEY $(BOB_MESSAGE) -end
```

BOB_ENCRYPT= "25,3,40,2,4,19,31"

Decryption is done in five steps.

```
NTRU_decrypt1:
  ᵁ $(ORBITER) -v 2 \n  ᵁ ᵁ -define F -finite_field -q $(NTRU_Q) -end \n  ᵁ ᵁ -with F -do \n  ᵁ ᵁ -finite_field_activity \n  ᵁ ᵁ -polynomial_mult_mod $(ALICE_PRIVATE_F) \n  ᵁ ᵁ ᵁ $(BOB_ENCRYPT) $(NTRUE_XN1) \n  ᵁ ᵁ -end
```

ALICE_C1="40,1,40,40,33,10,1"

```
NTRU_decrypt2:
  ᵁ $(ORBITER) -v 2 \n  ᵁ ᵁ -define F -finite_field -q $(NTRU_Q) -end \n```
\[ \text{ALICE}_C^2=\{-1,1,-1,-8,10,1\} \]

\[ \text{NTRU}\text{.decrypt}_3:\]
\[
\begin{align*}
\text{define F} & \text{-finite field } -q \ (\text{NTRU.P}) \text{-end} \\
\text{-with F} & \text{-do} \\
\text{-finite field activity} \\
\text{-polynomial reduce mod } p \ (\text{ALICE}_C^2) \text{-end}
\end{align*}
\]

\[ \text{ALICE}_C^3=\{2,1,2,2,1,1,1\} \]

\[ \text{NTRU}\text{.decrypt}_4:\]
\[
\begin{align*}
\text{define F} & \text{-finite field } -q \ (\text{NTRU.Q}) \text{-end} \\
\text{-with F} & \text{-do} \\
\text{-finite field activity} \\
\text{-polynomial mult mod } \ (\text{ALICE PRIVATE FP}) \\
\text{-end}
\end{align*}
\]

\[ \text{ALICE}_C^4=\{1,2,1,1,0,2\} \]

\[ \text{NTRU}\text{.decrypt}_5:\]
\[
\begin{align*}
\text{define F} & \text{-finite field } -q \ (\text{NTRU.P}) \text{-end} \\
\text{-with F} & \text{-do} \\
\text{-finite field activity} \\
\text{-polynomial center lift } \ (\text{ALICE}_C^4) \text{-end}
\end{align*}
\]

Decryption produces Bob’s message to Alice.

**ToDo:**

- RSA
- sqrt mod
• quadratic sieve
• pseudoprimes
Chapter 10

Coding Theory

10.1 Introduction

Orbiter supports research in coding theory. Global Orbiter commands for coding theory are summarized in Table 10.1. Additional commands, associated with objects of type code will be discussed below and in later sections.

The command

\begin{verbatim}
Allen_Gates_noise_1_percent:
  $(ORBITER) -v 3 \\
  -random_noise_in_bitmap_file \\
  allen_Gates.bmp \\
  allen_Gates_1.bmp \\
  1 100 \\
  open allen_Gates_1.bmp
\end{verbatim}

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

\begin{verbatim}
Hamming_space_4_2_distance_matrix:
  $(ORBITER) -Hamming_space_distance_matrix 4 2
\end{verbatim}

creates the distance matrix of the Hamming graph $H(n,q)$. The data is written to the file \texttt{Hamming_n4_q2.csv}. The command

\begin{verbatim}
Hamming_space_4_2_distance_matrix_draw:
  $(ORBITER) -v 2 -draw_matrix \\
  -input_csv_file Hamming_n4_q2.csv \\
  -box_width 20 -bit_depth 24
\end{verbatim}
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-make_macwilliams_system</td>
<td>q n k</td>
<td>Create the MacWilliams equations for the weight enumerator of the dual code.</td>
</tr>
<tr>
<td>-table_of_bounds</td>
<td>n_max q</td>
<td>Make a table of bounds for q-ary linear code for all k ≤ n ≤ n_max</td>
</tr>
<tr>
<td>-make_bounds_for_d_given_n_and_k_and_q</td>
<td>n k q</td>
<td>Make bounds for the minimum distance of a [n, k]_q code</td>
</tr>
<tr>
<td>-Hamming_space_distance_matrix</td>
<td>n q</td>
<td>Make the distance matrix of the Hamming graph H(n, q).</td>
</tr>
<tr>
<td>-random_noise_in_bitmap_file</td>
<td>f1 f2 n d</td>
<td>Apply random noise at the d/n level to the bitmap file f1 and write to f2.</td>
</tr>
<tr>
<td>-introduce_errors</td>
<td>CRC-options</td>
<td>Introduce errors to a file. See Table 10.6.</td>
</tr>
<tr>
<td>-check_errors</td>
<td>CRC-options</td>
<td>Find errors in a CRC coded file. See Table 10.6.</td>
</tr>
<tr>
<td>-extract_block</td>
<td>CRC-options</td>
<td>Extract a block from a CRC coded file. See Table 10.6.</td>
</tr>
</tbody>
</table>

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

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

code_5_2_3_diagram:
```
$\text{(ORBITER)} -v 2 \n\text{-define } F \text{-finite_field } -q 2 \text{-end } \n\text{-with } F \text{-do -coding_theoretic_activity } \n\text{-code_diagram } "\text{code}_5\text{.2}_3\text{" } \n\text{-metric_balls } 1 \n\text{-end } \n$\text{(ORBITER)} -v 2 \n\text{-draw_matrix } \n\text{-input_csv_file } \text{code}_5\text{.2}_3\text{.diagram.01}_5\text{.4}.csv \n\text{-box_width } 25 \text{-bit_depth } 24 \n\text{-partition } 4 8 4 \n\text{-end }
```

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

Hamming_5_2_graph:
```
$\text{(ORBITER)} -v 2 \n\text{-define } G \text{-graph } -\text{Hamming } 5 \text{ } 2 \text{-end } \n\text{-with } G \text{-do } \n\text{-graph_theoretic_activity } -\text{export_csv } -\text{end } \n```
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-BCH</td>
<td>$n \ q \ t$</td>
<td>Compute a table of BCH codes of length $n$ over $F_q$.</td>
</tr>
<tr>
<td>-BCH_dual</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-general_code_binary</td>
<td>$n \text{ text}$</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>$\text{fname}$</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>$n \text{ generators}$</td>
<td></td>
</tr>
<tr>
<td>-encode_text_5bits</td>
<td>input $\text{fname}$</td>
<td></td>
</tr>
<tr>
<td>-field_induction</td>
<td>$\text{fname-in} \ \text{fname-out nb-bits}$</td>
<td></td>
</tr>
<tr>
<td>-crc32</td>
<td>$\text{text}$</td>
<td></td>
</tr>
<tr>
<td>-crc32_hexdata</td>
<td>$\text{hexdata}$</td>
<td></td>
</tr>
<tr>
<td>-crc32_test</td>
<td>$\text{block-length}$</td>
<td></td>
</tr>
<tr>
<td>-crc256_test</td>
<td>$\text{message-length} \ \text{R k}$</td>
<td></td>
</tr>
<tr>
<td>-crc32_remainders</td>
<td>msg-length</td>
<td></td>
</tr>
<tr>
<td>-crc32_file_based</td>
<td>$\text{fname-in} \ \text{fname-out block-length}$</td>
<td></td>
</tr>
<tr>
<td>-crc_new_file_based</td>
<td>$\text{fname}$</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>
<tr>
<td>Command</td>
<td>Arguments</td>
<td>Purpose</td>
</tr>
<tr>
<td>---------</td>
<td>-----------</td>
<td>---------</td>
</tr>
<tr>
<td>-minimum_distance</td>
<td>code-object-label</td>
<td>Compute the minimum distance of the linear code object.</td>
</tr>
<tr>
<td>-generator_matrix_cyclic_code</td>
<td>n poly</td>
<td></td>
</tr>
<tr>
<td>-nth_roots</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>-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)
Figure 10.3: Drawing of the Hamming graph $H(5, 2)$

Using the unix dot program, this command sequence creates the drawing of $H(5, 2)$ shown in Figure 10.3.
<table>
<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:
  -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 \
```

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:  
\[
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 \\
\end{bmatrix}
\]
```
It is possible to create the Hamming code by taking the dual of the simplex code. The following command does so:

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 \n  >  >  >  > -end
```

The command also exports the code to magma by means of the magma file Hamming.magma, shown below:

```magma
K<w> := GF(2);
V := VectorSpace(K, 7);
C := LinearCode(sub<V | [1,1,1,0,0,0,0], [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 \n  >  > -end \
  >  > -with C -and F -do -coding_theoretic_activity \n  >  >  > -export_codewords RM_3_1_codewords.csv \
  >  >  > -end \
  >  >  > -with C -and F -do -coding_theoretic_activity \n  >  >  >  > -export_genma RM_3_1_genma.csv \
  >  >  >  > -end
```

296
Alternatively, we can store the generator matrix in a makefile variable:

```
CODE_RM_3_1_GENMA="\n11111111\n01010101\n00110011\n00001111"
```

The following command creates the Hamming code from its generator matrix directly:

```
RM_3_1_from_generator_matrix:
  $(ORBITER) -v 2 \
  -define F -finite_field -q 2 -end \
  -define genma -vector -format 8 -field F \
  -compact $(CODE_RM_3_1_GENMA) \
  -end \
  -define C -code -field F \
  -linear_code_through_generator_matrix genma \
  -end
  #pdflatex code_n8_k4_q2.tex
  #open code_n8_k4_q2.pdf
```

The following command creates the Hamming code and produces a list of codewords.

```
RM_3_1_and_codewords:
  $(ORBITER) -v 2 \
  -define F -finite_field -q 2 -end \
  -define C -code -field F -first_order_Reed_Muller 3 -end \
  -with C -and F -do -coding_theoretic_activity \
  -export_magma RM_3_1.magma \
  -end \
  -with C -and F -do -coding_theoretic_activity \
  -export_codewords RM_3_1_codewords.csv \
  -end \
  -with C -and F -do -coding_theoretic_activity \
  -export_genma RM_3_1_genma.csv \
  -end
```

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

This produces the following output:

Strong generators for a group of order 7:

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

0
1
2
3
4
5
6

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

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

```
SIMPLEX_CODE_GENMA_CYCLIC="\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 \\
▸ ▸ ▸ ▸ -dense $(HAMMING_CODE_GENERATOR) \\
▸ ▸ ▸ -end \\
▸ ▸ -define C -code -field F \\
▸ ▸ ▸ -linear_code_through_generator_matrix v \\
▸ ▸ ▸ -end \\
▸ ▸ -with C -do \\
▸ ▸ -coding_theoretic_activity \\
▸ ▸ ▸ -weight Enumerator \\
▸ ▸ ▸ -end
```
We find that the weight enumerator is

\[(1, 0, 0, 7, 7, 0, 0, 1)\].

The next command computes the minimum distance of the code:

```
Hamming_minimum_distance:
  ▶ $(ORBITER) -v 2 \
  ▶ ▶ -define F -finite_field -q 2 -end \
  ▶ ▶ -define v -vector -field F -format 4 \
  ▶ ▶ ▶ -dense $(HAMMING_CODE_GENERATOR) \
  ▶ ▶ -end \
  ▶ ▶ -with F -do \
  ▶ ▶ -coding_theoretic_activity \
  ▶ ▶ ▶ -minimum_distance v \
  ▶ ▶ -end
```

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

```
Golay23_minimum_distance:
  ▶ $(ORBITER) -v 2 \
  ▶ ▶ -define F -finite_field -q 2 -end \
  ▶ ▶ -define v -vector -field F -format 12 \
  ▶ ▶ ▶ -dense $(GOLAY23_CODE_GENERATOR) \
  ▶ ▶ -end \
  ▶ ▶ -with F -do \
  ▶ ▶ -coding_theoretic_activity \
  ▶ ▶ ▶ -minimum_distance v \
  ▶ ▶ -end
```
10.3 Golay Codes

The Golay code of length 23 is a perfect code of dimension 12 and minimum distance 7. The metric balls of radius three centered around codewords cover the whole Hamming space. We can create the code by listing the columns of a generator matrix in Orbiter ranks of points in PG(11,2). The following makefile variable does that:

\begin{verbatim}
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"
\end{verbatim}

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

\begin{verbatim}
Golay23_code_words:
▷ $(ORBITER) -v 2 \n▷ ▷ -define v -vector -dense $(GOLAY_23_COLUMN_RANKS_PROJECTIVELY) -end \n▷ ▷ -define F -finite_field -q 2 -end \n▷ ▷ -define C -code -field F \n▷ ▷ ▷ -linear_code_from_from_projective_set 12 v -end \n▷ ▷ ▷ -with C -and F -do -coding_theoretic_activity \n▷ ▷ ▷ ▷ -export_magma Golay23.magma \n▷ ▷ ▷ ▷ -end \n▷ ▷ ▷ -with C -and F -do -coding_theoretic_activity \n▷ ▷ ▷ ▷ -export_codewords Golay23_codewords.csv \n▷ ▷ ▷ ▷ -end \n▷ ▷ ▷ -with C -and F -do -coding_theoretic_activity \n▷ ▷ ▷ ▷ -export_genma Golay23_genma.csv \n▷ ▷ ▷ ▷ -end 
▷ #pdflatex code_n23_k12_q2.tex
▷ #open code_n23_k12_q2.pdf
\end{verbatim}
<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:**

```bash
$ (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 \\
```

This creates the following output:

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

11011011110000110111111010000101
==================================
1101 | 1010110100110101010111000010111100
1101
====
11110101101101010111000010111100
1101
====
10101001101010111000010111100
1101
====
1111001101010111000010111100
1101
====
10001101010111000010111100
1101
====
10111010100111000010111100
1101
====
110101010111000010111100
1101
====
1110101111000010111100
1101
```

304
The remainder after division by the check polynomial is 5, or the polynomial $X^2 + 1$, or the bit-sequence 101.

The following command investigates all 1-bit errors, to see which of them can be detected using the given CRC-polynomial:

```
encode_text_5bits_1error:
  $ (ORBITER) -encode_text_5bits \n  "Hithere" "text.csv"
  $(ORBITER) -v 2 \n  $ -define F -finite_field -q 2 -end \n  $ -with F -do \n```
The following output is created:

<table>
<thead>
<tr>
<th>Index</th>
<th>Binary String</th>
<th>Coefficient</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>01010110100110101010111000010111100</td>
<td>100</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>01010110100110101010111000010111101</td>
<td>111</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>01010110100110101010111000010111000</td>
<td>001</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>01010110101010101010111000010111100</td>
<td>000</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>01010110101010101010111000010111000</td>
<td>010</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>01010110101010101010111000010111100</td>
<td>110</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>01010110101010101010111000010111000</td>
<td>001</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>01010110101010101010111000010111000</td>
<td>000</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>01010110101010101010111000010111100</td>
<td>010</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>01010110101010101010111000010111000</td>
<td>001</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>01010110101010101010111000010111100</td>
<td>000</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>01010110101010101010111000010111100</td>
<td>010</td>
<td>2</td>
</tr>
<tr>
<td>12</td>
<td>01010110101010101010111000010111100</td>
<td>001</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>01010110101010101010111000010111100</td>
<td>000</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>01010110101010101010111000010111100</td>
<td>010</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>01010110101010101010111000010111100</td>
<td>001</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>01010110101010101010111000010111100</td>
<td>000</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>01010110101010101010111000010111100</td>
<td>010</td>
<td>2</td>
</tr>
<tr>
<td>18</td>
<td>01010110101010101010111000010111100</td>
<td>001</td>
<td>1</td>
</tr>
<tr>
<td>19</td>
<td>01010110101010101010111000010111100</td>
<td>000</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>01010110101010101010111000010111100</td>
<td>010</td>
<td>2</td>
</tr>
<tr>
<td>21</td>
<td>01010110101010101010111000010111100</td>
<td>001</td>
<td>1</td>
</tr>
<tr>
<td>22</td>
<td>01010110101010101010111000010111100</td>
<td>000</td>
<td>0</td>
</tr>
<tr>
<td>23</td>
<td>01010110101010101010111000010111100</td>
<td>010</td>
<td>2</td>
</tr>
<tr>
<td>24</td>
<td>01010110101010101010111000010111100</td>
<td>001</td>
<td>1</td>
</tr>
<tr>
<td>25</td>
<td>01010110101010101010111000010111100</td>
<td>000</td>
<td>0</td>
</tr>
<tr>
<td>26</td>
<td>01010110101010101010111000010111100</td>
<td>010</td>
<td>2</td>
</tr>
<tr>
<td>27</td>
<td>01010110101010101010111000010111100</td>
<td>001</td>
<td>1</td>
</tr>
<tr>
<td>28</td>
<td>01010110101010101010111000010111100</td>
<td>000</td>
<td>0</td>
</tr>
<tr>
<td>29</td>
<td>01010110101010101010111000010111100</td>
<td>010</td>
<td>2</td>
</tr>
<tr>
<td>30</td>
<td>01010110101010101010111000010111100</td>
<td>001</td>
<td>1</td>
</tr>
<tr>
<td>31</td>
<td>01010110101010101010111000010111100</td>
<td>000</td>
<td>0</td>
</tr>
<tr>
<td>32</td>
<td>01010110101010101010111000010111100</td>
<td>010</td>
<td>2</td>
</tr>
<tr>
<td>33</td>
<td>01010110101010101010111000010111100</td>
<td>001</td>
<td>1</td>
</tr>
</tbody>
</table>

The output shows various binary strings and their corresponding polynomial expressions, along with the resulting coefficients and values obtained from polynomial division operations on all possible k-bit error patterns.
It shows that 5 single bit errors are undetected.

The following command performs an exhaustive search over all binary CRC polynomials of degree $k = 10$ which can detect every error pattern of Hamming weight at most $t = 3$ in messages of length $n = 128$.

CRC$_{3, 128, 10}$:

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

The program finds 244 polynomials in about 1 minute.

Here is a collection of CRC polynomials from various sources:

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

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

CRC8_ATM="1,8,1,2,1,1,1,0"

CRC16_CCITT="1,16,1,12,1,5,1,0"

CRC32ETHERNET="1,32,1,26,1,23,1,22,1,16,1,12,1,11,1,10,1,8,1,7,1,5,1,4,1,2,1,1,1,0"

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

CRC64_ECMA182="1,64,1,62,1,57,1,55,1,54,1,53,1,52,1,47,1,46,1,45,1,40,1,39,1,38,1,37,1,35,1,33,1,32,1,31,1,29,1,27,1,26,1,24,1,23,1,22,1,21,1,19,1,17,1,13,1,12,1,10,1,9,1,7,1,4,1,1,1,0"

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"

307
We test whether the polynomial crc32 is irreducible:

```bash
# CRC32_Berlekamp_matrix:
$ (ORBITER) -v 2 \
  -define F -finite_field -q 2 -end \
  -define v -vector -field F -sparse 33 $(CRC32_ETHETERNET) -end \
  -with F -do \
  -finite_field_activity \
  -Berlekamp_matrix v \
  -end
```

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

```bash
# CRC_F256_roots_771:
$ (ORBITER) -v 3 \
  -define F -finite_field -q 256 -end \
  -with F -do -coding_theoretic_activity \
  -nth_roots 771 \
  -end
```

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

```bash
# CRC_F256_BCH_code_d2:
$ (ORBITER) -v 2 \
  -define F -finite_field -q 256 -end \
  -define C -code -field F \
  -BCH 771 2 \
  -end \
  -with C -and F -do -coding_theoretic_activity \
  -export_magma BCH_lq8_n771_d2.magma \
  -end \
pdflatex BCH_codes_q256_n771_d2.tex \
open BCH_codes_q256_n771_d2.pdf
```

The polynomial in dense coding

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

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

```bash
# CRC_F256_BCH_write_code_for_division_d2:
$ (ORBITER) -v 2 \
```

308
We create a BCH code of length 771 over $\mathbb{F}_{256}$ with designed distance 16:

\begin{verbatim}
F256_BCH_code_d16:
$($(ORBITER) -v 3 \n-define F -finite_field -q 256 -end \n-define A -vector -field F -sparse 772 "1,771,1,0" -end \n-define B -vector -field F -dense $(CRC_POLY Q256 DEG2 DENSE) -end \n-with F -do \ncoding_theoretic_activity \n  write_code_for_division \n  alfa A B \n-end

g++ crc_alfa.cpp -o crc_alfa.out
./crc_alfa.out
\end{verbatim}

The polynomial in sparse coding is:

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

The polynomial in dense coding is:

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

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

\begin{verbatim}
F256_BCH_write_code_for_division_d16:
$($(ORBITER) -v 2 \n-define F -finite_field -q 256 -end \n-define A -vector -field F -sparse 772 "1,771,1,0" -end \n-define B -vector -field F -dense $(POLY_Q256_DEG30_DENSE) -end \n-with F -do \n\end{verbatim}
We confirm that the polynomial divides $X^{771} - 1$ as it should:

F256_BCH_code_d16.division:

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

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

F256_BCH_code_d16.error:

```
$ (ORBITER) -v 2 \
  -define F -finite_field -q 256 -end \
  -define A -vector -field F -sparse 771 "2,30,3,31,55,770" -end \
  -define B -vector -field F -dense $(POLY_Q256_DEG30_DENSE) -end \
  -with F -do \
  -finite_field_activity \
  -polynomial_division A B -end
```
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 \
  > -with F -do \
  >  -coding_theoretic_activity \
  >  -code_diagram "RM_3_1" \
  >  $(REED_MULLER_3_1_CODEWORDS) 8 \
  >  -metric_balls 1 \
  >  -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 \
  >  -input_csv_file RM_3_1_holes_8_16.csv \
  >  -box_width 25 -bit_depth 8 \
  >  -partition 4 16 16 \
  >  -end \
  $(ORBITER) -v 2 \
  > -draw_matrix \
  >  -input_csv_file RM_3_1_diagram_01_8_16.csv \
  >  -box_width 25 -bit_depth 8 \
  >  -partition 4 16 16 \
  >  -end \
  $(ORBITER) -v 2 \
  > -draw_matrix \
  >  -input_csv_file RM_3_1_diagram_8_16.csv \
  >  -box_width 25 -bit_depth 8 \
  >  -partition 4 16 16 \
  >  -end \
  open RM_3_1_diagram_8_16_draw.bmp
```
Figure 10.5: Boolean function representation of $\text{RM}_{3,1}$ in $H(8, 2)$
10.6 BCH Codes

Let $\beta$ be an $n$-th root of unity over $\mathbb{F}_q$. The minimum polynomial of $\beta$ over $\mathbb{F}_q$ is denoted as $m_{\beta, \mathbb{F}_q}$. The BCH code of length $n$ and designed distance $d$ is the cyclic code with generator polynomial

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

To create the polynomial $m_{\beta^a, \mathbb{F}_q}$, we consider the $q$-cyclotomic set of $a$ modulo $n$, which is

$$\{aq^i \mod n \mid i \in \mathbb{Z}\}.$$

Suppose we want to make a BCH-code of length 21 over $\mathbb{F}_8$. In Section 3.3, we considered the $q$-cyclotomic sets modulo 21 for $q = 8$. Let us produce a pictorial representation. Omitting the singletons, a transversal is given by the sets containing 1, 2, 4, 5, 7, 10, 13. For this reason, we issue the command

```bash
draw_cyclootomic_mod_21_q8:
  ▷ $(ORBITER) -v 2 \n  ▷ -draw.options \n  ▷ -radius 100 \n  ▷ -line_width 1.0 -embedded \n  ▷ -end \n  ▷ -draw_mod_n -n 21 -file mod_21_cyclootomic \n  ▷ -cyclotomic_sets 8 "1,2,4,5,7,10,13" -end
  ▷ pdflatex mod_21_cyclootomic_draw.tex
  ▷ open mod_21_cyclootomic_draw.pdf
```

The output is shown in Figure 10.6. We will try BCH-codes with minimum distances 3, 5 and 7. Here is distance 3:

```bash
F_8.BCH_code_d3:
  ▷ $(ORBITER) -v 3 \n  ▷ -define F -finite_field -q 8 -override_polynomial 11 -end \n  ▷ -with F -do \n  ▷ -coding_theoretic_activity \n  ▷ -make_BCH_code 21 3 \n  ▷ -end
  ▷ pdflatex BCH_codes_q8_n21_d3.tex
  ▷ open BCH_codes_q8_n21_d3.pdf
```

The code is described in a latex output file:

313
BCH-code:
\( n = 21, \, k = 17, \, d_0 = 3, \, q = 8, \)
\( g(x) = m_1 m_2 = X^4 + 4X^3 + 4X^2 + 3X + 4 \)
Chosen cyclotomic sets:
\{ 1, 8 \}
\{ 2, 16 \}
The generator polynomial has degree 4

- dense "4,3,4,4,1"

- sparse "4,0,3,1,4,2,4,3,1,4"
The generator matrix is:

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

And now for \( d = 5 \):

\texttt{F\_8\_BCH\_code\_d5:}

\begin{verbatim}
▷ $(\text{ORBITER})$ -v 3 \n▷ ▷ -define F -finite_field -q 8 -override_polynomial 11 -end \n▷ ▷ -with F -do \n▷ ▷ ▷ -coding_theoretic_activity \n▷ ▷ ▷ ▷ -make_BCH_code 21 5 \n▷ ▷ ▷ -end \n▷ pdflatex BCH\_codes\_q8\_n21\_d5.tex \n▷ open BCH\_codes\_q8\_n21\_d5.pdf
\end{verbatim}

The output file is:

\begin{verbatim}
BCH-code:
n = 21, k = 14, d_0 = 5, q = 8,
g(x) = m_1 m_2 m_3 m_4 = X^7 + 3X^6 + 3X^5 + 2X^4 + X^3 + 2X^2 + X + 2
Chosen cyclotomic sets:
\{ 1, 8 \}
\{ 2, 16 \}
\end{verbatim}
The generator polynomial has degree 7

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

The generator matrix is:

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

We compute the minimum distance:

```bash
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_OVERRIDGE_POLYNOMIAL11) \ 
> -end \ 
> -with F -do \ 
> -coding_theoretic_activity \ 
> -minimum_distance v \ 
> -end
```

# important: use the same polynomial as when creating the code.
#
# d=5

316
The minimum distance turns out to be \( d = 5 \).

Finally, we create the BCH code with minimum distance \( d = 7 \):

\[ F_{8\text{-BCH\_code\_d7}}: \]
\[ \texttt{\$\(\text{ORBITER}\) -v 3 \backslash} \]
\[ \texttt{\quad -define F -finite\_field -q 8 -override\_polynomial 11 -end \backslash} \]
\[ \texttt{\quad -with F -do \backslash} \]
\[ \texttt{\quad \quad -coding\_theoretic\_activity \backslash} \]
\[ \texttt{\quad \quad -make\_BCH\_code 21 7 \backslash} \]
\[ \texttt{\quad \quad -end} \]

The output file is:

```
BCH-code:
\( n = 21, k = 11, d_0 = 7, q = 8, \)
\( g(x) = m_1m_2m_3m_4m_5m_6 = X^{10} + X^9 + 2X^8 + 5X^7 + 2X^6 + 4X^4 + 6X^3 + 5X^2 + 6X + 6 \)
\( \) Chosen cyclotomic sets:
\{ 1, 8 \}  
{ 2, 16 \}  
{ 3 \}  
{ 4, 11 \}  
{ 5, 19 \}  
{ 6 \}  
\( \) The generator polynomial has degree 10

-\text{dense} "6,6,5,6,4,0,2,5,2,1,1"

-\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:

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

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:

\[
\text{\$\text{ORBITER} \ -v \ 2 \ \backslash \ \\
\quad -\text{draw.options} \ -\text{nodes_empty} \ -\text{radius \ 10} \ \backslash \ \\
\quad -\text{line_width \ 0.4} \ -\text{embedded} \ -\text{end} \ \\
\quad -\text{draw.mod_n} \ -n \ 255 \ -\text{file \ mod_255_cyclotomic_1_and_3} \ \\
\quad -\text{cyclo\text{m}otic_sets \ 2 \ "1,3"} \ -\text{end} \ \\
\quad \text{pdflatex \ mod_255_cyclotomic_1_and_3\_draw.tex} \ \\
\quad \text{open \ mod_255_cyclotomic_1_and_3\_draw.pdf}
\]

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:

\[
\text{\$\text{ORBITER} \ -v \ 3 \ \backslash \ \\
\quad -\text{define F} \ -\text{finite_field} \ -\text{q \ 256} \ -\text{end} \ \\
\quad -\text{with F} \ -\text{do} \ -\text{coding_theoretic_activity} \ \\
\]

318
The next command creates a BCH-code of length 771 over \( \mathbb{F}_{256} \) with minimum distance at least 16:

```
\$ (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

\[
y^6 + 120x^3y^3 + 360x^4y^2 + 972x^5y + 948x^6.
\]
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 \\
0561000 \\
0056100 \\
0005610 \\
0000561"
```

```
RREF_RS_8_weight_enumerator:
\$\{ORBITER\} -v 2 \
   -define F -finite_field -q 8 -end \
   -define v -vector -format 5 -field F \
   -compact $\{CODE_RS_8\} \
   -end \
   -define C -code -field F \
   -linear_code_through_generator_matrix v \
   -end \
   -with C -do \
   -coding_theoretic_activity \
   -weight Enumerator \
   -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.
The reduced matrix is shown in Figure 10.8. Let us compute the weight enumerator of the reduced code. The command

```
RS_8.reduced="\n010001100000000000000000\n001110010000000000000000\n110011001000000000000000\n000010001100000000000000\n000001110010000000000000\n000001110010000000000000\n"
```
RREF_RS_8_reduced_weightEnumerator:
▷ $(ORBITER) -v 2 \\
▷ ▷ -define F -finite_field -q 2 -end \\
▷ ▷ -define v -vector -format 15 -field F \\
▷ ▷ ▷ -compact $(RS_8\_reduced) \\
▷ ▷ -end \\
▷ ▷ -define C -code -field F \\
▷ ▷ ▷ -linear_code_through_generator_matrix v \\
▷ ▷ -end \\
▷ ▷ -with C -do \\
▷ ▷ -coding_theoretic_activity \\
▷ ▷ ▷ -weightenumerator \\
▷ ▷ -end

computes the weight enumerator of the binary code. It is

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

In particular, the field reduced Reed-Solomon code is not optimal. It has minimum distance three, but there are codes of minimum distance 4. Here is one. We store the code to a file and then draw the generator matrix as bitmap.

CODE_21.15.4="\n111000100000000000000000  \
110100010000000000000000  \
101100001000000000000000  \
011100000100000000000000  \
110010000010000000000000  \
101010000001000000000000  \
011010000000100000000000 &
We compute the weight enumerator

\[
\begin{align*}
1y^{21} + 221x^4y^{17} + 1000x^6y^{15} + 6498x^8y^{13} + 10912x^{10}y^{11} + 9250x^{12}y^9 + \\
3584x^{14}y^7 + 669x^{16}y^5 + 32x^{18}y^3 + 1x^{20}y.
\end{align*}
\]

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:
To compute the minimum distance of the code, we do:

```
CODE_GV_N15_K6="\n11111111100000\n11111000010000\n11100110001000\n11010100000100\n101011000010\n101101000001"
```

GV_n15_k6_d5_weight Enumerator:

```
$\$(ORBITER) -v 2 \\
    -define F -finite_field -q 2 -end \\
    -define v -vector -format 6 -field F \\
    -compact $(CODE_GV_N15_K6) \\
    -end \\
    -define C -code -field F \\
    -linear_code_through_generator_matrix v \\
    -end \\
    -with C -do \\
    -coding_theoretic_activity \\
    -weight Enumerator \\
    -end
```

The weight enumerator is

\[
y^{15} + 27x^6y^9 + 24x^8y^7 + 9x^{10}y^5 + 3x^{12}y^3.
\]

From this, we see that the code has minimum distance 6, which is better than predicted.
10.9 Classification of Optimal Linear Codes

The classification problem of optimal codes in coding theory is the problem of determining the equivalence classes of codes for a given set of values of \( n \) and \( k \) and \( q \) with a lower bound on \( d \). Orbiter can be used to classify linear codes with given redundancy and bounded minimum distance. The redundancy of a linear \([n, k]\) code is the parameter \( r = n - k \). Codes with redundancy \( r \) can be identified with subsets of \( \text{PG}(r-1, q) \). Under this correspondence, a code with minimum distance at least \( d \) corresponds to a subset such that any \( d-1 \) elements are independent. We use the notation \( \Lambda_{r-1,s}(q) \) to denote the poset of subsets of \( \text{PG}(r-1, q) \) for which any \( d-1 \)-subset (if any) is independent. Under the correspondence, the action of \( \text{PGL}(r, q) \) on \( \Lambda_{r-1,s}(q) \) corresponds to the orbits of equivalent linear codes. For this reason, we are interested in determining the orbits of \( \text{PGL}(r, q) \) on \( \Lambda_{r-1,s}(q) \). An orbit of size \( n \) represents an isometry class of \([n, n-r, d; q]\) codes with \( d \geq s + 1 \). The projective stabilizer of the subset is the automorphism group of the code.

The Orbiter command

\begin{verbatim}
codes_8_4_4:
  $\text{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 \ 
  -linear_codes 3 8 \ 
  -end
\end{verbatim}

classifies linear codes with redundancy 4 and minimum distance at least 4. Orbiter confirms that there is exactly one such code, and it computes the code together with the projective stabilizer. Orbiter creates the action of the group \( \text{PGL}(4, 2) \) on the poset \( \Lambda_{3,3}(2) \). Using poset classification, Orbiter then produces the poset of orbits shown in Figure 10.9. In this diagram, the numbers stand for Orbiter ranks of points in \( \text{PG}(3, 2) \). All nodes except for the root node have a number attached to it. The nodes represent subsets. In order to determine the set associated to a node, follow the path from the root node to the node and collect the points according to their labels. The root node represents the empty set. The \([8, 4, 4; 2]\)-code is represented by the set \( \{0, 1, 2, 3, 8, 11, 13, 14\} \). The fact that there is only one node at level
Figure 10.9: Orbits of $\text{PGL}(4, 2)$ 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

```
Sym_10_conj_classes:
$ (ORBITER) -v 2 -conjugacy_classes_Sym_n 10
```

produces a list of the conjugacy classes of Sym(10). The list is written to a csv file. A pie chart of the class size distribution is shown in Fig. 11.1.

The next command computes the character table of the symmetric group Sym(4):

```
Char_Sym_4:
$ (ORBITER) -v 2 -character_table_symmetric_group 4
```

The command produces the following output:

```
The character table of Sym(4) is the matrix

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

The following command creates the character table of Sym(4).
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-random_permutation</td>
<td>n fname</td>
<td>Creates a random permutation in $\text{Sym}(n)$ and stores it in the given file.</td>
</tr>
<tr>
<td>-create_random_k_subsets</td>
<td>$n \ k \ N$</td>
<td>Creates $N$ random $k$-subsets of an $n$-set.</td>
</tr>
<tr>
<td>-read_poset_file</td>
<td>fname</td>
<td>Reads a poset from the given file.</td>
</tr>
<tr>
<td>-read_poset_file_with_grouping</td>
<td>fname x-stretch</td>
<td>Reads a poset from the given file and sets stretch factor for orbit grouping.</td>
</tr>
<tr>
<td>-list_parameters_of_SRG</td>
<td>$v_{\text{max}}$</td>
<td>Performs a sift for putative parameter sets of SRGs.</td>
</tr>
<tr>
<td>-conjugacy_classes_Sym_n</td>
<td>$n$</td>
<td>Compute a list of conjugacy classes of $\text{Sym}(n)$.</td>
</tr>
<tr>
<td>-tree_of_all_k_subsets</td>
<td>$n \ k$</td>
<td>Creates a tree-file for all $k$-subsets of an $n$-set.</td>
</tr>
<tr>
<td>-Delandtsheer_Doyen</td>
<td></td>
<td>See Section 11.7.</td>
</tr>
<tr>
<td>-tdo_refinement</td>
<td></td>
<td>See Section 11.8.</td>
</tr>
<tr>
<td>-tdo_print</td>
<td></td>
<td>See Section 11.8.</td>
</tr>
<tr>
<td>-convert_stack_to_tdo</td>
<td></td>
<td>See Section 11.8.</td>
</tr>
<tr>
<td>-maximal_arc_parameters</td>
<td></td>
<td>See Section 11.8.</td>
</tr>
<tr>
<td>-arc_parameters</td>
<td></td>
<td>See Section 11.8.</td>
</tr>
<tr>
<td>-pentomino_puzzle</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 11.1: Commands related to Combinatorics (Part 1)
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-draw_layered_graph</td>
<td>options</td>
<td>Draws a graph.</td>
</tr>
<tr>
<td>-make_elementary_symmetric_functions</td>
<td>n k_{max}</td>
<td>Computes the elementary symmetric functions in n variables of degree 1,...,k_{max}</td>
</tr>
<tr>
<td>-Dedekind_numbers</td>
<td>n_{min} n_{max} q_{min} q_{max}</td>
<td>Computes the Dedekind numbers $D_{n,q}$ for $n_{min} \leq n \leq n_{max}$ and $q_{min} \leq q \leq q_{max}$</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 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 $\text{Sym}(10)$ arranged by size

Char_Sym_4:
▶ $(\text{ORBITER})$ -v 2 -character_table_symmetric_group 4

The following command illustrates how to create random $k$-subsets of a set of size $n$. In the example, we create 20 5-subsets of a 10-element set:

random_k_subsets:
▶ $(\text{ORBITER})$ -v 4 \n▶ ▶ -create_random_k_subsets 10 5 20

Using the lexicographic order, the $k$-subsets of an $n$-element set are ranked. The following command computes the ranks of a number of 3-subsets of a 10-element set:

rank_k_subsets_test:
▶ $(\text{ORBITER})$ -v 2 \n▶ ▶ -rank_k_subset 10 3 0,1,2,0,3,4,1,3,5,2,4,5,3,6,7,1,6,8,0,6,9

Orbiter can create the Sylvester type Hadamard matrix of size $2^n$ (also called the Walsh matrix). The following command creates the matrix of size $2^4 \times 2^4$ and produces a graphical representation:
Walsh_matrix_4:

- $(\text{ORBITER})$ -v 3 \ 
- -define F -finite_field -q 2 -end \ 
- -with F -do -finite_field_activity \ 
- -Walsh_matrix 4 -end \ 
- $(\text{ORBITER})$ -v 2 -draw_matrix \ 
- -input_csv_file Walsh_01_4.csv \ 
- -box_width 10 -bit_depth 24 -partition 3 16 16 -end \ 
- #pdflatex GF_2.tex \ 
- #open GF_2.pdf

The following command creates the matrix of Dedekind numbers of order at most 10:

Dedekind_10_10:

- $(\text{ORBITER})$ -v 3 -Dedekind_numbers 2 10 2 10

The following command creates the elementary symmetric functions in 4 variables.

elementary_symmetric_functions_4:

- $(\text{ORBITER})$ -make_elementary_symmetric_functions 4 4

The output is:

k=1 :
\[ x_0 + x_1 + x_2 + x_3 \]

k=2 :
\[ x_0x_1 + x_0x_2 + x_0x_3 + x_1x_2 + x_1x_3 + x_2x_3 \]

k=3 :
\[ x_0x_1x_2 + x_0x_1x_3 + x_0x_2x_3 + x_1x_2x_3 \]

k=4 :
\[ x_0x_1x_2x_3 \]

Orbiter can compute domino portraits. To do so, we need an input file in r/g/b format of size \((D + 1)s \times Ds\), where \(D = 7\) for double-six dominos.

domino_portrait:

- $(\text{ORBITER})$ -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

```plaintext
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 TEST_SYSTEM for the coefficient matrix and TEST_RHS for the right hand side.

\[
\text{TEST\_SYSTEM}="\begin{array}{cccccccc}
0,1,0,1,0,0, & \backslash \\
0,0,1,0,1,0, & \backslash \\
1,0,1,0,0,0, & \backslash \\
0,1,0,1,0,1, & \backslash \\
1,0,0,0,0,1, & \backslash \\
1,0,1,0,0,0, & \backslash \\
0,1,0,0,1,1 & \end{array}"
\]

\[
\text{TEST\_RHS}="1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1"
\]

\[
\text{solve\_test\_system:}
\]

\[
\begin{array}{l}
\text{\( \text{(ORBITER) -v 4} \) } \\
\text{\( \backslash \backslash \text{define A} \text{-vector -format 7 -dense $(\text{TEST\_SYSTEM})$ -end} \) } \\
\text{\( \backslash \\backslash \text{define D} \text{-diophant} \) } \\
\text{\( \backslash \\\backslash \text{-label test\_system} \) } \\
\text{\( \backslash \\\\backslash \text{-coefficient\_matrix A} \) } \\
\text{\( \backslash \\\\\backslash \text{-RHS $(\text{TEST\_RHS})} \) } \\
\text{\( \backslash \\\\\\backslash \text{-x\_min\_global 0 -x\_max\_global 1} \) } \\
\text{\( \backslash \\\\\\\backslash \text{-end} \) }
\end{array}
\]
<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 \n  -label test_system \n  -coefficient_matrix A \n  -RHS $(TEST_RHS) \n  -x_min_global 0 -x_max_global 1 \n  -end \n  -with D -do \n  -diophant_activity -solve_mckay \n  -end
```

The solutions are written to the file DLX_test.sol. And now the dancing links solver:

DLX_test:
```
$ (ORBITER) -v 4 \
  -define A -vector -format 7 -dense $(TEST_SYSTEM) -end \
  -define D \n  -diophant -label test_system \n  -coefficient_matrix A \n  -RHS $(TEST_RHS) \n  -x_min_global 0 -x_max_global 1 \n  -end \n  -with D -do \n  -diophant_activity -solve_DLX \n  -end
```
11.3 Combinatorial Linear Spaces

A linear space is a pair \((S, \mathcal{L})\) where \(S\) is a set and \(\mathcal{L}\) is a set of subsets of \(S\) such that each set \(L \in \mathcal{L}\) satisfies \(|L| \geq 2\) and moreover for any two \(a, b \in S\) there is exactly one element \(L \in \mathcal{L}\) such that both \(a\) and \(b\) belong to \(L\). The usual notions of isomorphism and automorphism apply. For finite linear spaces, a first combinatorial property is the number \(a_i\) which counts the number of sets \(L \in \mathcal{L}\) of size \(i\). The vector \((a_2, \ldots, a_n)\) is the line type of \((S, \mathcal{L})\). The equation

\[
\binom{n}{2} = \sum_{j=2}^{n} a_j \binom{j}{2}
\]

(11.1)

is satisfied. The equation can be used to generate all possible line types of a putative linear space. Here is an example. For \(|S| = 6\), (11.1) becomes

\[
x_0 \binom{6}{2} + x_1 \binom{5}{2} + x_2 \binom{4}{2} + x_3 \binom{3}{2} + x_4 \binom{2}{2} = \binom{6}{2}.
\]

Here, \((x_0, x_1, \ldots, x_4)\) is the line type of a putative linear space on 6 points. That is, \(x_i = a_{6-i}\) is the number of lines of size \(6-i\). The extended coefficient matrix of the system is

\[
\begin{bmatrix}
15 & 10 & 6 & 3 & 1 \\
15 & 1 & 1
\end{bmatrix}.
\]

The Orbiter command

linsp6:

\[
\begin{align*}
\text{linsp6:} \\
& \texttt{\$ (ORBITER) -v 4 \ } \\
& \texttt{\textasciitilde -define A -vector -format 1 -dense "15,10,6,3,1" -end } \\
& \texttt{\textasciitilde -define D -diophant -label linsp6 } \\
& \texttt{\textasciitilde -coefficient matrix A } \\
& \texttt{\textasciitilde -RHS "15,15,1" } \\
& \texttt{\textasciitilde -x_min_global 0 } \\
& \texttt{\textasciitilde -x_max_global 15 } \\
& \texttt{\textasciitilde -end } \\
& \texttt{\textasciitilde -with D -do } \\
& \texttt{\textasciitilde \textasciitilde -diophant_activity -solve_mckay } \\
& \texttt{\textasciitilde -end } \\
\end{align*}
\]

# 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 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

\[
p_j = \#j\text{-lines though } P.
\]

342
We are trying to precompute the matrix of point types

\[(p_{ij})\]

where \( j = 7,5,4 \) and \( i \) belongs to an index set of all possible point types. Fixing a point \( P \), counting points \( Q \neq P \) collinear with \( P \) yields

\[6p_7 + 4p_5 + 3p_4 = 29, \quad p_7 \leq 1, \quad p_5 \leq 27, \quad p_4 \leq 24.\]

Using the Orbiter commands

\[
\text{linsp30\_pt\_types:} \\
\text{  \$\text{(ORBITER)} -v 4} \\
\text{  \text{\texttt{\$}} -define A -vector -format 1 -dense "6,4,3" -end \} } \\
\text{  \text{\texttt{\$}} -define D -diophant \} \\
\text{  \text{\texttt{\$}} -label linsp30\_pt\_types \} \\
\text{  \text{\texttt{\$}} -coefficient\_matrix A \} \\
\text{  \text{\texttt{\$}} -RHS "29,29,1" -x\_bounds "0,1,0,27,0,24" \} \\
\text{  \text{\texttt{\$}} -end \} \\
\text{  \text{\texttt{\$}} -with D -do \} \\
\text{  \text{\texttt{\$}} -diophant\_activity -solve\_mckay \} \\
\text{  \text{\texttt{\$}} -end
\]

we determine the possibilities

\[
(p_7,p_5,p_4) = \begin{cases} 
1 & 5 & 1 \\
1 & 2 & 5 \\
0 & 5 & 3 \\
0 & 2 & 7 
\end{cases}
\]

The rows in this matrix are called the point types \((i = 0,1,2,3)\). Let \( b_i \) be the number of points of type \( i \). By counting points, incident (point,line) pairs by \( j \)-lines and pairs of intersecting \( j \)-lines, we arrive at the following system:

\[
\begin{align*}
b_0 + b_1 + b_2 + b_3 &= 30 \\
b_0 + b_1 &= 7 \\
5b_0 + 2b_1 + 5b_2 + 2b_3 &= 135 = 27 \cdot 5 \\
b_0 + 5b_1 + 3b_2 + 7b_3 &= 96 = 24 \cdot 4 \\
10b_0 + b_1 + 10b_2 + b_3 &\leq 351 = \binom{27}{2} \\
10b_1 + 3b_2 + 21b_3 &\leq 276 = \binom{24}{2}
\end{align*}
\]

Using the Orbiter commands
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 $L = (V, B)$, where $B$ is a set of distinct subsets of $V$, called lines. The members of $V \cup B$ are called elements. For two elements $x, y$, we say that $x$ is incident with $y$, written $xIy$, 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 $V \cup B$ such that each $C_i$ either is a subset of $V$ or a subset of $B$. A decomposition is called tactical if for all $i$, the incidence number

$$\iota(C_i, C_j) = \# \{y \in C_j, xIy\}$$

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 $\text{Aut}(L)$ be the automorphism group of the linear space, which is the subgroup of $\text{Sym}(V)$ which preserves incidence. For $\alpha \in \text{Aut}(L)$ we say that the decomposition $\Pi$ preserves $\alpha$ if $\alpha$ fixes every class of $\Pi$. For $A \leq \text{Aut}(L)$, we say that $\Pi$ preserves $A$ is $\Pi$ preserves every element $\alpha \in A$. Mostly, we are interested in those decompositions $\Pi$ which preserve $\text{Aut}(L)$. In light of this, the discrete decomposition is not that interesting.

Any linear space has a coarsest tactical decomposition that preserves its automorphism group: The orbit partition of the automorphism group acting on $V \cup B$ will do. Up to ordering of the classes, the coarsest tactical refinement is unique. Computing the orbit decomposition is challenging as it involves computing the automorphism group. Computationally, there are easier ways to get to admissible decompositions. One is by means of successive refinements. If a class $C_i$ does not have the property that $\iota(C_i, C_j)$ is well-defined for all $x \in C_i$, then a refinement of $C_i$ will do. The coarsest refinement of $C_i$ has the property that if $C_i$ preserves some group $A$ then the refinement will do, too. This shows that there is an algorithm to compute a tactical decomposition of any given linear space $P$. Simply start with the decomposition of two classes, one the set of points and one the set of blocks, and refine. The output may or may not be equal to the decomposition arising from the orbit partition of $\text{Aut}(L)$.

Let us consider the opposite question. Given a tactical decomposition, does there exist a linear space whose coarsest tactical decomposition is the given one? If so, how many nonisomorphic partial linear spaces are there for a given tactical decomposition? in other words, we would like to classify the linear spaces which admit a given tactical decomposition. The -geometry_builder option can answer these kinds of questions. Table 11.5 shows the options for the geometry builder.

The command

```
geo_10_3:
▷ $(ORBITER) -v 2 \
▷ ▷ -define Test_lines -set -loop 4 11 1 -end \
▷ ▷ -define Geo -geometry_builder \
▷ ▷ ▷ -V 10 -B 10 -TDO 3 -fuse 1 \
▷ ▷ ▷ -fname_GEO 10_3 \
```

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 $\text{Test lines}$. The set of test lines is defined using a loop command. The command shows that there are exactly 10 configurations of this kind. One of them is the Desargues configuration. 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 $b \cdot i + j$, where $b$ is the number of blocks in the geometry. In this case, with $b = 10$, zero represents the incidence between point 0 and block zero. The number 99 represents the incidence between point 9 and block 9. Here is the file $10_3$.inc:

```
10 10 30
0  1  2 10  13 14  20  25  26  31  33  35  41  44  47  52  53  58  62  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  53  58  62  66  69  73  78  79  85  87  89  96  97  98
0  1  2 10  13 14  20  25  26  31  33  35  41  44  47  52  54  58  62  66  69  73  78  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  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  56  58  62  67  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  48  52  54  57  62  66  68  73  77  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  97  99
0  1  2 10  13 14  20  25  26  31  33  35  41  47  48  52  54  57  62  66  68  73  77  79  84  86  89  95  97  99
0  1  2 10  13 14  20  25  26  31  33  35  41  47  48  52  54  57  62  66  68  73  77  79  84  86  89  95  96  97  99
0  1  2 10  13 14  20  25  26  31  33  37  41  45  48  52  54  57  62  66  68  73  77  79  84  88  89  96  97  99
-1  10 120, 24, 12, 10, 6, 4^2, 3^2, 2
```

2. The option -output_to_sage_file writes $10_3$.sage. This file is meant to be read by Sage [64].

3. The option -output_to_blocks_file writes $10_3$.blocks. Here is the content of the file $10_3$.blocks:

```
10 10 3
0 15 26 44 51 68 81 109 114 116
0 15 26 46 49 68 81 109 114 116
0 15 26 46 49 68 83 106 115 116
0 15 26 46 52 69 77 106 114 116
0 15 26 46 56 69 80 101 106 119
0 15 26 46 56 69 80 103 104 119
```

347
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 \n  \> -define Test_lines -set -loop 0 11 1 -end \n  \> -define GEO -geometry_builder \n  \> -V 10 -B 10 -TDO 3 -fuse 1 \n  \> -fname_GEO 10_3 \n  \> -search_tree \n  \> -test Test_lines \n  \> -end
  $(ORBITER) -v 20 \n  \> -draw_options -embedded -radius 40 \n  \> -paperheight 220 \n  \> -paperwidth 330 \n  \> -xin 10000 -yin 10000 \n  \> -xout 1000000 -yout 500000 \n  \> -scale 2 -line_width 0.3 \n  \> -nodes_empty \n  \> -end
  -tree_draw 

348
Figure 11.4: Search tree for the classification of 10₃ configurations

```
▷▷▷ -file 10_3_tree.txt \
▷▷▷ -end
▷ pdflatex 10_3_tree_draw.tex
▷ open 10_3_tree_draw.pdf
```

The size of the tree can be determined by counting the lines of the file `10_3_tree.txt` and subtracting one:

```
wcat 10_3_tree.txt
```

The word count command yields the following output:

```
1471 13543 37633 10_3_tree.txt
```

This means that there are 1471 lines in the file. Hence the search tree has 1470 nodes. The resulting tree is shown in Figure 11.4.

Any incidence structure defines a graph on its underlying set of points. The vertices are the points of the incidence structure. Two vertices are adjacent if and only if the incidence structure contains a block which contains the associated points. In a geometric context, the graph is known as the collinearity graph of the geometry. The distance between two points is the distance of the associated vertices in the collinearity graph. The girth is the length of the shortest cycle. It is often desired to classify incidence structures with a given girth. This means that we are given an integer $g$ (the girth), and that we are looking for incidence structures whose collinearity graph has no cycles of length less than $g$. For instance, the following example classifies all cubic graphs on 10 vertices with girth at least 5:
geo_petersen:
  ▶ $(ORBITER) -v 8 \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

There is a unique graph with these properties. It is the Petersen graph. Its automorphism group is Sym(5) of order 120.

We can classify configurations with a given girth. For instance, while there are 245342 isomorphism classes of configurations 15_3, only one of them has girth 4. This is the Cremona Richmond configuration. It is associated to a cubic surface. The following command classifies all configurations 15_3:

15_3.inc:
  ▶ $(ORBITER) -v 2 \n  ▶ ▶ -define Test_lines -set -loop 4 16 1 -end \n  ▶ ▶ -define Geo -geometry_builder \n  ▶ ▶ ▶ -V 15 -B 15 -TDO 3 \n  ▶ ▶ ▶ -girth 4 \n  ▶ ▶ ▶ -search_tree \n  ▶ ▶ ▶ -test Test_lines \n  ▶ ▶ -end

This command takes about 8 minutes of time to complete.

The next command classifies the configurations 15_3 with girth 4. Only one configuration arises, the Cremona Richmond, with automorphism group Sym(6) of order 720.

geo_15_3_g4:
  ▶ $(ORBITER) -v 2 \n  ▶ ▶ -define Test_lines -set -loop 4 16 1 -end \n  ▶ ▶ -define Geo -geometry_builder \n  ▶ ▶ ▶ -V 15 -B 15 -TDO 3 \n  ▶ ▶ ▶ -girth 4 \n  ▶ ▶ ▶ -search_tree \n  ▶ ▶ ▶ -test Test_lines \n  ▶ ▶ -end
  ▶ $(ORBITER) -v 2 \n  ▶ ▶ -draw_options -embedded -radius 50 \n  ▶ ▶ ▶ -nodes_empty \n
350
The next command classifies the configurations $40_4$ with girth 4. Exactly two configuration arise, both with a group of order $51840$. Note the extra option `-special_test_not_orderly` to speed up the search.

40_4.g4.inc:

```
$\texttt{(ORBITER)} -v 5 \\n-define \texttt{Test\_lines} -set -loop 0 41 1 -end \n-define \texttt{Geo -geometry\_builder} \n-V 40 -B 40 -TDO 4 \n-fuse 1 \n-fname_{GEO} 40_4.g4 \ngirth 4 \n-search_tree \nspecial_test_not_orderly \ntest \texttt{Test\_lines} \n-output_to_sage_file \n-output_to_inc_file \n-end \n$\texttt{(ORBITER)} -v 2 \n-draw_options -embedded -radius 50 \n-xin 10000 -yin 10000 \n-xout 1000000 -yout 1000000 \n-nodes_empty \n-scale 0.5 -line_width 0.3 -end \n-tree_draw -file 40_4.g4_tree.txt -end \npdflatex 40_4.g4_tree_draw.tex \open 40_4.g4_tree_draw.pdf
```

The search tree is shown in Figure 11.5.

The next command classifies the configurations $63_3$ with girth 6. Exactly two configuration arise, both with a group of order $12096$. Note the extra option `-special_test_orderly` to speed up the search.

geo_63_3.g6:

```
$\texttt{(ORBITER)} -v 2 \n-define \texttt{Test\_lines} -set -loop 4 64 1 -end \n-define Geo -geometry\_builder \n-V 63 -B 63 -TDO 3 \n```

351
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_3 -end \
▷ ▷ ▷ -with D -do \
▷ ▷ ▷ ▷ -design_activity \n▷ ▷ ▷ ▷ ▷ -export_inc \n▷ ▷ ▷ -end
```

creates the design \( \text{PG}(2,3) \).

We have created the following design:

\( \{19, 79, 126, 219, 256, 284, 371, 392, 465, 541, 619, 627, 653\} \)
The stabilizer is generated by:

Strong generators for a group of order 5616:

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

The blocks of the design are encoded in the lexicographic ordering of \(k\)-subsets (here \(k = 4\))

The program also displays the tactical decomposition schemes of the design, which are

\[
\begin{array}{c|c|c}
\rightarrow & 13_1 & \downarrow 13_1 \\
13_0 & 4 & 13_0 & 4
\end{array}
\]

In Section 15.4, we will show how to compute further properties of the design.

The command

\[
\texttt{wreath_product_designs_n4_k2_inc.txt:}
\]

\[
\texttt{$(ORBITER) -v 8 \ \backslash}
\]

\[
\texttt{\ v -define D -design -wreath_product_designs 4 2 -end \ \\
\ v -with D -do \ \\
\ v -design_activity \ \\
\ v -export_inc \ \\
\ v -end}
\]

creates a design on 8 points invariant under the wreath product Sym(4) \(\rtimes\) Sym(2). The design has 12 blocks of size 4. The command

\[
\texttt{wreath_product_designs_n8_k6_inc.txt:}
\]

\[
\texttt{$(ORBITER) -v 8 \ \backslash}
\]

355
creates a design on 16 points invariant under the wreath product \( \text{Sym}(8) \wr \text{Sym}(2) \). The design has 3920 blocks of size 6. We will compute the automorphism groups of these two designs in Section 15.3.

One way to construct designs is by assuming a suitable group of symmetries. Let us consider an example. It is possible to construct \( t-(v,k,\lambda) \) designs invariant under a permutation group \( G \) acting on a set \( V \) with \( |V| = v \) as follows: Classify the orbits of \( G \) on subsets of size \( k \) and less. Construct a matrix which describes the relationship between the orbits on \( t \)-sets and the orbits on \( k \)-sets. This matrix is often referred to as the Kramer-Mesner matrix (cf. [40]). For each pair of \( t \)-orbit and \( k \)-orbit, for instance with representatives \( T \) and \( K \), say, we count the number of elements in the orbit of \( K \) which contain \( T \). The rows of the matrix are in correspondence to the \( t \)-orbits, while the columns are in correspondence to the \( k \)-orbits. The matrix entry \( a_{ij} \) is the number just defined where \( T \) is the representative of the \( i \)-th orbit on \( t \)-sets, and where \( K \) is the representative of the \( j \)-th orbit on \( k \)-sets. Let \( M_{t,k}(G) \) be the Kramer-Mesner matrix for the group \( G \leq \text{Sym}(V) \) defined in this way. The \( t-(v,k,\lambda) \) designs invariant under \( G \) are in one-to-one correspondence to the solutions of

\[
M_{t,k}(G) \cdot x = \lambda 1,
\]

where \( x \) is a column vector of zeros and ones and \( 1 \) is the column vector of all ones. The length of \( x \) is the number of \( k \)-orbits of \( G \) on \( V \), while the length of \( 1 \) is the number of \( t \)-orbits of \( G \) on \( V \). Any vector \( x \) satisfying the matrix equation corresponds to a design invariant under \( G \). Simply take the blocks of the design to be the union of those orbits of \( G \) on \( k \)-subsets whose associated entry in \( x \) is one. We assume the group \( \text{PGL}(2,32) \) in the action on points of the projective line \( \text{PG}(1,32) \) over the field \( \mathbb{F}_{32} \). 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 $M_{7,8}(\text{PGL}(2,32))$

```
KM_PGGL_2_32:
  ▷ $(ORBITER) -v 3 \n  ▷ ▷ -define Control -poset_classification_control \n  ▷ ▷ ▷ -problem_label KM_PGGL_2_32 -W -depth 8 \n  ▷ ▷ ▷ -Kramer_Mesner_matrix 7 8 \n  ▷ ▷ ▷ -draw_poset \n  ▷ ▷ ▷ -draw_options -embedded -sideways -radius 50 \n  ▷ ▷ ▷ ▷ -scale 0.5 -line_width 0.3 -end \n  ▷ ▷ ▷ -end \n  ▷ ▷ -define G -linear_group -PGGL 2 32 -end \n  ▷ ▷ -define Orb -orbits -group G \n  ▷ ▷ ▷ -on_subsets 8 Control \n  ▷ ▷ ▷ -end
  ▷ $(ORBITER) -v 2 -draw_matrix \n  ▷ ▷ -input_csv_file KM_PGGL_2_32_KM_7_8.csv \n  ▷ ▷ -box_width 20 -bit_depth 24 \n  ▷ ▷ -partition 3 32 97 -end
  ▷ pdflatex KM_PGGL_2_32_poset_lvl_8.tex
  ▷ open KM_PGGL_2_32_poset_lvl_8.pdf
  ▷ open KM_PGGL_2_32_KM_7_8_draw.bmp
  ▷ $(ORBITER) -v 4 \n  ▷ ▷ -define A -vector -file KM_PGGL_2_32_KM_7_8.csv -end \n  ▷ ▷ -define D -diophant \n  ▷ ▷ ▷ -label "KM_PGGL_2_32_KM_7_8_system" \n  ▷ ▷ ▷ -coefficient_matrix A \n  ▷ ▷ ▷ -RHS_constant "10,10,1" \n  ▷ ▷ ▷ -x_min_global 0 -x_max_global 1 \n  ▷ ▷ ▷ -end \n  ▷ ▷ ▷ -with D -do \n  ▷ ▷ ▷ ▷ -diophant_activity -solve_mckay \n```

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 \ 
  ▶ ▶ -define Geo -geometry_builder \ 
  ▶ ▶ ▶ -V 9 -B 12 \ 
  ▶ ▶ ▶ -TDO 4 -fuse 1 \ 
  ▶ ▶ ▶ -fname_GEO AG_2_3 \ 
  ▶ ▶ ▶ -test 3,4,5,6,7,8,9 \ 
  ▶ ▶ -end
```

The command produces the file `AG_2_3.inc`, which contains the following lines:

```
9 12 36
0 1 2 3 12 16 17 18 24 31 32 33 37 43 46 49 53 56 59 62 64 69 74 78 80 82 87 89 93 94 99 102 103 107
-1 1
432
```

This shows that the design is unique, and has an automorphism group of order 432. For the following commands, we will treat blocks of the design as sets of ranks of \( k \)-subsets. We can now create a table of all designs \( AG(2, 3) \), as orbit under the group \( Sym(9) \). The following command does that:

```
AG_2_3_BLOCKS="0,13,22,27,35,41,47,53,55,59,71,76"
```

```
LS_AG_2_3_design_table_create:
  ▶ $(ORBITER) -v 5 \ 
  ▶ ▶ -define B -vector -dense $(AG_2_3_BLOCKS) -end \ 
  ▶ ▶ -define D -design -list_of_blocks 9 3 B -end \ 
  ▶ ▶ -define Sym9 -permutation_group -symmetric_group 9 -end \ 
  ▶ ▶ -define T -design_table D "AG_2_3" Sym9 -end
```

The number of designs is \( |Sym(9)|/432 = 362880/432 = 840 \). 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:

```
```

359
LS_AG_2_3_disjoint_sets_graph_and_cliques:
  \[
  $(ORBITER) \ -v \ 2 \ \backslash
  \]\[\text{define Gamma -graph } \backslash\]
  \[
  \text{define AG}_2_3\text{design_table.csv } \backslash
  \]\[\text{-end } \backslash\]
  \[
  \text{-with Gamma -do } \backslash
  \]\[\text{-graph_theoretic_activity } \backslash\]
  \[
  \text{-save } \backslash
  \]\[\text{-end } \backslash\]
  \[
  \text{-find_cliques -target_size 7 -end } \backslash
  \]\[\text{-print_symbols } \backslash\]

The files \text{AG}_2_3\text{design_table_disjoint_sets_sol.txt} and \text{AG}_2_3\text{design_table_disjoint_sets_sol.csv} are created, each containing the cliques of size 7. There are exactly 15360 cliques of size 7. It remains to classify the resulting 15360 large sets up to isomorphism. To do that, we first need to create the actual large sets from the cliques. The following command does that:

LS_AG_2_3_export_solutions:
  \[
  $(ORBITER) \ -v \ 20 \ \backslash
  \]\[\text{-define B -vector -dense $(AG}_2_3\text{BLOCKS) -end } \backslash
  \]\[\text{-define D -design -list_of_blocks 9 3 B -end } \backslash
  \]\[\text{-define Sym9 -permutation_group -symmetric_group 9 -end } \backslash
  \]\[\text{-define T -design_table D "AG}_2_3" Sym9 -end } \backslash
  \]\[\text{-with D -do } \backslash
  \]\[\text{-design_activity } \backslash
  \]\[\text{-extract_solutions_by_index "AG}_2_3" Sym9 } \backslash
  \]\[\text{AG}_2_3\text{design_table_disjoint_sets_sol.csv } \backslash
  \]\[\text{solutions.csv } \backslash
  \]\["" } \backslash
  \]\[\text{-end } \backslash

The final step to classify the large sets up to isomorphism will be discussed in Section 15.4.
11.7 Design Theory – Delandtsheer-Doyen

Delandtsheer and Doyen in [23] study line-transitive and point-imprimitive designs and show that they are rare in a certain sense. Orbiter can be used to construct such designs assuming that there is a grid structure on the set of points and assuming that the design is invariant under a chosen group $G$. The group $G$ is assumed to be a subgroup of the group $\text{AGL}(d_1, q_1) \times \text{AGL}(d_2, q_2)$ acting on a grid of size $q_1^{d_1} \times a_2^{d_2}$ 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

$$(\tau_1, \tau_2), \mathbb{Z}_3 \times \mathbb{Z}_7 \rightarrow \mathbb{Z}_3 \times \mathbb{Z}_7,$$

where

$$\tau_1 : \mathbb{Z}_3 \rightarrow \mathbb{Z}_3, x \mapsto x + 1 \mod 3, \quad \tau_2 : \mathbb{Z}_7 \rightarrow \mathbb{Z}_7, y \mapsto y + 1 \mod 7.$$

With blocks of size 5, we cover 10 pairs each. The group of order 21 allows to cover each of the $210 = \binom{21}{2}$ pairs exactly once using a single orbit of a block. The question remains to construct all blocks and to classify the resulting designs. The Desarguesian plane $\text{PG}(2, 4)$ provides a solution. The question is to decide whether there are any other, nonisomorphic designs. The following Orbiter commands can be used:

```
PP4=-d1 1 -q1 3 -d2 1 -q2 7 -K 5 -search_control -W -end -problem_label PP4

PP4_GROUP1=-subgroup "1,1,1,1, " "21" -group_label "cyclic21"

PP4_MASK1=\n  ▷ -nb_orbits_on_blocks 1 \n  ▷ -depth 5 \n  ▷ -mask_label "no_mask"
```

The command `DD_PP4` sets up the orbits of the group on pairs and writes the file `PP4_pair_covering.csv`.

```
DD_PP4:
  ▷ $(ORBITER) -v 6 \n  ▷ ▷ -Delandtsheer_Doyen $(PP4) $(PP4_GROUP1) $(PP4_MASK1) \n  ▷ ▷ ▷ -end \n```

The command `DD_PP4_system` creates a diophantine system of Steiner type and solves it.
DD_PP4_system:
▷ $(ORBITER) -v 4 \n▷ ▷ -define D -diophant -label PP4 \n▷ ▷ -problem_of_Steiner_type_10 PP4.pair_covering.csv \n▷ ▷ -has_sum 1 \n▷ ▷ -end \n▷ ▷ -with D -do \n▷ ▷ ▷ -diophant_activity -solve_mckay \n▷ ▷ ▷ -end

It finds exactly one solution. This must be the PG(2, 4) design. Since there are no more designs, isomorphism testing is not needed.
11.8 Tactical Decompositions

Table 11.6 lists the Orbiter commands for decomposition refinement.

Suppose we want to study projective planes of order 16. It is a linear space with $16^2 + 16 + 1 = 273$ points and equally many lines. Each point lies on 17 lines and each line contains 17 points. Any two points lie on exactly one line and any two lines intersect in exactly one point.

We decide to study maximal arcs of degree 4 in this plane (the degree has to divide the order of the plane). A maximal arc of degree $d$ is a set of points so that each line intersects in either $d$ or zero points. A line which intersects in $d$ points is called a secant. A line which intersects in no point is called an external line. The command

```
max_arc_16_4_start:
> $(ORBITER) -v 4 -maximal_arc_parameters 16 4
```

creates a decomposition stack for the parameters of the arc and writes the file `max_arc_q16_r4.stack`

```<HTDO type=pt ptanz=2 btanz=2 fuse=simple>
   221 52
   52 17 0
   221 13 4
   1 1
</HTDO>
```

This is a point-tactical decomposition with 2 point-classes and 2 block-classes. The point classes are associated with the rows. The block-classes are associated with the columns. The first row and column indicates the size of the classes. The entries $a_{ij}$ count the number of blocks in the column class $j$ that are incident with a given point in the $i$th row class. The fuse information at the bottom (1 1) is a partition of the row classes which indicates the ancestor decomposition which was column tactical. The next step is to convert the stack file to a tdo file. The command

```
max_arc_16_4_convert_stack.tdo:
> $(ORBITER) -v 4 -convert_stack_to_tdo max_arc_q16_r4.stack
```

does that. It creates the file `max_arc_q16_r4.tdo`. It also prints the decomposition stack:

```
lambda_scheme at level 2 :
is 1 x 1
   | 273_{ 1}
   ===============
273_{ 0} |
```

363
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-lambda3</td>
<td>λᵢ s</td>
<td>Refine as 3-design with λ₃ and with block-size s</td>
</tr>
<tr>
<td>-solution</td>
<td>i fname</td>
<td>Use solutions to system i from file name.</td>
</tr>
<tr>
<td>-range</td>
<td>f l</td>
<td>Refine cases i with $f \leq i &lt; f + l$ only.</td>
</tr>
<tr>
<td>-select</td>
<td>label</td>
<td>Select the case for refinement by label.</td>
</tr>
<tr>
<td>-o1</td>
<td>s</td>
<td>Omit s variables from the first refinement system.</td>
</tr>
<tr>
<td>-o2</td>
<td>s</td>
<td>Omit s variables from the second refinement system.</td>
</tr>
<tr>
<td>-D1_upper_bound_x0</td>
<td>b</td>
<td>Add the bound $x₀ \leq b$ 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

```bash
max_arc_16_4_refine:
```

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:

```bash
max_arc_16_4r_print:
```

This produces the following output:

```plaintext
decomposition 0.1:
lambda_scheme at level 2 :
is 1 x 1
       | 273_{ 1}
-----------------------
273_{ 0} | 17 17
```
```
| 52_{ 0} | 17 0 |
| 221_{ 3} | 13 4 |

**col_scheme at level 4:**
is 2 x 2
```

```
| 221_{ 1} 52_{ 2} |
```

```
| 52_{ 0} | 4 0 |
| 221_{ 3} | 13 17 |

**extra_col_scheme at level 3:**
is 1 x 2
```

```
| 221_{ 1} 52_{ 2} |
```

```
| 273_{ 0} | 17 17 |
```
Chapter 12

Finite Geometry

12.1 Spreads

A $t$-spread of $\text{PG}(n,q)$ is a set of disjoint $\text{PG}(t,q)$ that cover all of $\text{PG}(n,q)$ pointwise. $t$-spreads in $\text{PG}(n,q)$ exist if and only if $t+1$ divides $n+1$. In order to create a spread, Orbiter offers several commands, as summarized in Table 12.1. The following two commands create the two spreads of order 9, relying on the Orbiter knowledge base.

create\_spread\_9a:
  $\text{create}\_\text{spread}\_\text{9a}$:
  $\text{define}\ F\ -\text{finite}\_\text{field}\ -q\ 3\ -\text{end}\$
  $\text{define}\ G\ -\text{linear}\_\text{group}\ -\text{PGL}\ 4\ F\ -\text{end}\$
  $\text{define}\ S\ -\text{spread}\ -\text{kernel}\_\text{field}\ F\$
  $\text{group}\ G\ -k\ 2\ -\text{catalogue}\ 0\$
  $\text{end}$

create\_spread\_9b:
  $\text{create}\_\text{spread}\_\text{9b}$:
  $\text{define}\ F\ -\text{finite}\_\text{field}\ -q\ 3\ -\text{end}\$
  $\text{define}\ G\ -\text{linear}\_\text{group}\ -\text{PGL}\ 4\ F\ -\text{end}\$
  $\text{define}\ S\ -\text{spread}\ -\text{kernel}\_\text{field}\ F\$
  $\text{group}\ G\ -k\ 2\ -\text{catalogue}\ 1\$
  $\text{end}$

The first spread is the Desarguesian spread, with automorphism group of order 5760. The second spread is the Hall spread with automorphism group of order 1920.

Spreads can be defined using spread sets. A spread set is a set of $q^k$ matrices of size $k \times k$ over $\mathbb{F}_q$ such that $A_i - A_j$ is nonsingular for all $i \neq j$. Let us look at an example. The spread due to Rao and Rao [54] can be defined using the following makefile variable.
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-kernel_field</td>
<td>$F$</td>
<td>Define the kernel of the spread. $F$ must be an object of type finite field.</td>
</tr>
<tr>
<td>-group</td>
<td>$G$</td>
<td>Define the group acting on the spread. Should be $\text{PGL}(2k; F)$.</td>
</tr>
<tr>
<td>-k</td>
<td>$k$</td>
<td>Set the dimension of the spread.</td>
</tr>
<tr>
<td>-catalogue</td>
<td>$i$</td>
<td>Pull spread number $i$ from the catalogue of spreads associated with the given field and the given dimension.</td>
</tr>
<tr>
<td>-family</td>
<td>$L$</td>
<td>Define a spread from a named family $L$. So far, no family has been provided.</td>
</tr>
<tr>
<td>-spread_set</td>
<td>$S$</td>
<td>Define a spread from the named spread-set $S$. The spreadset $S$ must be a vector object. It must contain $q^3k^2$ entries over $F$.</td>
</tr>
</tbody>
</table>

Table 12.1: Orbiter commands to define a spread

```
SPREAD_SET_27_RAO_RAO="\n 0,0,0,0,0,0,0,0,0,0, \n 1,1,0,2,1,1,0,0,2, \n 1,0,1,1,2,2,0,1,0, \n 1,2,2,1,2,0,2,2,2, \n 0,0,2,2,0,1,2,0, \n 1,1,2,0,2,1,2,1,0, \n 0,1,0,1,0,1,0,2,1, \n 2,0,2,0,0,2,1,1,0, \n 2,2,2,0,1,1,0,1,2, \n 2,0,0,1,0,2,1,2,1, \n 0,2,2,2,2,2,0,2, \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 0,1,1,1,0,1,2,2,2, \n 2,2,0,2,0,0,0,2,2, \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
```
1,2,1,1,0,0,1,1,1, \
0,2,1,1,1,1,2,2,0, \
1,1,1,0,0,1,1,0,2, \
1,0,2,2,1,2,2,1,1 \
"

Each line represents one matrix of the spread set, with matrix entries being listed consecutively. The following command can be used to define the spread:

```plaintext
create_spread_Rao_Rao_27:
▷ $(ORBITER) -v 3 \n▷ ▷ -define F -finite_field -q 3 -end \n▷ ▷ -define SS -vector -dense $(SPREAD_SET_27_RAO_RAO) -end \n▷ ▷ -define G -linear_group -PGL 6 F -end \n▷ ▷ -define S -spread -kernel_field F \n▷ ▷ ▷ -group G -k 3 -spread_set SS \n▷ ▷ ▷ -end
```

The following command creates the Desarguesian line-spread in PG(3,2):

```plaintext
desarguesian_spread_in_PG_3_2:
▷ $(ORBITER) -v 3 \n▷ ▷ -define FQ -finite_field -q 4 -end \n▷ ▷ -define Fq -finite_field -q 2 -end \n▷ ▷ -with FQ -and Fq -do -finite_field_activity \n▷ ▷ ▷ -cheat_sheet_desarguesian_spread 2 -end
▷ pdflatex Desarguesian_Spread_3_2.tex
▷ open Desarguesian_Spread_3_2.pdf
```

The cheat sheet contains the following spread:

- **Spread element 0 is (1,0)**
  
  $\begin{bmatrix}
    1 & 0 & 0 & 0 \\
    0 & 1 & 0 & 0 \\
    0 & 0 & 1 & 0 \\
    1 & 0 & 1 & 0
  \end{bmatrix}$

- **Spread element 1 is (0,1)**
  
  $\begin{bmatrix}
    0 & 0 & 1 & 0 \\
    0 & 0 & 0 & 1 \\
    1 & 0 & 1 & 0 \\
    0 & 1 & 0 & 1
  \end{bmatrix}$

- **Spread element 2 is (1,1)**
  
  $\begin{bmatrix}
    1 & 1 & 1 & 0 \\
    1 & 0 & 0 & 1 \\
    1 & 0 & 1 & 0 \\
    0 & 1 & 0 & 1
  \end{bmatrix}$

- **Spread element 3 is (2,1)**
  
  $\begin{bmatrix}
    1 & 0 & 0 & 1 \\
    1 & 0 & 0 & 1 \\
    1 & 0 & 0 & 1 \\
    1 & 0 & 0 & 1
  \end{bmatrix}$

369
Spread element 4 is (3, 1) = \[
\begin{bmatrix}
0 & 1 & 1 & 0 \\
1 & 1 & 0 & 1 \\
\end{bmatrix}
\]

Spread elements by rank: (0, 34, 9, 17, 22).

The following command creates the Desarguesian plane-spread in \(\text{PG}(5, 2)\):

```
\texttt{desarguesian\_spread\_in\_PG\_5\_2:}
\texttt{\$\langle ORBITER \rangle -v 3 \}
\texttt{\$ \$ -define FQ -finite\_field -q 8 \end \}
\texttt{\$ \$ -define Fq -finite\_field -q 2 \end \}
\texttt{\$ \$ -with FQ -and Fq -do -finite\_field\_activity \}
\texttt{\$ \$ -cheat\_sheet\_desarguesian\_spread 2 \end}
\texttt{\$ \$ pdflatex Desarguesian\_Spread\_5\_2.tex}
\texttt{\$ \$ open Desarguesian\_Spread\_5\_2.pdf}
```

Spread element 0 is (1, 0) = \[
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 & 0 & 0 \\
\end{bmatrix}
\]

Spread element 1 is (0, 1) = \[
\begin{bmatrix}
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 \\
1 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 \\
\end{bmatrix}
\]

Spread element 2 is (1, 1) = \[
\begin{bmatrix}
1 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 1 \\
1 & 1 & 0 & 1 & 0 & 0 \\
\end{bmatrix}
\]

Spread element 3 is (2, 1) = \[
\begin{bmatrix}
0 & 1 & 1 & 0 & 1 & 0 \\
1 & 1 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 1 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 \\
\end{bmatrix}
\]

Spread element 4 is (3, 1) = \[
\begin{bmatrix}
1 & 1 & 1 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 \\
1 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 \\
1 & 1 & 1 & 0 & 0 & 1 \\
1 & 1 & 1 & 0 & 0 & 1 \\
\end{bmatrix}
\]

Spread element 5 is (4, 1) = \[
\begin{bmatrix}
1 & 1 & 1 & 0 & 1 & 0 \\
1 & 1 & 1 & 0 & 0 & 1 \\
1 & 1 & 1 & 0 & 0 & 1 \\
1 & 1 & 1 & 0 & 0 & 1 \\
1 & 1 & 1 & 0 & 0 & 1 \\
1 & 1 & 1 & 0 & 0 & 1 \\
\end{bmatrix}
\]
Spread element 6 is $(5, 1) = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$

Spread element 7 is $(6, 1) = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$

Spread element 8 is $(7, 1) = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$

Spread elements by rank: $(0, 1394, 189, 671, 562, 1040, 792, 1161, 373)$

Two $t$-spreads are isomorphic if there is a collineation which maps one to the other. The classification problem for $t$-spreads is the problem of determining a complete set of pairwise non-isomorphic $t$-spreads. 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 \$
  ▶ ▶ -define F -finite_field -q 4 -end \$
  ▶ ▶ -define P -projective_space -n 3 -field F -v 0 -end \$
  ▶ ▶ -define C -spread_classifier \$
  ▶ ▶ ▶ -projective_space P \$
  ▶ ▶ ▶ -k 2 \$
  ▶ ▶ ▶ -starter_size 17 \$
  ▶ ▶ ▶ -poset_classification_control \$
  ▶ ▶ ▶ ▶ -draw_options \$
  ▶ ▶ ▶ ▶ ▶ -radius 20 \$
  ▶ ▶ ▶ ▶ ▶ -nodes_empty \$
  ▶ ▶ ▶ ▶ ▶ -line_width 0.2 \$
  ▶ ▶ ▶ ▶ ▶ -embedded \$
  ▶ ▶ ▶ ▶ ▶ -end \$
  ▶ ▶ ▶ ▶ -draw_poset \$
  ▶ ▶ ▶ ▶ ▶ -problem_label spreads_16_4 \$
```

371
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:

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

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

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

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

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}

374
Table 12.2: Spreads in PG(3, 4) in the Orbiter Catalogue

| OCN | |Aut| | Name |
|-----|-----|-----|-----|
| 0   | 1200|     | Hall spread |
| 1   | 81600| Desarguesian spread |
| 2   | 576 |     | Semifield spread |

Strong generators for a group of order 576:

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

1, 0, 0, 0, 2, 0, 0, 0, 0, 2, 0, 0, 0, 0, 3, 1,
1, 0, 0, 0, 1, 0, 0, 0, 3, 0, 3, 2, 1, 0, 0, 2, 0,
1, 0, 0, 3, 1, 0, 0, 3, 0, 2, 2, 1, 0, 1, 2, 0,
1, 1, 1, 2, 0, 2, 0, 2, 0, 2, 1, 0, 2, 2, 3, 0,
1, 0, 3, 1, 1, 3, 1, 0, 1, 0, 2, 2, 0, 0, 0, 1, 1,
0, 1, 1, 0, 0, 0, 1, 2, 0, 2, 1, 3, 2, 3, 2, 0,
There are 0 extensions
Number of generators 6

The three spreads in PG(3, 4) can be distinguished by their stabilizer orders. Table 12.2 lists the line spreads in PG(3, 4) according to their orbiter catalogue number (OCN).

Let us now look at a more difficult problem. We wish to classify the spreads in PG(3, 5). To this end, we will use the method of classification by substructure. We pick a size \( s \) of a partial spread, and classify all partial spreads of size \( s \). These are the substructures. Next, we perform the lifting, which means we construct all spreads of PG(3, 5) containing one of the orbit representatives of the substructures. In a final step, we perform an isomorph classification on the set of liftings. This will furnish the desired classification of spreads of PG(3, 5). From a computational point of view, the lifting process is the bottleneck in this procedure. Because of this, we use specialized algorithms from graph theory, which enhance the performance of the lifting. Specifically, we perform a search for rainbow cliques. We will go over some examples to illustrate the technique. To begin with, we choose the parameter \( s = 5 \).
The command

```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 \
  -with C -do -spread_classify_activity \
  -prepare_lifting_single_case 0 \
  -end
```

classifies the partial spreads of size $s = 5$ and prepares for the lifting of the first case only. In order to prepare for the lifting, a graph is constructed which describes the lines that can be added to the first partial spread. The vertices of the graph are the lines disjoint from the initial set of 5 lines in the partial spread. Two vertices are joined by an edge if the associated lines are disjoint. The vertices of the graph are colored according to the very first basis vector in the generator matrix of the subspace in reduced row echelon form. In order to find the rainbow clique in the graph, the command

```bash
spreads_25_starter_0_cliques:
  $(ORBITER) -v 2 \
```

376
can be used.

The command

```
classify_spreads_25_starter_lift_all_cases:
  $(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 
  -output_prefix "" 
  -end 
  -with C -do -spread_classify_activity 
  -compute_starter 
  -problem_label spreads_25 
  -end 
  -output_prefix "" 
  -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\textsubscript{25}_starter_cliques:
\begin{verbatim}
$\textit{ORBITER}$ -v 2 \
  -loop L 0 29 1 \
  -define G -graph -load spreads\textsubscript{25}\_graph\_L.bin -end \
  -with G -do \
  -graph\_theoretic\_activity \
  -find\_cliques -rainbow -target\_size 21 -end \
  -end \
  -end \end{verbatim}

The resulting cliques are again stored in files. The command

classify\textsubscript{spreads\textsubscript{25}\_isomorph}:
\begin{verbatim}
$\textit{ORBITER}$ -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\textsubscript{25} \
  -end \
  -output\_prefix "" \
  -end \
  -with C -do -spread\_classify\_activity \
  -compute\_starter \
  -problem\_label spreads\textsubscript{25} \
  -W -depth 5 \
  -report -end \
  -end \
  -end \
  -with C -do -spread\_classify\_activity \
  -isomorph \end{verbatim}

378
-prefix_iso "./spreads_25" \
-use_database_for_starter \
-build_db \
-solution_prefix "" \
-base_fname "" \
-end \
-with C -do -spread_classify_activity \
-isomorph \
-prefix_iso "./spreads_25" \
-use_database_for_starter \
-read_solutions \
-solution_prefix "" \
-base_fname "spreads_25_graph" \
-end \
-with C -do -spread_classify_activity \
-isomorph \
-prefix_iso "./spreads_25" \
-use_database_for_starter \
-compute_orbits \
-solution_prefix "" \
-base_fname "spreads_25_graph" \
-end \
-with C -do -spread_classify_activity \
-isomorph \
-prefix_iso "./spreads_25" \
-use_database_for_starter \
isomorph_testing \
-solution_prefix "" \
-base_fname "spreads_25_graph" \
-end \
-with C -do -spread_classify_activity \
-isomorph \
-prefix_iso "./spreads_25" \
-use_database_for_starter \
isomorph_report \
-solution_prefix "" \
-base_fname "spreads_25_graph" \
-end \
-end \
pdflatex spreads_25_isomorphism_types.tex \
open spreads_25_isomorphism_types.pdf
performs the final isomorph rejection on the spreads arising from the rainbow cliques in all cases. It results in a transversal of the isomorphism classes of spreads of $\text{PG}(3, 5)$. 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).
12.2 Translation Planes

Orbiter can create translation planes from spreads. The construction of translation planes from spreads is due to André and Bruck, Bose (cf. [3, 16]). In order to perform the construction, we need a field $F = \mathbb{F}_q$ which is the kernel of the plane, a spread of $k$-subspaces, and the groups $\text{PGL}(2k, F)$ and $\text{PGL}(2k + 1, F)$. For instance, the command

```
create_translation_plane_9b:
  ➤ $(ORBITER) -v 3 \\
  ➤  ➤ -define F -finite_field -q 3 -end \\
  ➤  ➤ -define G -linear_group -PGL 4 F -end \\
  ➤  ➤ -define G1 -linear_group -PGL 5 F -end \\
  ➤  ➤ -define S -spread -kernel_field F \\
  ➤  ➤  ➤ -group G -k 2 -catalogue 1 \\
  ➤  ➤  ➤ -end \\
  ➤  ➤ -define T -translation_plane S G G1 -end \\
  ➤  ➤  ➤ -with T -do -translation_plane_activity \\
  ➤  ➤  ➤ -export_incma \\
  ➤  ➤  ➤ -end \\
  ➤  ➤ -with T -do -translation_plane_activity \\
  ➤  ➤  ➤ -report \\
  ➤  ➤ -end \\
  ➤  ➤ -define A -linear_group -import_group_of_plane T -end \\
  ➤  ➤ -define Orb -orbits -group A \\
  ➤  ➤  ➤ -on_points \\
  ➤  ➤  ➤ -end \\
  ➤ $\linebreak (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 500 -spanning_tree -embedded \\
  ➤  ➤  ➤ -line_width 1.1 -x_stretch 1.4 -scale 0.25 \\
  ➤  ➤ -end
```

#open plane_catalogue_q3_k2_1_incma_draw.bmp
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 orbit_PGL_5_3_on_andre_3_draw.tex
#open orbit_PGL_5_3_on_andre_3_draw.pdf
pdflatex group_of_plane_plane_catalogue_q3_k2_1_stab_orb_3.report.tex
open group_of_plane_plane_catalogue_q3_k2_1_stab_orb_3_report.pdf
In the next example, we create a translation plane of order 16 with kernel of order 4:

```
create_translation_plane_16_4_0:
  $(ORBITER) -v 3 \
  ▶ ▶ -define F -finite_field -q 4 -end \
  ▶ ▶ -define G -linear_group -PGGL 4 F -end \
  ▶ ▶ -define G1 -linear_group -PGGL 5 F -end \n  ▶ ▶ -define S -spread -kernel_field F \
  ▶ ▶ ▶ -group G -k 2 -catalogue 0 \n  ▶ ▶ ▶ -end \
  ▶ ▶ -define T -translation_plane S G G1 -end
  $(ORBITER) -v 2 \
```

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:
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:

\begin{verbatim}
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
\end{verbatim}

The spread has a stabilizer of order 1008, which means that the associated translation plane has a stabilizer of order $1008 \cdot 256 = 258045$. According to [52], there are two planes whose associated spreads have this automorphism group order. They can be distinguished by the 2-rank of their incidence matrices. The Johnson-Walker plane has a 2-rank of 100. The Lorimer-Rahilly plane has a 2-rank of 106. Using Orbiter, we compute the 2-rank of the translation plane that we have created:

\begin{verbatim}
RREF_plane_16_2_0_rank_of_incma:
$(ORBITER) -v 2 
  > -define F -finite_field -q 2 -end 
  > -define v -vector -field F 
  >   > -file plane_catalogue_q2_k4_0_incma.csv 
  >   > -end 
  > -with F -do -finite_field_activity 
  > -RREF v -normalize_from_the_right 
  > -end
\end{verbatim}
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

```
create_translation_plane_27_Rao_Rao:
  ▷ $(ORBITER) -v 3 \n  ▷ ▷ -define F -finite_field -q 3 -end \n  ▷ ▷ -define SS -vector -dense $(SPREAD_SET_27_RAO_RAO) -end \n  ▷ ▷ -define G -linear_group -PGL 6 F -end \n  ▷ ▷ -define G1 -linear_group -PGL 7 F -end \n  ▷ ▷ -define S -spread -kernel_field F \n  ▷ ▷ ▷ -group G -k 3 -spread_set SS \n  ▷ ▷ ▷ -end \n  ▷ ▷ -define T -translation_plane S G G1 -end \n  ▷ ▷ ▷ -with T -do -translation_plane_activity \n  ▷ ▷ ▷ -export_incma \n  ▷ ▷ -end
```

creates the translation plane from the spread (there is an error message which we can ignore; this is because we did not create the stabilizer of the spread). To compute the 3-rank of the incidence matrix, we issue the following command:

```
RREF_Rao_Rao_plane_incma_rank:
  ▷ $(ORBITER) -v 2 \n  ▷ ▷ -define F -finite_field -q 3 -end \n  ▷ ▷ -define v -vector -field F \n  ▷ ▷ ▷ -file plane_incma.csv \n  ▷ ▷ ▷ -end \n  ▷ ▷ -with F -do -finite_field_activity \n  ▷ ▷ -RREF v -normalize_from_the_right \n  ▷ ▷ -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 PG(3, q) is a set of pairwise line-disjoint spreads of PG(3, q) of size $q^2 + q + 1$. Each spread contains $q^2 + 1$ lines. A simple counting argument shows that every line is contained in exactly one spread of the packing. The classification problem for packings is the problem of determining a complete set of pairwise non-isomorphic packings. Orbiter can be used to classify packings for small parameters. It is sometimes useful to make a symmetry assumption. This means that only those packings will be found that satisfy the symmetry assumption. The reason for making such an assumption is that the problem becomes easier and hence more tractable. Often, an assumption is made that the packings are invariant under a (nontrivial) group $H$. This section describes various ways in which Orbiter can help find and classify packings, with or without symmetry assumption.

Table 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 PG(3, 4). There are three isomorphism types of spreads in PG(3, 4). The command computes the orbits of each. In total, this gives 5096448 labeled spreads.

spread_table_PG_3_4:

There are 21 isomorphism types of spreads in PG(3, 5). The regular spread has Orbiter catalogue number equal to 12. The following command creates a table of all labeled regular spreads:

spread_table_PG_3_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
Table 12.6: Commands for creating BLT-sets

<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>

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

```
BLT_11_0:
  ▶ $(ORBITER) -v 2 \n  ▶ ▶ -define F -finite_field -q 11 -end \n  ▶ ▶ -define O -orthogonal_space 0 5 F -end \n  ▶ ▶ -with O -do -orthogonal_space_activity \n  ▶ ▶ ▶ -create_BLT_set -catalogue 0 -end \n  ▶ ▶ -end
  ▶ #pdflatex 0.1.6.2_report.tex
  ▶ #open 0.1.6.2_report.pdf
```

creates the BLT-set #0 in $Q(4, 11)$. The command

```
BLT_11_Mondello:
  ▶ $(ORBITER) -v 2 \n  ▶ ▶ -define F -finite_field -q 11 -end \n  ▶ ▶ -define O -orthogonal_space 0 5 F -end \n  ▶ ▶ -with O -do -orthogonal_space_activity \n```

388
<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>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)

<table>
<thead>
<tr>
<th>→</th>
<th>$18_1$</th>
<th>↓</th>
<th>$18_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>12_0</td>
<td>6</td>
<td>12_0</td>
<td>4</td>
</tr>
</tbody>
</table>

$C_0 = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}_{12}$
$C_1 = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17\}_{18}$

<table>
<thead>
<tr>
<th>→</th>
<th>$18_1$</th>
<th>↓</th>
<th>$18_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>12_0</td>
<td>6</td>
<td>12_0</td>
<td>4</td>
</tr>
</tbody>
</table>

$C_0 = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}_{12}$
$C_1 = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17\}_{18}$

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:**

- $(ORBITER) -v 2$
- $-define F -finite_field -q 13 -end$
- $-define O -orthogonal_space 0 5 F -end$
- $-define C -BLT_set_classifier 0 -starter_size 14 -end$
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 \
  with C -do -BLT_set.classifier O -starter_size 5 -end \
  compute_starter \
  problem_label BLT_q13 \
  -W -depth 5 \
  -end \
  -end \
  with C -do -BLT_set.classify_activity \
  create_graphs \
  -end
```

In the next step, we compute all rainbow cliques in each of the graphs:

```
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:
  $(ORBITER) -v 2 \n  -define F -finite_field -q 13 -end \n  -define O -orthogonal_space 0 5 F -end \n  -define C -BLT_set_classifier 0 -starter_size 5 -end \n  -with C -do -BLT_set_classify_activity \n  -compute_starter \n  -problem_label BLT_q13 \n  -W -depth 5 \n  -end \n  -end

BLT_13_isomorph_read_solutions:
  $(ORBITER) -v 2 \n  -define F -finite_field -q 13 -end \n  -define O -orthogonal_space 0 5 F -end \n  -define C -BLT_set_classifier 0 -starter_size 5 -end \n  -with C -do -BLT_set_classify_activity \n  -compute_starter \n  -problem_label BLT_q13 \n  -W -depth 5 \n  -end \n  -end
```

392
Then, we compute the stabilizer orbits, which are also known as the flag orbits:

```
BLT_13_isomorph_stabilizer_orbits:
  ▶ $ (ORBITER) -v 2 \n  ▶ ▶ -define F -finite_field -q 13 -end \n  ▶ ▶ -define 0 -orthogonal_space 0 5 F -end \n  ▶ ▶ -define C -BLT_set_classifier 0 -starter_size 5 -end \n  ▶ ▶ -with C -do -BLT_set_classify_activity \n  ▶ ▶ ▶ -compute_starter \n  ▶ ▶ ▶ ▶ -problem_label BLT_q13 \n  ▶ ▶ ▶ ▶ -W -depth 5 \n  ▶ ▶ ▶ ▶ -end \n  ▶ ▶ ▶ -end \n  ▶ ▶ ▶ -with C -do -BLT_set_classify_activity \n  ▶ ▶ ▶ -isomorph \n  ▶ ▶ ▶ ▶ -prefix_iso "./BLT_q13" \n  ▶ ▶ ▶ ▶ -use_database_for_starter \n  ▶ ▶ ▶ ▶ -compute_orbits \n  ▶ ▶ ▶ ▶ -list_of_cases BLT_q13_list_of_cases_5_0_1.csv \n  ▶ ▶ ▶ ▶ -solution_prefix "" \n  ▶ ▶ ▶ ▶ -base_fname "BLT_q13_graph" \n  ▶ ▶ ▶ ▶ -end \n  ▶ ▶ ▶ -end
```

Finally, we perform isomorphism testing:

```
BLT_13_isomorph_testing:
  ▶ $ (ORBITER) -v 4 \n  ▶ ▶ -define F -finite_field -q 13 -end \n  ▶ ▶ -define 0 -orthogonal_space 0 5 F -end \n  ▶ ▶ -define C -BLT_set_classifier 0 -starter_size 5 -end \n  ▶ ▶ -with C -do -BLT_set_classify_activity \n  ▶ ▶ ▶ -compute_starter \n  ▶ ▶ ▶ ▶ -problem_label BLT_q13 \n  ▶ ▶ ▶ ▶ -W -depth 5 \n  ▶ ▶ ▶ ▶ -report -end \n  ▶ ▶ ▶ ▶ -end \n  ▶ ▶ ▶ -end \n  ▶ ▶ ▶ -with C -do -BLT_set_classify_activity \n  ▶ ▶ ▶ -isomorph \n  ▶ ▶ ▶ ▶ -prefix_iso "./BLT_q13" \n  ▶ ▶ ▶ ▶ -use_database_for_starter \n  ▶ ▶ ▶ ▶ -isomorph_testing \n  ▶ ▶ ▶ ▶ -solution_prefix "" \n  ▶ ▶ ▶ ▶ -base_fname "BLT_q13_graph" \n```

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

```bash
Cycle_graph_13:
  $(ORBITER) -v 2 \
  -define Gamma -graph \
  -cycle 13 \
  -end
```

creates the cycle graph of degree 13.

There are many ways to read graphs from file. One way is by means of an adjacency matrix stored as a csv file. Consider an example. The `-load_csv_no_border` command can be used to create a graph from a csv file containing the adjacency matrix. The following command sequence uses a makefile variable to store the adjacency matrix of a graph. The matrix is then copied into a file and the graph is created from the file. Here is the makefile variable containing the adjacency matrix:

```
TRIANGLE_GRAPH="0,1,1\n1,0,1\n1,1,0"
```

And here is the command to create the csv file from the makefile variable and to create the graph from the csv file:

```bash
make_triangle_graph:
  echo $(TRIANGLE_GRAPH) >triangle_graph.csv
  $(ORBITER) -v 6 \
  -define G -graph \
  -load_csv_no_border \
  triangle_graph.csv \
```

395
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-load</code></td>
<td>filename</td>
<td>Read a graph from file in Orbiter format.</td>
</tr>
<tr>
<td><code>-load_csv_no_border</code></td>
<td>filename</td>
<td>Read a graph from a csv file. Ignore the first row and first column.</td>
</tr>
<tr>
<td><code>-load_dimacs</code></td>
<td>filename</td>
<td>Read a graph from file in dimacs format.</td>
</tr>
<tr>
<td><code>-edge_list</code></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><code>-edges_as_pairs</code></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><code>-cycle</code></td>
<td>$n$</td>
<td>Cycle graph on $n$ vertices.</td>
</tr>
<tr>
<td><code>-Hamming</code></td>
<td>$n$ $q$</td>
<td>Hamming graph $H(n,q)$</td>
</tr>
<tr>
<td><code>-Johnson</code></td>
<td>$n$ $k$ $s$</td>
<td>Johnson graph</td>
</tr>
<tr>
<td><code>-Paley</code></td>
<td>$q$</td>
<td>Paley graph</td>
</tr>
<tr>
<td><code>-Sarnak</code></td>
<td>$p$ $q$</td>
<td>Lubotzky-Phillips-Sarnak graph [46]</td>
</tr>
<tr>
<td><code>-Schlaefli</code></td>
<td>$q$</td>
<td>Schlaefli graph</td>
</tr>
<tr>
<td><code>-Shrikhande</code></td>
<td></td>
<td>Shrikhande graph</td>
</tr>
<tr>
<td><code>-Winnie_Li</code></td>
<td>$q$ $i$</td>
<td>Winnie-Li graph [45]</td>
</tr>
<tr>
<td><code>-Grassmann</code></td>
<td>$n$ $k$ $q$ $r$</td>
<td>Grassmann graph</td>
</tr>
<tr>
<td><code>-coll_orthogonal</code></td>
<td>$\epsilon$ $d$ $q$</td>
<td>Collinearity graph of $O(\epsilon(d,q))$</td>
</tr>
<tr>
<td><code>-trihedral_pair_disjointness_graph</code></td>
<td></td>
<td>Triheral pair disjointness graph</td>
</tr>
<tr>
<td><code>-non_attacking_queens_graph</code></td>
<td>$n$</td>
<td>Create the graph for non-attacking queens on a $n \times n$ chess board.</td>
</tr>
</tbody>
</table>

Table 13.1: Orbiter commands to define graphs (Part 1)
Table 13.2: Orbiter commands to define graphs (Part 2)

This will create the three-cycle graph.

The command

```
Chain_232:
▷ $(ORBITER) -v 2 \n▷ ▷ -define P1 -vector -dense 2,3,2 -end \n▷ ▷ -define P2 -vector -dense 2,3,2 -end \n▷ ▷ -define Gamma -graph \n▷ ▷ ▷ -chain_graph P1 P2 \n▷ ▷ -end
```

creates the chain graph with respect to the partitions (2, 3, 2) and (2, 3, 2).

The command

```
Paley_13_graph:
▷ $(ORBITER) -v 2 \n▷ ▷ -define Gamma -graph -Paley 13 -end \n```

creates the Paley graph on 13 vertices.
trihedral_pair_graph:
▷ $(ORBITER) -v 2 \n▷ ▷ -define Gamma \n▷ ▷ ▷ -graph -trihedral_pair_disjointness_graph \n▷ ▷ -end

creates the graph of trihedral pairs. Two vertices are adjacent if the associated trihedral pairs are line-disjoint.

The command

small_graph:
▷ $(ORBITER) -v 2 \n▷ ▷ -define G -graph -edges_as_pairs \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

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:
\[\texttt{\$ (ORBITER) -v 2 \ \\
\texttt{\textasciitilde define G -graph -Johnson 6 2 0 -end} \]

creates the Johnson graph \(J(6, 2, 0)\).

The command
\begin{verbatim}
Hamming_graph_3:
\texttt{\$ (ORBITER) -v 2 \ \\
\texttt{\textasciitilde define G -graph -Hamming 3 2 -end}
\end{verbatim}
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 \texttt{halljanko315.csv} is present, containing the incidence
matrix of the graph. The following command creates the graph from the file:
\begin{verbatim}
HJ_graph:
\texttt{\$ (ORBITER) -v 6 \ \\
\texttt{\textasciitilde define G -graph \ \\
\texttt{\textasciitilde load_csv_no_border \ \\
\texttt{\textasciitilde halljanko315.csv \ \\
\texttt{\textasciitilde -end \ \\
\end{verbatim}

In Section 15.7, we will compute the automorphism group of the graph (of order 1209600).
This will create a file \texttt{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:
\begin{verbatim}
HJ315_orbital_graph_3:
\texttt{\$ (ORBITER) -v 2 \ \\
\texttt{\textasciitilde define gens -vector -file \ \\
\texttt{\textasciitilde halljanko315\_gens.csv -end \ \\
\texttt{\textasciitilde define G -permutation_group \ \\
\texttt{\textasciitilde bsgs halljanko315 "File\_halljanko315" \ \\
\texttt{\textasciitilde 315 1209600 "0,1,2" 6 gens \ \\
\texttt{\textasciitilde -end \ \\
\texttt{\textasciitilde define Gamma -graph \ \\
\texttt{\textasciitilde \textasciitilde orbital_graph G 3 \ \\
\texttt{\textasciitilde \textasciitilde -end \ \\
\end{verbatim}

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</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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:
  $ (ORBITER) -v 6 \
  -define G -graph \
  -load_csv_no_border halljanko315.csv \
  -distance_2 \
  -end
```

Let us look at some examples of Cayley graphs. The first graph has $G = \mathbb{Z}_{11}$ and connection set all elements congruent 1 mod 3. We create the group as a subgroup of the one-dimensional affine group over $F_{11}$. This means that the map $x \mapsto ax + b \mod 11$ is encoded as a pair $(a, b)$.

```
Cayley_Z11_1mod3:
  $ (ORBITER) -v 2 \
  -define F -finite_field -q 11 -end \
  -define S -vector -dense \
  "1,1, 1,4, 1,7, 1,10" -end \
  -define G -linear_group -AGL 1 F \
  -subgroup_by_generators "Z11" 1 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
Cayley_Sym4_coxeter:
  ▶ $(ORBITER) -v 2 \n  ▶ ▶ -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 \n  ▶ ▶ -define Gamma -graph \n  ▶ ▶ ▶ -Cayley_graph G S \n  ▶ ▶ -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
Cayley_Sym4_star:
  ▶ $(ORBITER) -v 2 \n  ▶ ▶ -define S -vector -dense "1,0,2,3, 2,1,0,3, 3,1,2,0" -end \n  ▶ ▶ -define G -permutation_group -symmetric_group 4 \n  ▶ ▶ -end \n  ▶ ▶ -define Gamma -graph \n  ▶ ▶ ▶ -Cayley_graph G S \n  ▶ ▶ -end
```
<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:

```bash
Cycle_13.draw:
  $(ORBITER) -v 2 
  define Gamma -graph -cycle 13 -end 
  with Gamma -do 
  -graph_theoretic_activity -export_csv -end 
  with Gamma -do 
  -graph_theoretic_activity -export_graphviz -end 
  $(ORBITER) -v 2 -draw_matrix 
  -input_csv_file Cycle_13.csv 
  -box_width 20 -bit_depth 8 -partition 4 13 13 -end 
  dot -Tpng Cycle_13.gv >Cycle_13.png 
  #twopi -Tpng Cycle_13.gv >Cycle_13.png 
  #open Cycle_13_draw.bmp 
  #pdflatex Cycle_13_report.tex 
  #open Cycle_13_report.pdf
```

The command first creates the cycle graph of order 13, and then exports the adjacency matrix as csv file. It then draws the adjacency matrix as a bitmap graphics.

Suppose we want to compute the eigenvalues of the adjacency matrix of a graph. In the following example, the command `-eigenvalues` is used to compute the eigenvalues (both regular and Laplace) of the 9-cycle:

```bash
Cycle_9_eigenvalues:
  $(ORBITER) -v 2 
  define Gamma -graph 
  with Gamma -do 
  -cycle 9 
  -end 
  with Gamma -do 
  -graph_theoretic_activity -eigenvalues -end 
  pdflatex Cycle_9_eigenvalues.tex 
  open Cycle_9_eigenvalues.pdf
```

The following output is produced:
The energy is 11.5175
Eigenvalues: $\lambda_i$
Laplace eigenvalues: $\theta_i$

<table>
<thead>
<tr>
<th>$i$</th>
<th>$\lambda_i$</th>
<th>$\theta_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>3.87939</td>
</tr>
<tr>
<td>1</td>
<td>1.53209</td>
<td>3.87939</td>
</tr>
<tr>
<td>2</td>
<td>1.53209</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>0.347296</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>0.347296</td>
<td>1.6527</td>
</tr>
<tr>
<td>5</td>
<td>$-1$</td>
<td>1.6527</td>
</tr>
<tr>
<td>6</td>
<td>$-1$</td>
<td>0.467911</td>
</tr>
<tr>
<td>7</td>
<td>$-1.87939$</td>
<td>0.467911</td>
</tr>
<tr>
<td>8</td>
<td>$-1.87939$</td>
<td>$-2.26243e-16$</td>
</tr>
</tbody>
</table>

The command

\begin{verbatim}
Paley_13_draw:
$\text{(ORBITER)} -v 2 \ \
$-define Gamma -graph -Paley 13 -end \ 
$-with Gamma -do \ 
$-graph_theoretic_activity -export_csv -end \ 
$-with Gamma -do \ 
$-graph_theoretic_activity -export_graphviz -end \ 
$\text{(ORBITER)} -v 2 -draw_matrix \ 
$-input_csv_file Paley_13.csv \ 
$-box_width 20 -bit_depth 8 -partition 4 13 13 -end \ 
$\text{dot -Tpng Paley_13.gv >Paley_13.png} \ 
$\text{open Paley_13_draw.bmp}$
\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:

\begin{verbatim}
Cayley_Z11_1mod3_eigenvalues_and_draw:
$\text{(ORBITER)} -v 2 \ 
$-draw_options -xin 2000000 \ 
$-yin 2000000 -embedded -radius 20000 -end \ 
$-define F -finite_field -q 11 -end \ 
\end{verbatim}
The drawing is shown in Figure 13.1. Let us draw the Cayley graph of \( \text{Sym}(5) \) with respect to the Coxeter generators:

Cayley_Sym5.coxeter.draw:

```
$ (ORBITER) -v 2 \
-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-def G -permutation_group -symmetric_group 5 \n-end \\
```

405
The drawing is shown in Figure 13.2.

It is possible to create the collinearity graph of an incidence structure. The incidence structure must be encoded by means of an incidence matrix. Let us continue an example from Section 4.7, where the incidence matrix of $Q(4,2)$ was created. At that time, we wrote the incidence matrix to file. Here, we read the incidence matrix from file and create the collinearity graph of it:

```
PGO_5_2_collinearity_graph: 0_5_2_incidence_matrix.csv

$ (ORBITER) -v 3 \n
 define Inc -vector -file 0_5_2_incidence_matrix.csv -end \n
 define Gamma -graph -collinearity_graph Inc -end \n
 with Gamma -do \n
 -graph_theoretic_activity \n
 properties \n
 -end
```
The command also computes properties of the graph. The graph has 15 vertices of degree 6. This is because in the geometry, each point lies on three lines, and hence is collinear with 6 other points.
### Table 13.5: Options for classifying graphs

<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>

#### 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:

```bash
graph_classify_5:
  ▷ $(ORBITER) -v 2 \n  ▷ ▷ -orbi_path $(ORBITER_PATH) \n  ▷ ▷ -define GC -graph_classification \n  ▷ ▷ ▷ -n 5 \n  ▷ ▷ ▷ -poset_classification_control \n  ▷ ▷ ▷ ▷ -problem_label graphs_v5 \n  ▷ ▷ ▷ ▷ -depth 10 -draw_poset \n  ▷ ▷ ▷ ▷ -draw_options -radius 250 \n  ▷ ▷ ▷ ▷ -embedded -end \n  ▷ ▷ ▷ ▷ -report -end \n  ▷ ▷ ▷ -end \n  ▷ ▷ -end \n  ▷ ▷ -with GC -do \n  ▷ ▷ -graph_classification_activity \n  ▷ ▷ ▷ -list_graphs_at_level 5 5 \n  ▷ ▷ ▷ -end \n  ▷ ▷ -with GC -do \n  ▷ ▷ -graph_classification_activity \n  ▷ ▷ ▷ -draw_options \n  ▷ ▷ ▷ ▷ -radius 300 -nodes_empty \n  ▷ ▷ ▷ ▷ -line_width 1.5 \n  ▷ ▷ ▷ ▷ -scale 0.1 \n  ▷ ▷ ▷ -end \n  ▷ ▷ ▷ -draw_graphs_at_level 5 \n  ▷ ▷ ▷ -end \n  ▷ ▷ -print_symbols \n  ▷ pdflatex graphs_v5_level_5_reps.tex \n  ▷ open graphs_v5_level_5_reps.pdf \n  ▷ pdflatex graphs_v5_poset.tex \n  ▷ open graphs_v5_poset.pdf
```

408
After the classification, the graphs with 5 edges are shown. The file contains the following graph drawings:

![Graphs with 5 edges]

The next command classifies all tournaments on 4 vertices:

```
tournament_classify_4:
  ▶ $ (ORBITER) -v 2 \
  ▶   ‡ -define GC -graph_classification \n  ▶   ‡ -n 4 -tournament \n  ▶   ‡ -poset_classification_control \n  ▶   ‡   ‡ -problem_label tournament_4 \n  ▶   ‡   ‡ -depth 6 -draw_poset \n  ▶   ‡   ‡ -draw_options \n  ▶   ‡   ‡   ‡ -radius 250 -embedded \n  ▶   ‡   ‡   ‡ -end \n  ▶   ‡   ‡ -end \n  ▶   ‡ -end \n  ▶ -with GC -do \n  ▶ -graph_classification_activity \n  ▶   ‡ -draw_options \n  ▶   ‡   ‡ -radius 400 \n  ▶   ‡   ‡ -line_width 2 -scale 0.10 \n  ▶   ‡   ‡ -end \n  ▶   ‡ -draw_graphs_at_level 6 \n  ▶   ‡ -end \n  ▶ -print_symbols
  ▶ pdflatex tournament_4_level_6_reps.tex
  ▶ open tournament_4_level_6_reps.pdf
```

There are four tournaments. The following graph drawings are produced:
The next command classifies all cubic graphs on 8 vertices:

\begin{verbatim}
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 \\
  \$ -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
\end{verbatim}

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 \n  $with G -do \n  $-graph_theoretic_activity \n  $find_cliques -target_size 3 \n  $end
```

This command finds three cliques of size 3.

It is also possible to classify all cliques under the automorphism group of the graph. This is a multi-step process, though. At first, the automorphism group of the graph has to be computed. Then, poset classification can be invoked to classify the cliques of a certain size. Here is an example. We wish to classify the cliques in the Paley graph of order 13. The first command creates the graph and computes the automorphism group:

```bash
Paley_13_aut:
  $(ORBITER) -v 2 \n  $define Gamma -graph -Paley 13 -end \n```

412
The command writes a file `Paley_13_group.makefile`, shown below:

```
Paley_13:
  $(ORBITER_PATH)orbiter.out -v 2 \
  -define gens -vector -file Paley_13_gens.csv -end \
  -define G -permutation_group \
  -bsgs Paley_13 "Paley\_13" 13 78 "0,1" 3 gens -end \
```

The group is given using a base and strong generating set. The base consists of the two points 0, 1. Three strong generators with respect to this base are given in a csv file. Using this group, the next command classifies all cliques of size at most 5 in the Paley graph of order 13 under the action of the automorphism group. It turns out that there are no 5-cliques, and that the largest cliques have size 3. The command shows that there is a unique orbit of 3-cliques:

```
Paley_13_cliques_classify:
  $(ORBITER) -v 4 \
  -define Control -poset_classification_control \
  -W \
  -problem_label Paley13_cliques \
  -clique_test Gamma \
  -depth 5 \
  -end \
  -define gens -vector -file Paley_13_gens.csv -end \
  -define G -permutation_group \
  -bsgs Paley_13 "Paley\_13" 13 78 "0,1" 3 gens -end \
  -define Gamma -graph -Paley 13 -end \
  -define Orb -orbits -group G \
  -on_subsets 5 Control \
  -end
```

For comparison, the command

```
Paley_13_cliques_all:
  $(ORBITER) -v 10 \
  -define Gamma -graph -Paley 13 -end \
  -with Gamma -do \
  -graph_theoretic_activity \
```

413
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_3.colored_graph \n> -end \n> -with Gamma -do \n> -graph_theoretic_activity \n> -find_cliques -target_size 5 -end \n> -end
```

It turns out that there are exactly 1008 5-cliques. Concerning the classification with respect to the automorphism group of the graph, we apply the following command:

```
HJ64_cliques5_classify:
> $(ORBITER) -v 6 \
> -define Control -poset_classification_control \
> -W \n> -problem_label HJ64_cliques \n> -clique_test Gamma \n> -depth 5 \n> -end \n> -define Gamma -graph \n> -load \n> Group_Perms315_3.colored_graph \n> -end \n> -define gens -vector \n> -file halljanko315 gens.csv \n> -end \n> -define G -permutation_group \n> -bsgs halljanko315 "File\halljanko315" \n> 315 1209600 "0,1,42,95" 6 gens -end \n> -define Orb -orbits -group G \n> -on_subsets 5 Control \n> -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:

```plaintext
PGO_5_2_cliques: 0_5_2_incidence_matrix.csv
  $(ORBITER) -v 3 \ 
  -define Inc -vector -file 0_5_2_incidence_matrix.csv -end \ 
  -define Gamma -graph -collinearity_graph Inc -end \ 
  -with Gamma -do \ 
  -graph_theoretic_activity \ 
  -find_cliques -target_size 3 -end \ 
  -end
```

There are 15 cliques of size 3. They correspond to the lines in the geometry.
Chapter 14

Combinatorial Objects

14.1 Combinatorial Objects

Combinatorial objects are objects that are defined by means of a finite group action. The isomorphism problem for combinatorial objects is the question to decide whether two objects of the same type belong to the same orbit under the relevant group action. Orbiter offers a unified treatment of such questions for various types of objects. The main tool is the computation of a canonical form, as well as the automorphism group.

Combinatorial objects are coded as sequences of integers. Each type of object has its own coding. Coding of objects as integer sequences allows easy handling of objects. For instance, objects can be specified in a command line argument, or they can be stored in a file. Large numbers of objects can be stored in files.

In order to apply Orbiter commands, an input stream is defined. An input stream is a sequence of objects, all of the same kind. The objects can be defined using any of the commands listed in Table 14.1. The file types will be discussed in more detail in the next section. Here are some examples. First, we create the Hirschfeld surface. Since the Hirschfeld surface is a cubic surface, the object is defined using point ranks in the relevant projective space as described in Section 4.2. For instance, the Hirschfeld surface in PG(3, 4) is defined as 45 points, coded as 45 integers which are point ranks. A makefile variable is employed to define the set. The makefile variable is then used to define a set-object:

HIRSCHFELD_SURFAC_E_Q4_SET_OF_POINTS="0,1,2,3,4,5,6,7,8,9,\
10,11,12,13,14,23,26,27,30,31,34,35,38,39,42,47,48,51,52,\
53,54,59,60,61,62,67,68,69,70,75,76,79,80,81,82"

Hirschfeld
text

Hirschfeld_q4_from_set:
  $(ORBITER) -v 4 \
  -define H -set -here \
  $(HIRSCHFELD_SURFAC_E_Q4_SET_OF_POINTS) \n  -end \
  -define C -combinatorial_objects \n
417
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-set_of_points</td>
<td></td>
<td>A set consisting of points.</td>
</tr>
<tr>
<td>-set_of_lines</td>
<td></td>
<td>A set consisting of lines.</td>
</tr>
<tr>
<td>-set_of_points_and_lines</td>
<td></td>
<td>A set consisting of points and a second set consisting of lines.</td>
</tr>
<tr>
<td>-set_of_packing</td>
<td></td>
<td>A set of packings.</td>
</tr>
<tr>
<td>-file_of_points</td>
<td></td>
<td>A set consisting of points read from file.</td>
</tr>
<tr>
<td>-file_of_lines</td>
<td></td>
<td>A set consisting of lines read from file.</td>
</tr>
<tr>
<td>-file_of_packings</td>
<td></td>
<td>A set consisting of packings read from file.</td>
</tr>
<tr>
<td>-file_of_packings_through_spread_table</td>
<td></td>
<td>A file of packings.</td>
</tr>
<tr>
<td>-file_of_point_set</td>
<td></td>
<td>A file containing point sets.</td>
</tr>
<tr>
<td>-file_of_designs</td>
<td></td>
<td>A file containing designs or large sets.</td>
</tr>
<tr>
<td>-file_of_incidence_geometries</td>
<td>v b f</td>
<td>A file of incidence geometries defined by their set of flags. Here, v is the number of points, b is the number of blocks and f is the number of flags.</td>
</tr>
<tr>
<td>-file_of_incidence_geometries_by_row_ranks</td>
<td></td>
<td>A file describing incidence geometries defined by their row ranks.</td>
</tr>
<tr>
<td>-incidence_geometry</td>
<td>flags v b f</td>
<td>An incidence geometry defined by a set of flags. Here, v is the number of points, b is the number of blocks and f is the number of flags.</td>
</tr>
<tr>
<td>-incidence_geometry_by_row_ranks</td>
<td></td>
<td>An incidence geometry defined by row ranks.</td>
</tr>
</tbody>
</table>

Table 14.1: Defining Combinatorial Objects
The next example creates the two hyperovals in PG(2, 16). The hyperovals are stored in makefile variables:

```
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 \\
  -spreads.csv \\
  -SPREAD_TABLES_5_REG/spread_25_spreads.csv \\
  -5 \\
  -end
```

The following command reads a file of large sets of designs:
LS_AG_2.3_read:
▷ $(ORBITER) -v 2 \
▷ ▷ -define C -combinatorial_objects \
▷ ▷ ▷ -file_of_designs \
▷ ▷ ▷ solutions.csv 9 84 3 12 \
▷ ▷ -end

The next command reads incidence geometries from a file containing the flags:

dev_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
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

```bash
geo_pasch_read:
```

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 $\mathbb{F}_{11}$ and stored the set of $\mathbb{F}_q$-points in the file

```
elliptic_curve_b1_c3_q11.txt.
```

The following example computes the set stabilizer of the curve. This means it computes the set stabilizer of the points on the curve. In order to do so, an input stream is created which referst to the file containing the Orbiter point ranks of points on the curve.

```
EC_canon: elliptic_curve_b1_c3_q11.txt
  \$ \text{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
  open \text{EC\_classification.pdf}
$ \text{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 \text{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

Hirschfeld_q4_set_c:
In the next example, we compute the canonical form of the two hyperovals in PG(2, 16).

hyperoval_16.canonical_form:
> $(ORBITER) -v 2 \
> -define C -combinatorial_objects \
> -set.of.points $(HYPEROVAL_16_16320) \
> -set.of.points $(HYPEROVAL_16_144) \
> -end \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> -combinatorial_object_activity \n> -canonical_form PG P \n> -classification_prefix hyperoval_q16 \n> -label hyperoval_q16 \n> -save_agr \n> -save_transversal \n> -max_TDO_depth 10 \n> -end \n> -report \n> -prefix hyperoval_q16 \n> -export_flag_orbits \n> -export_TDO \n> -export_TDA \n> -dont_show_incidence_matrices \n> -export_group_GAP \

429
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 \texttt{PG\_3\_2\_subsets}. These orbits are relevant for Section 7.5. Concerning the work in Dickson [24] only subsets whose size is odd are relevant, so we restrict to those subsets:

\textbf{Dickson\_sets.stabilizer:}
\begin{verbatim}
$\$ (ORBITER) -v 3 \$
  \$ -define C -combinatorial_objects \$
  \$ -set_of_points "0,1,2,5,6" \$
  \$ -set_of_points "0,1,2,3,6" \$
  \$ -set_of_points "0,1,2,3,4" \$
  \$ -set_of_points "0,1,2,3,8" \$
  \$ -set_of_points "0,1,2,5,6,7,8" \$
  \$ -set_of_points "0,1,2,3,5,6,7" \$
  \$ -set_of_points "0,1,2,3,5,6,9" \$
  \$ -set_of_points "0,1,2,3,5,6,10" \$
  \$ -set_of_points "0,1,2,3,5,6,4" \$
  \$ -set_of_points "0,1,2,3,5,6,4" \$
  \$ -set_of_points "0,1,2,3,8,11,13" \$
  \$ -set_of_points "3,6,9,7,10,12,8,11,13,14,4" \$
  \$ -set_of_points "3,5,6,9,7,10,12,11,13,14,4" \$
  \$ -set_of_points "0,1,2,3,5,6,9,7,10,12,4" \$
  \$ -end \$
  \$ -define F -finite_field -q 2 -end \$
  \$ -define P -projective_space -n 3 -field F -v 0 -end \$
  \$ -with C -do \$
  \$ -combinatorial_object_activity \$
  \$ -canonical_form PG P \$
\end{verbatim}
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:

```
$\text{ovoid}_q8\_canon: \text{ovoid}_q8\.txt$

```

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 $q = 8$.

```
$\text{ovoid}_ST_q8\_canon: \text{ovoid}_ST_q8\.txt$

```

431
We can store the generators in a makefile variable as follows:

```
SUZUKI_8_GENERATORS="
1,0,0,0,0,1,0,0,0,1,0,0,0,1,1, \
1,0,0,0,0,6,0,0,0,0,2,0,0,0,0,3,0, \
1,0,0,0,1,1,0,0,0,1,0,0,0,1,0, \
1,0,0,0,3,6,2,2,5,0,2,0,3,0,6,3,2, \
0,1,0,0,1,0,0,0,0,0,0,1,0,0,1,0,2"
```

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```
-with G -do \
- group_theoretic_activity \
-report \
-end 
\pdflatex PGGL_4_8_Subgroup_Sz8_87360_report.tex 
open PGGL_4_8_Subgroup_Sz8_87360_report.pdf
15.3 Canonical Forms of Incidence Geometries

Let us consider a system of subsets. These 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 if 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 $v_k$ is an incidence geometry on a set of size $v$ and with $b$ lines such that each line has size $k$ and each point is contained in exactly $r$ lines. In the special case where $b = v$ and $k = r$, the name symmetric configuration $v_r$ is used (the term symmetric is somewhat misleading because the incidence matrix of a symmetric configuration need not be symmetric). Orbiter can be used to classify incidence geometries. One of the important steps in this process is computing a canonical form of the incidence geometry.

We will also be producing drawings of the incidence matrices of geometries. In these drawings, flags are indicated as heavy squares while anti-flags are drawn as small squares. The coloring will indicate the orbits of the automorphism group on flags and anti-flags. Objects with the same color belong to the same orbit. For a flag-transitive geometry, there is only one color for the incidences.

The following command computes the canonical form and a report of the projective plane PG(2, 2), which is a configuration 73.

```
geo_7_3_c:
\% $(ORBITER) -v 10 \n\% -draw_incidence_structure_description \n\% -width 60 -with_10 6 -end \
```

434
A bitmap drawing is produced, as shown in Figure 15.1. The command also produces the following report of the geometry:

**Summary of Orbits**

<table>
<thead>
<tr>
<th>Rep</th>
<th>#</th>
<th>Ago</th>
<th>Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>168</td>
</tr>
</tbody>
</table>

Ago: 168
Isomorphism type 0 / 1

Isomorphism type 0 / 1 is original object 0 and appears 1 times:
This isomorphism type appears 1 times, namely for the following 1 input objects:
{0}
incidence structure:
( 0, 1, 2, 7, 10, 11, 14, 19, 20, 22, 24, 26, 29, 32, 34, 37, 38, 41, 44, 46, 47 )
Column sets of the encoded object:
{ 0, 1, 2 }
{ 0, 3, 4 }
{ 0, 5, 6 }
{ 1, 3, 5 }
{ 1, 4, 6 }
{ 2, 3, 6 }
{ 2, 4, 5 }
Row sets of the encoded object:
{ 0, 1, 2 } = 0
{ 0, 3, 4 } = 9
{ 0, 5, 6 } = 14
{ 1, 3, 5 } = 20
{ 1, 4, 6 } = 23
{ 2, 3, 6 } = 27
{ 2, 4, 5 } = 28
Generators for the automorphism group:

The stabilizer of order 168 is generated by:

- \( g_1 = (3, 5)(4, 6)(8, 9)(12, 13) \) of order 2 and with 6 fixed points.
- \( g_2 = (3, 4)(5, 6)(10, 11)(12, 13) \) of order 2 and with 6 fixed points.
- \( g_3 = (1, 2)(5, 6)(10, 12)(11, 13) \) of order 2 and with 6 fixed points.
- \( g_4 = (1, 3)(2, 4)(7, 8)(11, 12) \) of order 2 and with 6 fixed points.
- \( g_5 = (0, 1)(4, 5)(8, 10)(9, 11) \) of order 2 and with 6 fixed points.

Canonical labeling:

- canonical row = 6
- canonical orbit number = 0
- Flags: ( 0, 1, 2, 7, 10, 11, 14, 19, 20, 22, 24, 26, 30, 31, 34, 36, 39, 41, 44, 46, 47 )

Flag orbits:
- orbit length : number of orbits of that length:
  - 21 1

Anti-Flag orbits:
- orbit length : number of orbits of that length:
  - 28 1

The following command computes the canonical form and a report of the affine plane \( AG(2, 3) \), which is a configuration \( 9_412_3 \).

```
AG_2_3_c: AG_2_3.inc
  $\text{ORBITER} \ -v 2 \ \
  \ -define C -combinatorial_objects \ 
  \ -file_of_incidence_geometries \ 
  \ -AG_2_3.inc 9 12 36 \ 
  \ -end \ 
  \ -with C -do \ 
  \ -combinatorial_object_activity \ 
  \ -canonical_form \ 
  \ -classification_prefix AG_2_3 \ 
  \ -label AG_2_3 \ 
```

437
A bitmap drawing is produced, shown in Figure 15.2. Because the geometry is flag transitive, there is only one color being used for the incidence. In fact, all incidences are in black. The geometry is also anti-flag transitive. This can be seen from the fact that there is only one color in the picture for the smaller boxes, which represent anti-flags. Orbiter also produces the following report of the geometry:
Summary of Orbits

<table>
<thead>
<tr>
<th>Rep</th>
<th>#</th>
<th>Ago</th>
<th>Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>432</td>
</tr>
</tbody>
</table>

Ago :432

**Isomorphism type 0 / 1**

Isomorphism type 0 / 1 is original object 0 and appears 1 times:
This isomorphism type appears 1 times, namely for the following 1 input objects:
{0}

incidence structure:
( 0, 1, 2, 3, 12, 16, 17, 18, 24, 31, 32, 33, 37, 40, 43, 46, 49, 53, 56, 59, 62, 64, 69, 71, 74, 78, 80, 82, 87, 89, 93, 94, 99, 102, 103, 107 )

Generators for the automorphism group:
The stabilizer of order 432 is generated by:
g_1 = (3, 4)(5, 7)(6, 8)(11, 12)(13, 14)(16, 17)(19, 20) of order 2 and with 7 fixed points.
g_2 = (3, 5)(4, 6)(7, 8)(10, 11)(14, 15)(16, 18)(19, 20) of order 2 and with 7 fixed points.
g_3 = (1, 3)(2, 4)(7, 8)(9, 10)(14, 16)(15, 19)(18, 20) of order 2 and with 7 fixed points.
g_4 = (0, 1)(4, 5)(6, 7)(10, 13)(11, 14)(12, 15)(17, 18) of order 2 and with 7 fixed points.

Decomposition by combinatorial refinement:

\[
\begin{array}{c|c}
\rightarrow & 12_1 \\
9_0 & 4 \\
\downarrow & 12_1 \\
9_0 & 3 \\
\end{array}
\]
Decomposition by automorphism group:

Canonical labeling:
canonical row = 6
canonical orbit number = 0
Flags: (0, 1, 2, 3, 12, 16, 17, 18, 24, 31, 32, 33, 37, 40, 43, 46, 49, 53, 56, 59, 62,
64, 69, 71, 75, 78, 79, 83, 87, 89, 93, 94, 98, 102, 104, 106)

Flag orbits:
orbit length : number of orbits of that length:

36 1

Anti-Flag orbits:
orbit length : number of orbits of that length:

72 1

It is possible to perform isomorph classification for configurations based on incidence files. Suppose we want to check that the configurations in 10_3 are in fact all nonisomorphic. We apply the canonical form algorithm given by Nauty. This produces a transversal of the isomorphism types of incidence geometries from the given list of input objects. The objects are specified by means of the combinatorial_objects command. The classification algorithm can print a report which lists the transversal and all elements in it in latex form.

geo_10_3_c:
- $(ORBITER) -v 10 \$
- -draw_incidence_structure_description \$
- -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 \$

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, 3², 4², 6, 10, 12, 24, 120

Isomorphism type 0 / 10

Isomorphism type 0 / 10 is original object 9 and appears 1 times:
This isomorphism type appears 1 times, namely for the following 1 input objects:
{9}
incidence structure:
( 0, 1, 2, 10, 13, 14, 20, 25, 26, 31, 33, 35, 41, 44, 47, 52, 53, 58, 62, 66, 69, 74, 78, 79, 85, 87, 89, 96, 97, 98 )

Generators for the automorphism group:
The stabilizer of order 3 is generated by:
g₁ = (0, 1, 3)(2, 5, 4)(6, 7, 8)(10, 13, 11)(12, 14, 15)(16, 18, 17) of order 3 and with 2 fixed points.
Decomposition by automorphism group:

1013112141516181719

Canonical labeling:
  canonical row = 5
  canonical orbit number = 1
  Flags : 0,1,2,16,17,18,25,27,29,34,38,39,40,43,45,51,53,56,62,63,64,70,74,77,82,86,89,91,95,98,

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:

![Graphical representation of the automorphism group]

Canonical labeling:
- canonical row = 0
- canonical orbit number = 0
- Flags: 0, 1, 2, 15, 18, 19, 24, 26, 29, 33, 37, 40, 43, 44, 50, 55, 56, 61, 67, 68, 72, 75, 77, 82, 84, 88, 91, 93, 96

The following command computes the canonical form for the three triangle free configurations
24 of found by Abdullah Alazemi. These configurations have 24 points, 24 lines, each line
consists of 3 points and each point is on 3 lines.

```
FILE_24_3_TFC_INC="24 24 72"
\n0 1 2 24 27 28 48 53 54 73 79 80 97 105 106 122 131
132 146 157 158 171 175 183 195 203 208 220 225 233 244
258 259 269 272 282 293 300 308 318 324 327 342 354 358
367 379 381 392 398 400 417 428 429 442 443 450 466 471
479 492 497 502 517 519 521 542 548 551 571 574 575
48\n\n0 1 2 24 27 28 48 53 54 73 79 80 97 105 106 122 131
132 146 157 158 171 175 183 195 203 208 220 225 233 244
258 259 269 272 281 293 301 308 318 324 327 342 354 357
```
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
Figure 15.3: A flag transitive $24_3$ configuration

the three geometries is flag transitive. A bitmap drawing is produced, shown in Figure 15.3. Because the geometry is flag transitive, there is only one color being used for the incidence. In fact, all incidences are in black.
15.4 Canonical Forms of Objects from Design Theory

In Section 11.5, designs have been created. In order to compute properties of the design, we export the incidence matrix to file. After that, we compute the canonical form of the design, which allows us to determine many properties. The following example computes the properties of PG(2,3):

design_PG_2_3_canonical:
  > $(ORBITER) -v 3 \
  >   -define F -finite_field -q 3 -end \
  >   -define D -design -field F -family PG_2_q -end \
  >   -with D -do \
  >   > -design_activity \
  >   >   -export_inc \
  >   > -end \
  >   -end \
  > $(ORBITER) -v 3 \
  >   -draw_incidence_structure_description \
  >   > -width 60 -with_10 6 -end \
  >   -define C -combinatorial_objects \
  >   > -file_of_incidence_geometries PG_2_3_inc.txt 13 13 52 \
  >   -end \
  >   -with C -do \
  >   > -combinatorial_object_activity \
  >   >   -canonical_form \
  >   >   > -classification_prefix PG_2_3 \
  >   >   > -label PG_2_3 \
  >   >   > -save_ago \
  >   >   > -save_transversal \
  >   >   -end \
  >   > -report \
  >   >   -prefix PG_2_3 \
  >   >   > -export_flag_orbits \
  >   >   > -show_incidence_matrices \
  >   >   > -export_group_GAP \
  >   >   -end \
  >   -end \
  > pdfflatex 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

447
The command

\texttt{wreath\_product\_designs\_n4\_k2\_c: wreath\_product\_designs\_n4\_k2\_inc.txt}
\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 \}
\texttt{\> \> \> \> wreath\_product\_designs\_n4\_k2\_inc.txt \}
\texttt{\> \> \> \> 8 12 24 \}
\texttt{\> \> \> \> -end \}
\texttt{\> \> \> -with C -do \}
\texttt{\> \> \> \> -combinatorial\_object\_activity \}
\texttt{\> \> \> \> \> -canonical\_form \}
\texttt{\> \> \> \> \> \> -classification\_prefix wreath\_4\_2 \}
\texttt{\> \> \> \> \> \> \> -label wreath\_4\_2 \}
\texttt{\> \> \> \> \> \> \> \> -save\_ago \}
\texttt{\> \> \> \> \> \> \> \> \> -save\_transversal \}
\texttt{\> \> \> \> \> \> \> \> \> -end \}
\texttt{\> \> \> \> \> \> \> \> \> -report \}
\texttt{\> \> \> \> \> \> \> \> \> \> -prefix wreath\_4\_2 \}
\texttt{\> \> \> \> \> \> \> \> \> \> \> -export\_flag\_orbits \}
\texttt{\> \> \> \> \> \> \> \> \> \> \> \> -show\_incidence\_matrices \}
\texttt{\> \> \> \> \> \> \> \> \> \> \> \> \> -export\_group\_GAP \}
\texttt{\> \> \> \> \> \> \> \> \> \> \> \> \> \> -end \}
\texttt{\> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> pdflatex wreath\_4\_2.classification.tex}
\texttt{\> open wreath\_4\_2.classification.pdf}

computes the automorphism group of the design on 8 points created in Section 11.5. The group is $\text{Sym}(4) \wr \text{Sym}(2)$. The command

\texttt{wreath\_product\_designs\_n8\_k6\_c: wreath\_product\_designs\_n8\_k6\_inc.txt}
\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 \}
\texttt{\> \> \> \> wreath\_product\_designs\_n8\_k6\_inc.txt \}
\texttt{\> \> \> \> 16 3920 23520 \}
\texttt{\> \> \> \> -end \}

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 $\text{AG}(2, 3)$.

\texttt{LS\_AG\_2\_3\_solutions\_classify:}

\begin{verbatim}
\$ (ORBITER) -v 2 \n\$ \$ -draw.incidence_structure.description \n\$ \$ -width 20 -width_10 2 -end \n\$ \$ -define C -combinatorial.objects \n\$ \$ -file_of_designs \n\$ \$ -solutions.csv 9 84 3 12 \n\$ \$ -end \n\$ \$ -with C -do \n\$ \$ -combinatorial.object.activity \n\$ \$ \$ -canonical_form \n\$ \$ \$ -save_ago \n\$ \$ \$ -save_transversal \n\$ \$ \$ -classification.prefix LS\_AG\_2\_3 \n\$ \$ \$ -label LS\_AG\_2\_3 \n\$ \$ \$ -max_TDO_depth 10 \n\$ \$ \$ -end \n\$ \$ \$ -report \n\$ \$ \$ \$ -prefix LS\_AG\_2\_3 \n\end{verbatim}
It turns out that there are exactly two isomorphism types, with automorphism groups of order 54 and 42, respectively.
15.5 Canonical Forms of Linear Codes

Orbiter can compute canonical forms and automorphism groups of codes using Nauty. For linear codes, the semilinear automorphism group can be computed.

Consider the \([3, 2, 2]\) code generated by

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

The semilinear automorphism group can be computed using the following command:

code 3 2 aut:
▷ $(ORBITER) -v 20 \\
▷ ▷ -define F -finite_field -q 2 -end \\
▷ ▷ -define genma -vector -field F -format 2 \\
▷ ▷ ▷ -dense $(CODE_N3_K2_Q2_GENMA) \\
▷ ▷ -end \\
▷ ▷ -define P -projective_space -n 1 -field F -v 0 -end \\
▷ ▷ -with P -do \\
▷ ▷ -projective_space_activity \\
▷ ▷ ▷ -canonical_form_of_code \\
▷ ▷ ▷ ▷ "3_2" genma -save_ago -label "3_2" \\
▷ ▷ ▷ ▷ -classification_prefix "3_2" \\
▷ ▷ ▷ ▷ -end \\
▷ ▷ ▷ -end
▷ ▷ pdflatex 3_2_classification.tex
▷ ▷ open 3_2_classification.pdf
▷ $(ORBITER) -v 2 -draw_matrix \\
▷ ▷ -input_csv_file 3_2_object0_TDA_flag_orbits.csv \\
▷ ▷ -secondary_input_csv_file 3_2_object0_TDA.csv \\
▷ ▷ -box_width 16 -bit_depth 24 \\
▷ ▷ -end
▷ open 3_2_object0_TDA_flag_orbits_draw.bmp

The code has a semilinear automorphism group of order 6. The following report is written:

<table>
<thead>
<tr>
<th>Rep</th>
<th># Ago</th>
<th>Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Isomorphism type 0 / 1

Isomorphism type 0 / 1 is original object 0 and appears 1 times:
This isomorphism type appears 1 times, namely for the following 1 input objects:
{0}
set of points of size 3: ( 0, 1, 2 )

<table>
<thead>
<tr>
<th>i</th>
<th>Rank</th>
<th>Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>(1,0)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>(0,1)</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>(1,1)</td>
</tr>
</tbody>
</table>

Column sets of the encoded object:
{ 0, 1, 2 }
{ 0, 1, 2, 3 }
Row sets of the encoded object:
{ 0, 1 } = 0
{ 0, 1 } = 0
{ 0, 1 } = 0
{ 1 } = 1

Generators for the automorphism group:
The stabilizer of order 6 is generated by:
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 = \[
\begin{bmatrix}
1 & 0 \\
1 & 1
\end{bmatrix}
\] of order 2 and with 1 fixed points.
g_2 = \[
\begin{bmatrix}
0 & 1 \\
1 & 0
\end{bmatrix}
\] of order 2 and with 1 fixed points.

Decomposition by combinatorial refinement:
Decomposition by automorphism group:
\[
\begin{array}{c|c}
40 & 2 \\
\hline
21 & 2 \\
\end{array}
\]
\[
\begin{array}{c|c}
40 & 3 \\
\hline
21 & 3 \\
\end{array}
\]

Canonical labeling:
canonical row = 3
canonical orbit number = 1
Flags : ( 0, 1, 2, 3, 4, 5, 7 )
\[
\begin{array}{c|c}
45 & \star \\
\hline
3 & \star \\
\end{array}
\]

Flag orbits:
orbit length : number of orbits of that length:
\[
\begin{array}{c|c}
1 & 1 \\
3 & 2 \\
\end{array}
\]

Anti-Flag orbits:
orbit length : number of orbits of that length:
\[
\begin{array}{c|c}
1 & 1 \\
\end{array}
\]

The command

```bash
CODE_RM_3_1_GENMA="\\
11111111\\
01010101\\
00110011\\
00001111"
```

```
RM_3_1_group:
$ (ORBITER) -v 2 \n  -define F -finite_field -q 2 -end \n  -define genma -vector -field F -format 4 \n  -compact $(CODE_RM_3_1_GENMA) \n  -end \n```

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
$ (ORBITER) -v 2
$ define F -finite_field -q 2 -end
$ define genma -vector -field F -format 4
$ -compact $(CODE_RM_3_1_GENMA)
$ -define P -projective_space -n 3 -field F -v 0 -end
$ -with P -do
$ -canonical_form_of_code
$ "RM_3_1" genma -save_ago -label "RM_3_1"
$ -classification_prefix "RM_3_1"
$ -end
$ -end
$ pdflatex RM_3_1_classification.tex
$ open RM_3_1_classification.pdf
```

```bash
$ (ORBITER) -v 2 -draw_matrix
$ -input_csv_file RM_3_1_object0_INP_flag_orbits.csv
$ -secondary_input_csv_file RM_3_1_object0_INP.csv
$ -box_width 16 -bit_depth 24
$ -end
$ $(ORBITER) -v 2 -draw_matrix
$ -input_csv_file RM_3_1_object0_TDA_flag_orbits.csv
$ -secondary_input_csv_file RM_3_1_object0_TDA.csv
$ -box_width 16 -bit_depth 24
$ -end
$ open RM_3_1_object0_INP_flag_orbits_draw.bmp
```
Figure 15.4: PG(3, 2) with the Reed-Muller code distinguished

```
open RM.3.1_object0_TDA_flag_orbits_draw.bmp
```

The drawing in Figure 15.4 is created.

The command

```
RS_6.4_7_group:
  $(ORBITER) -v 20 \
  -define F -finite_field -q 7 -end \n  -define genma -vector -field F -format 4 \n  -compact $(CODE_RS_6_4_7) \n  -define P -projective_space -n 3 -field F -v 0 -end \n  -with P -do \n  -projective_space_activity \n  -canonical_form_of_code \n  "RS_6" genma -save_ago -label "RS_6" \n  -classification_prefix "RS_6" \n  -end \n  -end
```

shows that the automorphism group has order 12. After some shortening, the output is:

Isomorphism type 0 / 1 is original object 0 and appears 1 times:
set of points of size 6: {(0, 9, 51, 344, 253, 3)}
<table>
<thead>
<tr>
<th>i</th>
<th>Rank</th>
<th>Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>(1,0,0,0)</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>(5,1,0,0)</td>
</tr>
<tr>
<td>2</td>
<td>51</td>
<td>(6,5,1,0)</td>
</tr>
<tr>
<td>3</td>
<td>344</td>
<td>(0,6,5,1)</td>
</tr>
<tr>
<td>4</td>
<td>253</td>
<td>(0,0,4,1)</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>(0,0,0,1)</td>
</tr>
</tbody>
</table>

Group order 12
This isomorphism type appears 1 times, namely for the following 1 input objects:
{0}
Stabilizer:
Strong generators for a group of order 12:
\[
\begin{pmatrix}
1 & 0 & 0 & 0 \\
2 & 0 & 0 & 1 \\
5 & 0 & 6 & 0 \\
5 & 1 & 0 & 0 \\
\end{pmatrix},
\begin{pmatrix}
0 & 0 & 0 & 3 \\
4 & 0 & 0 & 6 \\
0 & 4 & 0 & 1 \\
0 & 0 & 4 & 1 \\
\end{pmatrix}
\]

\[1,0,0,0,2,0,0,1,5,0,6,0,5,1,0,0,0,0,0,1,6,0,0,2,0,6,0,5,0,0,6,5,\]

\[\rightarrow \ 2850_1 \ 1_2 \]
\[401_0 \ 57 \ 1 \]

The command

GV_n15_k6_d5_group:
\[
\$\text{(ORBITER)} -v 20 \ \\
\ \ \ \ -\text{define F} -\text{finite_field} -q 2 -\text{end} \ \\
\ \ \ \ -\text{define genma} -\text{vector} -\text{field F} -\text{format 6} \ \\
\ \ \ \ \ -\text{compact $\text{CODE_GV_{N15_K6}}$} \ \\
\ \ \ \ \ -\text{end} \ \\
\ \ \ \ -\text{define P} -\text{projective_space} -n 5 -\text{field F} -v 0 -\text{end} \ \\
\ \ \ \ \ -\text{with P} -\text{do} \ \\]

456
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

```sh
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 $2688 = 16 \cdot 168$.  

458
15.7 Canonical Forms of Graphs

Orbiter can compute isomorphism and automorphism between graphs. Here are some examples.

Suppose we want to compute the automorphism group of the cycle graph of order 13:

```
Cycle_13_aut:
  $(ORBITER) -v 2 \
  -define Gamma -graph -cycle 13 -end \
  -with Gamma -do \
  -graph_theoretic_activity -automorphism_group \
  -end \
```

The output is two files: The first one, `Cycle_13_group.makefile` is a makefile containing an Orbiter command to create the automorphism group: The second file is `Cycle_13_gens.csv`, which contains the permutation representation of the group, and which is needed for the makefile.

The next command computes the automorphism group of the chain graph with respect to the partition $(2, 3, 2)$.

```
Chain_232_aut:
  $(ORBITER) -v 2 \
  -define P1 -vector -dense 2,3,2 -end \
  -define P2 -vector -dense 2,3,2 -end \
  -define Gamma -graph \
  -chain_graph P1 P2 \
  -with Gamma -do \
  -graph_theoretic_activity -automorphism_group \
  -end \
  pdflatex chain_graph_report.tex \
  open chain_graph_report.pdf
```

The following report is written:

```
The automorphism group of `chain_graph` has order 1152 and is generated by:
Strong generators for a group of order 1152:

(12, 13),
(3, 4),
(2, 3),
```

459
Junntila and Kaski maintain a collection of graphs that can be used as test cases. The graphs are stored in Dimacs format and can be read in through the Orbiter -load_dimacs command. For instance, the following command computes the automorphism group of the Levi graph of the desarguesian projective plane of order 16:

```
JK_graph_pp16_1:
  $(ORBITER) -v 2 \
  -define Gamma -graph -load_dimacs \
  ../JUNTTILA_KASKI/benchmarks/pp/pp16-1 \
  -end \
  -with Gamma -do \n  -graph_theoretic_activity -save -end \n  -with Gamma -do \n  -graph_theoretic_activity -automorphism_group -end \
```

The command shows a group of order 34217164800. As a measurement of the complexity, the number of backtrack nodes in Nauty is recorded:

```
nb_backtrack1 = 6
nb_backtrack2 = 134
nb_backtrack3 = 134
nb_backtrack4 = 1
```

Here, nb_backtrack1 is the number of calls to firstpathnode, nb_backtrack2 is the number of calls to othernode, nb_backtrack3 is the number of calls to processnode,
nb_backtrack4 is the number of calls to firstterminal. These are the four recursive functions in Nauty.

Unfortunately, the complexity of graph isomorphism is not well-understood. We can see this here. While the first projective plane of order 16 can be handled relatively easily, the second one causes problems. The following command hardly finishes:

```
JK_graph_pp16_2:
  $ (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.

```
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:

```
HJ_aut:
  $(ORBITER) -v 6 \
  $ -define G -graph \
  $ $-load_csv_no_border \n  $ $-define G -graph \
  $ -with G -do \
  $ $-graph_theoretic_activity -automorphism_group \
  $ -end \
  $ -with G -do \
  $ $-graph_theoretic_activity -properties \
  $ -end
```

The next step is to compute the orbits of the automorphism group on pairs, using the following command:

```
HJ_group_and_orbits:
  $(ORBITER) -v 2 \
  $ $-define Control -poset_classification_control \
  $ $-define Control -poset_classification_control \
  $ -W \n  $ -problem_label HJ_orbits \n  $ $-depth 2 \n  $ -end \n  $ $-define gens -vector -file \
  $ $-define gens -vector -file \
  $ $-define G -permutation_group \
  $ $-define G -permutation_group \
  $ $-bsgs halljanko315 "File\_halljanko315" \n  $ $-bsgs halljanko315 "File\_halljanko315" \n  $ $-on_subsets 2 Control \n  $ $-on_subsets 2 Control \n  $ -end
```

There are 4 orbits on pairs. We decide to pick the fourth orbit to create a new graph. Because indexing is zero-based, we give the orbit index of 3:

```
HJ_orbital_graph_3:
  $(ORBITER) -v 2 \
  $ $-define gens -vector -file \
  $ $-define gens -vector -file 
  $ $-define G -permutation_group \
  $ $-define G -permutation_group 
  $ $-bsgs halljanko315 "File\_halljanko315" 
  $ $-bsgs halljanko315 "File\_halljanko315" 
  $ $-on_subsets 2 Control \
  $ $-on_subsets 2 Control 
  $ -end
```
\begin{verbatim}
\$define G -permutation_group \\
\$define Gamma -graph \\
\$with Gamma -do \\
\$graph theoric_activity \\
\$properties \\
\$end \\
\$with Gamma -do \\
\$graph theoric_activity \\
\$save \\
\$end

The graph is regular of degree 64.

The next command computes the automorphism group of the collinearity graph of the $Q(4,2)$ quadric.

\texttt{PGO\_5\_2\_graph\_group: 0\_5\_2\_incidence\_matrix.csv}

\begin{verbatim}
\$\texttt{(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 theoric_activity \\
\$automorphism_group \\
\$end \\
\$with Gamma -do \\
\$graph theoric_activity \\
\$eigenvalues \\
\$end

pdflatex collinearity\_graph\_eigenvalues.tex
open collinearity\_graph\_eigenvalues.pdf
\end{verbatim}

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.
\end{verbatim}
15.8 Canonical Forms of Quartic Curves

We wish to study the automorphism groups of certain quartic curves introduced by Edge. We start by creating a cheat sheet of the field $\mathbb{F}_{17}$

\begin{verbatim}
F_17_edge:
  $\text{(ORBITER)} \ -v \ 3 \ \
  \text{-define F -finite_field -q 17 -end}\ 
  \text{-with F -do -finite_field_activity}\ 
  \text{-cheat_sheet_GF -end}\ 
  \text{pdflatex GF}\ 
  \text{open GF_17.pdf}\ 
\end{verbatim}

Next, we compute the canonical form of the Edge quartic. This command also computes generators for the automorphism group of the curve.

\begin{verbatim}
Edge_curve_17_nauty:
  $\text{(ORBITER)} \ -v \ 3 \ \
  \text{-define C -combinatorial_objects}\ 
  \text{-file of points Edge_q17.txt}\ 
  \text{-end}\ 
  \text{-define F -finite_field -q 17 -end}\ 
  \text{-define P -projective_space -n 2 -field F -v 0 -end}\ 
  \text{-with C -do}\ 
  \text{-combinatorial_object_activity}\ 
  \text{-canonical_form_PG P}\ 
  \text{-classification_prefix Edge_curve_q17}\ 
  \text{-label Edge_curve_q17}\ 
  \text{-save ago}\ 
  \text{-save_transversal}\ 
  \text{-max_TDO_depth 10}\ 
  \text{-end}\ 
  \text{-report}\ 
  \text{-prefix Edge_curve_q17}\ 
  \text{-export_flag_orbits}\ 
  \text{-show_TDO}\ 
  \text{-show_TDA}\ 
  \text{-dont_show_incidence_matrices}\ 
  \text{-export_group_GAP}\ 
  \text{-end}\ 
  \text{pdflatex Edge_curve_q17_classification.tex}\ 
  \text{open Edge_curve_q17_classification.pdf}\ 
  $\text{(ORBITER)} \ -v \ 2 \ -draw_matrix\ 
  \text{-input_csv_file Edge_curve_q17_object0_TDA_flag_orbits.csv}\ 
\end{verbatim}
Using the generators that have just been computed, we can recreate the group of the quartic curve:

\begin{verbatim}
Edge_curve_17_group:
\$ (ORBITER) -v 3 \
\$ -define G -linear_group -PGL 3 17 \
\$ -subgroup_by_generators "Stab_Edge" "24" 3 \
\$   "1,0,0,0,13,0,0,0,4" \
\$   "1,0,0,0,0,16,0,16,0" \
\$   "0,1,16,2,4,4,15,4,4" \
\$ -end \
\$ -with G -do \
\$ -group_theoretic_activities \
\$ -print_elements_tex \
\$ -group_table \
\$ -report \
\$ -end
pdflatex PGL_3_17_Subgroup_Stab_Edge_24_report.tex
open PGL_3_17_Subgroup_Stab_Edge_24_report.pdf
\end{verbatim}
Chapter 16

Interfaces

16.1 Graphical Output

Orbiter can produce graphical output in a variety of formats:

1. TikZ / Latex [67],
2. Metapost [34],
3. Bitmap files (.bmp) [70],
4. Povray, see Section 16.2.

Bitmaps can be created using the `-draw_matrix` command. The input is an integer-valued matrix in csv format. The matrix entries are translated into colors. The possible commands after `-draw_matrix` are shown in Table 16.1. Suppose we want to create a graphical representation of the addition table of the finite field $\mathbb{F}_7$. 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 width $w$.</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
F.7.tables:
▷ $(ORBITER) -v 3 \\
▷ ▷ -define F -finite_field -q 7 -end \\
▷ ▷ -with F -do -finite_field_activity \\
▷ ▷ ▷ -cheat_sheet_GF \\
▷ ▷ -end \\
▷ $(ORBITER) -v 2 \\
▷ ▷ -draw_matrix \\
▷ ▷ ▷ -input_csv_file GF_q7_addition_table.csv \\
▷ ▷ ▷ -box_width 40 \\
▷ ▷ ▷ -bit_depth 24 \\
▷ ▷ ▷ -partition 3 7 7 \\
▷ ▷ -end \\
▷ open GF_q7_addition_table_draw.bmp

The finite field activity `-cheat_sheet_GF` creates the file

```
GF_q7_addition_table.csv
```

which is used as the input for the second command. The file content is:

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

The second command creates the diagram in Figure 16.1. The `-partition` command is used to define an outline of width 3 pixes. The all-in-one partition 7 is used as both row-partition and column-partition.

The planes PG(2,q) admit a cyclic automorphism group known as the Singer cycle. The command

```
PG.2.4.cyclic_incma:
▷ $(ORBITER) -v 2 \\
▷ ▷ -define F -finite_field -q 4 -end \\
▷ ▷ -define P -projective_space -n 2 -field F -v 0 -end \\
▷ ▷ -with P -do -projective_space_activity \\
▷ ▷ ▷ -cheat_sheet_for_decomposition_by_element_PG \\
```

468
Figure 16.1: Addition table of $\mathbb{F}_7$

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

produces a cyclically ordered incidence matrix of the plane $\text{PG}(2,4)$, shown in Figure 16.2. The Singer cycle is the projectivity defined by the companion matrix of an irreducible polynomial. We may pick the irreducible polynomial $X^2 + X + \omega$ over $\mathbb{F}_4$. The associated Singer cycle is the projectivity defined by the companion matrix

$$\begin{bmatrix}
0 & 1 & 0 \\
0 & 0 & 1 \\
2 & 1 & 1
\end{bmatrix}.$$
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-file</td>
<td>fname</td>
<td>Use the given file name for output files.</td>
</tr>
<tr>
<td>-xin</td>
<td>$a$</td>
<td>Assume input $x$-coordinates are in the interval $[0, a]$. Default value: 10000.</td>
</tr>
<tr>
<td>-yin</td>
<td>$a$</td>
<td>Assume input $y$-coordinates are in the interval $[0, a]$. Default value: 10000.</td>
</tr>
<tr>
<td>-xout</td>
<td>$a$</td>
<td>Assume output $x$-coordinates are in the interval $[0, a]$. Default value: 1000000.</td>
</tr>
<tr>
<td>-yout</td>
<td>$a$</td>
<td>Assume output $y$-coordinates are in the interval $[0, a]$. Default value: 1000000.</td>
</tr>
<tr>
<td>-spanning_tree</td>
<td></td>
<td>Place nodes according to a spanning tree. Default value: off.</td>
</tr>
<tr>
<td>-circle</td>
<td></td>
<td>Circle all nodes. Default value: on.</td>
</tr>
<tr>
<td>-corners</td>
<td></td>
<td>Draw corners at the outside of the drawing. Default value: off.</td>
</tr>
<tr>
<td>-rad</td>
<td>$r$</td>
<td>Use radius $r$ for drawing circles around nodes. Default value: 50.</td>
</tr>
<tr>
<td>-embedded</td>
<td></td>
<td>Create latex headers for stand-alone latex files. Default value: off.</td>
</tr>
<tr>
<td>-sideways</td>
<td></td>
<td>Create latex figure sideways. Default value: off.</td>
</tr>
<tr>
<td>-label_edges</td>
<td></td>
<td>Label the edges in Schreier trees. Default value: off.</td>
</tr>
<tr>
<td>-x_stretch</td>
<td>$s$</td>
<td>Apply $x$-axis scaling by a factor of $s$. Default value: $s = 1.0$. This option does not affect the drawing of Schreier trees.</td>
</tr>
<tr>
<td>-y_stretch</td>
<td>$s$</td>
<td>Apply $y$-axis scaling by a factor of $s$. Default value: $s = 1.0$. This option does not affect the drawing of Schreier trees.</td>
</tr>
</tbody>
</table>

Table 16.2: Drawing Options for Layered Graph Files (Part 1)
Figure 16.2: A cyclic ordering of the incidence matrix of PG(2, 4)

<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-scale</td>
<td>$s$</td>
<td>Use tikz global scale-factor of $s$. Default value: $s = 0.45$.</td>
</tr>
<tr>
<td>-line_width</td>
<td>$s$</td>
<td>Set tikz line width to $s$. Default value: $s = 1.5$.</td>
</tr>
<tr>
<td>-nodes_empty</td>
<td></td>
<td>Draw nodes empty. Do not label. Default value: off.</td>
</tr>
<tr>
<td>-select_layers</td>
<td>$S$</td>
<td>Draw layers whose index is given in the list $S$ only.</td>
</tr>
<tr>
<td>-paths_in_between</td>
<td>$l_{1} i_{1} l_{2} i_{2}$</td>
<td>Draw all paths from node $(l_{1}, i_{1})$ to node $(l_{2}, i_{2})$. Here, $(l, i)$ is the $i$-th node at layer $l$ (counting from zero). Delete all other edges between layers $l_{1}$ and $l_{2}$.</td>
</tr>
</tbody>
</table>

Table 16.3: Drawing Options for Layered Graph Files (Part 2)
Figure 16.3: The first basic orbit of PGL(4, 2) as a subgroup of PGO⁺(6, 2)

```
PGL_4_2_Wedge_4_0_graphical_output:
  ▶ $(ORBITER) -v 4 \
  ▶ ▶ -define G -linear_group -PGL 4 2 \
  ▶ ▶ ▶ -wedge_detached \
  ▶ ▶ ▶ -end \
  ▶ ▶ -with G -do \
  ▶ ▶ ▶ -group_theoretic_activity \
  ▶ ▶ ▶ ▶ -report \
  ▶ ▶ ▶ -end \
  ▶ pdflatex PGL_4_2_Wedge_4_2_detached_report.tex \
  ▶ open PGL_4_2_Wedge_4_2_detached_report.pdf
```

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 \
  ▶ ▶ -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 \
```
Figure 16.4: A Schreier tree in the action on polynomials

```plaintext
$\text{on}_\text{polynomials} 3 \ -\text{draw}_\text{tree} 6 \$
$\text{-end}$

$\text{pdflatex poly_orbits_d3_n3_q2_orbit_6_tree.tex}$
$\text{open poly_orbits_d3_n3_q2_orbit_6_tree.pdf}$
$\text{#pdflatex poly_orbits_d3_n3_q2.tex}$
$\text{#open poly_orbits_d3_n3_q2.pdf}$
```

draws the 6th Schreier tree in the classification of orbits of PGL(4, 2) on homogeneous polynomials of degree 3 in 4 variables. The drawing is shown in Figure 16.4. This particular orbit has length 420, so there are 420 nodes in the tree.
16.2 The Povray Interface

Orbiter can be used to create raytracing 3D-graphics. Orbiter serves as a front end for the raytracing software Povray [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 )</td>
<td>In round ( r ), produce running text \text ( ) with sustain value ( a ).</td>
</tr>
<tr>
<td>-label</td>
<td>( r \ s \ a \ g \ text )</td>
<td>In round ( r ), produce running text \text ( ) with start value ( s ), sustain ( s ) and gravity ( g ).</td>
</tr>
<tr>
<td>-latex</td>
<td>( r \ s \ a \ praemable \ g \ text \ l \ fname )</td>
<td>In round ( r ), produce running latex text \text ( ) with start value ( s ), sustain ( s ) and gravity ( g ). Put \text {praemable} \ in the latex source code. Use \text {fname} \ for the latex file names (no extension).</td>
</tr>
<tr>
<td>-global_picture_scale</td>
<td>( d )</td>
<td>Set scaling factor to ( d ).</td>
</tr>
<tr>
<td>-picture</td>
<td>( r \ d \ fname \ options )</td>
<td>In round ( r ), place picture \text {fname} \ scaled by ( d ) using options.</td>
</tr>
<tr>
<td>-picture</td>
<td>( r \ d \ fname \ options )</td>
<td>In round ( r ), place picture \text {fname} \ scaled by ( d ) using options.</td>
</tr>
<tr>
<td>-look_at</td>
<td>( L )</td>
<td>Override camera look-at value to ( L ). ( L ) is a three-dimensional vector.</td>
</tr>
<tr>
<td>-default_angle</td>
<td>( a )</td>
<td>Set default camera angle to ( a ).</td>
</tr>
<tr>
<td>-clipping_radius</td>
<td>( f )</td>
<td>Set default clipping radius to ( f ).</td>
</tr>
<tr>
<td>-scale_factor</td>
<td>( s )</td>
<td>Set default scale factor to ( s ).</td>
</tr>
<tr>
<td>-line_radius</td>
<td>( s )</td>
<td>Set default line radius to ( s ).</td>
</tr>
</tbody>
</table>

Table 16.5: Options for Orbiter 3D-graphics (Part 2)
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-cubic_lex</td>
<td>coeffs</td>
<td>Cubic surface given by 20 coefficients in lexicographic ordering</td>
</tr>
<tr>
<td>-cubic_orbiter</td>
<td>coeffs</td>
<td>Cubic surface given by 20 coefficients in Orbiter ordering</td>
</tr>
<tr>
<td>-cubic_Goursat</td>
<td>A B C</td>
<td>Cubic surface with tetrahedral symmetry given by 3 Goursat coefficients as $Axyz + B(x^2 + y^2 + z^2) + C = 0$</td>
</tr>
<tr>
<td>-quadric_lex_10</td>
<td>coeffs</td>
<td>Quadric surface given by 10 coefficients in lexicographic ordering</td>
</tr>
<tr>
<td>-quartic_lex_35</td>
<td>coeffs</td>
<td>Quartic surface given by 35 coefficients in lexicographic ordering</td>
</tr>
<tr>
<td>-octic_lex_165</td>
<td>coeffs</td>
<td>Octic surface given by 165 coefficients in lexicographic ordering</td>
</tr>
<tr>
<td>-point</td>
<td>coeffs</td>
<td>Point given by three coordinates</td>
</tr>
<tr>
<td>-point_list_from_csv_file</td>
<td>fname</td>
<td>List of points with coordinates given in a csv file</td>
</tr>
<tr>
<td>-line_through_two_points_recentered_from_csv_file</td>
<td>fname</td>
<td>List of lines through two points with point coordinates given in a csv file</td>
</tr>
<tr>
<td>-line_through_two_points_from_csv_file</td>
<td>fname</td>
<td>List of lines through two points with point coordinates given in a csv file</td>
</tr>
<tr>
<td>-point_as_intersection_of_two_lines</td>
<td>$i_1$ $i_2$</td>
<td>Create a point from the intersection of two lines $i_1$ and $i_2$</td>
</tr>
<tr>
<td>-edge</td>
<td>$i_1$ $i_2$</td>
<td>Create an edge (line segment) between points $i_1$ and $i_2$</td>
</tr>
<tr>
<td>-text</td>
<td>$i_1$ s</td>
<td>Create a label s located at the point $i$</td>
</tr>
<tr>
<td>-triangular_face_given_by_three_lines</td>
<td>$i_1$ $i_2$ $i_3$</td>
<td>Create a triangular face give by three lines $i_1, i_2, i_3$</td>
</tr>
<tr>
<td>-face</td>
<td>pts</td>
<td>Create a face through the vertices pts, ordered cyclically</td>
</tr>
<tr>
<td>-quadric_through_three_skew_lines</td>
<td>$i_1$ $i_2$ $i_3$</td>
<td>Create a quadric through three skew lines</td>
</tr>
<tr>
<td>-plane_defined_by_three_points</td>
<td>$i_1$ $i_2$ $i_3$</td>
<td>Create a plane through three noncollinear points</td>
</tr>
<tr>
<td>-line_through_two_points_recentered</td>
<td>pt-coords</td>
<td>Create a line through two points given by 6 coordinates, recentered</td>
</tr>
</tbody>
</table>

Table 16.6: Scene definition commands (part 1)
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-line_through_two_points</td>
<td>pt-coords</td>
<td>Create a line through two points given by 6 coordinates</td>
</tr>
<tr>
<td>-line_through_two_existing_points</td>
<td>$i_1$ $i_2$</td>
<td>Create a line through two points</td>
</tr>
<tr>
<td>-line_through_point_with_direction</td>
<td>$x$ $y$ $z$ $u_x$ $u_y$ $u_z$</td>
<td>Create a line through a point $(x, y, z)$ with a given direction $(u_x, u_y, u_z)$</td>
</tr>
<tr>
<td>-plane_by_dual_coordinates</td>
<td>$a$ $b$ $c$ $d$</td>
<td>Create the plane $ax + by + cz + d = 0$ given in dual coordinates</td>
</tr>
<tr>
<td>-dodecahedron</td>
<td></td>
<td>Create a Dodecahedron centered at the origin (20 points, 30 edges, 12 faces)</td>
</tr>
<tr>
<td>-Hilbert_Cohn_Vossen_surface</td>
<td></td>
<td>Create the Hilbert, Cohn-Vossen surface (1 cubic surface, 45 tritangent planes, 27 lines)</td>
</tr>
<tr>
<td>-obj_file</td>
<td>fname</td>
<td>Read points and faces from the given .obj file</td>
</tr>
<tr>
<td>-group_of_things</td>
<td>list</td>
<td>Create a group of things from the given list</td>
</tr>
<tr>
<td>-group_of_things_with_offset</td>
<td>list offset</td>
<td>Create a group of things from the given list, each value is increase by offset</td>
</tr>
<tr>
<td>-group_of_things_as_interval</td>
<td>$a$ $b$</td>
<td>Create a group of things indexed by the interval $a, \ldots, a+b-1$</td>
</tr>
<tr>
<td>-group_of_things_as_interval_with_exceptions</td>
<td>$a$ $b$ $ex$</td>
<td>Create a group of things indexed by the interval $a, \ldots, a+b-1$ with the exceptional elements in the list ex removed</td>
</tr>
<tr>
<td>-group_of_all_points</td>
<td></td>
<td>Create a group of things from all points currently defined</td>
</tr>
<tr>
<td>-group_of_all_faces</td>
<td></td>
<td>Create a group of things from all faces currently defined</td>
</tr>
<tr>
<td>-group_subset_at_random</td>
<td>$i$ $f$</td>
<td>Create a group of things from the existing group $i$ by picking a random subset with probability $f$</td>
</tr>
<tr>
<td>-create_regulus</td>
<td>$i$ $N$</td>
<td>Create a regulus for quadric $i$ with $N$ lines</td>
</tr>
</tbody>
</table>

Table 16.7: Scene definition commands (part 2)
<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-spheres</td>
<td>$i \ r \ \text{prop}$</td>
<td>For each element in point group $i$, create a sphere of radius $r$ with given Povray properties.</td>
</tr>
<tr>
<td>-cylinders</td>
<td>$i \ r \ \text{prop}$</td>
<td>For each element in edge group $i$, create a cylinder of radius $r$ with given Povray properties.</td>
</tr>
<tr>
<td>-prisms</td>
<td>$i \ d \ \text{prop}$</td>
<td>For each element in face group $i$, create a prism of half-thickness $d$ with given Povray properties.</td>
</tr>
<tr>
<td>-planes</td>
<td>$i \ \text{prop}$</td>
<td>For each element in plane group $i$, create a plane with given Povray properties.</td>
</tr>
<tr>
<td>-lines</td>
<td>$i \ r \ \text{prop}$</td>
<td>For each element in line group $i$, create a line of radius $r$ with given Povray properties.</td>
</tr>
<tr>
<td>-cubics</td>
<td>$i \ \text{prop}$</td>
<td>For each element in group $i$ of cubics, create a surface with given Povray properties.</td>
</tr>
<tr>
<td>-quadrics</td>
<td>$i \ \text{prop}$</td>
<td>For each element in group $i$ of quadrics, create a surface with given Povray properties.</td>
</tr>
<tr>
<td>-quartics</td>
<td>$i \ \text{prop}$</td>
<td>For each element in group $i$ of quartics, create a surface with given Povray properties.</td>
</tr>
<tr>
<td>-octics</td>
<td>$i \ \text{prop}$</td>
<td>For each element in group $i$ of octics, create a surface with given Povray properties.</td>
</tr>
<tr>
<td>-texts</td>
<td>$i \ d \ s \ \text{prop}$</td>
<td>For each element in group $i$ of labels, create a text element with half-thickness $d$ and size $s$ with given Povray properties.</td>
</tr>
</tbody>
</table>

Table 16.8: Graphical output commands
This command instructs Orbiter to create 30 povray files (extension .pov), one for each frame of a rotating scene. The scene contains a cube whose vertices are shown in chrome, whose edges are in red, and whose faces are yellow and transparent. The cube turns around a vertical axis of symmetry. Here is the first frame of the result:

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
```

480
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.

\texttt{MONKEY\_SADDLE\_CUBIC="1,0,0,0,-3,0,0,0,0,0,0,0,0,0,0,0,0,-1,0"}

\texttt{monkey:\n\ & $(\text{ORBITER}) -v 2 -\text{povray} \ \&\ \&\ \& -\text{round 0 -nb\_frames\_default 30} \ \&\ \& -\text{output\_mask monkey\_d\_03d.pov} \ \&\ \& -\text{video\_options -W 1024 -H 768} \ \&\ \& -\text{global\_picture\_scale 0.8} \ \&\ \& -\text{default\_angle 75} \ \&\ \& -\text{clipping\_radius 0.8} \ \&\ \& -\text{camera 0 "0,0,1" "1,1,0.5" "0,0,0"} \ \&\ \& -\text{rotate\_about\_z\_axis} \ \&\ \& -\text{end} \ \&\ \& -\text{scene\_objects} \ \&\ \& -\text{cubic\_lex $(\text{MONKEY\_SADDLE\_CUBIC})} \ \&\ \& -\text{plane\_by\_dual\_coordinates "0,0,1,0"} \ \&\ \& -\text{group\_of\_things "0"} \ \&\ \& -\text{group\_of\_things "0"} \ \&\ \& -\text{cubics 0 $(\text{COLOR\_GOLD})} \ \&\ \& -\text{planes 1 $(\text{COLOR\_BLUE})} \ \&\ \& -\text{scene\_objects\_end} \ \&\ \& -\text{povray\_end} \ \&\ \& -\text{rm -rf POV} \ \&\ \& \text{mkdir POV} \ \&\ \& \text{mv monkey_0*.pov POV} \ \&\ \& \text{mv makefile\_animation POV}
Here is one of the frames that are created:

The Eckardt surface is given by the equation

$$\frac{5}{2}xyz - (x^2 + y^2 + z^2) + 1 = 0.$$  

We use the following code to plot the surface and the lines on it. The Schläfli labeling of the lines is indicated.

**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 \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" \n
```

482
Figure 16.5 shows the final product.

The Endrass octic [26] is the algebraic surface given by the equation

\[ x^8 := 64 (\sqrt{2} z^2 + 2 (2 + 7 \sqrt{2}) w^2) (x^2 + y^2) - 16 z^4 + 8 (1 - 2 \sqrt{2}) z^2 w^2 - (1 + 12 \sqrt{2}) w^4 \]

The following Orbiter command creates a povray graphics of the octic, shown in Figure 16.6:
Figure 16.5: The Eckardt surface
ENDRASS_OCTIC_LEX_165="-93.2548,0,0,0,-309.019,0,0,527.529,0,395.647,\n0,0,0,0,0,0,0,0,-687.529,0,0,1582.59,0,1186.94,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-309.019,\n0,0,1582.59,0,1186.94,0,0,0,0,0,0,-2110.12,0,-3165.17,0,-1186.94,0,0,0,0,0,0,874.039,0,1560.63,0,1677.92,0,343.362,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-256,0,-468.077,0,-789.019,0,\n-525.726,0,0.941125"

eindrass8:
  ▶ $(ORBITER) -v 2 -povray \n  ▶ -round 0 -nb_frames_default 30 \n  ▶ -output_mask endrass_otic\%d\%03d.pov \n  ▶ -video_options -W 1024 -H 768 \n  ▶ -global_picture_scale 0.75 \n  ▶ -default_angle 75 \n  ▶ -clipping_radius 3.7 \n  ▶ -no_bottom_plane \n  ▶ -camera 0 "1,1,1" "6,6,3" "0,0,0" \n  ▶ -rotate_about 111 \n  ▶ -end \n  ▶ -scene_objects \n  ▶ ▶ -line_through_two_points_recentered_from_csv_file \n  ▶ ▶ ▶ coordinate_grid.csv \n  ▶ ▶ ▶ -group_of_things "0" \n  ▶ ▶ ▶ -group_of_things "1" \n  ▶ ▶ ▶ -group_of_things "2" \n  ▶ ▶ ▶ -group_of_things_as_interval 3 39 \n  ▶ ▶ ▶ -lines 0 0.15 $(COLOR_RED_SHINY) \n  ▶ ▶ ▶ -lines 1 0.15 $(COLOR_GREEN_SHINY) \n  ▶ ▶ ▶ -lines 2 0.15 $(COLOR_BLUE_SHINY) \n  ▶ ▶ ▶ -lines 3 0.05 $(COLOR_BLACK_SHINY) \n  ▶ ▶ ▶ -octic_lex_165 $(ENDRASS_OCTIC_LEX_165) \n  ▶ ▶ ▶ -plane_by_dual_coordinates "0,0,1,0" \n  ▶ ▶ ▶ -group_of_things "0" \n  ▶ ▶ ▶ -group_of_things "0" \n  ▶ ▶ ▶ -octics 4 $(SURFACE_COLOR_SEETHROUGH) \n  ▶ ▶ ▶ -planes 5 "texture{ pigment{ color Blue transmit 0.5 } \n    finish { diffuse 0.9 phong 1}}"
  ▶ ▶ -scene_objects_end
  ▶ ▶ -povray_end
  ▶ -rm -rf POV
  ▶ mkdir POV

485
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

```bash
monkey_video:
▷ - rm -r FRAMES
▷ - mkdir FRAMES
▷ - rm monkey.mp4
▷ $(ORBITER) \n▷ ▷ -prepare_frames \n▷ ▷ ▷ -i 0 30 monkey_0_%03d.png \n▷ ▷ ▷ -output_starts_at 0 \n▷ ▷ ▷ -o FRAMES/frame%04d.png \n▷ ▷ -end
▷ ffmpeg -r 5 -f image2 -i FRAMES/frame%04d.png \n▷ ▷ -f mp4 -q:v 0 -vcodec mpeg4 monkey.mp4
```

creates a video `monkey.mp4` from a set of 30 files. The individual filenames are created using the printf format string `monkey_0_%03d.png`, with an integer index that is drawn from the interval $[0, 29]$. The part that starts with a percent sign and ends with a “d” character defines the way in which the integer is formatted. The number three before the “d” indicates that three characters will be printed. The zero indicates the use of leading zeros. So, the first file would be `monkey_0_000.png` and the very last file is `monkey_0_029.png`. The description of the printf format string can be found in the documentation of the C standard library [39].
16.4 Continuous Function Plotter

Orbiter can plot functions using a built-in function tracker. The functions must be continuous apart from a finite number of poles. The function can have multiple components, each described using an expression. Each expression is specified in Reverse Polish Notation (RPN). Consider an example. A Lissajous curve is defined using coordinate functions of the form

\[ x = r \sin(at + c), \quad y = r \sin(bt), \quad a, b, c, r \in \mathbb{R}. \]

The terms

\[ r \sin(at + c), \quad r \sin(bt) \]

are the expressions of the two coordinate functions. RPN means that the operator is listed after the operands. A stack data structure is used to hold temporary values. Operators are pushed to the top of the stack using the push commands. A binary operator pops the two elements from the stack, performs the operation, and pushes the resulting value back onto the stack. For a unary operator, only one element is popped and replaced by the result. Here are some examples of expressions rewritten in RPN:

\[
\begin{align*}
\sin(x) & \mapsto \text{push } x \sin, \\
 a + b & \mapsto \text{push } a \text{ push } b \text{ add}, \\
 a \cdot b & \mapsto \text{push } a \text{ push } b \text{ mult}.
\end{align*}
\]

The coordinate functions are enclosed between \texttt{-code} and \texttt{-code_end} commands. Each coordinate function is described in RPN and terminated using a \texttt{return} keyword. By the time the \texttt{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 \texttt{-const} and \texttt{-const_end} keywords. Likewise, variables are declared between the \texttt{-var} and \texttt{-var_end} keywords. Picking \(a = 3, b = 2, c = \pi/2\) and \(r = 7\), the function is computed using

\[
\text{lissajous:}
\[
\begin{align*}
\text{\texttt{\$}} \text{(ORBITER)} & \text{ -v 2 } \text{"} \\
\text{\texttt{\$}} & \text{-smooth_curve } "\text{lissajous" 0.07 2000 15 0 18.85 \} \\
\text{\texttt{\$}} & \text{-const a 3 b 2 c 1.57 r 7 -const_end \} \\
\text{\texttt{\$}} & \text{-var t -var_end \} \\
\text{\texttt{\$}} & \text{-code \} \\
\text{\texttt{\$}} & \text{push t push a mult push c add sin push r mult return \} \\
\text{\texttt{\$}} & \text{push t push b mult sin push r mult return \} \\
\text{\texttt{\$}} & \text{-code_end \} \\
\end{align*}
\]

The sequence

\[
\text{push t push a mult push c add sin push r mult}
\]

is \(r \sin(at + c)\) expressed in RPN. The constants are defined in the line

\[
\text{-const a 3 b 2 c 1.57 r 7 -const_end}
\]
The input variable is defined using the line

```
-var t -var_end
```

The sequence

```
-smooth_curve "lissajous" 0.07 2000 15 0 18.85
```

defines the name of the output file, the fact that two consecutive points are never further than $\epsilon = 0.07$ away, the fact that points that are 15 or more away from the origin should be ignored, and the fact that the variable $t$ loops over the range $[0, 18.85]$ with a default of 2000 steps. The evaluator automatically reduces the step-size if consecutive image points are more than $\epsilon$ apart. The code to produce the plot is

```
lissajous_plot:
  $(ORBITER) -v 2 -povray 
  -round 0 -nb_frames_default 1 
  -output_mask lissajous_%d_%03d.pov 
  -video_options -W 1024 -H 768 
  -global_picture_scale 0.40 
  -default_angle 45 
  -clipping_radius 5 
  -omit_bottom_plane 
  -camera 0 "0,-1,0" "0,0,12" "0,0,0" 
  -rotate_about_z_axis 
  -end 
  -scene_objects 
  -line_through_two_points_recentered_from_csv_file 
  coordinate_grid.csv
  -group_of_things "0"
  -group_of_things "1"
  -group_of_things "2"
  -lines 0 0.09 "texture{ pigment{ color Yellow } }"
  -lines 1 0.09 "texture{ pigment{ color Yellow } }"
  -lines 2 0.09 "texture{ pigment{ color Yellow } }"
  -group_of_things_as_interval 3 39
  -lines 3 0.02 "texture{ pigment{ color Black } }"
  -point_list_from_csv_file
  function_lissajous_N2000_points.csv
  -group_of_things_as_interval 0 6524
  -spheres 4 0.1 "texture{ pigment{ color Red } } finish { diffuse 0.9 phong 1}"
  -plane_by_dual_coordinates "0,0,1,0"
  -group_of_things "0"
  -planes 5 "texture{ pigment{ color Blue*0.5 transmit 0.5 } }
```

489
The plot is shown in Figure 16.7.

We can turn it into a 3D plot by using the $t$ value for the $z$ coordinate. The function is computed using the command

```
lissajous 3d:
  $(ORBITER) -v 2 /
  -smooth_curve "lissajous_3d" 0.07 2000 50 0 18.85 /
  -const a 3 b 2 c 1.57 r 7 -const_end /
  -var t -var_end /
  -code /
  push t push a mult push c add sin push r mult return /
  push t push b mult sin push r mult return /
  push t return /
  -code_end /
```

The code to produce the 3D plot is

```
lissajous_3d_plot:
  $(ORBITER) -v 2 -povray /
  -round 0 -nb_frames_default 30 /
```

490
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 \texttt{-csv_file_select_rows} can be used to select rows from a csv file. The command \texttt{-csv_file_select_cols} can be used to select columns from a csv file. The command \texttt{-csv_file_select_rows_and_cols} selects rows and columns. Here is an example. We create the multiplication table of the finite field $\mathbb{F}_7$, ordered according to the powers of a primitive element:

$$\alpha^0, \alpha^1, \alpha^2, \alpha^3, \alpha^4, \alpha^5.$$  

After that, we pull the rows and columns corresponding to even powers $\alpha^0, \alpha^2, \alpha^4$.

```
misc_select:
> $(ORBITER) -v 3 \n> -define F -finite_field -q 7 -end \n> -with F -do -finite_field_activity -cheat_sheet_GF -end \n> $(ORBITER) -v 4 -csv_file_select_rows_and_cols \n> GF_q7_multiplication_table_reordered.csv \n> "0,2,4" "0,2,4"
```

The even powers of $\alpha$ create a multiplicative subgroup. Figure 17.1 shows the table of the multiplicative group $\mathbb{F}_7^*$ and the subgroup of squares (compare with Figure 3.2 in Section 3.2). Here is the file $GF_q7_multiplication_table_reordered.csv$

<table>
<thead>
<tr>
<th>Row, C0, C1, C2, C3, C4, C5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0, 1, 3, 2, 6, 4, 5</td>
</tr>
<tr>
<td>1, 3, 2, 6, 4, 5, 1</td>
</tr>
<tr>
<td>2, 2, 6, 4, 5, 1, 3</td>
</tr>
<tr>
<td>3, 6, 4, 5, 1, 3, 2</td>
</tr>
<tr>
<td>4, 4, 5, 1, 3, 2, 6</td>
</tr>
<tr>
<td>Command</td>
</tr>
<tr>
<td>--------------------------------</td>
</tr>
<tr>
<td><code>-csv_file_select_rows</code></td>
</tr>
<tr>
<td><code>-csv_file_select_cols</code></td>
</tr>
<tr>
<td><code>-csv_file_select_rows_and_cols</code></td>
</tr>
<tr>
<td><code>-csv_file_join</code></td>
</tr>
<tr>
<td><code>-csv_file_latex</code></td>
</tr>
<tr>
<td><code>-store_as_csv_file</code></td>
</tr>
</tbody>
</table>

Table 17.1: Miscellaneous Orbiter Commands

![Image](image1.png)  ![Image](image2.png)

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>C0</th>
<th>C2</th>
<th>C4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>1</td>
<td>2</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 \texttt{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 \texttt{MAX_NUMBER_OF_POINTS_FOR_POINT_TABLE} and the number of lines is less than \texttt{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 \texttt{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 \texttt{MAX_NB_POINTS_FOR_LINE_INTERSECTION_TABLE} which is defined in src/lib/foundations/geometry/projective_space.cpp. If the number of points or lines is too big, the table is not created. In this case, the projective geometry class may behave slow.

8. For Windows users: Cygwin by default uses 32 bit integers for both int and long int. Using Cygwin 64 to compile Orbiter recommended.

9. A limited list of primitive polynomials are hard-coded in Orbiter. For large fields, the user must provide their own primitive polynomial. The polynomials encoded in orbiter are not guaranteed to be compatible with the subfield relationship.
Chapter 18

Orbiter on Windows

18.1 Using Windows Subsystem Linux

The following quote from https://docs.microsoft.com/en-us/windows/wsl/ summarizes the function of the Windows Subsystem for Linux:

Windows Subsystem for Linux (WSL) lets developers run a GNU/Linux environment – including most command-line tools, utilities, and applications – directly on Windows, unmodified, without the overhead of a traditional virtual machine or dual-boot setup. You can:

1. Choose your favorite GNU/Linux distributions from the Microsoft Store.
2. Run common command-line tools such as grep, sed, awk, or other ELF-64 binaries.
3. Run Bash shell scripts and GNU/Linux command-line applications including:
4. Tools: vim, emacs, tmux
5. Languages: NodeJS, Javascript, Python, Ruby, C/C++, C# & F#, Rust, Go, etc.
6. Services: SSHD, MySQL, Apache, lighttpd, MongoDB, PostgreSQL.
7. Install additional software using your own GNU/Linux distribution package manager.
8. Invoke Windows applications using a Unix-like command-line shell.
9. Invoke GNU/Linux applications on Windows.

The following set of slides will illustrate the installation of Orbiter under WSL.
Resources

• Many of the steps will be taken from the following sources:
  • https://okunhardt.github.io/documents/Installing_WSL.pdf
  • https://docs.microsoft.com/en-us/windows/wsl/basic-commands
• Consult the two links for further help and suggestions.

Installing WSL

• Search “Turn Windows features on or off” in the Windows search bar
• Search for “Windows Subsystem for Linux”, the box must be checked
• Restart the computer
Update

- The Windows Subsystem for Linux kernel does not automatically update due to system settings
- Updates must be done manually
- To update, first you need to command prompt as admin
  - Press Windows + R to open the “Run” box
  - Type “cmd” into the box
  - Press Ctrl + Shift + Enter
  - When the window prompt opens, click “Yes”
  - Command prompt will now open as admin
- In command prompt
  - Type `wsl --update`
  - Type `wsl --shutdown`

WSL1, WSL2

- When using WSL, you can adjust the configurations according to the Linux distribution that you are using
- To run Ubuntu distribution, we need the WSL1 configuration
- To check the status, in the command prompt enter
  - `wsl --status`
- To change WSL configuration type
  - `wsl --set-default-version 1`
  - `wsl --shutdown`
Ubuntu - installation

- Generally, the Ubuntu distribution is installed by default when WSL is installed
  - `wsl --status`
    - Displays the default distribution
- If you find that Ubuntu was not installed, you can find it in the Microsoft store
- Launch Ubuntu after installation

Ubuntu - launching

- After launching Ubuntu, allow the installation to be initiated
- If you receive an error, this could be a result of the configuration
  - Set configuration to WSL1
    - `wsl --set-default-version 1`
  - Make sure to terminate Ubuntu and reboot
    - `wsl --terminate Ubuntu`
  - Start Ubuntu again
- Once Ubuntu starts correctly
  - Create Username & Password to complete installation
  - Note: the password will not appear when you type it
Ubuntu - update

- Ubuntu does not update automatically, to update run the command
  - `sudo apt update && sudo apt upgrade`
- You will be prompted to enter your password
- When update are ready to be installed the message will appear
  - Do you want to continue? [Y/n]
    - Y + enter

Ubuntu – g++ and make

- At this point, you have successfully installed and setup WSL, and now you can use the terminal as you would on Ubuntu
- Terminate and reboot Ubuntu
- Run the command in Ubuntu
  - `sudo apt install g++`
  - You can now compile C++ in WSL
- Run the command in Ubuntu
  - `sudo apt install make`
  - You can now use makefiles in WSL
Orbiter - installation

- The easiest way to run make is through the command prompt, not Ubuntu
- To run WSL commands in command prompt, use either
  - `wsl <command>`
  - `wsl.exe <command>`
- Open command prompt
- Change directory to Users\username
  - `cd C:\Users\"your username"`

---

Orbiter - installation

- In web, go to https://github.com/abetten/orbiter
- Click on the green icon “Code” that opens a drop-down menu
- You want to copy HTTPS URL
Orbiter - installation

- In command prompt, once you are in C:\Users\Joel type the command
  - wsl.exe git clone https://github.com/abetten/orbiter.git
  - Hit enter
- Now, orbiter will begin the cloning process

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

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
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30

The Makefile

#MY PATH=../orbiter
MY PATH=~/DEV.22/orbiter
#MY PATH=/scratch/betten/COMPILE/orbiter

# uncomment exactly one of the following two lines.
# uncomment the first if you want to run orbiter through docker.
# uncomment the second if you have an installed copy of orbiter and you want to r
un it directly
#ORBITER PATH=docker run -it --volume ${PWD}:/mnt -w /mnt abetten/orbiter
ORBITER PATH=$(MY PATH)/src/apps/orbiter/
ORBITER=$(ORBITER PATH)orbiter.out
SANDBOX=$(MY PATH)/src/apps/sandbox/sandbox.out
###############################################################################
# additional configurations for when you want to
# compile automatically generated code
###############################################################################
SRC=$(MY PATH)/src
MY CPP = g++
MY CC = gcc
CPPFLAGS = -Wall -I../../DEV.22/orbiter/src/lib -std=c++14
LIB = $(SRC)/lib/liborbiter.a -lpthread
LFLAGS = -lm -Wl,-rpath -Wl,/usr/local/gcc-8.2.0/lib64

###############################################################################
# End of configuration part
###############################################################################

507


GINAC_PATH=$(/path)/src/apps/ginac
SANDBOX_PATH=$(/path)/src/apps/sandbox

update:
  - cd $(ORBITER_PATH); make clean;
  - cd $(MY_PATH); make cleana; git pull; make

sandbox:
  - $(SANDBOX_PATH)/sandbox.out

# Makefile Variables

MAGMA_PATH=/usr/local/magma

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

DOILY="Row,C0,C1,C2\n\n0,0,12,13\n1,1,12,14\n2,8,9,12\n3,4,6,8\n4,6,10,14\n5,3,7,8\n6,7,10,13\n7,4,11,13\n8,3,11,14\n9,0,5,6"
CONWAY
GEN1="
1101110001000001010000
1111010111110100001011
0000001000000100010101
1111100110110001001110
010101000000010011101
000001000000100010101\n
CONWAY
GEN2="
0101000010111010111111
0110010100011110110000
0011010000111111010111
0001101110001011010011
1010010000100001011110
1101000000001010100011
1100101010001111010101
000000000000110011101
0000000000001100010101
0000000000000101011111
0001000110000010011010
0000000000000110111111
0000000000000101011111
01101111110101111111\n
509
# 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="0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0"

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

\textbf{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)

\textbf{GENERATORS\_HESSE\_GROUP}="\
3000300030 \\
2000201230 \\
1000100111 \\
1000220200 \\
1002312010 \\
0331003211 \\
2200011331"

\textbf{GENERATORS\_WEYL\_GROUP\_E8}="\
-1,-1,-1,-1,0,0,0,0, \\
0,0,0,1,0,0,0,0, \\
1,0,0,0,0,0,0,0, \\
0,0,1,0,0,0,0,0, \\
0,1,0,1,1,0,0,0, \\
0,0,0,0,0,1,0,0, \\
0,0,0,0,0,0,1,0, \\
0,0,0,0,0,0,0,1, \\
-1,0,-1,-1,-1,-1,-1,-1, \\
0,1,0,1,1,1,1,1, \\
1,0,0,0,0,0,0,0, \\
0,0,1,0,0,0,0,0, \\
0,0,0,1,0,0,0,0, \\
0,0,0,0,1,0,0,0, \\
0,0,0,0,0,1,0,0, \\
0,0,0,0,0,0,1,0" \\
Ree\_gen1="21,5,1,6,17,1,1, 3,13,5,21,6,6,18, 21,3,21,21,22,6,14, \\
14,18,1,5,13,6,7, 3,3,2,1,24,16,3, 17,3,22,10,16,24,26, \\
21,21,6,18,20,2,5"

\textbf{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, \\
511
\[219\quad 24,15,2,13,11,14,24\]

\[\text{HIRSCHFELD\_SURFACE\_Q4\_SET\_OF\_POINTS}=\{0,1,2,3,4,5,6,7,8,9,\}
\[\quad 10,11,12,13,14,23,26,27,30,31,34,35,38,39,42,47,48,51,52,\]
\[\quad 53,54,59,60,61,62,67,68,69,70,75,76,79,80,81,82\}\]

\[\text{HYPEROVAL\_16\_144}=\{0,1,2,3,52,67,89,106,126,\}
\[\quad 141,159,176,184,199,220,235,245,262\}\]

\[\text{HYPEROVAL\_16\_16320}=\{0,1,2,3,52,70,83,109,127,\}
\[\quad 139,156,174,186,199,217,229,256,264\}\]

\[\text{FILE\_24\_3\_TFC\_INC}=\"24\ 24\ 72\"
\[\quad 48\}\]

\[\text{\textbackslash n0}\ 1\ 2\ 24\ 27\ 28\ 48\ 53\ 54\ 73\ 79\ 80\ 97\ 105\ 106\ 122\ 131\ \]
\[\quad 132\ 146\ 157\ 158\ 171\ 175\ 183\ 195\ 203\ 208\ 220\ 225\ 233\ 244\ \]
\[\quad 258\ 259\ 269\ 272\ 278\ 293\ 300\ 308\ 318\ 325\ 333\ 342\ 352\ 358\ \]
\[\quad 367\ 379\ 381\ 392\ 398\ 400\ 417\ 428\ 429\ 442\ 443\ 450\ 466\ 471\ \]
\[\quad 479\ 492\ 497\ 502\ 517\ 519\ 521\ 542\ 548\ 551\ 571\ 574\ 575\ \]
\[\quad 48\}\]

\[\text{\textbackslash n0}\ 1\ 2\ 24\ 27\ 28\ 48\ 53\ 54\ 73\ 79\ 80\ 97\ 105\ 106\ 122\ 131\ \]
\[\quad 132\ 146\ 157\ 158\ 171\ 175\ 183\ 195\ 203\ 208\ 220\ 225\ 233\ 244\ \]
\[\quad 258\ 259\ 269\ 272\ 278\ 293\ 300\ 308\ 318\ 325\ 333\ 342\ 352\ 358\ \]
\[\quad 367\ 373\ 378\ 392\ 400\ 403\ 417\ 428\ 442\ 444\ 466\ 472\ \]
\[\quad 479\ 492\ 500\ 518\ 525\ 526\ 545\ 549\ 551\ 571\ 572\ 574\ \]
\[\quad 48\}\]

\[\text{\textbackslash n0}\ 1\ 2\ 24\ 27\ 28\ 48\ 53\ 54\ 73\ 79\ 80\ 97\ 105\ 106\ 122\ 131\ \]
\[\quad 132\ 146\ 157\ 158\ 171\ 175\ 179\ 195\ 201\ 207\ 220\ 226\ 232\ 244\ \]
\[\quad 257\ 258\ 269\ 274\ 277\ 293\ 300\ 307\ 318\ 323\ 329\ 342\ 352\ 356\ \]
\[\quad 367\ 374\ 381\ 392\ 397\ 406\ 416\ 423\ 431\ 441\ 450\ 454\ 468\ 476\ \]
\[\quad 477\ 494\ 499\ 503\ 519\ 521\ 525\ 544\ 547\ 550\ 570\ 572\ 575\ \]
\[\quad 144\}\]

\[\text{n-1\ 3}\]

\[\text{ELEMENTARY\_SYMMETRIC\_3\_1}=\"x0 + x1 + x2\"
\[\quad \text{ELEMENTARY\_SYMMETRIC\_3\_2}=\"x0*x1 + x0*x2 + x1*x2\"
\[\quad \text{ELEMENTARY\_SYMMETRIC\_3\_3}=\"x0*x1*x2\"
\[\quad \text{ELEMENTARY\_SYMMETRIC\_4\_1}=\"x0 + x1 + x2 + x3\"\]
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="0,1,2,3,4,5,6,7,8,9,10,16,17,28,35,36,59,60,61,62,63,64,65,76,80,91,95,106,107,111,119,121,122,130,138,139,141,146,150,154,155,157,162,170,175,184,186,199,203,204,206,226,231,234,239,240,252,254,278,279,282,287,299,301,302,319,320,330,338,343,345,350,351,357,364,370,371,376,378,382,385,388,392,394,399"

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="
621000 \
062100 \
006210 \
000621"

CODE_RS_10_8_11="
7,2,1,0,0,0,0,0,0,0, \
0,7,2,1,0,0,0,0,0,0, \
0,0,7,2,1,0,0,0,0,0, \
0,0,0,7,2,1,0,0,0,0, \
0,0,0,0,7,2,1,0,0,0, \
0,0,0,0,0,7,2,1,0,0, \
0,0,0,0,0,0,7,2,1,0, \
0,0,0,0,0,0,0,7,2,1 , "

# Normal form for cubic surfaces with 15 rational lines:
305  \text{F\_ALPHA\_BETA\_GAMMA\_DELTA} = "beta*(gamma + 1)*x0*x0*x2 \\
306  + (alpha*delta - beta*gamma + alpha - beta - delta - 1)*x0*x1*x2 \\
307  -1*(alpha*beta -alpha*delta + delta)*(gamma + 1)*x0*x1*x3 \\
308  + (0-alpha*delta + alpha*gamma -beta*gamma -beta + delta -gamma)*x0*x2*x2 \\
309  - (alpha*delta + beta -delta)*(gamma +1)*x0*x2*x3 \\
310  -(delta + 1)*(alpha - 1)*x1*x1*x2 \\
311  - (delta + 1)*(alpha - 1)*x1*x1*x3 \\
312  + (alpha*delta - alpha*gamma + beta*gamma + beta - delta + gamma)*x1*x2*x2 \\
313  + alpha*beta*(gamma + 1)*x1*x3*x3"

# general normal form for cubic surfaces with 27 rational lines:

321  \text{F\_abcd\_eqn} = "-(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*(b - d)*X0*X0*X2 \\
322  + (a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*(a + b - c - d)*X0*X1*X2 \\
323  + (a*a*c - a*a*d + a*c*c + a*d - b*c)*(b - d)*X0*X1*X3 \\
324  - (a*d - b*c)*(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*X0*X2*X2 \\
325  - (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 \\
326  - (a - c)*(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*X1*X1*X2 \\
327  - (a - c)*(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*X1*X1*X3 \\
328  + (a*d - b*c)*(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*X1*X2*X2 \\
329  + (1+1)*a*a*b*c*d - a*a*b*d*d - (1+1)*a*a*c*d*d \\
330  - (1+1)*a*a*b*c*d + a*b*b*c*d + (1+1)*a*b*b*c*d + a*b*c*c*d \\
331  - (1+1)*a*b*b*c*c + a*b*b*c*d + a*a*c*d + a*b*b*c + a*b*c*c \\
332  + 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:

340  \text{F\_abcd\_eqn\_with\_exponents} = "-(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*(b - d)*X0*X0*X2 \\
341  + (a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*(a + b - c - d)*X0*X1*X2 \\
342  + a^2*c - a^2*d - a*c^2 + b*c^2 + a*d - b*c)*(b - d)*X0*X1*X3 \\
343  - (a*d - b*c)*(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*X0*X2*X2 \\
344  - (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 \\
345  - (a - c)*(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*X1*X2*X2 \\
346  - (a - c)*(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*X1*X2*X3 \\
347  + (a*d - b*c)*(a*b*c - a*b*d - a*c*d + b*c*d + a*d - b*c)*X1*X2*X3 \\
348  + (1+1)*a^2*b*c*d - a^2*b*d^2 - (1+1)*a^2*c*d^2 \\
349  - (1+1)*a*b^2*c^2 + a*b^2*c*d + (1+1)*a*b*c^2*d + a*b*c*d^2 \\
350  - b^2*c^2*d - a^2*b^2*c + a^2*b^2*c*d + a^2*d^2 + a*b^2*c + a*b*c^2"
- (1+1+1+1)*a*b*c*d - a*c^2*d + a*c*d^2 + b^2*c^2)*X1*X2*X3 \
+ c*a*(a*d - b*c - a + b + c - d)*(b - d)*X1*X3^2"

KNECHT.13.1_AS_PAIRS="1,0,1,1,2,12,9"
KNECHT.13.1_AS_VECTOR="1,1,1,0,0, 0,0,0,12, 0,0,0,0,0"
KNECHT.13.2_AS_PAIRS="1,0,1,1,2,8,9,8,10,8,11"
KNECHT.13.2_AS_VECTOR="1,1,1,0,0, 0,0,0,8, 0,8,0,0,0"

# coding theory
CRC4="1,4,1,2,1,1,1,0"
CRC7="1,7,1,3,1,0"
CRC8_ATM="1,8,1,2,1,1,1,0"
CRC16_CCITT="1,16,1,12,1,5,1,0"
CRC32 ETHERNET="1,32,1,26,1,23,1,22,1,16,1,12,1,11,1,10,1,8,1,7,\n1,5,1,4,1,2,1,1,1,0"
CRC32_CASTAGNOLI="1,32,1,28,1,27,1,26,1,25,1,23,1,22,1,20,1,19,1,\n18,1,14,1,13,1,11,1,10,1,9,1,8,1,6,1,0"
CRC64_ECMA182="1,64,1,62,1,57,1,55,1,54,1,53,1,52,1,47,1,46,1,45,\n1,40,1,39,1,38,1,37,1,35,1,33,1,32,1,31,1,29,1,27,1,24,1,23,1,22,\n1,21,1,19,1,17,1,13,1,12,1,10,1,9,1,7,1,4,1,1,1,0"
CRC64_ROCKSOFT="1,64,1,63,1,61,1,59,1,58,1,56,1,55,1,52,1,49,1,48,\n1,47,1,46,1,44,1,41,1,37,1,36,1,34,1,32,1,31,1,28,1,26,1,23,1,22,1,\n19,1,16,1,13,1,12,1,10,1,9,1,6,1,4,1,3,1,0"
GOLAY.23.COLUMN_RANKS_PROJECTIVELY="0, 1, 2, 3, 4, 5, 6, 7, \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 \n0561000 \n0056100 \n0005610 \n0000561"

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"
RS_8_reduced="
010001100000000000000000
011100100000000000000000
110011001000000000000000
000010001100000000000000
000001110010000000000000
CODE_21_15_4="\n111001000000000000000 \n101000100000000000000 \n101100010000000000000 \n011000001000000000000 \n110100000010000000000 \n010100000001000000000 \n100110000000100000000 \n010010000000010000000 \n110000000001000000000 \n010010000000001000000 \n1010100000000000000001" 

# there are 5 [15,6,6] 

# ago=12 
CODE_15_6_6_A="\n111111111100000 \n111110000010000 \n111001100001000 \n110101010000100 \n101010110000010 \n10110100100001" 

# ago=12 
CODE_15_6_6_B="\n111111111100000 \n111110000010000 \n111001100001000 \n110101010000100 \n110101010000100 \n110101010000100 \n110101010000100 \n110101010000100
101010110000010 \
011011001000001"
BCH_21.6_GENERATOR_MATRIX=" 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0,
0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
"

POLY.Q256.DEG30.SPARSE="1,0,26,1,210,2,24,3,
533 138,4,148,5,160,6,58,7,108,8,199,9,95,10,56,
534 11,9,12,205,13,194,14,193,15,3,16,248,17,110,
535 18,150,19,24,20,169,21,192,22,212,23,112,24,
536 144,25,97,26,109,27,174,28,253,29,1,30"

POLY.Q256.DEG30.DENSE="1,26,210,24,138,148,
540 160,58,108,199,95,56,9,205,194,193,3,248,110,\n541 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*x3*
x5 + x0*x3*x6 + x0*x3*x7 + x0*x4*x5 + x0*x4*x6 + x0*x4*x7 + x0*x5*x6 + x0*x5*x7 +
x0*x6*x7 + x1*x2*x3 + x1*x2*x4 + x1*x2*x5 + x1*x2*x6 + x1*x2*x7 + x1*x3*x4 + x1*
x3*x5 + x1*x3*x6 + x1*x3*x7 + x1*x4*x5 + x1*x4*x6 + x1*x4*x7 + x1*x5*x6 + x1*x5*
x7 + x1*x6*x7 + x2*x3*x4 + x2*x3*x5 + x2*x3*x6 + x2*x3*x7 + x2*x4*x5 + x2*x4*x6 +
x2*x4*x7 + x2*x5*x6 + x2*x5*x7 + x2*x6*x7 + x3*x4*x5 + x3*x4*x6 + x3*x4*x7 + x3*
x5*x6 + x3*x5*x7 + x3*x6*x7 + x4*x5*x6 + x4*x5*x7 + x4*x6*x7 + x5*x6*x7"
# elt order, class size, centralizer order

#2A: 2 48960 40320 Baer involution

#2B: 2 5355 368640 one block of 10,11

#2C: 2 64260 30720 two blocks of 10,11 (problem group)

# elements of order 2:
# conjugacy class reps:
# elt order, class size, centralizer order

#2 48960 40320 Baer involution

#2 5355 368640 one block of 10,11

#2 64260 30720 two blocks of 10,11 (problem group)

CLASS_2A=-centralizer_of_element \ 
"1,0,0,0, 0,1,0,0, 0,0,1,0, 0,0,0,1, 1" \ 
-label "2A"

# Baer involution

CLASS_2B=-centralizer_of_element \ 
"1,0,0,0, 1,1,0,0, 0,0,1,0, 0,0,0,1, 0" \ 
-label "2B"

CLASS_2C=-centralizer_of_element \ 
"1,0,0,0, 1,1,0,0, 0,0,1,0, 0,0,1,1, 0" \ 
-label "2C"

# problem group

# 3 classes of elements of order 3

# 4 classes of elements of order 4

# Baer involution:

PGGL_4_4_SUBGROUP_2A=-PGGL 4 4 \ 
▷ -subgroup_by_generators "2A" 2 1 \ 
▷ "1,0,0,0, 0,1,0,0, 0,0,1,0, 0,0,0,1, 1"

⊿ pdflatex PGGL_4_4_classes_out.tex

⊿ open PGGL_4_4_classes_out.pdf

521
PGGL_4_4_SUBGROUP_2A_NORMALIZER=-PGGL 4 4 \\
> -subgroup_by_generators "centralizer_2A" "40320" 10 \ 
> "1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1, \\
> 1,0,0,0,0,1,0,0,0,0,1,0,1,1,1,1, \\
> 1,0,0,0,0,1,0,0,0,0,1,0,1,1,1,0, \\
> 1,0,0,0,0,1,0,0,0,1,1,0,1,1,1,0, \\
> 1,0,0,0,0,1,0,0,0,1,1,1,0,1,1,1,0, \\
> 1,0,0,0,0,1,0,0,1,1,0,1,1,1,0,0, \\
> 1,0,0,0,0,1,0,0,1,0,0,1,0,1,1,1,1, \\
> 1,0,0,0,0,1,0,0,1,0,0,1,0,1,1,0,0, \\
> 1,0,0,0,0,1,0,0,0,0,0,1,1,0,1,1,0, \\
> 1,0,0,0,0,1,0,0,1,0,1,0,0,1,1,0,0, \\
> 1,0,0,0,0,1,0,0,0,0,0,0,1,1,1,0,0, \\
> 1,0,0,0,0,1,0,0,0,1,0,0,1,1,1,1,0, \\
> 1,0,0,0,0,1,0,0,0,0,0,1,1,0,1,0,0, \\
> 1,0,0,0,0,1,0,0,0,1,0,0,1,0,1,0,0, \\
> 1,0,0,0,0,1,0,0,0,0,1,0,1,0,1,0,0, \\
> 0,1,0,0,0,1,0,0,0,0,0,0,1,1,1,1,1,1

# 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, \\
> 1,0,0,0,0,1,0,0,0,1,0,0,0,0,2,0, \\
> 1,0,0,0,0,1,0,0,0,0,1,0,0,1,1,0, \\
> 1,0,0,0,0,1,0,0,0,0,0,2,0,0,0,2,0, \\
> 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 \\
> -subgroup_by_generators "5B" 5 1 \ 
> "0,2,0,0,1,1,0,0,0,0,1,0,0,0,2,0,0,1,1,0"

PGGL_4_4_SUBGROUP_5B_NORMALIZER=-PGGL 4 4 \\
> -subgroup_by_generators "normalizer_5B" "81600" 6 \ 
> "1,0,0,0,0,1,0,0,0,0,1,0,0,0,2,0,0,1,1,0, \\
> 1,0,0,0,1,0,0,0,0,1,1,0,0,3,2,0, \\
> 1,0,0,0,2,2,0,0,0,0,3,0,0,0,1,1,1, \\
> 0,1,0,0,3,3,0,0,0,0,1,0,0,3,3,0, \\
> 0,0,1,0,0,0,1,2,2,0,0,2,3,0,0,1"

PGGL_4_4_SUBGROUP_2Cx2_0=-PGGL 4 4 \\
> -subgroup_by_generators "2Cx2_0" 4 2 \ 
> "1,0,0,0,0,1,1,0,0,0,0,1,0,0,0,1,1,0 \ 
> 1,0,0,0,1,0,0,1,0,1,0,0,1,0,1,0"

PGGL_4_4_SUBGROUP_2Cx2_0_NORMALIZER=-PGGL 4 4 \\
> -subgroup_by_generators "normalizer_2Cx2_0" "768" 8 \ 
> "1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1 \ 
> 1,0,0,0,1,0,0,1,0,1,0,3,1,0,1,1 \ 
> 1,0,0,0,1,0,0,3,0,1,0,2,3,0,1,1 \ 
> 1,0,0,0,1,0,0,1,1,0,0,1,0,1,0,1,0 \ 
> 1,0,0,0,2,1,1,0,1,0,1,0,1,1,2,1,1 \ 
> 1,0,0,0,0,1,0,0,1,0,0,0,0,1,0,1,0 \ 
> 1,0,0,0,3,1,0,0,1,0,1,0,1,0,1,3,1,1"

#PGL_4_5_SUBGROUP_3B=-PGL 4 5 \\
> -subgroup_by_generators "3B" 3 1 \ 
> "1,0,0,0, 0,1,0,0, 0,0,2,1, 0,0,3,2"

#PGL_4_5_SUBGROUP_3B_NORMALIZER=-PGL 4 5 \\
> -subgroup_by_generators "normalizer_3B" "5760" 8 \ 
> "1,0,0,0,0,4,0,0,0,0,1,0,0,0,0,1," \ 
> "1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,4," \ 
> "1,0,0,0,0,4,0,0,0,0,4,0,0,0,4," \ 
> "1,0,0,0,0,1,0,0,0,0,3,0,0,0,3," \ 
> "1,0,0,0,0,3,0,0,0,0,1,0,0,0,1,"

523
elementary abelian subgroups of order 4 with 3 elements of class 2C:

# nice generators, from Michael Epstein:

PGL_4_5_SUBGROUP_3B_ME=-PGL 4 5 \
  -subgroup_by_generators "3B" 3 1 \
  "1,0,0,0, 0,1,0,0, 0,0,2,2, 0,0,4,2"

PGL_4_5_SUBGROUP_3B_ME_NORMALIZER=-PGL 4 5 \
  -subgroup_by_generators "normalizer_3B" "5760" 8 \
  "1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,\n  1,0,0,0,0,4,0,0,0,0,4,0,0,0,0,4,\n  1,0,0,0,0,3,0,0,0,0,3,0,0,0,0,3,\n  1,0,0,0,0,4,0,0,0,0,4,0,0,0,0,4,"

PGL_4_5_SUBGROUP_31_ME=-PGL 4 5 \
  -subgroup_by_generators "31" 31 1 \
  "1,0,0,0, 0,3,4,3, 0,3,3,4, 0,3,2,3"

PGL_4_5_SUBGROUP_31_ME_NORMALIZER=-PGL 4 5 \
  -subgroup_by_generators "normalizer_31" "372" 4 \
  "1,0,0,0,0,4,0,0,0,0,4,0,0,0,0,4,\
  1,0,0,0,0,3,0,0,0,0,3,0,0,0,0,3,\n  1,0,0,0,0,4,0,0,0,0,4,0,0,0,0,4,\
  1,0,0,0,0,3,4,3, 0,3,3,4, 0,3,2,3,"

PGL_4_5_SUBGROUP_31=-PGL 4 5 \

# 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 \\
-subgroup_by_generators "normalizer_31" "372" 4 \\
"1,0,0,0,0,4,0,0,0,0,0,4,0,0,0,0,4, \\
1,0,0,0,0,3,0,0,0,0,3,0,0,0,0,3, \\
1,0,0,0,0,4,0,0,0,0,2,1,0,3,2,4, \\
1,0,0,0,0,0,1,0,0,0,1,0,1,1,3,"

#372:
"1,0,0,0,0,4,0,0,0,0,4,0,0,0,0,4,0,0,0,4, \\
1,0,0,0,0,3,0,0,0,0,3,0,0,0,0,3, \\
1,0,0,0,0,4,0,0,0,0,2,1,0,3,2,4, \\
1,0,0,0,0,0,1,0,0,0,1,0,1,1,3,"

#Exterior square roots:

#elt of order 3:
the exterior square root of f is X=
[1 0 0 0]
[0 3 4 3]
[0 3 3 4]
[0 3 2 3]

#elt of order 31:
the exterior square root of g is Z=
[1 0 0 0]
[0 3 4 3]
[0 3 3 4]

#Michael

HAMMING_CODE_CODEWORDS="0, 67, 37, 102, 22, 85, 51, 112, 15, 76, 42, 105, 25, 90, 60, 127"
SIMPLEX_CODE_GENERATOR="
1,0,1,0,1,0,1, \"
HAMMING_CODE_GENERATOR="\n 1,0,0,0,0,1,1, \n 0,1,0,0,1,1,1, \n 0,0,1,0,1,1,0, \n 0,0,0,1,1,1,1" 
GOLAY23_CODE_GENERATOR="\n 1,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,1,0,1,0, \n 0,1,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,1,1,0, \n 0,0,1,0,0,0,0,0,0,0,0,0,1,1,0,1,0,0,1,1, \n 0,0,0,1,0,0,0,0,0,0,0,0,1,1,1,0,1,1,1,0, \n 0,0,0,0,1,0,0,0,0,0,0,0,1,0,1,0,1,1,1,0, \n 0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,1,0,1,1,0, \n 0,0,0,0,0,0,1,0,0,0,0,0,1,1,0,1,0,0,1,0, \n 0,0,0,0,0,0,0,1,0,0,0,0,1,0,1,1,0,1,1,0, \n 0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,0,0,1,1,1, \n 0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,0,0,1,1, \n 0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,1" 
HAMMING_CODE_ROWS_IN_BINARY_RANKS="67,37,22,15" 
SIMPLEX_CODE_GEMMA_CYCLIC="\n 1,0,0,1,1,1,0, \n 0,1,0,1,1,1, \n 0,0,1,1,1,1,1, \n 0,0,0,1,1,1,1" 
CODE_GV_N15_K6="\n 111111111100000\n 11110000010000\n 11100110000100\n 11010101000010\n 10101011000001\n 10110100100001" 
CODE_GV_N15_K6_CHECK="\n 100000000111111\n 010000000111100"
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"
LARGE_SET_AG_2_3_NEIGHBOR_SET="129,130,133,134,136,139,141,142,153,154,156,160,16
5,166,178,179,183,184,185,190,192,194,197,203,204,206,218,221,222,225,227,231,248
TEST_SYSTEM="\n 0,1,0,1,0,0, \n 0,0,1,0,1,0, \n 1,0,1,0,0,0, \n 0,1,0,1,0,1, \n 1,0,0,0,0,1, \n 1,0,1,0,0,0, \n 0,1,0,0,1,1,"

527
TEST_RHS="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 \ 
\> -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 \ 
\> -depth 6 \ 
\> -mask.label "no_mask"

DELANDTSHEER_DOYEN_PROBLEM_27.53= -d1 1 -q1 27 -d2 1 -q2 53 -K 11 -DDx 2 -DDy 1 -search_control -W -end

DELANDTSHEER_DOYEN_PROBLEM_27.53_GROUP1=-subgroup \ \
\> "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 \ 
\> -masktest 4 x+y ge 3 \ 
\> -depth 4 \ 
\> -mask.label "mask1"

DELANDTSHEER_DOYEN_PROBLEM_3.7= -d1 1 -q1 3 -d2 1 -q2 7 -K 5 -DDx 3 -DDy 1 -search_control -W -end
DELANDTSHEER_DOYEN_PROBLEM_3_7_GROUP1=-subgroup \\ "1,1,1,0, 1,0,1,1 " "21" -group_label "group_cyclic"

DELANDTSHEER_DOYEN_PROBLEM_3_7_MASK1= -mask_label "mask1" -depth 5

PENTTILA_WILLIAMS_PRINCE_REG_PACKING_0="444,43313,154402,46682,108254,\
75363,27729,32139,5244,60442,142811,111115,94209,120678,89533,13798,\
103994,129953,82168,136838,19253,23017,145985,134996,54705,36267,\
55066,117542,96699,69154,72460"

PENTTILA_WILLIAMS_PRINCE_REG_PACKING_1="616,42728,152655,48576,105431,\
79607,28634,32817,9799,62356,141176,110085,92557,122136,86312,13975,\
101942,126869,81478,139352,18028,24325,147284,130370,52074,36843,\
55602,118454,95973,69642,74036"

PENTTILA_WILLIAMS_PRINCE_REG_PACKING_0_DUAL="3938, 66740, 56555, \\ 93538, 107785, 64917, 47567, 54483, 141012, 138602, 18308, 6880, \\ 131351, 88788, 125484, 102075, 21234, 99392, 119149, 80640, 124839, \\ 148843, 71862, 11468, 35950, 27050, 75338, 113337, 40002, 154102, \\ 30567"

PG_3_5_PACKING_0_WITH_AG03="0,5201,60427,86602,11453,121452,46663,\
19716,32921,108680,23456,91963,68386,26921,74601,57067,36188,42312,\
78780,53117,118488,114700,83960,99669,104791,126662,130960,145179,\
137230,150626,140216"

PG_3_5_PACKING_0_WITH_AG03_FIXP444="444,5001,12957,18194,23485,26817,\
34667,38299,41249,47472,50450,56601,62638,68986,71833,75369,80805,\
87025,92577,95676,104509,109718,114948,116333,124391,127498,133240,\
137711,144777,148059,150175"

# consider the binary code with generator matrix:
# 1 0 1
# 0 1 1
CODE_N3_K2_Q2_GENMA="1,0,1, 0,1,1"

CODE_N6_K3_Q2_GENMA="111100"
TRIANGLE_GRAPH="0,1,1
1,0,1
1,1,0"

# q=17:
# 3 is p.e. mod 17.
# so we pick f=3.
# then, 2f^2=18=1
# 4f = 12
# X^4 -Y^4 -Z^4 +2f^2Y^2Z^2 +4fX^2YZ

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,"$(EDGE_CURVE_Q17_EQUATION)","$(EDGE_CURVE_Q17_AS_POINTS)","",-1\n\nEND"

DESARGUES_PATH_LEX_LEAST="10 10 3\n0\n0\n15\n15\n26\n15\n26\n46\n5 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, 530
# Povray:

# povray colors:

POLISHED_CHROME_WHITE=
    "texture{ Polished_Chrome pigment{quick_color White} }"

YELLOW_TRANSPARENT=
    "texture{ pigment{ color Yellow transmit 0.7 } \
    finish {diffuse 0.9 phong 0.6} }"

COLOR_RED=
    "texture{ pigment{ color Red } \
    finish {diffuse 0.9 phong 0.6} }"

COLOR_RED_SHINY=
    "texture{ pigment{ color Red } \
    finish { diffuse 0.9 phong 1}"
COLOR_GREEN_SHINY=
 "texture{ pigment{ color Green } \ 
 finish { diffuse 0.9 phong 1}}"

COLOR_BLUE_SHINY=
 "texture{ pigment{ color Blue } \ 
 finish { diffuse 0.9 phong 1}}"

COLOR_YELLOW_SHINY=
 "texture{ pigment{ color Yellow } \ 
 finish { diffuse 0.9 phong 1}}"

COLOR_BLACK_SHINY=
 "texture{ pigment{ color Black } \ 
 finish { diffuse 0.9 phong 1}}"

COLOR_RED_SEE_THROUGH=
 "texture{ pigment{ color Red transmit 0.5 } \ 
 finish { diffuse 0.9 phong 1}}"

COLOR_GREEN_SEE_THROUGH=
 "texture{ pigment{ color Green transmit 0.5 } \ 
 finish { diffuse 0.9 phong 1}}"

COLOR_BLUE_SEE_THROUGH=
 "texture{ pigment{ color Blue transmit 0.5 } \ 
 finish { diffuse 0.9 phong 1}}"

COLOR_YELLOW_SEE_THROUGH=
 "texture{ pigment{ color Yellow transmit 0.5 } \ 
 finish { diffuse 0.9 phong 1}}"

COLOR_YELLOW_THICK=
 "texture{ pigment{ color Yellow } \ 
 finish { diffuse 0.9 phong 1}}"

COLOR_BLACK_NO_SHADOW=
 "texture{ pigment{Black} } no_shadow"

SURFACE_COLOR=
 "texture{ pigment{ White*0.5 } \ 
 finish {ambient 0.4 diffuse 0.5 roughness 0.001 \ 
 reflection 0.1 specular .8} }"

SURFACE_COLOR_SEETHROUGH=
 "texture{ pigment{ White*0.5 transmit 0.5 } \ 

finish {ambient 0.4 diffuse 0.5 roughness 0.001 \
reflection 0.1 specular .8} "}
COLOR_GOLD=\ 
texture{ pigment{ Gold } finish \\n{ambient 0.4 diffuse 0.5 roughness 0.001 \\nreflection 0.1 specular .8} }
COLOR_TURQUOISE=\ 
texture{ pigment{Cyan*1.3} \\
finish {ambient 0.4 diffuse 0.6 roughness 0.001 \\
reflection 0 specular .8} }
MONKEY_SADDLE_CUBIC="1,0,0,0,-3,0,0,0,0,0,0,0,0,0,0,0,-1,0"
ECKARDT_CUBIC_DEFORM1.LEX="0, 10, 0, -8, 10, 25, 2, 0, -20, -8, -20, -10, -24, 10 \\
, -2, 12, 0, -8, 8, 16"
ECKARDT_CUBIC_DEFORM2.LEX="0, -5, 0, -5, -5, 10, -1, 0, 10, 4, 10, 5, 3, -5, 1, -6, 0, -5, -4, 1"
KUMMER_QUARTIC.LEX_35="-2,0,0,0,2,0,0,2,0,2,0,0,\\n0,0,0,0,0,0,0,0,-2,0,0,2,0,2,0,0,0,0,-2,0,2,0,-2"
BEAUVILLE_QUINTIC.LEX_56="-44, 228, 400, 315, -396, -852, \\
-512, -553, -1050, -354, 284, 504, -62, -707, -1390, -1010, \\
281, -167, -1644, -1024, -72, -196, 192, 373, 322, 78, 150, \\
966, 1540, 348, -475, -492, 1063, 1550, 390, 0, 96, 3, -337, \\
263, -250, -919, 557, 1800, 741"
ENDRASS_OCTIC.LEX_165="-93.2548, 0, 0, 0, -309.019, 0, 0, 527.529, 0, 395.647, \\
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-309.019, \\
0,1582.59,0,1186.94,0,0,0,0,-1055.06,0, \\
-1582.59,0,-593.47,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-309.019, \\
0,0,1582.59,0,1186.94,0,0,0,0,-2110.12,0,-3165.17,0,-1186.94,0,0,0,0,0, \\
281, -167, -1644, -1024, -72, -196, 192, 373, 322, 78, 150, \\
966, 1540, 348, -475, -492, 1063, 1550, 390, 0, 96, 3, -337, \\
263, -250, -919, 557, 1800, 741"
ENDRASS_OCTIC.LEX_165="-93.2548, 0, 0, 0, -309.019, 0, 0, 527.529, 0, 395.647, \\
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-309.019, \\
0,1582.59,0,1186.94,0,0,0,0,-1055.06,0, \\
-1582.59,0,-593.47,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-309.019, \\
0,0,1582.59,0,1186.94,0,0,0,0,-2110.12,0,-3165.17,0,-1186.94,0,0,0,0,0, \\
281, -167, -1644, -1024, -72, -196, 192, 373, 322, 78, 150, \\
966, 1540, 348, -475, -492, 1063, 1550, 390, 0, 96, 3, -337, \\
263, -250, -919, 557, 1800, 741"
# Chapter 2 - Getting Started

### Section 2.2: Orbiter Session

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

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

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

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

SECTION_VECTOR_BUILDER:

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

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

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

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

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

matrix_example1:
> $\$(ORBITER) -v 2 \ 

536
matrix_example_co_1:

SECTION_FORMULA_BUILDER:

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

formula_example:

formula_evaluate:

formulas

Chapter 3 - Basic Algebra

# should evaluate to 1, since (-2+3*3)*4+2+3*4=2+4=6=1 mod 5

# but: problem with the leading minus sign

# Chapter 3 - Basic Algebra

# should evaluate to 1, since (-2+3*3)*4+2+3*4=2+4=6=1 mod 5

# but: problem with the leading minus sign

# Chapter 3 - Basic Algebra

# should evaluate to 1, since (-2+3*3)*4+2+3*4=2+4=6=1 mod 5

# but: problem with the leading minus sign
# 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
```

```
$ (ORBITER) -v 2 -square_root_mod 5 11
```

```
$ (ORBITER) -v 2 -square_root_mod 5 19
```

```
$ (ORBITER) -v 2 -square_root_mod 20 31
```

order_of_2_mod_n:

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

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

```
pdflatex order_of_q_mod_n_q2_3_151.tex
```

```
open order_of_q_mod_n_q2_3_151.pdf
```

```
Eulerfunction_150:
```

```
$ (ORBITER) -v 1 -eulerfunction.interval 1 150
```

```
$ (ORBITER) -v 1 -csv_file_latex 1 \ table_eulerfunction_1_150.csv
```

```
pdflatex table_eulerfunction_1_150.tex
```

```
open table_eulerfunction_1_150.pdf
```

```
Eulerfunction_900:
```

```
Eulerfunction_10000:
- $(\text{ORBITER}) -v 1 -eulerfunction_interval 10000 10150
- $(\text{ORBITER}) -v 1 -csv_file_latex 1 \table_eulerfunction_10000_10150.csv
- pdflatex table_eulerfunction_10000_10150.tex
- open table_eulerfunction_10000_10150.pdf

PR_1000:
- $(\text{ORBITER}) -v 1 -smallest_primitive_root_interval 2 1000
- $(\text{ORBITER}) -v 1 -csv_file_latex 1 \primitive_element_table_2_1000.csv
- pdflatex primitive_element_table_2_1000.tex
- open primitive_element_table_2_1000.pdf

PE_number_1000:
- $(\text{ORBITER}) -v 1 -number_of_primitive_roots_interval 2 1000
- $(\text{ORBITER}) -v 1 -csv_file_latex 1 table_number_of_pe_2_1000.csv
- pdflatex table_number_of_pe_2_1000.tex
- open table_number_of_pe_2_1000.pdf

number_of_primitive_roots_10000:
- $(\text{ORBITER}) -v 1 -number_of_primitive_roots_interval 10000 10001

power_function_2_mod_11:
- $(\text{ORBITER}) -v 5 -power_function_mod_n 2 11
- $(\text{ORBITER}) -v 1 -csv_file_latex 1 power_function_k2_n11.csv
- pdflatex power_function_k2_n11.tex
- open power_function_k2_n11.pdf

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_draw.tex
- open mod_13_draw.pdf

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

Chinese_remainders_B:
\begin{verbatim}
$\text{(ORBITER)} -v 2 \\
\end{verbatim}
-define R -vector -dense "38,2" -end \\
-define M -vector -dense "74,27" -end \\
\begin{verbatim}
-Chinese_remainders R M
\end{verbatim}

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

# The solution is 5684294357108828365 modulo 12703626939758759219 (computed in longinteger)

# checking with Maple:
5684294357108828365 mod 2147483647;  # 2
5684294357108828365 mod 5915587277;  # 3

Chinese_remainders_C3:
\begin{verbatim}
$\text{(ORBITER)} -v 2 \\
\end{verbatim}
-define R -vector -dense "2,3,4" -end \\
-define M -vector -dense "2147483647,5915587277,3267000013" -end \\
\begin{verbatim}
-Chinese_remainders R M
\end{verbatim}

# The solution is 31431541759324477327451572539 modulo 41502749373901658536869847 (computed in longinteger)

# checking with Maple:
31431541759324477327451572539 mod 2147483647;  # 2
31431541759324477327451572539 mod 5915587277;  # 3
31431541759324477327451572539 mod 3267000013;
SECTION_PRIME_FIELDS:

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

F_3:
$\$(ORBITER) \ -v \ 3 \ \$
$\$-define \ F \ \ -finite_field \ -q \ 3 \ -end \$
$\$-with \ F \ \ -do \ \ -finite_field_activity \ \ -cheat_sheet_GF \ \ -end\$
$\$#pdflatex \ GF_3.tex\$
$\$#open \ GF_3.pdf\$

F_5:
$\$(ORBITER) \ -v \ 3 \ \$
$\$-define \ F \ \ -finite_field \ -q \ 5 \ -end \$
$\$-with \ F \ \ -do \ \ -finite_field_activity \ \ -cheat_sheet_GF \ \ -end\$
$\$pdflatex \ GF_5.tex\$
$\$open \ GF_5.pdf\$

F_5_add_table:
$\$(ORBITER) \ -v \ 3 \ \$
$\$-define \ F \ \ -finite_field \ -q \ 5 \ -end \$
$\$-with \ F \ \ -do \ \ -finite_field_activity \ \ -cheat_sheet_GF \ \ -end\$
$\$-draw_matrix \ \ -input_csv_file \ \ GF_q5_addition_table.csv \$
$\$-box_width \ 40 \ \ -bit_depth \ 24 \ \ -partition \ \ 3 \ \ 5 \ \ 5 \ \ -end \$

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

542
\begin{verbatim}
...
...(code)
...
\end{verbatim}

# Section 3.3: Extension Fields

SECTION EXTENSION_FIELDS:

# Section 3.3: Extension Fields
1632
1633 F.4:
1634 $\text{\$\textit{ORBITER} \textmd{-v 3}}$
1635 $\text{\$\textit{define F -finite_field -q 4 -end}}$
1636 $\text{\$\textit{-with F -do -finite_field_activity -cheat_sheet_GF -end}}$
1637 $\text{\$\textit{pdflatex GF_4.tex}}$
1638 $\text{\$\textit{open GF_4.pdf}}$
1639
1640 F.4\_tables:
1641 $\text{\$\textit{ORBITER} \textmd{-v 3}}$
1642 $\text{\$\textit{define F -finite_field -q 4 -end}}$
1643 $\text{\$\textit{-with F -do -finite_field_activity -cheat_sheet_GF -end}}$
1644 $\text{\$\textit{pdflatex GF_4.tex}}$
1645 $\text{\$\textit{open GF_4.pdf}}$
1646
1647 F.16:
1648 $\text{\$\textit{ORBITER} \textmd{-v 3}}$
1649 $\text{\$\textit{define F -finite_field -q 16 -end}}$
1650 $\text{\$\textit{-with F -do -finite_field_activity -cheat_sheet_GF -end}}$
1651 $\text{\$\textit{pdflatex GF_16.tex}}$
1652
1653 F.16\_tables:
1654 $\text{\$\textit{ORBITER} \textmd{-v 3}}$
1655 $\text{\$\textit{define F -finite_field -q 16 -end}}$
1656 $\text{\$\textit{-with F -do -finite_field_activity -cheat_sheet_GF -end}}$
1657 $\text{\$\textit{pdflatex GF_16.tex}}$
1658
1659
1660
1661
1662 F.16\_tables:
1663 $\text{\$\textit{ORBITER} \textmd{-v 3}}$
1664 $\text{\$\textit{define F -finite_field -q 16 -end}}$
1665 $\text{\$\textit{-with F -do -finite_field_activity -cheat_sheet_GF -end}}$
1666 $\text{\$\textit{pdflatex GF_16.tex}}$
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
2000
544
# Section 3.4: Linear Algebra over Finite Fields

SECTION LINEAR_ALGEBRA:

RREF:

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

RREF_V7:

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

nullspace:

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

```bash
$ (ORBITER) -v 6 \n$ (ORBITER) -define F -finite_field -q 5 -end \n$ (ORBITER) -define F -finite_field -q 2 -end \n$ (ORBITER) -all_rational_normal_forms F 3
```

defines a finite field $F$ and calculates all rational normal forms.

```bash
pdflatex Class_reps.GL.3.2.tex
open Class_reps.GL.3.2.pdf
```

Calculates classes of $GL_3(F)$.

```bash
RREF.demo_4.4_q5:
$ (ORBITER) -v 2 \n$ (ORBITER) -define F -finite_field -q 5 -end \n$ (ORBITER) -define F -finite_field_activity -RREF.demo 4 4 -end
```

Calculates RREF for a 4x4 matrix over $F_5$.

```bash
gs -sDEVICE=png16 -dFIXEDMEDIA \n-dDEVICEWIDTHPOINTS=500 -dDEVICEHEIGHTPOINTS=450 \nr240 -oRREF.example.q5.4.4_page%02d.png
```

Generates a PNG image of the RREF example.

```bash
# 252 classes
```

Calculates classes of $GL_4(F)$.

```bash
RREF.demo_4.6_q7:
```

Calculates RREF for a 6x6 matrix over $F_7$. 

```bash
#-dDEVICEWIDTHPOINTS=100 -dDEVICEHEIGHTPOINTS=100
```

Adjusts PNG image size.
SECTION_ADVANCED_TOPICS_INFINITE_FIELDS:

normal_basis_2.3:
> $(ORBITER) -v 2 \n> ^ define F -finite_field -q 2 -end \n> ^ with F -do -finite_field_activity \n> ^ normal_basis 3 -end

normal_basis_2.6:
> $(ORBITER) -v 2 \n> ^ define F -finite_field -q 2 -end \n> ^ with F -do -finite_field_activity \n> ^ normal_basis 6 -end

F8_over_F2_field_reduction:
> $(ORBITER) -v 2 \n> ^ define F -finite_field -q 8 -end \n> ^ loop L 0 8 1 \n> ^ with F -do \n> ^ finite_field_activity \n> ^ field_reduction "F8_red.%L" 2 1 1 "%L" \n> ^ end \n> ^ end_loop
> $(ORBITER) -v 2 -loop L 0 8 1 
> ^ draw_matrix -input_csv_file F8_red.%L.csv 
> ^ box_width 40 -bit_depth 24 -partition 4 3 3 -end 
> ^ end_loop
> #pdflatex field_reduction_Q8_q257.tex

F64_over_F8_field_reduction:
> $(ORBITER) -v 2 \n> ^ define F -finite_field -q 64 -end \n> ^ define elts -vector -field F -loop 0 64 1 -end \n> ^ with F -do \n> ^ finite_field_activity -field_reduction "F64_over_F8" 8 8 8 \n> ^ elts -end
> $(ORBITER) -v 2 -draw_matrix \n
548
F_64_over_F8_field_reduction:

F_64_over_F4_field_reduction:

F_8_Nth_roots_21:
1914 ▷ ▷ ▷ -nth_roots 21 \n1915 ▷ ▷ -end
1916 ▷ pdflatex Nth_roots_q8_n21.tex
1917 ▷ open Nth_roots_q8_n21.pdf
1918
1919
1920
1921
1922 F_8.vandermonde:
1923 ▷ $(ORBITER) -v 3 \n1924 ▷ ▷ -define F -finite_field -q 8 -end \n1925 ▷ ▷ -with F -do -finite_field_activity \n1926 ▷ ▷ ▷ -Vandermonde_matrix \n1927 ▷ ▷ -end
1928
1929
1930
1931 F_1024.vandermonde:
1932 ▷ $(ORBITER) -v 3 \n1933 ▷ ▷ -define F -finite_field -q 1024 -end \n1934 ▷ ▷ -with F -do -finite_field_activity \n1935 ▷ ▷ ▷ -Vandermonde_matrix \n1936 ▷ ▷ -end
1937 ▷ rm Vandermonde_1024.csv
1938 ▷ rm Vandermonde_inv_1024.csv
1939
1940 #User time: 0:46
1941
1942
1943
1944
1945 NTT_k4_q17.cpp:
1946 ▷ $(ORBITER) -v 3 \n1947 ▷ ▷ -define F -finite_field -q 17 -end \n1948 ▷ ▷ -with F -do -coding_theoretic_activity \n1949 ▷ ▷ ▷ -NTT 4 17 \n1950 ▷ ▷ -end
1951
1952 F_17_NTTCompile: NTT_k4_q17.cpp
1953 ▷ $(MY_CPP) NTT_k4_q17.cpp $(CPPFLAGS) \n1954 ▷ ▷ $(LIB) $(LFLAGS) -o NTT_k4_q17.out
1955 ▷ ./NTT_k4_q17.out
1956
1957 ▷
1958
1959 # Section 3.6: Basic Ring Theory
1968 Polynomial ring:
1969 $\text{>>(ORBITER) -v 3 \ }
1970 $\text{>>(ORBITER) -define F -finite_field -q 4 -end \ }
1971 $\text{>>(ORBITER) -define R -polynomial_ring -field F \ }
1972 $\text{>>(ORBITER) -number_of_variables 4 \ }
1973 $\text{>>(ORBITER) -homogeneous_of_degree 3 \ }
1974 $\text{>>(ORBITER) -variables "x0,x1,x2,x3" "x.0,x.1,x.2,x.3" \ }
1975 $\text{>>(ORBITER) -end}
1976
1977
1978
1979
1980
1981 # Chapter 4 - Geometry
1982
1983
1984
1985 # Section 4.1: Finite Projective Spaces
1986
1987
1988 SECTION_FINITE_PROJECTIVE_SPACES:
1989
1990
1991 PG_3_2_easy:
1992 $\text{>>(ORBITER) -v 2 \ }
1993 $\text{>>(ORBITER) -define F -finite_field -q 2 -end \ }
1994 $\text{>>(ORBITER) -define P -projective_space -n 3 -field F -end}
1995
1996
1997
1998
1999 PG_1_16:
2000 $\text{>>(ORBITER) -v 2 \ }
2001 $\text{>>(ORBITER) -define F -finite_field -q 16 -end \ }
2002 $\text{>>(ORBITER) -define P -projective_space -n 1 -field F -v 0 -end \ }
2003 $\text{>>(ORBITER) -with P -do -projective_space_activity \ }
2004 $\text{>>(ORBITER) -cheat_sheet \ }
2005 $\text{>>(ORBITER) -end}
2006 pdflatex PG_1_16.tex
2007 open PG_1_16.pdf
551
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 ▶ $(ORBITER) -v 2 \\
2061 ▶ ▶ -define F -finite_field -q 4 -end \\
2062 ▶ ▶ -define P -projective_space -n 3 -field F -v 0 -end \\
2063 ▶ ▶ -with P -do -projective_space_activity \\
2064 ▶ ▶ ▶ -cheat_sheet \\
2065 ▶ ▶ -end
2066 ▶ pdflatex PG_3.4.tex
2067 ▶ open PG_3.4.pdf
2068
2069 PG_3.5:
2070 ▶ $(ORBITER) -v 2 \\
2071 ▶ ▶ -define F -finite_field -q 5 -end \\
2072 ▶ ▶ -define P -projective_space -n 3 -field F -v 0 -end \\
2073 ▶ ▶ -with P -do -projective_space_activity \\
2074 ▶ ▶ ▶ -cheat_sheet \\
2075 ▶ ▶ -end
2076 ▶ pdflatex PG_3.5.tex
2077 ▶ open PG_3.5.pdf
2078
2079
2080 PG_3.7:
2081 ▶ $(ORBITER) -v 2 \\
2082 ▶ ▶ -define F -finite_field -q 7 -end \\
2083 ▶ ▶ -define P -projective_space -n 3 -field F -v 0 -end \\
2084 ▶ ▶ -with P -do -projective_space_activity \\
2085 ▶ ▶ ▶ -cheat_sheet \\
2086 ▶ ▶ -end
2087 ▶ pdflatex PG_3.7.tex
2088 ▶ open PG_3.7.pdf
2089
2090
2091
2092
2093 PG_3.8:
2094 ▶ $(ORBITER) -v 2 \\
2095 ▶ ▶ -define F -finite_field -q 8 -end \\
2096 ▶ ▶ -define P -projective_space -n 3 -field F -v 0 -end \\
2097 ▶ ▶ -with P -do -projective_space_activity \\
2098 ▶ ▶ ▶ -cheat_sheet \\
2099 ▶ ▶ -end
2100 ▶ pdflatex PG_3.8.tex
2101 ▶ open PG_3.8.pdf

553
PG_3.16:
\[
\{(ORBITER) -v 2 \\
\quad \quad -define F -finite_field -q 16 -end \\
\quad \quad -define P -projective_space -n 3 -field F -v 0 -end \\
\quad \quad -with P -do -projective_space_activity \\
\quad \quad \quad -cheat_sheet \\
\quad \quad -end \\
\]
pdflatex PG_3.16.tex
open PG_3.16.pdf

PG_3.25:
\[
\{(ORBITER) -v 2 \\
\quad \quad -define F -finite_field -q 25 -end \\
\quad \quad -define P -projective_space -n 3 -field F -v 0 -end \\
\quad \quad -with P -do -projective_space_activity \\
\quad \quad \quad -cheat_sheet \\
\quad \quad -end \\
\]
pdflatex PG_3.25.tex
open PG_3.25.pdf

PG_4.3:
\[
\{(ORBITER) -v 2 \\
\quad \quad -define F -finite_field -q 3 -end \\
\quad \quad -define P -projective_space -n 4 -field F -v 0 -end \\
\quad \quad -with P -do -projective_space_activity \\
\quad \quad \quad -cheat_sheet \\
\quad \quad -end \\
\]
pdflatex PG_4.3.tex
open PG_4.3.pdf

PG_8.2:
\[
\{(ORBITER) -v 2 \\
\quad \quad -define F -finite_field -q 2 -end \\
\quad \quad -define P -projective_space -n 8 -field F -v 0 -end \\
\quad \quad -with P -do -projective_space_activity \\
\quad \quad \quad -cheat_sheet \\
\quad \quad -end \\
\]
pdflatex PG_8.2.tex
Section 4.2: Indexing Points

SECTION_INDEXING_POINTS:

PG_2.4_rank_point:

PG_2.4_unrank_point:

elliptic_curve_b1_c3_q11.txt:

PG_2.2.incidence_matrix:
$(ORBITER) -v 2 \ 
-define all_one -vector -repeat 1 7 -end \ 
draw_matrix \ 
-input_csv_file PG_n2_q2.incidence_matrix.csv \ 
-box_width 20 -bit_depth 8 \ 
-partition 3 \ 
-all_one all_one \ 
-end \ 
open PG_n2_q2.incidence_matrix_draw.bmp

PG_2_4.incidence_matrix:

$(ORBITER) -v 2 \ 
-define F -finite_field -q 4 -end \ 
-define P -projective_space -n 2 -field F -v 0 -end \ 
-with P -do -projective_space_activity \ 
-export_point_line_incidence_matrix \ 
-end \ 
open PG_n2_q4.incidence_matrix_draw.bmp

# writes PG_n2_q4_incidence_matrix.csv

PG_2_8.incidence_matrix:

$(ORBITER) -v 2 \ 
-define F -finite_field -q 8 -end \ 
-define P -projective_space -n 2 -field F -v 0 -end \ 
-with P -do -projective_space_activity \ 
-export_point_line_incidence_matrix \ 
-end \ 
open PG_n2_q8.incidence_matrix_draw.bmp

# writes PG_n2_q8_incidence_matrix.csv
Section 4.3: Finite Desarguesian Projective Planes

SECTION FINITE DESARGUESIAN PROJECTIVE PLANES:

PG_2.16:

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

pdflatex PG_2.16.tex
open PG_2.16.pdf

PG_2.4 with decomposition:

$(ORBITER) -v 2 \n
-define F -finite_field -q 4 -end \n
-define P -projective_space -n 2 -field F -v 0 -end \n
-with P -do -projective_space_activity \n
-cheat_sheet_for_decomposition_by_element_PG \n
1 "0,1,0, 0,0,1, 2,1,1, 0" \n
"PG_2.4_singer" \n
-end

pdflatex PG_2.4_singer.tex
open PG_2.4_singer.pdf

#PG_2.4_singer_incma_cyclic.csv
#PG_2.4_singer_incma_subgroup_index_3.csv
#PG_2.4_singer_incma_subgroup_index_7.csv

557
PG_2.4_incma_cyclic:
$\text{ORBITER} -v 2 \$
- list_arguments \
- define R -vector -repeat 1 21 -end \
- define C -vector -repeat 1 21 -end \
- draw_matrix \
- input_csv_file PG_2.4_singer_incma_cyclic.csv \
- box_width 40 -bit_depth 24 \
- partition 3 R C \
- end

open PG_2.4_singer_incma_cyclic_draw.bmp

PG_2.4_incma_singer_sub_3:
$\text{ORBITER} -v 2 \$
- list_arguments \
- define R -vector -repeat 3 7 -end \
- define C -vector -repeat 3 7 -end \
- draw_matrix \
- input_csv_file PG_2.4_singer_incma_subgroup_index_3.csv \
- box_width 40 -bit_depth 24 \
- partition 3 R C \
- end

open PG_2.4_singer_incma_subgroup_index_3_draw.bmp

PG_2.4_incma_singer_sub_7:
$\text{ORBITER} -v 2 \$
- draw_matrix \
- input_csv_file PG_2.4_singer_incma_subgroup_index_7.csv \
- box_width 20 -bit_depth 24 \
- partition 3 3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3 -end

open PG_2.4_singer_incma_subgroup_index_7_draw.bmp

# Section 4.4: The Grassmannian
\textbf{SECTION GRASSMANNIAN:}

\textbf{GR\_3\_2\_2:}
\begin{verbatim}
\$ (ORBITER) -v 2 \ \
-define F -finite_field -q 2 -end \ 
-define v1 -vector -format 3 \ 
-dense "1,0,2,2,0,1,1,2, 1,0,2,0,0,1,1,2, 1,0,2,2,0,1,2,1" \ 
-end \ 
-define v2 -vector -format 3 \ 
-dense "1,0,0,0,0,1,0,0, 1,0,0,0,0,0,0,1, 0,1,0,0,0,0,2,1" \ 
-end \ 
-define F -finite_field -q 3 -end \ 
-define P -projective_space -n 3 -field F -v 0 -end \ 
-define v -vector \ 
-dense "0,36,72,108,144,805" \ 
-end \ 
-define F -finite_field -q 5 -end \ 
-define P -projective_space -n 3 -field F -v 0 -end \ 
-end \ 
-define v -vector \ 
-dense "0,36,72,108,144,805" \ 
-end \ 
-define F -finite_field -q 5 -end \ 
-define P -projective_space -n 3 -field F -v 0 -end \ 
-end \ 
\end{verbatim}
planes_in_pencil:

```bash
$ (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:

EC11.txt:

```bash
$ (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:

```bash
$ (ORBITER) -v 2 \
define F -finite_field -q 4 -end 
define R -polynomial_ring -field F 
number_of_variables 4 
homogeneous_of_degree 3 
end
```
-define P -projective_space -n 3 -field F -v 0 -end \
-define H4 -geometric_object P \
-define H16 -geometric_object P \

Hirschfeld\_surface\_q4.txt:

561

# Section 4.6: The Klein Quadric and Pluecker coordinates

SECTION\_KLEIN\_QUADRIC\_AND\_PLUECKER\_COORDINATES:

GR\_4.2.2:

# the coefficient vector is given as a list of pairs.
# 165 = binomial(11,3)
# Section 4.7: Orthogonal spaces

SECTION ORTHOGONAL SPACES:

Op.4.2:

- $(\text{ORBITER}) -v 2 \
  -$define F -finite_field -q 2 -end \
  -$define O -orthogonal_space 1 4 F -without_group -end \
  -with O -do -orthogonal_space_activity \
  -cheat_sheet.orthogonal -end

- pdflatex O_1_4_2_report.tex
- open O_1_4_2_report.pdf

O_5_2.incidence_matrix.csv:

- $(\text{ORBITER}) -v 2 \
  -$define F -finite_field -q 2 -end \
  -$define O -orthogonal_space 0 5 F -without_group -end \
  -with O -do -orthogonal_space_activity \
  -export_point_line_incidence_matrix \
  -end

- $($(\text{ORBITER}) -v 2 \
  -$define all_one_r -vector -repeat 1 15 -end \
  -$define all_one_c -vector -repeat 1 15 -end \
  -draw_matrix \
  -input_csv_file O_5_2.incidence_matrix.csv \
  -box_width 20 -bit_depth 8 \
  -partition 2 \
  -all_one_r all_one_c \
  -end

- open O_5_2.incidence_matrix_draw.bmp

#O(5,2) projectively = Q(4,2) = (dual of) W(3,2) = W(3,2)

# recall that W(3,2) and Q(4,q) are self dual if q is even

Op.6.2:

- $(\text{ORBITER}) -v 2 \

562
2525 ▷ ▷ -define F -finite_field -q 2 -end \
2526 ▷ ▷ -define O -orthogonal_space 1 6 F -without_group -end \
2527 ▷ ▷ -with O -do -orthogonal_space_activity \n2528 ▷ ▷ ▷ -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 \n2535 ▷ ▷ -define F -finite_field -q 2 -end \n2536 ▷ ▷ -define O -orthogonal_space 1 6 F -end \n2537 ▷ ▷ -with O -do -orthogonal_space_activity \n2538 ▷ ▷ ▷ -export.point_line_incidence_matrix \n2539 ▷ ▷ ▷ -end
2540 ▷ ▷ $(ORBITER) -v 2 \n2541 ▷ ▷ ▷ -define all_one_r -vector -repeat 1 35 -end \n2542 ▷ ▷ ▷ -define all_one_c -vector -repeat 1 105 -end \n2543 ▷ ▷ ▷ -draw_matrix \n2544 ▷ ▷ ▷ ▷ -input_csv_file Op_6_2_incidence_matrix.csv \n2545 ▷ ▷ ▷ ▷ -box_width 20 -bit_depth 8 \n2546 ▷ ▷ ▷ ▷ -partition 2 \n2547 ▷ ▷ ▷ ▷ ▷ all_one_r all_one_c \n2548 ▷ ▷ ▷ ▷ -end
2549 ▷ ▷ open Op_6_2.incidence_matrix_draw.bmp
2550
2551
2552 Op_6_2.with_group:
2553 ▷ $(ORBITER) -v 2 \n2554 ▷ ▷ -define F -finite_field -q 2 -end \n2555 ▷ ▷ -define O -orthogonal_space 1 6 F -end \n2556 ▷ ▷ -with O -do -orthogonal_space_activity \n2557 ▷ ▷ ▷ -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 \n2568 ▷ ▷ -define F -finite_field -q 8 -end \n2569 ▷ ▷ -define O -orthogonal_space 1 6 F -end \n2570 ▷ ▷ -with O -do -orthogonal_space_activity \n2571 ▷ ▷ ▷ -cheat_sheet.orthogonal \n
563
Op_8_2:

- $(\text{ORBITER}) -v 2 \$
- -define F -finite_field -q 2 -end \n- -define 0 -orthogonal_space 1 8 F -without_group -end \n- -with 0 -do -orthogonal_space_activity \n- -cheat_sheet_orthogonal -end
- \n- \n- 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 \n- -define 0 -orthogonal_space 1 6 F -without_group -end \n- -with 0 -do -orthogonal_space_activity \n- -cheat_sheet_orthogonal -end
- \n- \n- 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 \n- -define 0 -orthogonal_space 1 6 F -end \n- -with 0 -do -orthogonal_space_activity \n- -unrank_line_through_two_points 15447347 15225451 \n- -end

# this will create a basic report without the group:

Op_6_64_line_rank:
Op_64_report:

 taraf $(ORBITER) -v 4 \\
 taraf taraf -define F -finite_field -q 64 -end \\
 taraf taraf -define O -orthogonal_space 1 6 F -without_group -end \\
 taraf taraf -with O -do -orthogonal_space_activity \\
 taraf taraf -cheat_sheet_orthogonal \\
 taraf -end

 taraf pdflatex O 1 6 64 report.tex
 taraf open O 1 6 64 report.pdf

eccentric quadric subspace:

 taraf $(ORBITER) -v 3 \\
 taraf taraf -define F -finite_field -q 5 -end \\
 taraf taraf -define v -vector -format 4 \\
 taraf taraf -dense "1,3,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,1,1" \\
 taraf taraf -end \\
 taraf taraf -with O -do -orthogonal_space_activity \\
 taraf taraf -intersect_with_subspace v \\
 taraf -end

 taraf We found that the intersection contains 26 points:

eccentric quadric subspace:

 taraf $(ORBITER) -v 2 \\
 taraf taraf -define F -finite_field -q 7 -end \\
 taraf taraf -define P -projective_space -n 4 -field F -v 0 -end \\
 taraf taraf -define S -geometric_object P \\
 taraf taraf -BLT_database 1 \\
 taraf -end \\
 taraf -with S -do -combinatorial_object_activity -save \\
 taraf -end \\
 taraf # writes BLT_7_1.txt

eccentric quadric subspace:

 taraf $(ORBITER) -v 2 \\
 taraf taraf -define F -finite_field -q 7 -end \\
 taraf taraf -define O -orthogonal_space 0 5 F -without_group -end \\
 taraf taraf -define S -set -file_orbiter_format BLT_7_1.txt -end \\
 taraf taraf -with O -do -orthogonal_space_activity \\
 taraf taraf -print_points S -end
\begin{verbatim}
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 O -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 O -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 \# the output defines doily.csv
2688
2689
2690
2691
2692
2693
2694
2695 \# Section 4.8: Hermitian varieties
2696
2697
2698 SECTION_HERMITIAN_VARIETIES:
2699
2700
2701 H_2.4:
2702 \> $(\text{ORBITER}) -v 2 \\
2703 \> -define F -finite_field -q 4 -end \\
2704 \> -with F -do -finite_field.activity \\
2705 \> -cheat_sheet_hermitian 2 -end \\
2706 \> pdflatex H_2.4.tex
2707 \> open H_2.4.pdf
2708
2709
2710
2711 H_2.9:
\end{verbatim}
2712  ▷ $(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
2720  # 28 points: 6, 11, 9, 7, 14, 19, 17, 15, 80, 75, 78, 74, 35, 30, 33, 29, 62, 57, 60, 56, 26, 21, 24, 3, 37, 82, 64, 46
2721
2722
2723 H_3.4:
2724  ▷ $(ORBITER) -v 2 \
2725  ▷ ▷ -define F -finite_field -q 4 -end \
2726  ▷ ▷ -with F -do -finite_field_activity \
2727  ▷ ▷ ▷ -cheat_sheet_hermitian 3 -end
2728  ▷ pdflatex H_3.4.tex
2729  ▷ open H_3.4.pdf
2730
2731
2732  # H_3.4 = the Hirschfeld surface
2733
2734
2735  # Section 4.9: Projective Space Advanced Topics
2736
2737
2738
2739  SECTION_PROJECTIVE_SPACE_ADVANCED_TOPICS:
2740
2741
2742
2743 fix_structure_2A:
2744  ▷ $(ORBITER) -v 2 \
2745  ▷ ▷ -define F -finite_field -q 4 -end \
2746  ▷ ▷ -define P -projective_space -n 3 -field F -v 0 -end \
2747  ▷ ▷ -with P -do \
2748  ▷ ▷ ▷ -projective_space_activity \
2749  ▷ ▷ ▷ ▷ -cheat_sheet_for_decomposition_by_element_PG 1 \n2750  ▷ ▷ ▷ ▷ ▷ "1,0,0,0, 0,1,0,0, 0,0,1,0, 0,0,0,1, 1" \n2751  ▷ ▷ ▷ ▷ ▷ ▷ fix_structure_2A \n2752  ▷ ▷ ▷ ▷ -end
2753  ▷ pdflatex fix_structure_2A.tex
2754  ▷ open fix_structure_2A.pdf
2755
2756
2757 fix_structure_2B:
fix_structure_2C:

\$(\text{ORBITER}) -v 2 \$
\define F -finite_field -q 4 -end \define P -projective_space -n 3 -field F -v 0 -end \with P -do -projective_space_activity -cheat_sheet_for_decomposition_by_element_PG 1 "1,0,0,0, 1,1,0,0, 0,0,1,0, 0,0,0,1, 0" fix_structure_2B -end
\pdflatex \text{fix_structure_2B.tex}
\open \text{fix_structure_2B.pdf}

trans:
\$(\text{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 -cheat_sheet_for_decomposition_by_element_PG 1 "1,0,0,0, 1,1,0,0, 0,0,1,0, 0,0,0,1, 0" fix_structure_2C -end
\pdflatex \text{fix_structure_2C.tex}
\open \text{fix_structure_2C.pdf}

\define f3 -formula "del\ Pezzo\ F13a" "del\ Pezzo\ F13a" "x,y,z"
\[ x^4 + y^4 + z^4 + 8x^2y^2 + 8x^2z^2 + 8y^2z^2 \]

```
define f4 -formula \( \text{del} \_\text{Pezzo}\_F13b \) "\text{del}\_\text{Pezzo}\_F13b\( x,y,z \)"
```

```
x^4 + y^4 + z^4 + x^2y^2 + x^2z^2 + y^2z^2
```

```
define del_Pezzo13 -collection "f3,f4"
```

```
with P -do \( \text{-projective}\_\text{space}\_\text{activity} \)
```

```
pdflatex del\_Pezzo\_F13b\_report.tex
```

```
open del\_Pezzo\_F13b\_report.pdf
```

```
del\_Pezzo\_F13a\_points.txt:
```

```
$\text{(ORBITER)} -v 3 \( \text{-define F} \) -finite_field -q 13 -end \n```

```
$\text{-define P} -projective_space -n 3 -field F -v 0 -end \n```

```
$\text{-define f1 -formula "del}\_\text{Pezzo}\_F9\)"
```

```
$\text{-define f2 -formula "del}\_\text{Pezzo}\_F11\)"
```

```
$\text{-define f3 -formula "del}\_\text{Pezzo}\_F13a\)"
```

```
$\text{-define f4 -formula "del}\_\text{Pezzo}\_F13b\)"
```

```
$\text{-define del}\_\text{Pezzo9} -collection "f1"
```

```
$\text{-define del}\_\text{Pezzo11} -collection "f2"
```

```
$\text{-define del}\_\text{Pezzo13} -collection "f3,f4"
```

```
$\text{-with P -do} \( \text{-projective}\_\text{space}\_\text{activity} \)
```

```
$\text{-analyze del}\_\text{Pezzo}\_\text{surface del}\_\text{Pezzo13} ""
```

```
pdflatex del\_Pezzo\_F13a\_report.tex
```

```
pdflatex del\_Pezzo\_F13b\_report.tex
```

```
open del\_Pezzo\_F13a\_report.pdf
```

```
open del\_Pezzo\_F13b\_report.pdf
```

```
#dot -Tpng del\_Pezzo\_F13a.gv >del\_Pezzo\_F13a.png
```

```
#open del\_Pezzo\_F13a.png
```

```
# writes del\_Pezzo\_F13a\_points.txt
```

```
```
```
\documentclass{article}
\usepackage{amsmath,amssymb}
\begin{document}
\section*{Section 4.10: Geometric Objects}

\section{SECTION GEOMETRIC OBJECTS:}

\subsection{elliptic_quadric_ellipsis_q8:}
\begin{verbatim}
\$ (ORBITER) -v 2 \n\$define F -finite_field -q 8 -end \n\$define P -projective_space -n 3 -field F -v 0 -end \n\$define f3 -formula "delPezzoF169a" \n\$define f4 -formula "delPezzoF169b" \n\$define delPezzo13 -collection "f3,f4" \n\$define O -geometric_object P
\$define elliptic_quadric_ellipsis_q8 -end
\$with O -do -combinatorial_object_activity -save
\end{verbatim}

\subsection{ovoid_q8:}
\begin{verbatim}
\$ (ORBITER) -v 2 \n\$define F -finite_field -q 8 -end \n\$define P -projective_space -n 3 -field F -v 0 -end \n\$define P -projective_space -n 3 -field F -v 0 -end \n\$define O -geometric_object P
\$define elliptic_quadric_ellipsis_q8 -end
\$with O -do -combinatorial_object_activity -save
\end{verbatim}

\subsection{ovoid_ST_q8:}
\begin{verbatim}
\$ (ORBITER) -v 2 \n\$define F -finite_field -q 8 -end \n\$define P -projective_space -n 3 -field F -v 0 -end \n\$define O -geometric_object P
\$define elliptic_quadric_ellipsis_q8 -end
\$with O -do -combinatorial_object_activity -save
\end{verbatim}
\end{document}
-define P -projective_space -n 3 -field F -v 0 -end \ 
-define O -geometric_object P \ 
-ovoid_ST \ 
-end \ 
-with O -do -combinatorial_object_activity -save \ 
-end  
#ovoid_ST_q8.txt  

Baer_PG_2.4:  
$\text{ORBITER}$ -v 2 \ 
-define F -finite_field -q 4 -end \ 
-define P -projective_space -n 2 -field F -v 0 -end \ 
-define O -geometric_object P \ 
-Baer_substructure \ 
-end \ 
-with O -do -combinatorial_object_activity -save \ 
-end  

Baer_PG_3.4:  
$\text{ORBITER}$ -v 2 \ 
-define F -finite_field -q 4 -end \ 
-define P -projective_space -n 3 -field F -v 0 -end \ 
-define O -geometric_object P \ 
-Baer_substructure \ 
-end \ 
-with O -do -combinatorial_object_activity -save \ 
-end  

BLT_database_5.0:  
$\text{ORBITER}$ -v 2 \ 
-define F -finite_field -q 5 -end \ 
-define P -projective_space -n 4 -field F -v 0 -end \ 
-define O -geometric_object P \ 
-BLT_database 0 \ 
-end \ 
-with O -do -combinatorial_object_activity -save \ 
-end  

# writes BLT_5.0.txt  

BLT_database_7.0:  
$\text{ORBITER}$ -v 2 \ 
-define F -finite_field -q 7 -end \ 
-define P -projective_space -n 4 -field F -v 0 -end \ 

571
2946  >  >  -define O  -geometric_object P \
2947  >  >  >  -BLT_database 0 \
2948  >  >  >  -end \
2949  >  >  >  -with 0 -do -combinatorial_object_activity -save \
2950  >  >  >  -end 

2952  # writes BLT_7_0.txt
2953
2954
2955
2956
2957  BLT_database_67_4:
2958  >  $(ORBITER) -v 2 \
2959  >  >  -define F  -finite_field -q 67 -end \
2960  >  >  -define P  -projective_space -n 4 -field F -v 0 -end \
2961  >  >  -define Obj  -geometric_object P \
2962  >  >  >  -BLT_database 4 \
2963  >  >  >  -end \
2964  >  >  >  -with Obj -do -combinatorial_object_activity -save \
2965  >  >  >  -end \
2966  >  >  >  -define 0  -orthogonal_space 0 5 F  -without_group  -end  \
2967  >  >  >  -define BLT_67_4  -set  -file_orbiter_format  BLT_67_4.txt  -end  \
2968  >  >  >  -with 0  -do  -orthogonal_space_activity  \
2969  >  >  >  >  -print_points  BLT_67_4  -end  
2970  >  >  >  >  pdflatex  BLT_67_4_set_report.tex
2971  >  >  >  >  open  BLT_67_4_set_report.pdf
2972
2973
2974
2975
2976
2977
2978
2979
2980  Doily_disjoint_sets_graph_cliques_3:
2981  >  echo $(DOILY) >doily.csv
2982  >  $(ORBITER) -v 2 \
2983  >  >  -define G  -graph  -disjoint_sets_graph  \
2984  >  >  >  doily.csv \ 
2985  >  >  >  -end \ 
2986  >  >  >  -with G -do  \
2987  >  >  >  >  -graph_theoretic_activity  \
2988  >  >  >  >  -find_cliques \ 
2989  >  >  >  >  >  -target_size 3 \ 
2990  >  >  >  >  >  -output_file  doily_cliques  \
2991  >  >  >  >  >  -end \ 
2992  >  >  >  >  -end \ 

572
2993 ▶ ▶ -print_symbols
2994 ▶ $\$(ORBITER) -v 2 \n2995 ▶ ▶ -union doily.csv doily_cliques.txt doily_cliques_union.csv
2996
2997 # 80 cliques
2998
2999 Doily_disjoint_sets_graph_cliques_5:
3000 ▶ echo $\$(DOILY) >doily.csv
3001 ▶ ▶ $\$(ORBITER) -v 2 \n3002 ▶ ▶ ▶ -define G -graph -disjoint_sets_graph \n3003 ▶ ▶ ▶ ▶ doily.csv \n3004 ▶ ▶ ▶ -end \n3005 ▶ ▶ ▶ -with G -do \n3006 ▶ ▶ ▶ ▶ -graph-theoretic_activity \n3007 ▶ ▶ ▶ ▶ ▶ -find_cliques \n3008 ▶ ▶ ▶ ▶ ▶ ▶ -target_size 5 \n3009 ▶ ▶ ▶ ▶ ▶ ▶ ▶ -output_file doily_cliques_5 \n3010 ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ -end \n3011 ▶ ▶ ▶ ▶ -end \n3012 ▶ ▶ ▶ -print_symbols
3013
3014 # 6 cliques
3015 # doily_cliques_5.csv
3016
3017
3018 PG_3_2_test:
3019 ▶ $\$(ORBITER) -v 2 \n3020 ▶ ▶ -define F -finite_field -q 2 -end \n3021 ▶ ▶ -define P -projective_space -n 3 -field F -v 0 -end \n3022 ▶ ▶ -with P -do -projective_space_activity \n3023 ▶ ▶ ▶ -cheat_sheet \n3024 ▶ ▶ -end
3025 ▶ pdflatex PG_3_2.tex
3026 ▶ open PG_3_2.pdf
3027
3028
3029 Edge_curve_17:
3030 ▶ $\$(ORBITER) -v 2 \n3031 ▶ ▶ -define F -finite_field -q 17 -end \n3032 ▶ ▶ -define R -polynomial_ring -field F \n3033 ▶ ▶ ▶ -number_of_variables 3 \n3034 ▶ ▶ ▶ -homogeneous_of_degree 4 \n3035 ▶ ▶ ▶ -end \n3036 ▶ ▶ -define P -projective_space -n 2 -field F -v 0 -end \n3037 ▶ ▶ -define C -geometric_object P \n3038 ▶ ▶ ▶ -projective_variety R \n3039 ▶ ▶ ▶ ▶ "Edge_q17" "Edge\_q17" \

573
Edge_curve_17_line.type:
> echo $(FILE_Q17) >edge_q17.csv
$ (ORBITER) -v 2 \
> define F -finite_field -q 17 -end \
> define R -polynomial_ring -field F \
> number_of_variables 3 \
> homogeneous_of_degree 4 \
> -end \
> define P -projective_space -n 2 -field F -v 0 -end \
> define C -geometric_object P \
> projective_variety R \
> "Edge_q17" "Edge\_q17" \
> $(EDGE_CURVE_Q17_EQUATION) \
> -end \
> with C -do \
> combinatorial_object_activity \
> -line_type \
> -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 \

#(Edge\_q17.txt)
#combinatorial_object_create::init created a set of size 12
#( 4, 7, 16, 19, 20, 23, 32, 35, 89, 100, 244, 251 )
Section 5.1: Permutation groups

Cyclic_6:

$\text{ORBITER} -v 3$

-define G -permutation_group -cyclic_group 6 -end

-with G -do

-group_theoretic_activity

-report

-end

pdflatex Perm6_report.tex

open Perm6_report.pdf

Cyclic_6_group_table:

$\text{ORBITER} -v 3$

-define G -permutation_group -cyclic_group 6 -end

-with G -do

-group_theoretic_activity

-export_group_table
Symmetric_3:

Symmetric_3_group_table:

Symmetric_3_elements:
3181  $(ORBITER) -v 2 \
3182  $define Sym3_elts -vector -load_csv_data_column \n3183  $define Sym3_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 \n3189  $input_csv_file Sym3_elts_matrix.csv \
3190  $box_width 50 -bit_depth 8 \
3191  $partition 3 \n3192  $all_one_r all_one_c 
3193  $-end 
3194  $open Sym3_elts_matrix.draw.bmp
3195
3196  Symmetric_3_long:
3197  $(ORBITER) -v 3 \
3198  $define G -permutation_group -symmetric_group 3 -end \
3199  $with G -do \
3200  $group_theoretic_activity \
3201  $export_orbiter \
3202  $-end \
3203  $with G -do \
3204  $group_theoretic_activity \
3205  $-export_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  $define Symmetric3_elts.csv 1 -end 
3227  $-save_matrix.csv M
3228 \> \$\text{(ORBITER)} \text{-v} 2  \\
3229 \> \> \text{-define} all\_one\_r \text{-vector} \text{-repeat} 1 \, 6 \text{-end}  \\
3230 \> \> \text{-define} all\_one\_c \text{-vector} \text{-repeat} 1 \, 3 \text{-end}  \\
3231 \> \> \text{-draw_matrix}  \\
3232 \> \> \> \text{-input_csv_file} M\_matrix.csv  \\
3233 \> \> \> \text{-box_width} 50 \text{-bit_depth} 8  \\
3234 \> \> \> \text{-partition} 3  \\
3235 \> \> \> \> \text{all\_one\_r all\_one\_c}  \\
3236 \> \> \> \> \text{-end}  \\
3237 \> \text{pdflatex Perm3\_elements_tree\_draw.tex}  \\
3238 \> \text{open Perm3\_elements_tree\_draw.pdf}  \\
3239 \> \#pdflatex Perm3\_report.tex  \\
3240 \> \#open Perm3\_report.pdf  \\
3241  \\
3242  \\
3243 Symmetric\_4:  \\
3244 \> \$\text{(ORBITER)} \text{-v} 3  \\
3245 \> \> \text{-define} G \text{-permutation_group} \text{-symmetric_group} 4 \text{-end}  \\
3246 \> \> \text{-with} G \text{-do}  \\
3247 \> \> \text{-group_theoretic_activity}  \\
3248 \> \> \> \text{-report}  \\
3249 \> \> \> \text{-end}  \\
3250 \> \text{pdflatex Perm4\_report.tex}  \\
3251 \> \text{open Perm4\_report.pdf}  \\
3252  \\
3253  \\
3254 Symmetric\_4\_group\_table:  \\
3255 \> \$\text{(ORBITER)} \text{-v} 3  \\
3256 \> \> \text{-define} G \text{-permutation_group} \text{-symmetric_group} 4 \text{-end}  \\
3257 \> \> \text{-with} G \text{-do}  \\
3258 \> \> \text{-group_theoretic_activity}  \\
3259 \> \> \> \text{-export_group_table}  \\
3260 \> \> \> \text{-end}  \\
3261 \> \$\text{(ORBITER)} \text{-v} 2  \\
3262 \> \> \text{-define} all\_one\_r \text{-vector} \text{-repeat} 1 \, 24 \text{-end}  \\
3263 \> \> \text{-define} all\_one\_c \text{-vector} \text{-repeat} 1 \, 24 \text{-end}  \\
3264 \> \> \text{-draw_matrix}  \\
3265 \> \> \> \text{-input_csv_file} Perm4\_group\_table.csv  \\
3266 \> \> \> \text{-box_width} 50 \text{-bit_depth} 24  \\
3267 \> \> \> \text{-partition} 3 \, all\_one\_r \, all\_one\_c  \\
3268 \> \> \> \text{-end}  \\
3269 \> \text{open Perm4\_group\_table\_draw.bmp}  \\
3270  \\
3271  \\
3272  \\
3273 Symmetric\_4\_long:  \\
3274 \> \$\text{(ORBITER)} \text{-v} 3  \\
3275

define G -permutation_group -symmetric_group 4 -end \
with G -do \
-group_theoretic_activity \n-export_orbiter \n-end \
with G -do \
-group_theoretic_activity \n-export_group_table \n-end \
with G -do \
-group_theoretic_activity \n-print_elements_tex \n-end \
with G -do \
-group_theoretic_activity \n-report \n-end \
with G -do \
-group_theoretic_activity \n-save_elements_csv "Symmetric4_elts.csv" \n-end \
with G -do \
-group_theoretic_activity \n-export_inversion_graphs "Symmetric4_inversion_graphs.csv" \n-end \
$(ORBITER) -v 2 \n-draw_options \n-nodes \n-embedded -radius 175 \n-xin 10000 -yin 10000 \n-xout 1500000 -yout 600000 \n-scale 0.3 -line_width 1.0 \n-end \
-tree_draw -file Perm4_elements_tree.txt -end 
$(ORBITER) -v 2 -draw_matrix \n-input_csv_file Perm4_group_table.csv \n-box_width 50 -bit_depth 24 -end 
$(ORBITER) -v 2 \n-define M -vector -load_csv_data_column \n-Symmetric4_elts.csv 1 -end 
-save_matrix.csv M 
$(ORBITER) -v 2 \n-define all_one_r -vector -repeat 1 24 -end 
-define all_one_c -vector -repeat 1 4 -end 
-draw_matrix \n-input_csv_file M_matrix.csv 
-box_width 50 -bit_depth 8 

3322 ▶ ▶ ▶ -partition 3 \n3323 ▶ ▶ ▶ ▶ all_one_r all_one_c \n3324 ▶ -end
3325 ▶ pdflatex Perm4_elements_tree_draw.tex
3326 ▶ open Perm4_elements_tree_draw.pdf
3327 ▶ #pdflatex Perm4_report.tex
3328 ▶ #open Perm4_report.pdf
3329
3330
3331
3332
3333 # ToDo:
3334
3335 Symmetric_4_stab:
3336 ▶ $(ORBITER) -v 2 \n3337 ▶ ▶ -define G -permutation_group -symmetric_group 4 -end \n3338 ▶ ▶ -define Orb -orbits -group G \n3339 ▶ ▶ ▶ -on_points \n3340 ▶ ▶ -end
3341 ▶ $(ORBITER) -v 2 \n3342 ▶ ▶ -define gens -vector -file Perm4_stab_orb_0_gens.csv -end \n3343 ▶ ▶ -define G -permutation_group \n3344 ▶ ▶ -bsgs Perm4_stab_orb_0 "Sym3" 4 6 "0,1,2" 2 gens -end \n3345 ▶ ▶ -define Gr -modified_group -from G \n3346 ▶ ▶ ▶ -restricted_action "1,2,3" \n3347 ▶ ▶ -end \n3348 ▶ ▶ -with Gr -do \n3349 ▶ ▶ -group_theoretic_activity \n3350 ▶ ▶ ▶ -report \n3351 ▶ ▶ -end
3352
3353
3354 #▷ ▶ -with G -do \n3355 #▷ ▶ -group_theoretic_activity \n3356 #▷ ▶ ▶ -orbits_on_points \n3357 #▷ ▶ ▶ -stabilizer_of_orbit_rep 0 \n3358 #▷ ▶ -end
3359
3360
3361
3362
3363
3364 #------------------------------------------------------------------------------------
3365 # Section 5.2: Linear Groups
3366
3367
3368 SECTION_LINEAR_GROUPS:
PGL_3_2:

$\texttt{\$(ORBITER) -v 2 \ }
\texttt{\define F -finite_field -q 2 -end \ }
\texttt{\define G -linear_group -PGL 3 F -end \ }
\texttt{\with G -do \ }
\texttt{\group_theoretic_activity \ }
\texttt{\report \ }
\texttt{-end \ }
\texttt{pdflatex PGL_3_2_report.tex \ }
\texttt{open PGL_3_2_report.pdf \ }

PGL_4_2:

$\texttt{\$(ORBITER) -v 2 \ }
\texttt{\define F -finite_field -q 2 -end \ }
\texttt{\define G -linear_group -PGL 4 F -end \ }
\texttt{\with G -do \ }
\texttt{\group_theoretic_activity \ }
\texttt{\report \ }
\texttt{-end \ }
\texttt{pdflatex PGL_4_2_report.tex \ }
\texttt{open PGL_4_2_report.pdf \ }

# created by PGL_4_2_export

PGL_4_2_export:

$\texttt{\$(ORBITER) -v 2 \ }
\texttt{\define F -finite_field -q 2 -end \ }
\texttt{\define G -linear_group -PGL 4 F -end \ }
\texttt{\with G -do \ }
\texttt{\group_theoretic_activity \ }
\texttt{\report \ }
\texttt{-end \ }
\texttt{\with G -do \ }
\texttt{\group_theoretic_activity \ }
\texttt{\export_orbiter \ }
\texttt{-end \ }
\texttt{pdflatex PGL_4_2_report.tex \ }
\texttt{open PGL_4_2_report.pdf \ }

# created by PGL_4_2_export

PGL_4_2_generated:

$\texttt{\$(ORBITER) -v 2 \ }
\texttt{\define gens -vector -file PGL_4_2_gens.csv -end \ }
# PSL(5,3): Order 237783237120 = 121 * 120 * 117 * 108 * 81 * 16

# PSL(4,5): Order 7254000000

# PGL(4,5):
\$\text{ORBITER}$ -v 2 \n-define G -linear_group -PGGL 3 4 -end \n-with G -do \n-group_theoretic_activity \n-do \n-report \nsylow \nclasses \n-end \n-pdflatex PGGL 3 4 report.tex \n-open PGGL 3 4 report.pdf \n
PGGL:3.4:

\$\text{ORBITER}$ -v 2 \n-define G -linear_group -PGGL 3 8 -end \n-with G -do \n-group_theoretic_activity \n-report \n-sylow \n-classes \n-end \n-pdflatex PGGL 3 8 report.tex \n-open PGGL 3 8 report.pdf

PGGL:3.8:

\$\text{ORBITER}$ -v 3 \n-define G -linear_group -PGGL 3 8 -end \n-with G -do \n-group_theoretic_activity \n-report \n-end \n-pdflatex PGGL 3 8 report.tex \n-open PGGL 3 8 report.pdf

AGL:1.27:

\$\text{ORBITER}$ -v 2 \n-define F -finite_field -q 27 -end \n-define G -linear_group -AGL 1 F -end \n-with G -do \n-group_theoretic_activity \n-report \n-end \n-pdflatex AGL 1.27 report.tex \n-open AGL 1.27 report.pdf

#> -group_table \

583
3510
3511 SP_4_2:
3512 \> $(\text{ORBITER}) -v 2 \\
3513 \> \> \> -define F -finite_field -q 2 -end \\
3514 \> \> \> -define G -linear_group -GL 4 F \\
3515 \> \> \> \> -symplectic_group \\
3516 \> \> \> -end \\
3517 \> \> \> -with G -do \\
3518 \> \> \> \> -group_theoretic_activity \\
3519 \> \> \> \> -report \\
3520 \> \> \> -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 \> $(\text{ORBITER}) -v 2 \\
3529 \> \> -define F -finite_field -q 4 -end \\
3530 \> \> -define G -linear_group -PGL 4 F \\
3531 \> \> \> -symplectic_group \\
3532 \> \> \> -end \\
3533 \> \> \> -with G -do \\
3534 \> \> \> -group_theoretic_activity \\
3535 \> \> \> \> -report \\
3536 \> \> \> \> -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 \> $(\text{ORBITER}) -v 2 \\
3546 \> \> -define F -finite_field -q 2 -end \\
3547 \> \> -define G -linear_group -PGO 5 F -end \\
3548 \> \> \> -with G -do \\
3549 \> \> \> -group_theoretic_activity \\
3550 \> \> \> -report \\
3551 \> \> \> -end
3552 \> pdflatex PGO_5_2_report.tex
3553 \> open PGO_5_2_report.pdf
3554
3555 PGG_5_4:
3556 \> $(\text{ORBITER}) -v 2 \\
584
# the following two groups are isomorphic:

```
3594  # the following two groups are isomorphic:
3595
3596  PSP_6_2:
3597    $(ORBITER) -v 2 \
3598    -define F -finite_field -q 2 -end \n3599    -define G -linear_group -PGL 6 F -end \n3600    -with G -do \n3601    -symplectic_group \n3602    -end \n3603    -group_theoretic_activity \n```
# group order 1451520, acting on 63 points

PGO\_7.2:

```bash
$\text{ORBITER} -v 2 \\
define F -finite \_finite\_field -q 2 -end \\
define G -linear\_group -PGO 7 F -end \\
with G -do \\
group\_theoretic\_activity \\
report \\
end
```

# group order 1451520, acting on 63 points

# Section 5.3: Subgroups

SECTION\_SUBGROUPS:

C13:

```bash
$\text{ORBITER} -v 2 \\
\text{define \_gens \_vector \_dense }$(\text{GEN\_C13}) \text{ -end } \\
\text{define G -permutation\_group } \\
\text{bsgs C13 C\{}13\\} 13 13 0 1 \\
gens \\
\end
```

# Section 5.3: Subgroups
3651 ▶ ▶ -group_theoretic_activity \  
3652 ▶ ▶ ▶ -export_group_table \  
3653 ▶ ▶ -end \  
3654 ▶ ▶ -with G -do \  
3655 ▶ ▶ -group_theoretic_activity \  
3656 ▶ ▶ ▶ -report \  
3657 ▶ ▶ -end \  
3658 ▶ ▶ -with G -do \  
3659 ▶ ▶ -group_theoretic_activity \  
3660 ▶ ▶ ▶ -save_elements_csv "C13_elts.csv" \  
3661 ▶ ▶ -end \  
3662 ▶ pdflatex C13_report.tex  
3663 ▶ open C13_report.pdf  
3664  
3665  
3666 C13_generated:  
3667 ▶ $(ORBITER) -v 2 \  
3668 ▶ ▶ -define gens -vector -file C13_gens.csv -end \  
3669 ▶ ▶ -define G -permutation_group \  
3670 ▶ ▶ -bsgs C13 "C_{\{13\}}" 13 13 "0" 1 gens -end \  
3671  
3672  
3673 C13_as_subgroup:  
3674 ▶ $(ORBITER) -v 2 \  
3675 ▶ ▶ -define G -permutation_group -symmetric_group 13 \  
3676 ▶ ▶ ▶ -subgroup_by_generators C13 13 1 $(GEN_C13) -end \  
3677 ▶ ▶ -with G -do \  
3678 ▶ ▶ -group_theoretic_activity \  
3679 ▶ ▶ ▶ -export_orbiter \  
3680 ▶ ▶ -end \  
3681 ▶ ▶ -with G -do \  
3682 ▶ ▶ -group_theoretic_activity \  
3683 ▶ ▶ ▶ -report \  
3684 ▶ ▶ -end \  
3685 ▶ ▶ -with G -do \  
3686 ▶ ▶ -group_theoretic_activity \  
3687 ▶ ▶ ▶ -save_elements_csv "C13_elts.csv" \  
3688 ▶ ▶ -end \  
3689 ▶ #pdflatex Perm13_Subgroup_C13_13_report.tex  
3690 ▶ #open Perm13_Subgroup_C13_13_report.pdf  
3691  
3692  
3693  
3694  
3695 J1:  
3696 ▶ $(ORBITER) -v 2 \  
3697 ▶ ▶ -define G -linear_group -PGL 7 11 -Janko1 -end \  

587
3698 \[ \triangleright \triangleright \ -with \ G \ -do \ \] 
3699 \[ \triangleright \triangleright \ -group_theoretic_activity \ \] 
3700 \[ \triangleright \triangleright \ -report \ \] 
3701 \[ \triangleright \ -end \ \] 
3702 \[ \triangleright \ pdflatex \ PGL_7_{11}_Subgroup_Janko1_{report}.tex \ \] 
3703 \[ \triangleright \ open \ PGL_7_{11}_Subgroup_Janko1_{report}.pdf \ \] 
3704 
3705 PGL_3_{11}_singer: 
3706 \[ \triangleright \ (ORBITER) -v \ 2 \ \] 
3707 \[ \triangleright \ -define \ G \ -linear_group \ -PGL \ 3 \ 11 \ -singer \ 19 \ -end \ \] 
3708 \[ \triangleright \ -with \ G \ -do \ \] 
3709 \[ \triangleright \ -group_theoretic_activity \ \] 
3710 \[ \triangleright \ -report \ \] 
3711 \[ \triangleright \ -end \ \] 
3712 \[ \triangleright \ pdflatex \ PGL_3_{11}_Singer_{3}_{11}_{19}_{report}.tex \ \] 
3713 \[ \triangleright \ open \ PGL_3_{11}_Singer_{3}_{11}_{19}_{report}.pdf \ \] 
3714 
3715 PGL_3_{11}_singer_and_frobenius: 
3716 \[ \triangleright \ (ORBITER) -v \ 2 \ \] 
3717 \[ \triangleright \ -define \ G \ -linear_group \ -PGL \ 3 \ 11 \ -singer_and_frobenius \ 19 \ -end \ \] 
3718 \[ \triangleright \ -with \ G \ -do \ \] 
3719 \[ \triangleright \ -group_theoretic_activity \ \] 
3720 \[ \triangleright \ -report \ \] 
3721 \[ \triangleright \ -end \ \] 
3722 \[ \triangleright \ pdflatex \ PGL_3_{11}_Singer_and_Frob3_{11}_{19}_{report}.tex \ \] 
3723 \[ \triangleright \ open \ PGL_3_{11}_Singer_and_Frob3_{11}_{19}_{report}.pdf \ \] 
3724 
3725 PG_2_{4}_order_21: 
3726 \[ \triangleright \ (ORBITER) -v \ 2 \ \] 
3727 \[ \triangleright \ -define \ G \ -linear_group \ -PGL \ 3 \ 4 \ -end \ \] 
3728 \[ \triangleright \ -with \ G \ -do \ \] 
3729 \[ \triangleright \ -group_theoretic_activity \ \] 
3730 \[ \triangleright \ -search_element_of_order \ 21 \ \] 
3731 \[ \triangleright \ -end \ \] 
3732 
3733 
3734 
3735 quaternion: 
3736 \[ \triangleright \ (ORBITER) -v \ 2 \ \] 
3737 \[ \triangleright \ -define \ G \ -linear_group \ -SL \ 2 \ 3 \ \] 
3738 \[ \triangleright \ -subgroup_by_generators "quaternion" "8" \ 3 \ \] 
3739 \[ \triangleright \ -"1,1,1,2, 2,1,1,1, 0,2,1,0" \ \] 
3740 \[ \triangleright \ -end \ \] 
3741 \[ \triangleright \ -with \ G \ -do \ \] 
3742 \[ \triangleright \ -group_theoretic_activity \ \] 
3743 
3744 
3745
cube_group:

$\$(ORBITER) -v 2 \\
define gens -vector -dense \\
"0,1,0,2,0,0,0,1, \\
0,0,1,0,1,0,2,0,0, \\
2,0,0,0,1,0,0,1" \\
end \\
define G -linear_group -GL 3 3 \\
subgroup_by_generators "cube" "48" 3 \\
gens \\
end \\
with G -do \\
group_theoretic_activity \\
print_elements.tex \\
report \\
end \\
pdflatex GL_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 \\

tetra_group:

$\$(ORBITER) -v 3 \\
define G -linear_group -GL 3 3 \\
subgroup_by_generators "tetra" "12" 2 \\
"0,1,0,0,0,1,1,0,0, 0,0,1,2,0,0,0,2,0" \\
end \\
with G -do \\
group_theoretic_activity \\
print_elements.tex \\
report \\
end \\
pdflatex GL_3_3_Subgroup_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 \\

tetra_group:
Hesse group:

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

$-\text{define gens -vector -compact}$

$-\text{define G -linear_group -PGGL 3 4}$

$-\text{define } G -\text{linear_group -PGGL 3 4}$

$-\text{subgroup_by_generators "Hesse" "432" 7 gens}$

$-\text{end}$

$-\text{define G -linear}$

$-\text{group -theoretic_activity}$

$-\text{-print_elements.tex}$

$-\text{-report}$

$-\text{-end}$

$\text{pdflatex PGGL_3_4_Subgroup_Hesse_432_report.tex}$

$\text{open PGGL_3_4_Subgroup_Hesse_432_report.pdf}$

$\text{#Hesse group:}$

$\text{#1,0,0,0,1,0,0,0,1,0,3,2,3,2,0,}$

$\text{#1,0,0,0,1,0,0,3,1,2,0,1,0,1,3,0,}$

$\text{#1,0,0,0,1,0,0,0,1,1,0,0,0,0,1,1,}$

$\text{#1,0,0,0,2,2,0,0,2,0,0,0,0,0,1,0,}$

$\text{#1,0,0,0,2,3,1,0,2,0,1,0,3,1,3,1,0,}$

$\text{#0,1,1,0,2,0,0,0,1,3,2,0,2,1,1,2,1,}$

$\text{#1,1,0,0,0,0,3,0,3,2,2,0,1,2,3,3,1,}$

$\text{Weyl E8:}$

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

$-\text{define gens -vector -dense}$

$-\text{define } G -\text{linear_group -GL 8 3}$

$-\text{subgroup_by_generators}$

$-\text{"Weyl_E8" "696729600" 2 gens}$

$-\text{end}$

$-\text{-with G -do}$

$-\text{-group-theoretic_activity}$

$-\text{-report}$

$-\text{-end}$

$\text{pdflatex GL_8_3_Subgroup_Weyl_E8_696729600_report.tex}$

$\text{open GL_8_3_Subgroup_Weyl_E8_696729600_report.pdf}$

$\text{# group generators from http://www.math.rwth-aachen.de/~Gabriele.Nebe/LATTICES/E8}$
test_subgroup:
\$(ORBITER) -v 2 \n\$-define F -finite_field -q 2 -end \n\$-define G1 -linear_group -PGOp 6 F -end \n\$-define G2 -linear_group -PGL 6 F \n\$-define CR -vector_ge -action G2 \n\$-read_csv \n\$-PGOp.6.2.coset_reps.csv Element \n\$-end

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\$-define CR -vector_ge -action G2 \n\$-read_csv \n\$-PGOp.6.2.coset_reps.csv Element

\$-end

coset_reps_read:
\$(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\$-define CR -vector_ge -action G2 \n\$-read_csv \n\$-PGOp.6.2.coset_reps.csv Element

\$-end

SP.6.2_point_stab_subgroup:
\$(ORBITER) -v 2 \n\$-define F -finite_field -q 2 -end \n\$-define G -linear_group -PGL 6 F \n
591
$\rhd \rhd \rhd -\text{symplectic}\_\text{group} \\$
$\rhd \rhd -\text{end} \$
$\rhd \rhd -\text{define G0 -modified}\_\text{group -from G} \$
$\rhd \rhd \rhd -\text{point}\_\text{stabilizer 0} \$
$\rhd \rhd -\text{end} \$
$\rhd \rhd -\text{with G0 -do} \$
$\rhd \rhd -\text{group}\_\text{theoretic}\_\text{activity} \$
$\rhd \rhd \rhd -\text{report} \$
$\rhd \rhd -\text{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{\rhd -define F -finite_field -q 2 -end} \$
\text{\rhd -define G -linear_group -PGOp 6 F -end} \$
\text{\rhd -with G -do} \$
\text{\rhd -group\_theoretic\_activity} \$
\text{\rhd \rhd -report} \$
\text{\rhd \rhd -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{\rhd -define F -finite_field -q 2 -end} \$
\text{\rhd -define G -linear_group -PGOp 6 F -end} \$
\text{\rhd -define G0 -modified\_group -from G} \$
\text{\rhd \rhd -point\_stabilizer 0} \$
\text{\rhd \rhd -end} \$
\text{\rhd -with G0 -do} \$
\text{\rhd -group\_theoretic\_activity} \$
\text{\rhd \rhd -report} \$
\text{\rhd \rhd -end} \$
\text{pdflatex PGOp\_6\_2\_report.tex}
\text{open PGOp\_6\_2\_report.pdf}

\# group of order 1152

592
PGOp_6_2_GENS="$
1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
SECTION_LINEAR_GROUPS_ADVANCED_TOPICS:

U_3_3:
$\$(ORBITER) -v 3 \\
-define F -finite_field-q 9 -override_polynomial "17" -end \\
-define G -linear_group -PGL 3 F \\
-subgroup_by_generators "U_3_3" "6048" 2 \\
"1,6,4, 5,0,6, 8,5,1, \\
6,2,1, 7,8,4, 0,6,6" \\
-end \\
-with G -do \\
-group_theoretic_activity \\
-report \\
-end

PGL_2_3:
$\$(ORBITER) -v 3 \\
-define G -linear_group -PGL 2 3 -end \\
-with G -do \\
-group_theoretic_activity \\
-report \\
-group_table \\
-end

#Co3 from Conway et al., 1985 (ATLAS)
Co3: $(\textit{ORBITER}) -v 2$

-define F -finite_field -q 2
-define g1 -vector -field F -format 22 -compact $(\textit{CONWAY\_GEN1})
-define g2 -vector -field F -format 22 -compact $(\textit{CONWAY\_GEN2})
-define gens -vector -concatenate g1 -concatenate g2
-define G -linear_group -PGL 22 2
-subgroup_by_generators "Co3" "495766656000" gens
-with G -do
-group_theoretic_activity
-report
-end

# needs a lot of memory to run!

Ree27: $(\textit{ORBITER}) -v 2$

-define F -finite_field -q 27 -override_polynomial "34"
-define g1 -vector -field F -format 7 -dense $(\textit{Ree\_gen1})
-define g2 -vector -field F -format 7 -dense $(\textit{Ree\_gen2})
-define gens -vector -concatenate g1 -concatenate g2
-define G -linear_group -PGL 7 F
-subgroup_by_generators "Ree27" "10073444472" gens
-with G -do
-group_theoretic_activity
-report
-end

# needs a lot of memory to run!

595
define G -permutation_group -symmetric_group 4 -end \
define G_on_2subsets -modified_group -from G \\
on_k_subsets 2 \\
end \\
-define G

on k subsets 2 \\
end \\
with G_on_2subsets -do \\
group_theoretic_activity \\
-report \\
-end

pdflatex Perm4_on_2_subsets_report.tex
open Perm4_on_2_subsets_report.pdf

T3_on_tensors:

define G \\
linear_group -GL_d_q_wr_Sym_n 2 2 3 \\
on_tensors -end \\
with G -do \\
group_theoretic_activity \\
-report \\
-end

pdflatex GL_2_2_wreath_Sym3_report.tex
open GL_2_2_wreath_Sym3_report.pdf

T3r1:

define G \\
linear_group -GL_d_q_wr_Sym_n 2 2 3 \\
on_rank_one_tensors -end \\
with G -do \\
group_theoretic_activity \\
-report \\
-end

pdflatex GL_2_2_wreath_Sym3_report.tex
open GL_2_2_wreath_Sym3_report.pdf

T4_on_tensors:

define G \\
linear_group -GL_d_q_wr_Sym_n 2 2 4 \\
on_tensors -end \

4120 \[ \[ - \text{with } G \text{ -do} \] \\
4121 \[ \[ - \text{group-theoretic activity} \] \\
4122 \[ \[ \[ - \text{report} \] \\
4123 \[ \[ - \text{end} \] \\
4124 \[ \text{pdflatex GL}_2 \_wreath_Sym4-report.tex} \\
4125 \[ \text{open GL}_2 \_wreath_Sym4-report.pdf} \\
4126 \] \\
4127 \] \\
4128 T4r1: \\
4129 \[ \[ $(\text{ORBITER}) -v 4 \] \\
4130 \[ \[ - \text{define } G \] \\
4131 \[ \[ - \text{linear group -GL}_d q \_wr \_\text{Sym}_n \_2 \_2 \_4 \] \\
4132 \[ \[ \[ - \text{on rank one tensors -end} \] \\
4133 \[ \[ - \text{with } G \text{ -do} \] \\
4134 \[ \[ - \text{group-theoretic activity} \] \\
4135 \[ \[ \[ - \text{report} \] \\
4136 \[ \[ - \text{end} \] \\
4137 \[ \text{pdflatex GL}_2 \_wreath_Sym4-report.tex} \\
4138 \[ \text{open GL}_2 \_wreath_Sym4-report.pdf} \\
4139 \\
4140 \] \\
4141 \text{PGGL}_2 \_8 \_on\_conic: \\
4142 \[ \[ $(\text{ORBITER}) -v 4 \] \\
4143 \[ \[ - \text{define } G \] \\
4144 \[ \[ - \text{linear group -PGGL}_2 \_8 \_PGL2\_\text{OnConic} -\text{end} \] \\
4145 \[ \[ - \text{with } G \text{ -do} \] \\
4146 \[ \[ - \text{group-theoretic activity} \] \\
4147 \[ \[ \[ - \text{report} \] \\
4148 \[ \[ - \text{end} \] \\
4149 \[ \text{pdflatex PGGL}_2 \_8 \_OnConic}_2 \_8\_report.tex} \\
4150 \[ \text{open PGGL}_2 \_8 \_OnConic}_2 \_8\_report.pdf} \\
4151 \\
4152 \] \\
4153 \text{SURFACE}_q13\_\text{STAB}="1,0,0,0,0,0,12,0,0,0,0,12,0,0,0,0,0,1, \] \\
4154 1,0,0,0,0,12,0,0,0,0,1,0,0,0,0,12, \] \\
4155 1,0,0,0,0,12,0,0,12,0,0,0,0,0,1, \] \\
4156 0,1,0,0,1,0,0,0,0,0,1,0,0,0,0,1"} \\
4157 \\
4158 \] \\
4159 \] \\
4160 \text{surface}_q13\_\text{stab.on.tritangents_orbits:} \\
4161 \[ \[ $(\text{ORBITER}) -v 3 \] \\
4162 \[ \[ - \text{define } F \_\text{finite_field} -q 13 \_\text{-end} \] \\
4163 \[ \[ - \text{define } P \_\text{-projective_space} -n 3 \_\text{-field } F \_\text{-v 0} \_\text{-end} \] \\
4164 \[ \[ - \text{define } S \_\text{-cubic_surface} \_\text{-space } P \_\text{-arc_lifting} "0,1,2,3,43,113" \_\text{-end} \] \\
4165 \[ \[ - \text{with } S \_\text{-do} \] \\
4166 \[ \[ - \text{cubic_surface_activity} \] \\

597
-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 \n-end \
-define G -linear_group -PGL 4 13 \
-define G_on_planes -modified_group -from G \n-on_k_subspaces 3 
-end \
-define G_on_planes -modified_group -from G_on_planes \
-restricted_action TriP \n-end \
-with Gr -do \
-define Gr -modified_group -from G_on_planes \
-restricted_action TriP \n-end \
-with Gr -do \
-define G -linear_group -PGL 4 13 -wedge_detached -end \
-with G -do \n-group_theoretic_activity \
-report 
-end \
-define Orb -orbits -group Gr \
-on_points 
-end \
-with Gr -do \
-group_theoretic_activity \
-orbits_on_points \
-stabilizer \
-end 

PGL_4_2_wd: 
\$(ORBITER) -v 3 \
-define G -linear_group -PGL 4 2 -wedge_detached -end \
-with G -do \
-group_theoretic_activity \
-report 
-end \
pdflatex PGL_4_2_Wedge_4_0_detached_report.tex 
open PGL_4_2_Wedge_4_0_detached_report.pdf
PGL_4_2_wd_reverse:

```bash
$(ORBITER) -v 3 \
  -linear_group -PGL 4 2 -wedge_detached -end \n  -group_theoretic_activity \n  -reverse_isomorphism_exterior_square \n  -end
```

# Section 5.6: Group Theoretic Activities

SECTION_GROUP_THEORETIC_ACTIVITIES:

PGL_3_2_elements:

```bash
$(ORBITER) -v 5 \
  -define G -linear_group -PGL 3 2 -end \n  -with G -do \n  -group_theoretic_activity \n  -reverse_isomorphism_exterior_square
```

Sym_3_elements:

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

```bash
$(ORBITER) -v 2 \
  -draw_options \n  -nodes \n  -embedded -radius 250 \n  -xin 10000 -yin 10000 \n  -xout 1000000 -yout 600000 \n  -scale 0.3 -line_width 1.0 \
  -end
```

# Section 5.6: Group Theoretic Activities
4261 \> \> -tree_draw -file Perm3_elements_tree.txt -end
4262 \> \> pdflatex Perm3_elements_tree_draw.tex
4263 \> \> open Perm3_elements_tree_draw.pdf
4264
4265
4266
4267 Cycle_13_power:
4268 \> \> $(ORBITER) -v 5 \>
4269 \> \> \> -define G -permutation_group -symmetric_group 13 -end \>
4270 \> \> \> -with G -do \>
4271 \> \> \> -group_theoretic_activity \>
4272 \> \> \> \> -consecutive_powers \>
4273 \> \> \> \> "1,2,3,4,5,6,7,8,9,10,11,12,0" 13 \>
4274 \> \> \> -end
4275 \> \> \> pdflatex Perm13_all_powers.tex
4276 \> \> \> open Perm13_all_powers.pdf
4277
4278
4279 Cycle_12_power:
4280 \> \> $(ORBITER) -v 5 \>
4281 \> \> \> -define G -permutation_group -symmetric_group 12 -end \>
4282 \> \> \> -with G -do \>
4283 \> \> \> -group_theoretic_activity \>
4284 \> \> \> \> -consecutive_powers \>
4285 \> \> \> \> "1,2,3,4,5,6,7,8,9,10,11,0" 12 \>
4286 \> \> \> -end
4287 \> \> \> pdflatex Perm12_all_powers.tex
4288 \> \> \> open Perm12_all_powers.pdf
4289
4290
4291
4292
4293 PGL_3.4_singer:
4294 \> \> $(ORBITER) -v 5 \>
4295 \> \> \> -define G -linear_group -PGL 3 4 -end \>
4296 \> \> \> -with G -do \>
4297 \> \> \> -group_theoretic_activity \>
4298 \> \> \> \> -find_singer_cycle \>
4299 \> \> \> -end
4300
4301
4302 GL_2.8_multiply:
4303 \> \> $(ORBITER) -v 5 \>
4304 \> \> \> -define G -linear_group -GL 2 8 -end \>
4305 \> \> \> -with G -do \>
4306 \> \> \> -group_theoretic_activity \>
4307 \> \> \> \> -multiply "0,1,2,3" "4,5,6,7" \>
GL_2.7.multiply:
\$(ORBITER) -v 5 \ndefine G -linear_group -GL 2 7 -end \nwith G -do \ngroup_theoretic_activity \nmultiply "0,1,2,3" "4,5,6,0" \nend

GL_2.7.inv:
\$(ORBITER) -v 5 \ndefine G -linear_group -GL 2 7 -end \nwith G -do \ngroup_theoretic_activity \ninverse "0,1,2,3" \nend

GL_2.7.power:
\$(ORBITER) -v 5 \ndefine G -linear_group -GL 2 7 -end \nwith G -do \ngroup_theoretic_activity \nraise_to_the_power "0,1,2,3" 2 \nend

PGL_3.2.classes:
\$(ORBITER) -v 3 \ndefine G -linear_group -PGL 3 2 -end \nwith G -do \ngroup_theoretic_activity \nclasses_based_on_normal_form \nend

PGL_3.2.classes:
\$(ORBITER) -v 3 \ndefine G -linear_group -PGL 3 2 -end \nwith G -do \ngroup_theoretic_activity \nclasses_based_on_normal_form \nend

PGL_3.2.classes:
\$(ORBITER) -v 3 \ndefine G -linear_group -PGL 3 2 -end \nwith G -do \ngroup_theoretic_activity \nclasses_based_on_normal_form \nend

#pdflatex PGL_3.2_classes_out.tex
#open PGL_3_2_classes_out.pdf

PGL_4_2_classes_based_on_normal_form:

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

-define G -linear_group -PGL 4 2 -end \n
-with G -do \n
-group_theoretic_activity \n
-classes_based_on_normal_form \n
-end

pdflatex PGL_4_2_classes_normal_form.tex

open PGL_4_2_classes_normal_form.pdf

PGL_10_2_classes_based_on_normal_form:

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

-define G -linear_group -PGL 10 2 -end \n
-with G -do \n
-group_theoretic_activity \n
-classes_based_on_normal_form \n
-end

pdflatex PGL_10_2_classes_normal_form.tex

open PGL_10_2_classes_normal_form.pdf

normal_forms_PGL_4_4:

$\text{(ORBITER)} -v 7 \$

-define G -linear_group -PGGL 4 4 -end \n
-with G -do \n
-group_theoretic_activity \n
-classes_based_on_normal_form \n
-end

pdflatex PGGL_4_4_classes_normal_form.tex

open PGGL_4_4_classes_normal_form.pdf

PGL_4_4_2A_rank:

$\text{(ORBITER)} -v 6 \$

-define G -linear_group -PGGL 4 4 -end \n
-with G -do \n
-group_theoretic_activity \n
-element_rank \n

4402 ▷ ▷ ▷ "1,0,0,0, 0,1,0,0, 0,0,1,0, 0,0,0,1, 1" \  
4403 ▷ ▷ -end  
4404  
4405  
4406 PGL_4_4_2A_unrank:  
4407 ▷ $(ORBITER) -v 6 \  
4408 ▷ ▷ -define G -linear_group -PGGL 4 4 -end \  
4409 ▷ ▷ ▷ -with G -do \  
4410 ▷ ▷ ▷ -group_theoretic_activity \  
4411 ▷ ▷ ▷ ▷ -element_unrank "1" \  
4412 ▷ ▷ ▷ -end  
4413 ▷ ▷  
4414  
4415  
4416  
4417  
4418  
4419  
4420 PGL_4_5_3B_rank:  
4421 ▷ $(ORBITER) -v 6 \  
4422 ▷ ▷ -define G -linear_group -PGGL 4 5 -end \  
4423 ▷ ▷ -with G -do \  
4424 ▷ ▷ -group_theoretic_activity \  
4425 ▷ ▷ ▷ -element_rank "0,0,0,1, 2,3,0,1, 0,3,4,4, 0,1,2,1" \  
4426 ▷ ▷ ▷ -end  
4427  
4428  
4429 PGL_4_5_3B_unrank:  
4430 ▷ $(ORBITER) -v 6 \  
4431 ▷ ▷ -define G -linear_group -PGGL 4 5 -end \  
4432 ▷ ▷ -with G -do \  
4433 ▷ ▷ -group_theoretic_activity \  
4434 ▷ ▷ ▷ -element_unrank "701459351" \  
4435 ▷ ▷ ▷ -end  
4436 ▷ ▷  
4437  
4438  
4439  
4440 normal_forms_PGL_4_5:  
4441 ▷ $(ORBITER) -v 7 \  
4442 ▷ ▷ -define G -linear_group -PGGL 4 5 -end \  
4443 ▷ ▷ -with G -do \  
4444 ▷ ▷ -group_theoretic_activity \  
4445 ▷ ▷ ▷ -classes_based_on_normal_form \  
4446 ▷ ▷ ▷ -end  
4447 ▷ pdflatex PGL_4_5_classes_normal_form.tex  
4448 ▷ open PGL_4_5_classes_normal_form.pdf

603
related to planes_in_pencil:

# we are computing the action on the planes through the line 0.

on_planes:

$$(\text{ORBITER}) -v 2 \$

$$(\text{define F -finite_field -q 8 -end})$$

$$(\text{define P -projective_space -n 3 -field F -v 0 -end})$$

$$(\text{define G -linear_group -PGL 4 F -end})$$

$$(\text{define G_on_planes -modified_group -from G})$$

$$(\text{on_k_subspaces 3 \})$$

$$(\text{end})$$

$$(\text{with G_on_planes -do \})$$

$$(\text{group_theoretic_activity \})$$

$$(\text{apply "0,8,1,6,4,3,7,2,5" \})$$

$$(\text{"1,0,0,0, 0,1,0,0, 0,0,0,2, 0,0,1,1" \})$$

$$(\text{-end})$$

$$(\text{pdflatex PGL_4_8_Gr_4_3_apply.tex})$$

$$(\text{open PGL_4_8_Gr_4_3_apply.pdf})$$

# Section 5.7: Group Theoretic Activities Based on Magma

SECTION_GROUP_THEORETIC_ACTIVITIES_BASED_ON_MAGMA:

PGGL_2_4_classes:

$$(\text{define G \})$$

$$(\text{-linear_group -PGGL 2 4 \})$$

$$(\text{-end \})$$

$$(\text{with G -do \})$$

$$(\text{-group_theoretic_activity \})$$

$$(\text{-classes \})$$

$$(\text{-end \})$$

$$(\text{$(\text{MAGMA\_PATH})magma PGGL_2_4_classes.magma})$$

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

$$(\text{-define G \})$$

$$(\text{-linear_group -PGGL 2 4 \})$$

$$(\text{-end \})$$

$$(\text{-with G -do \})$$
PGL_7_2_classes:
$(ORBITER) -v 3 \
define G \n-linear_group -PGL 7 2 \n-end \n-with G -do \ngroup_theoretic_activity \nclasses \n-end

$(MAGMA_PATH)magma PGL_7_2_classes.magma

PGL_8_2_classes:
$(ORBITER) -v 3 \
define G \n-linear_group -PGL 8 2 \n-end \n-with G -do \ngroup_theoretic_activity \nclasses \n-end

$(MAGMA_PATH)magma PGL_8_2_classes.magma
PGL_10_2_classes:

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

$\text{-define G }$

$\text{-linear_group -PGL 10 2 }$

$\text{-end }$

$\text{-with G -do }$

$\text{-group_theoretic_activity }$

$\text{-classes }$

$\text{-end }$

$(\text{MAGMA_PATH})\text{magma PGL_10_2_classes.magma}$

$\text{-define G }$

$\text{-linear_group -PGL 10 2 }$

$\text{-end }$

$\text{-with G -do }$

$\text{-group_theoretic_activity }$

$\text{-classes }$

$\text{-end }$

$(\text{MAGMA_PATH})\text{magma PGL_10_2_classes.magma}$

$\text{-define G }$

$\text{-linear_group -PGL 10 2 }$

$\text{-end }$

$\text{-with G -do }$

$\text{-group_theoretic_activity }$

$\text{-classes }$

$\text{-end }$

$(\text{MAGMA_PATH})\text{magma PGL_10_2_classes.magma}$

$\text{-define G }$

$\text{-linear_group -PGL 10 2 }$

$\text{-end }$

$\text{-with G -do }$

$\text{-group_theoretic_activity }$

$\text{-classes }$

$\text{-end }$

$(\text{MAGMA_PATH})\text{magma PGL_10_2_classes.magma}$

$\text{-define G }$

$\text{-linear_group -PGL 10 2 }$

$\text{-end }$

$\text{-with G -do }$

$\text{-group_theoretic_activity }$

$\text{-classes }$

$\text{-end }$

$(\text{MAGMA_PATH})\text{magma PGL_10_2_classes.magma}$

PGGL_2_4_cent_2A:

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

$\text{-define G }$

$\text{-linear_group -PGGL 2 4 -end }$

$\text{-with G -do }$

$\text{-group_theoretic_activity }$

$\text{-centralizer_of_element "2A" "1,0, 0,1, 1" }$

$\text{-report }$

$\text{-end }$

$(\text{MAGMA_PATH})\text{magma element_2A_centralizer.magma}$

$\text{-define G }$

$\text{-linear_group -PGGL 2 4 -end }$

$\text{-with G -do }$

$\text{-group_theoretic_activity }$

$\text{-centralizer_of_element "2A" "1,0, 0,1, 1" }$

$\text{-report }$

$\text{-end }$

$(\text{MAGMA_PATH})\text{magma element_2A_centralizer.magma}$

Normalizer of H5:

$\text{(ORBITER)} -v 2 \$

606
define G -permutation_group -symmetric_group 13 \\
subgroup_by_generators H5 5 1 \\
$(GENERATORS_H5) -end \\
-with G -do \\
-group_theoretic_activity \\
-normalizer \\
-end \\
pdflatex Perm13_Subgroup_H5_5_normalizer.tex \\
open Perm13_Subgroup_H5_5_normalizer.pdf \\

PGGL_3_4_classes: \\
$(ORBITER) -v 3 \\
define G \\
-linear_group -PGGL 3 4 \\
-end \\
-with G -do \\
-group_theoretic_activity \\
-classes \\
-end \\
pdflatex PGGL_3_4_classes_out.tex \\
open PGGL_3_4_classes_out.pdf \\

classes_PGGL_4_4: \\
$(ORBITER) -v 3 \\
-magma_path $(MAGMA_PATH) \\
define G \\
-linear_group -PGGL 4 4 -end \\
-with G -do \\
-group_theoretic_activity \\
-classes \\
-end \\

# group order 1974067200 = 2^{13} * 3^4 * 5^2 * 7 * 17
the -find_subgroup command is too specialized

subgroups_PGL_4_5:
$($ORBITER) -v 6 \
-define G \
-linear_group -PGL 4 5 -end \
-with G -do \
-group_theoretic_activity \
-find_subgroup 3 \
-end

pdflatex PGL_4_5_report.tex
open PGL_4_5_report.pdf

classes_PGL_4_5:
$($ORBITER) -v 6 \
-define G \
-linear_group -PGL 4 5 -end \
-with G -do \
-group_theoretic_activity \
-clases \
-end

pdflatex PGL_4_5_classes_out.tex
open PGL_4_5_classes_out.pdf

# 163 classes

# two classes of elements of order 3
#Order of element = 3 Class size = 310000 Centralizer order = 93600 Normalizer order = 187200
#of order 3 and with 0 fixed points.
#0,1,0,2,0,1,2,1,4,2,3,1,2,0,4,3,  

#Class size = 10075000 Centralizer order = 2880 Normalizer order = 5760
#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:
$($ORBITER) -v 6 -define G \
-linear_group -PGL 4 5 -end \
-with G -do \
-group_theoretic_activity \
-conjugacy_class_of \
"0,0,0,1, 2,3,0,1, 0,3,4,4, 0,1,2,1" \
-end
searchPrimitive_poly_q5_deg3:

$\text{(ORBITER)} -v 6 \$

- search_for_primitive_polynomial_in_range 5 5 3 3

#OK, we found an irreducible and primitive polynomial $X^{-3} + X^{-2} + 2$

GL_3_5_singer_power:

$\text{(ORBITER)} -v 6 -\text{define G} \$

$\text{-linear_group -GL 3 5 -end} \$

$\text{-with G -do} \$

$\text{-group_theoretic_activity} \$

$\text{-raise_to_the_power} \$

$\text{"0,1,0, 0,0,1, 3,0,4" 31} \$

$\text{-end} \$

pdflatex GL_3_5_power.tex

open GL_3_5_power.pdf

PGL_4_5_norm_31:

$\text{(ORBITER)} -v 6 -\text{define G} \$

$\text{-linear_group -PGL 4 5 -end} \$

$\text{-with G -do} \$

$\text{-group_theoretic_activity} \$

$\text{-normalizer_of_cyclic_subgroup "31"} \$

$\text{"2,0,0,0, 0,0,1,0, 0,0,0,1, 0,3,0,4"} \$

$\text{-end} \$

pdflatex normalizer_of_31_in_PGL_4_5.tex

open normalizer_of_31_in_PGL_4_5.pdf

Normalizer_of_Z22_in_PGL_2_9:

$\text{(ORBITER)} -v 2 \$

$\text{-define G -linear_group -PGL 2 9} \$

$\text{-subgroup_by_generators Z22 4 2} \$

$\text{"2,0,0,1, 0,1,1,0" -end} \$

$\text{-with G -do} \$

$\text{-group_theoretic_activity} \$

$\text{-normalizer} \$

$\text{-end} \$

pdflatex PGL_2_9_Subgroup_Z22_4_normalizer.tex

open PGL_2_9_Subgroup_Z22_4_normalizer.pdf

# Chapter 6 - Orbit Algorithms
## Section 6.1: Orbit Algorithms

### Schreier Trees:

orbits_PGL_4_2_on_points:

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

```bash
$(ORBITER) -v 3 \
  -draw_layered_graph \
  -PGL_4_2_0.layered_graph \
  -radius 500 -spanning_tree -embedded \
  -line_width 1.1 -x_stretch 1.4 -scale 0.25 \
  -end
```

```bash
pdflatex PGL_4_2_0_draw.tex
open PGL_4_2_0_draw.pdf
```

```bash
T3r1_orbits:
```

```bash
$(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
```

```bash
pdflatex GL_2_2_wreath_Sym3_orbits_report.tex
open GL_2_2_wreath_Sym3_orbits_report.pdf
```

```bash
```
T3r1.orbits_draw:

```
# write GL_2_2_wreath_Sym3_res27_0.layered_graph

2C_orbit_under_PGGL_4_4.elements_coded.csv:
```

```
# class of size 64260
# creates:
# 2C_orbit_under_PGGL_4_4.csv
# 2C_orbit_under_PGGL_4_4.txt
# 2C_orbit_under_PGGL_4_4.elements_coded.csv
# 2C_orbit_under_PGGL_4_4.transporter.csv
# 1:33 on Mac
```

PGGL_4_4_subgroups_of_type_2C_2C: 2C_orbit_under_PGGL_4_4.elements_coded.csv

```
# User time: 2:59 on Mac
```

```
```

```
```
-subgroup_by_generators "centralizer_2C" "30720" 9 \n"1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1,"
"1,0,0,0,0,1,0,0,0,0,1,0,2,0,3,1,0,"
"1,0,0,0,0,1,0,0,0,0,1,0,1,0,1,1,1,1,"
"1,0,0,0,0,1,0,0,0,0,1,0,0,0,1,1,0,"
"1,0,0,0,0,2,1,0,0,0,0,1,0,1,0,1,0,"
"1,0,0,0,0,1,1,2,0,0,0,1,0,0,0,0,1,0,"
"1,0,3,0,1,1,1,3,0,0,2,0,0,0,0,2,1,"
-end \n-with G -do \n-group_theoretic_activity \n-orbits.on_group_elements.under_conjugation \n-2Corbit_under_PGGL_4_4_elements.coded.csv \n-2Corbit_under_PGGL_4_4.transporter.csv \n-end
-pdflatex subgroups_of_order_4.tex
-open subgroups_of_order_4.pdf

#The distribution of orbit lengths is: ( 1, 2, 15, 20, 24^3, 30, 40, 240, 256, 480, 512, 960^2, 1280, 1920^2, 2560^4, 3840, 5120, 6144^3, 7680, 10240 )
#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:
$ (ORBITER) -v 4 \n-define G -linear_group -PGL 3 13 -end \n-define Orb -orbits -group G \n-on_polynomials 2 \n-end
#pdflatex poly_orbits_d2_n2_q13.tex
#open poly_orbits_d2_n2_q13.pdf

orbits.cubic_curves.q2:
$ (ORBITER) -v 4 \n-define G -linear_group -PGL 3 2 -end \n-define Orb -orbits -group G \n-on_polynomials 3 \n-end
#pdflatex poly_orbits_d3_n3_q2.tex
4869 ▶ #open poly_orbits_d3_n3_q2.pdf
4870
4871
4872 orbits_cubic_curves_q2_with_draw_tree:
4873 ▶ $(ORBITER) -v 4 \
4874 ▶ ▶ -draw_options -yout 500000 -radius 15 -nodes_empty \n4875 ▶ ▶ ▶ -line_width 0.5 -y_stretch 0.25 -embedded -end \n4876 ▶ ▶ ▶ -define G -linear_group -PGL 3 2 -end \n4877 ▶ ▶ ▶ -define Orb -orbits -group G \n4878 ▶ ▶ ▶ -on_polynomials 3 \n4879 ▶ ▶ ▶ -draw_tree 6 \n4880 ▶ ▶ -end
4881
4882
4883 poly_orbits_d3_n3_q2.csv:
4884 ▶ $(ORBITER) -v 4 \
4885 ▶ ▶ -draw_options -yout 500000 -radius 15 -nodes_empty \n4886 ▶ ▶ ▶ -line_width 0.5 -y_stretch 0.25 -embedded -end \n4887 ▶ ▶ ▶ -define G -linear_group -PGL 4 2 -end \n4888 ▶ ▶ ▶ -define Orb -orbits -group G \n4889 ▶ ▶ ▶ -on_polynomials 3 \n4890 ▶ ▶ ▶ -draw_tree 6 \n4891 ▶ ▶ -end
4892
4893 poly_orbits_d3_n3_q2_get_ranks:
4894 ▶ $(ORBITER) -v 4 \
4895 ▶ ▶ -csv_file_select_cols poly_orbits_d3_n3_q2.csv 0
4896 ▶ #pdflatex poly_orbits_d3_n3_q2.tex
4897 ▶ #open poly_orbits_d3_n3_q2.pdf
4898
4899
4900 T4_orbits:
4901 ▶ $(ORBITER) -v 4 \
4902 ▶ ▶ -define G \n4903 ▶ ▶ ▶ -linear_group -GL_d_q.wr_Sym_n 2 2 4 \n4904 ▶ ▶ ▶ -on_tensors -end \n4905 ▶ ▶ ▶ -define Orb -orbits -group G \n4906 ▶ ▶ ▶ -on_points \n4907 ▶ ▶ ▶ -end
4908 ▶ ▶ -end
4909 ▶ pdflatex GL_2_2.wreath_Sym4_res65535.orbits.tex
4910 ▶ open GL_2_2.wreath_Sym4_res65535.orbits.pdf
4911 ▶ #pdflatex GL_2_2.wreath_Sym4_report.tex
4912 ▶ #open GL_2_2.wreath_Sym4_report.pdf
4913
4914
4915 ▶ ▶ -with G -do \n
613
T4r1 orbits:

\$(\text{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

pdflatex GL\(_2\)_2\_wreath\_Sym4\_orbits\_report\_tex
open GL\(_2\)_2\_wreath\_Sym4\_orbits\_report\_pdf

T4r1 orbits draw:

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

- draw layered graph
- GL\(_2\)_2\_wreath\_Sym4\_res81\_0.layered\_graph
- radius 400 -spanning tree -embedded
- line_width 1.1 -x_stretch 2.5 -scale 0.15
- end

pdflatex GL\(_2\)_2\_wreath\_Sym3\_report\_tex
open GL\(_2\)_2\_wreath\_Sym3\_report\_pdf

T4r1 orbits 4:

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

- orbiter_path $(\text{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 \(G\)
PGGL_2_8_on_conic_orbits:
$\text{(ORBITER)} -v 4 \$
$\text{-define G }$
$\text{-linear_group -PGGL 2 8 -PGL2OnConic -end }$
$\text{-define Orb -orbits -group G }$
$\text{-on_points }$
$\text{-end}$
$\# \text{pdflatex PGGL_2_8_OnConic_2_8_orbits_report.tex}$
$\# \text{open PGGL_2_8_OnConic_2_8_orbits_report.pdf}$

# example from the Fining manual, page 107:

PGGL_7_8_orbits:
$\text{(ORBITER)} -v 4 \$
$\text{-define G }$
$\text{-linear_group -PGGL 7 8 -end }$
$\text{-define Orb -orbits -group G }$
$\text{-on_points }$
$\text{-end}$

# 1 min 31 sec on Mac

# Section 6.2: Poset Classification

SECTION_POSET_CLASSIFICATION:
poset_of_4subsets:

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

\$\text{-orbiter\_path}\ $(\text{ORBITER\_PATH}) \$

\$\text{-define}\ \text{Control}\ -\text{poset\_classification\_control}\$

\$\text{-problem\_label}\ \text{poset\_4}\$

\$\text{-W}\ -\text{depth}\ 4 \$

\$\text{-draw\_options}\ -\text{radius}\ 200\ -\text{end}\$

\$\text{-report}\ -\text{end}\$

\$\text{-draw\_poset}\$

\$\text{-end}\$

\$\text{-define}\ \text{G}\ -\text{linear\_group}\ -\text{PGL}\ 2\ 3\ -\text{identity\_group}\ -\text{end}\$

\$\text{-define}\ \text{Orb}\ -\text{orbits}\ -\text{group}\ \text{G}\$

\$\text{-on\_subsets}\ 4\ \text{Control}\$

\$\text{-end}\$

pdflatex PGL_2_3_Identity_2_3_report.tex

pdflatex poset_4.poset.tex

open PGL_2_3_Identity_2_3_report.pdf

open poset_4.poset.pdf

poset_of_4subsets_draw:

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

\$\text{-draw\_layered\_graph}\ \$

\$\text{poset\_4\_poset\_lvl\_4\_layered\_graph}\ \$

\$\text{-radius}\ 300\ -\text{embedded}\ -\text{line\_width}\ 1.1\ \$

\$\text{-y\_stretch}\ 0.9\ -\text{scale}\ 0.25\ \$

\$\text{-end}\$

pdflatex poset_4.poset_lvl_4.draw.tex

open poset_4.poset_lvl_4.draw.pdf

poset_of_5subsets:

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

\$\text{-orbiter\_path}\ $(\text{ORBITER\_PATH}) \$

\$\text{-define}\ \text{Control}\ -\text{poset\_classification\_control}\$

\$\text{-problem\_label}\ \text{poset\_5}\$

\$\text{-W}\ -\text{depth}\ 5\ -\text{draw\_options}\ -\text{radius}\ 150\ -\text{end}\$

\$\text{-report}\ -\text{end}\ -\text{draw\_poset}\$

\$\text{-end}\$

\$\text{-define}\ \text{G}\ -\text{linear\_group}\ -\text{PGL}\ 2\ 4\ -\text{identity\_group}\ -\text{end}\$

\$\text{-define}\ \text{Orb}\ -\text{orbits}\ -\text{group}\ \text{G}\$

\$\text{-on\_subsets}\ 5\ \text{Control}\$

\$\text{-end}\$

pdflatex poset_5.poset.tex

open poset_5.poset.pdf

poset_of_5subsets_draw:
Symmetric_4_on_pairs_poset:

Symmetric_4_on_pairs_poset:

V_3_2_trivial:
V_{4,2} \text{trivial:}

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

-\text{orbiter\_path} \$(\text{ORBITER\_PATH}) \$

-\text{define Control -poset\_classification\_control} \$

-\text{problem\_label V_{4,2} trivial} \$

-\text{-W -depth 3 -node\_label\_is\_element} \$

-\text{-draw\_options} \$

-\text{-radius 200 -embedded} \$

-\text{-end} \$

-\text{-report -end} \$

-\text{-draw\_poset} \$

-\text{-end} \$

-\text{-define G -linear\_group -PGL 4 2 -identity\_group -end} \$

-\text{-define Orb -orbits -group G} \$

-\text{-on\_subspaces 4 Control} \$

-\text{-end} \\ 

# Section 6.3: Orbits on Subsets

$\text{pdflatex PGL}_4.2.\text{Identity}_4.2.\text{report}.\text{tex}$

$\text{open PGL}_4.2.\text{Identity}_4.2.\text{report}.\text{pdf}$

$\text{pdflatex PGL}_4.2.\text{Identity}_4.2.\text{tree\_lvl}_4.\text{tex}$

$\text{open PGL}_4.2.\text{Identity}_4.2.\text{tree\_lvl}_4.\text{pdf}$

$\text{pdflatex PGL}_4.2.\text{Identity}_4.2.\text{poset}.\text{tex}$

$\text{open PGL}_4.2.\text{Identity}_4.2.\text{poset}.\text{pdf}$
SECTION_ORBITS_ON_SUBSETS:

PG_2.2_subsets:

\$\text{ORBITER} -v 3 \$

- \text{orbiter\_path} \$(\text{ORBITER\_PATH}) \$

- \text{define Control} -\text{poset\_classification\_control} \$

- \text{problem\_label} PGL_3.2 \$

- \text{-depth} 7 \$

- \text{-draw\_options} \$

- \text{-radius} 200 -\text{embedded} \$

- \text{-end} \$

- \text{-report} -\text{-end} \$

- \text{-draw\_poset} \$

- \text{-end} \$

- \text{-define F} -\text{finite\_field} -q 2 -\text{-end} \$

- \text{-define G} -\text{linear\_group} -PGL 3 F -\text{-end} \$

- \text{-define Orb} -\text{orbits} -\text{group G} \$

- \text{-on\_subsets} 7 Control \$

- \text{-end} \$

\text{pdflatex} PGL_3.2\_poset\_lvl.7.tex

\text{open} PGL_3.2\_poset\_lvl.7.pdf

\text{pdflatex} PGL_3.2\_poset.tex

\text{open} PGL_3.2\_poset.pdf

\text{#pdflatex} PGL_3.2\_poset\_detailed\_lvl.7.tex

\text{#open} PGL_3.2\_poset\_detailed\_lvl.7.pdf

PG(3,2) has $2^3+2^2+2^1+1=15$ points:

PG(3,3) has $3^3+3^2+3^1+1=27+9+3+1=40$ points.

PG_3.2_subsets:

\$\text{ORBITER} -v 3 \$

- \text{orbiter\_path} \$(\text{ORBITER\_PATH}) \$

- \text{define Control} -\text{poset\_classification\_control} \$

- \text{problem\_label} PGL_4.2 \$

- \text{-depth} 15 \$

- \text{-draw\_options} \$

- \text{-radius} 200 -\text{embedded} \$

- \text{-end} \$

- \text{-report} -\text{-end} \$

- \text{-draw\_poset} \$

- \text{-end} \$

- \text{-define F} -\text{finite\_field} -q 2 -\text{-end} \$

619
define G -linear_group -PGL 4 F -end \r
\r
define Orb -orbits -group G \r
\r
\r
\r
\r
\r
pdflatex PGL_4_2_poset.tex \r
open PGL_4_2_poset.pdf 
 \r
\r
\r
\r
\r
\r
define G -linear_group -PGL 3 2 -singer 1 -end \r
\r
\r
\r
\r
\r
\r
\r
pdflatex PGL_3_2_singer_1_poset.tex \r
open PGL_3_2_singer_1_poset.pdf 
 \r
\r
\r
\r
\r
\r
\r
\r
pdflatex PGL_3_2_on_lines_poset.tex \r
open PGL_3_2_on_lines_poset.pdf 
 \r
\r
\r
\r
\r
\r
\r
\r
$\text{PGL}_2.5$ on subsets: \r
\r
$\text{PGL}_2.5$ on subsets: 

5245 ▷ ▷ -define Control -poset_classification_control \n5246 ▷ ▷ ▷ -problem_label PGL_2.5 -W -depth 6 \n5247 ▷ ▷ ▷ -draw_poset \n5248 ▷ ▷ ▷ -draw_options -radius 200 -end \n5249 ▷ ▷ ▷ -report -end \n5250 ▷ ▷ -end \n5251 ▷ ▷ ▷ -define G -linear_group -PGL 2 5 -end \n5252 ▷ ▷ ▷ -define Orb -orbits -group G \n5253 ▷ ▷ ▷ ▷ -on_subsets 6 Control \n5254 ▷ ▷ ▷ -end \n5255 ▷ pdflatex PGL_2.5_poset.tex \n5256 ▷ open PGL_2.5_poset.pdf \n5257 \n5258 PGL_2.7_on_subsets: \n5259 ▷ $(ORBITER) -v 10 \n5260 ▷ ▷ -orbiter_path $(ORBITER_PATH) \n5261 ▷ ▷ -define Control -poset_classification_control \n5262 ▷ ▷ ▷ -problem_label PGL_2.7 -W -depth 8 \n5263 ▷ ▷ ▷ -draw_poset \n5264 ▷ ▷ ▷ -draw_options -radius 200 -end \n5265 ▷ ▷ ▷ -report -end \n5266 ▷ ▷ -end \n5267 ▷ ▷ ▷ -define G -linear_group -PGL 2 7 -end \n5268 ▷ ▷ ▷ -define Orb -orbits -group G \n5269 ▷ ▷ ▷ ▷ -on_subsets 8 Control \n5270 ▷ ▷ ▷ -end \n5271 ▷ pdflatex PGL_2.7_poset.tex \n5272 ▷ open PGL_2.7_poset.pdf \n5273 \n5274 PGGL_2.8_on_subsets: \n5275 ▷ $(ORBITER) -v 10 \n5276 ▷ ▷ -orbiter_path $(ORBITER_PATH) \n5277 ▷ ▷ -define Control -poset_classification_control \n5278 ▷ ▷ ▷ -problem_label PGGL_2.8 -W -depth 9 \n5279 ▷ ▷ ▷ -draw_poset \n5280 ▷ ▷ ▷ -draw_options -radius 200 -end \n5281 ▷ ▷ ▷ -report -end \n5282 ▷ ▷ -end \n5283 ▷ ▷ -define G -linear_group -PGGL 2 8 -end \n5284 ▷ ▷ -define Orb -orbits -group G \n5285 ▷ ▷ ▷ -on_subsets 9 Control \n5286 ▷ ▷ ▷ -end \n5287 ▷ pdflatex PGGL_2.8_poset.tex \n5288 ▷ open PGGL_2.8_poset.pdf \n5289 \n5290 \n5291 PGGL_2.9_on_subsets:
\begin{verbatim}
5292  \$\{(ORBITER) -v 10 \ \\
5293  \orbiter_path \$(ORBITER_PATH) \ \\
5294  \define Control -poset_classification_control \ \\
5295  \problem_label PGGL_2.9 -W -depth 10 \ \\
5296  \draw_poset \ \\
5297  \draw_options -radius 200 -end \ \\
5298  \report -end \ \\
5299  -end \ \\
5300  \define G -linear_group -PGGL 2 9 -end \ \\
5301  \define Orb -orbits -group G \ \\
5302  \on_subsets 10 Control \ \\
5303  -end \ \\
5304  pdflatex PGGL_2.9_poset.tex
5305  open PGGL_2.9_poset.pdf
5306
5307
5308  PGL_2.11_on_subsets:
5309  \$\{(ORBITER) -v 10 \ \\
5310  \orbiter_path \$(ORBITER_PATH) \ \\
5311  \define Control -poset_classification_control \ \\
5312  \problem_label PGL_2.11 -W -depth 12 \ \\
5313  \draw_poset \ \\
5314  \draw_options -radius 200 -end \ \\
5315  \report -end \ \\
5316  -end \ \\
5317  \define G -linear_group -PGL 2 11 -end \ \\
5318  \define Orb -orbits -group G \ \\
5319  \on_subsets 12 Control \ \\
5320  -end \ \\
5321  pdflatex PGL_2.11_poset.tex
5322  open PGL_2.11_poset.pdf
5323
5324
5325
5326  PGGL_2.16_on_subsets:
5327  \$\{(ORBITER) -v 3 \ \\
5328  \orbiter_path \$(ORBITER_PATH) \ \\
5329  \define Control -poset_classification_control \ \\
5330  \problem_label PGGL_2.16 -W -depth 10 \ \\
5331  \draw_poset \ \\
5332  \report -end \ \\
5333  -end \ \\
5334  \define G -linear_group -PGGL 2 16 -end \ \\
5335  \define Orb -orbits -group G \ \\
5336  \on_subsets 10 Control \ \\
5337  -end \ \\
5338  pdflatex PGGL_2.16_poset.tex
\end{verbatim}
5339 \> open PGGL_2.16_poset.pdf
5340
5341
5342 PGGL_2.32_on_subsets:
5343 \> $(ORBITER) -v 3 \$
5344 \> \> -orbiter_path $(ORBITER_PATH) \$
5345 \> \> \> -define Control -poset_classification_control \$
5346 \> \> \> \> -problem_label PGGL_2.32 -W -depth 8 \$
5347 \> \> \> \> \> -draw_poset \$
5348 \> \> \> \> \> \> -report -end \$
5349 \> \> \> \> \> \> \> -end \$
5350 \> \> \> \> \> \> \> \> -define G -linear_group -PGGL 2 32 -end \$
5351 \> \> \> \> \> \> \> \> \> -define Orb -orbits -group G \$
5352 \> \> \> \> \> \> \> \> \> \> -on_subsets 8 Control \$
5353 \> \> \> \> \> \> \> \> \> \> \> -end \$
5354 \> \> \> \> \> \> \> \> \> \> \> \> -pdflatex PGGL_2.32_poset.tex
5355 \> \> \> \> \> \> \> \> \> \> \> \> \> open PGGL_2.32_poset.pdf
5356
5357
5358 PG_3.4_subsets:
5359 \> $(ORBITER) -v 3 \$
5360 \> \> -orbiter_path $(ORBITER_PATH) \$
5361 \> \> \> -define Control -poset_classification_control \$
5362 \> \> \> \> -problem_label PGGL_4.4 \$
5363 \> \> \> \> \> -depth 5 \$
5364 \> \> \> \> \> \> -draw_poset \$
5365 \> \> \> \> \> \> \> -draw_options \$
5366 \> \> \> \> \> \> \> \> \> -radius 200 \$
5367 \> \> \> \> \> \> \> \> \> \> -end \$
5368 \> \> \> \> \> \> \> \> \> \> \> -report -end \$
5369 \> \> \> \> \> \> \> \> \> \> \> \> -end \$
5370 \> \> \> \> \> \> \> \> \> \> \> \> \> -define G -linear_group -PGGL 4 4 -end \$
5371 \> \> \> \> \> \> \> \> \> \> \> \> \> \> -define Orb -orbits -group G \$
5372 \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> -on_subsets 5 Control \$
5373 \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> -end \$
5374 \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> -pdflatex PGGL_4.4_poset.tex
5375 \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> \> open PGGL_4.4_poset.pdf
5376
5377
5378 PGGL_2.9_orbits:
5379 \> $(ORBITER) -v 3 \$
5380 \> \> -orbiter_path $(ORBITER_PATH) \$
5381 \> \> \> -define Control -poset_classification_control \$
5382 \> \> \> \> -problem_label PGGL_2.9 -W -depth 5 \$
5383 \> \> \> \> \> -report -end \$
5384 \> \> \> \> \> \> -draw_poset \$
5385 \> \> \> \> \> \> \> -draw_options -radius 200 -end \$

623
Section 6.4: Orbits on Subspaces:

```
subspaces_Op_4_2:
  $(ORBITER) -v 5 
  -oritzer_path $(ORBITER_PATH) 
  -define Control -poset_classification_control 
  -define node_label_is_element 
  -draw_poset -draw_options -radius 200 
  -problem_label Op_4_2 -W -depth 4 
```

5433 ▶ ▶ -end \\ 5434 ▶ ▶ -define G -linear_group -PGL 4 2 -orthogonal 1 -end \\
5435 ▶ ▶ -define Orb -orbits -group G \\
5436 ▶ ▶ ▶ -on_subspaces 4 Control \\
5437 ▶ ▶ -end \\
5438
5439 ▶ #pdflatex PGL_4_2_Orthogonal_plus_4_2_poset.tex \\
5440 ▶ #open PGL_4_2_Orthogonal_plus_4_2_poset.pdf \\
5441 \\
5442
5443 PGL_4_2_on_subspaces: \\
5444 ▶ $(ORBITER) -v 5 \ \\
5445 ▶ ▶ -orbiter_path $(ORBITER_PATH) \ \\
5446 ▶ ▶ -define Control -poset_classification_control \ \\
5447 ▶ ▶ ▶ -problem_label PGL_4_2 -W -depth 4 \ \\
5448 ▶ ▶ ▶ -report -end \ \\
5449 ▶ ▶ -end \ \\
5450 ▶ ▶ -define G -linear_group -PGL 4 2 -end \\
5451 ▶ ▶ -define Orb -orbits -group G \ \\
5452 ▶ ▶ ▶ -on_subspaces 4 Control \ \\
5453 ▶ ▶ -end \\
5454 ▶ pdflatex PGL_4_2_poset.tex \\
5455 ▶ open PGL_4_2_poset.pdf \\
5456 \\
5457
5458 #▶ ▶ -with G -do \\
5459 #▶ ▶ -group_theoretic_activity \\
5460 #▶ ▶ ▶ -orbits_on_subspaces 4 \\
5461 #▶ ▶ ▶ -report \ \\
5462 #▶ ▶ ▶ -end \\
5463 #▶ pdflatex PGL_4_2_report.tex \\
5464 #▶ open PGL_4_2_report.pdf \\
5465 \\
5466 \\
5467 ▶ \\
5468 PGL_4_2_singer_on_subspaces: \\
5469 ▶ $(ORBITER) -v 5 \ \\
5470 ▶ ▶ -orbiter_path $(ORBITER_PATH) \ \\
5471 ▶ ▶ -define Control -poset_classification_control \ \\
5472 ▶ ▶ ▶ -node_label_is_element \ \\
5473 ▶ ▶ ▶ -draw_poset \ \\
5474 ▶ ▶ ▶ -draw_options -end \ \\
5475 ▶ ▶ ▶ -problem_label PGL_4_2_singer -W -depth 4 \ \\
5476 ▶ ▶ ▶ -report -end \ \\
5477 ▶ ▶ -end \ \\
5478 ▶ ▶ -define G -linear_group -PGL 4 2 -singer 1 -end \\
5479 ▶ ▶ -define Orb -orbits -group G \ 

625
May 7, 2020: 16 sec on Mac
1643 orbits in total

Op_6_2.orbits_on_subspaces:

$\text{pdflatex} \ PGL\_8\_2\_singer\_on\_subspaces$:

```bash
$\text{pdflatex} \ PGL\_8\_2\_Singer\_8\_2\_1\_poset.tex$
```

```bash
$\text{open} \ PGL\_8\_2\_Singer\_8\_2\_1\_poset.pdf$
```

```
-define G -linear_group -PGL 8 2 -singer 1 -end \\
-define Orb -orbits -group G \\
-define $\text{control}$ -node_label_is_element \\
-draw poset \\
-draw_options -radius 150 -end \\
-problem_label PGL\_8\_2\_singer \\
-w -depth 8 -report -end \\
-end \\
```

```
-define $\text{control}$ -node_label_is_element \\
-draw poset \\
-draw_options -radius 150 -end \\
-problem_label PGL\_8\_2\_singer \\
-w -depth 8 -report -end \\
-end \\
```

```
-define $\text{control}$ -node_label_is_element \\
-draw poset \\
-draw_options -radius 150 -end \\
-problem_label PGL\_8\_2\_singer \\
-w -depth 8 -report -end \\
-end \\
```

\begin{verbatim}
5527 \[ \[ \[ \text{\textit{-draw_options -radius 200 -end \}} \]
5528 \[ \[ \[ \text{\textit{-problem_label Op_6.2 -W \}} \]
5529 \[ \[ \[ \text{\textit{-depth 6 -report -end \}} \]
5530 \[ \[ \[ \text{\textit{-end \}} \]
5531 \[ \[ \[ \text{\textit{-define G -linear_group -PGL 6 2 -orthogonal 1 -end \}} \]
5532 \[ \[ \[ \text{\textit{-define Orb -orbits -group G \}} \]
5533 \[ \[ \[ \text{\textit{-on_subspaces 6 Control \}} \]
5534 \[ \[ \[ \text{\textit{-end \}} \]
5535 \[ \[ \[ \text{\textit{-with G -do \}} \]
5536 \[ \[ \[ \text{\textit{-group_theoretic_activity \}} \]
5537 \[ \[ \[ \text{\textit{-orbits_on_subspaces 6 \}} \]
5538 \[ \[ \[ \text{\textit{-report \}} \]
5539 \[ \[ \[ \text{\textit{-end \}} \]
5540 \[ \[ \[ \text{\textit{-pdflatex PGL_6.2_Orthogonal_plus_6.2_report.tex \}} \]
5541 \[ \[ \[ \text{\textit{-open PGL_6.2_Orthogonal_plus_6.2_report.pdf \}} \]
5542 \[ \[ \[ \text{\textit{-pdflatex PGL_6.2_Orthogonal_plus_6.2_poset.tex \}} \]
5543 \[ \[ \[ \text{\textit{-open PGL_6.2_Orthogonal_plus_6.2_poset.pdf \}} \]
5544 \[ \[ \[ \text{\textit{-June 3, 2020 on Mac: 0 sec \}} \]
5545 \[ \[ \[ \text{\textit{-627}} \]
5546 \[ \[ \[ \text{\textit{-Orbiter}} \]
5547 \[ \[ \[ \text{\textit{-v 5 \}} \]
5548 \[ \[ \[ \text{\textit{-orbiter_path $(ORBITER\_PATH)$ \}} \]
5549 \[ \[ \[ \text{\textit{-define Control -poset_classification_control \}} \]
5550 \[ \[ \[ \text{\textit{-node_label_is_element \}} \]
5551 \[ \[ \[ \text{\textit{-draw_poset \}} \]
5552 \[ \[ \[ \text{\textit{-draw_options -radius 200 -end \}} \]
5553 \[ \[ \[ \text{\textit{-problem_label Op_6.3 -W \}} \]
5554 \[ \[ \[ \text{\textit{-depth 6 -report -end \}} \]
5555 \[ \[ \[ \text{\textit{-end \}} \]
5556 \[ \[ \[ \text{\textit{-define G -linear_group -PGL 6 3 -orthogonal 1 -end \}} \]
5557 \[ \[ \[ \text{\textit{-define Orb -orbits -group G \}} \]
5558 \[ \[ \[ \text{\textit{-on_subspaces 6 Control \}} \]
5559 \[ \[ \[ \text{\textit{-end \}} \]
5560 \[ \[ \[ \text{\textit{-with G -do \}} \]
5561 \[ \[ \[ \text{\textit{-group_theoretic_activity \}} \]
5562 \[ \[ \[ \text{\textit{-orbits_on_subspaces 6 \}} \]
5563 \[ \[ \[ \text{\textit{-report \}} \]
5564 \[ \[ \[ \text{\textit{-end \}} \]
5565 \[ \[ \[ \text{\textit{-pdflatex PGL_6.3_Orthogonal_plus_6.3_report.tex \}} \]
5566 \[ \[ \[ \text{\textit{-open PGL_6.3_Orthogonal_plus_6.3_report.pdf \}} \]
5567 \[ \[ \[ \text{\textit{-June 3, 2020 on Mac: 0 sec \}} \]
5568 \[ \[ \[ \text{\textit{-627}} \]
\end{verbatim}
5574  Op_6.11_orbits_on_subspaces:
5575  $(ORBITER) -v 5 \n5576  -orbiter_path $(ORBITER_PATH) \n5577  -define Control -poset_classification_control \n5578  -node_label_is_element \n5579  -draw_poset \n5580  -draw_options -radius 200 -end \n5581  -problem_label Op_6.11 -W \n5582  -depth 6 -report -end \n5583  -end \n5584  -draw_options -nodes_empty -end \n5585  -define G -linear_group -PGL 6 11 -orthogonal 1 -end \n5586  -define Orb -orbits -group G \n5587  -on_subspaces 6 Control \n5588  -end \n5589  -end \n5590  -with G -do \n5591  -group_theoretic_activity \n5592  -orbits_on_subspaces 6 \n5593  -report \n5594  -end \n5595  -end \n5596  pdflatex PGL_6_11_Orthogonal_plus_6_11_report.tex
5597  open PGL_6_11_Orthogonal_plus_6_11_report.pdf
5598
5599
5600  # June 3, 2020 on Mac: 12 sec
5601
5602  Op_8.2_orbits_on_subspaces:
5603  $(ORBITER) -v 5 \n5604  -orbiter_path $(ORBITER_PATH) \n5605  -define Control -poset_classification_control \n5606  -node_label_is_element \n5607  -draw_poset -draw_options -radius 200 -end \n5608  -problem_label Op_8.2 -W -depth 8 -report -end \n5609  -end \n5610  -define G -linear_group -PGL 8 2 -orthogonal 1 -end \n5611  -define Orb -orbits -group G \n5612  -on_subspaces 8 Control \n5613  -end \n5614  -end \n5615
5616  # with G -do \n5617  -group_theoretic_activity \n5618  -orbits_on_subspaces 8 \n5619  -report \n5620  #
SECTION ORBITS ON SET PARTITIONS:

C6_on_partition:

```bash
$ (ORBITER) -v 5 \n$orbiter_path $(ORBITER_PATH) \n$define Control -poset_classification_control \n$node_label_is_element \n$draw_poset \n$draw_options -radius 200 -end \n$problem_label 0_7_2 \n$W -depth 7 \n$report -end \n-end \n$define F -finite_field -q 2 -end \n$define G -linear_group -PGL 7 F -orthogonal 0 -end \n$define Orb -orbits -group G \n$on_subspaces 7 Control \n-end
```

pdflatex PGL_7_2_Orthogonal_7_2_poset.tex

open PGL_7_2_Orthogonal_7_2_poset.pdf

# Section 6.5: Orbits on set partitions
SECTION_ARCS_AND_CAPS_IN_PROJECTIVE_SPACES:

PGL_2.17:

\$\text{PGL}_2.17\text{.on.partition:}$

```latex
\begin{align*}
&\text{-define } G \text{ -permutation_group -cyclic_group } 6 -\text{end} \\
&\text{-define } \text{Orb -orbits -group } G \\
&\text{-on_partition } 2 \text{ Control} \\
&\text{-end}
\end{align*}
```

PGL_3.27:

\$\text{PGL}_3.27\text{.on.partition:}$

```latex
\begin{align*}
&\text{-define } G \text{ -linear_group -PGL } 2 \text{ 17 -end} \\
&\text{-define } \text{Orb -orbits -group } G \\
&\text{-on_partition } 6 \text{ Control} \\
&\text{-end}
\end{align*}
```
AGGL_2_27:
  ▶ $(ORBITER) -v 5 \
  ▶ ▶ -define G \
  ▶ ▶ -linear_group -AGGL 2 27 -end \
  ▶ ▶ -with G -do \
  ▶ ▶ -group_theoretic_activity \
  ▶ ▶ ▶ -report \
  ▶ ▶ -end
  ▶ pdflatex AGGL_2_27_report.tex
  ▶ open AGGL_2_27_report.pdf

hyperoval_4_classify:
  ▶ $(ORBITER) -v 4 \
  ▶ ▶ -define F -finite_field -q 4 -end \
  ▶ ▶ -define P -projective_space -n 2 -field F -v 0 -end \
  ▶ ▶ -with P -do \
  ▶ ▶ -projective_space_activity \
  ▶ ▶ ▶ -classify_arcs \
  ▶ ▶ ▶ ▶ -poset_classification_control \
  ▶ ▶ ▶ ▶ ▶ -problem_label hyperoval_q4 \
  ▶ ▶ ▶ ▶ ▶ ▶ -W -depth 6 \
  ▶ ▶ ▶ ▶ ▶ ▶ ▶ -report -end \
  ▶ ▶ ▶ ▶ ▶ ▶ -end \
  ▶ ▶ ▶ ▶ -target_size 6 \
  ▶ ▶ ▶ -d 2 \
  ▶ ▶ -end \
  ▶ pdflatex hyperoval_q4_poset.tex
  ▶ open hyperoval_q4_poset.pdf

hyperoval_8_classify:
  ▶ $(ORBITER) -v 4 \
  ▶ ▶ -orbiter_path $(ORBITER_PATH) \
  ▶ ▶ -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_arcs \
  ▶ ▶ ▶ ▶ -poset_classification_control \
  ▶ ▶ ▶ ▶ ▶ -problem_label hyperoval_q8 \

631
frame_stabilizer_PGGL:
$(ORBITER) -v 4 \
-define G \
-linear_group -PGGL 3 8 -end \
-with G -do \
-group_theoretic_activity \
-poset_classification_control \
-problem_label frame_q8 -W -depth 4 \
-draw_options -end \
-report -end \
-end \
-classify_arcs \
-target_size 4 \
-q 8 \
-n 3 \
-d 2 \
-end \
-end

frame_stabilizer_PGL:
$(ORBITER) -v 4 \
-define G \
-linear_group -PGL 3 8 -end \
-with G -do \
-group_theoretic_activity \
-poset_classification_control \
-problem_label frame_q8 -W -depth 4 \
-draw_options -end \

5823 \texttt{hyperoval\_16\_classify:}:

\begin{verbatim}
  > $(ORBITER) -v 4 \\
  > -orbiter_path $(ORBITER\_PATH) \\
  > -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 hyperoval\_q16 -W -depth 18 \\
  > -report -end \\
  > -target_size 18 \\
  > -d 2 \\
  > -end \\
  > pdflatex hyperoval\_q16\_poset.tex \\
  > open hyperoval\_q16\_poset.pdf
\end{verbatim}

5843 \#> -draw_poset -draw_options -end \

5847 \texttt{hyperoval\_16\_1\_conic_type:}

\begin{verbatim}
  > $(ORBITER) -v 2 \\
  > -define F -finite_field -q 16 -end \\
  > -define P -projective_space -n 2 -field F -v 0 -end \\
  > -define H\_16\_1 -geometric_object P \\
  > -set $(HYPEROVAL\_16\_144) \\
  > -end \\
  > -with H\_16\_1 -do \\
  > -combinatorial_object_activity \\
\end{verbatim}
hyperoval_16_1_nonconical_type:
\$(ORBITER) -v 2 \
define F -finite_field -q 16 -end 
define P -projective_space -n 2 -field F -v 0 -end 
define H_16_1 -geometric_object P 
> -set \$(HYPEROVAL\_16\_144) \n> -end 
> -with H_16_1 -do 
> -combinatorial_object_activity 
> -save 
> -end 
> -with H_16_1 -do 
> -combinatorial_object_activity 
> -non_conical_type 
> -end 
> -print_symbols

#We found 17028 non-conical 6 subsets
#Eckardt point number distribution : $13^\{252\},\,$ $9^\{720\},\,$ $5^\{2304\},\,$ $3^\{13752\}$

hyperoval_16_2_nonconical_type:
\$(ORBITER) -v 2 \
define F -finite_field -q 16 -end 
define P -projective_space -n 2 -field F -v 0 -end 
define H_16_2 -geometric_object P 
> -set \$(HYPEROVAL\_16\_16320) \n> -end 
> -with H_16_2 -do 
> -combinatorial_object_activity 
> -save 
> -end 
> -with H_16_2 -do 
> -combinatorial_object_activity 
> -non_conical_type 
> -end 
> -print_symbols
We found 6188 = {17 choose 5} non-conical 6 subsets

Eckardt point number distribution: $45^{68}, 13^{2040}, 5^{4080}$

neighbors of 0 with 4 removed.csv

Row,C0,C1,C2,C3
0,2,3,9,10
1,1,3,7,8
2,10,12,13,15
3,1,5,10,11
4,3,5,6,13
5,8,9,11,12
6,7,11,13,17
7,7,10,14,16
8,1,9,13,16
9,2,8,13,14
10,1,2,15,17
11,6,8,10,17
12,6,7,9,15
13,2,6,11,16
14,5,9,14,17
15,5,8,15,16
16,1,6,12,14
17,2,5,7,12
18,3,12,16,17
19,3,11,14,15

END

hyperoval_16_stab_0_disjoint_sets_graph:
   $(ORBITER) -v 2 -define G -graph -disjoint_sets_graph
   neighbors_of_0_with_4_removed.csv
   $-end$
   $-with G -do$
   $-graph_theoretic_activity$
   $-find_cliques$
   $-target_size 4$
   $-end$
   $-end$
   $-print_symbols$

# 5 cliques of size 4
#ROW,C0,C1,C2,C3
#0,0,6,15,16
5949 #1,1,2,13,14
5950 #2,3,9,12,18
5951 #3,4,5,7,10
5952 #4,8,11,17,19
5953 #END
5954
5955 # clique 0:
5956 #0,2,3,9,10
5957 #6,7,11,13,17
5958 #15,5,8,15,16
5959 #16,1,6,12,14
5960 # partition: (1,6,12,14|2,3,9,10|5,8,15,16|7,11,13,17)
5961 # 4 is missing, it is the nucleus
5962 # 0 is missing is the chosen point
5963
5964
5965
5966
5967
5968
5969
5970 # nonconical 6-arcs are used for classifying cubic surfaces:
5971
5972
5973
5974
5975
5976 nc_arcs_16:
5977 ▶ $(ORBITER) -v 4 \
5978 ▶ ▶ -define F -finite_field -q 16 -end \
5979 ▶ ▶ -define P -projective_space -n 2 -field F -v 0 -end \
5980 ▶ ▶ -with P -do \
5981 ▶ ▶ -projective_space_activity \
5982 ▶ ▶ ▶ -classify_arcs \
5983 ▶ ▶ ▶ ▶ -poset_classification_control \
5984 ▶ ▶ ▶ ▶ -problem_label nc_arcs_q16_d2 \
5985 ▶ ▶ ▶ ▶ -W -depth 6 \
5986 ▶ ▶ ▶ ▶ -report -end \
5987 ▶ ▶ ▶ -end \
5988 ▶ ▶ -target_size 6 \
5989 ▶ ▶ ▶ -d 2 \
5990 ▶ ▶ ▶ -conic_test \
5991 ▶ ▶ -end \
5992 ▶ -end
5993 ▶ pdflatex nc_arcs_q16_d2_poset.tex
5994 ▶ open nc_arcs_q16_d2_poset.pdf
5995
nc_arcs_32_E13:
  $(ORBITER) -v 4 \\
  -orbiter_path $(ORBITER_PATH) \\
  -define F -finite_field -q 32 -end \\
  -define P -projective_space -n 2 -field F -v 0 -end \\
  -with P -do \\
  -projective_space_activity \\
  -classify arcs \\
  -poset.classification_control \\
  -problem_label nc_arcs_q32_d2 \\
  -W -depth 6 \\
  -draw_poset -draw_options -end \\
  -report -end \\
  -end \\
  -target_size 6 \\
  -test_nb_Eckardt_points 13 \\
  -d 2 \\
  -conic_test \\
  -end

open nc_arcs_q32_d2_poset.pdf

c_5_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*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 \n   -define P -projective_space -n 3 -field F -v 0 -end \n   -define S -cubic_surface -space P -catalogue 0 -end \n   -with S -do \n   -cubic_surface_activity \n   -report \n   -report_with_group \n   -end
   pdflatex surface_catalogue_q64.iso0_with_group.tex
   open surface_catalogue_q64.iso0_with_group.pdf

#makes it slow:
#test_nb_Eckardt_points 13 \n# -report -select_orbits_by_level 6 -select_orbits_by_stabilizer_order_multiple_of 24 -end \n#User time: 0:3

nc_arcs_128:
   $(ORBITER) -v 4 \
   -define F -finite_field -q 128 -end \n   -define P -projective_space -n 2 -field F -use_projectivity_subgroup -v 0 -end \n   -with P -do \n   -projective_space_activity \n   -classify_arcs \n   -poset_classification_control \n   -problem_label nc_arcs_q128_d2 -W -depth 6 \n   -report -select_orbits_by_level 6 \n   -select_orbits_by_stabilizer_order_multiple_of 24 \n   -end \n   -target_size 6 \n   -d 2 \n   -conic_test \n   -end
   pdflatex nc_arcs_q128_d2.poset.tex
```
open nc_arcs_q128_d2_poset.pdf

$\text{Example_F64:}$

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

$\text{define F -finite_field -q 64 -end}$

$\text{define P -projective_space -n 3 -field F -v 0 -end}$

$\text{define S64_abcd_52_8_8_52 -cubic.surface -space P -family.general.abcd 52 8 8 52 -end}$

$\text{with S64_abcd_52_8_8_52 -do}$

$\text{cubic.surface_activity}$

$\text{report}$

$\text{end}$

$\text{pdflatex surface_family_general_abcd_q64_a52_b8_c8_d52_report.tex}$
six_arcs_4_nbE13:
$\text{(ORBITER)} -v 3 \$
$\text{define } F \text{ -finite_field -q 4 -end } \$
$\text{define } P \text{ -projective_space -n 2 -field } F -v 0 \text{ -end } \$
$\text{with } P \text{ -do } \$
$\text{-projective_space_activity } \$
$\text{control_six_arcs -problem_label sixarcs_q4 -end } \$
$\text{-six_arcs_not_on_conic -filter_by_nb_Eckardt_points 13 -end } \$

six_arcs_8_nbE13:
$\text{(ORBITER)} -v 3 \$
$\text{define } F \text{ -finite_field -q 8 -end } \$
$\text{define } P \text{ -projective_space -n 2 -field } F -v 0 \text{ -end } \$
$\text{with } P \text{ -do } \$
$\text{-projective_space_activity } \$
$\text{control_six_arcs -problem_label sixarcs_q8 -end } \$
$\text{-six_arcs_not_on_conic -filter_by_nb_Eckardt_points 13 -end } \$

six_arcs_16_nbE13:
$\text{(ORBITER)} -v 3 \$
$\text{define } F \text{ -finite_field -q 16 -end } \$
$\text{define } P \text{ -projective_space -n 2 -field } F -v 0 \text{ -end } \$
$\text{with } P \text{ -do } \$
$\text{-projective_space_activity } \$
$\text{control_six_arcs -problem_label sixarcs_q16 -end } \$
$\text{-six_arcs_not_on_conic -filter_by_nb_Eckardt_points 13 -end } \$

six_arcs_32_nbE13:
$\text{(ORBITER)} -v 3 \$
$\text{define } F \text{ -finite_field -q 32 -end } \$
$\text{define } P \text{ -projective_space -n 2 -field } F -v 0 \text{ -end } \$
$\text{with } P \text{ -do } \$
$\text{-projective_space_activity } \$
$\text{control_six_arcs -problem_label sixarcs_q32 -end } \$
$\text{-six_arcs_not_on_conic -filter_by_nb_Eckardt_points 13 -end } \$

six_arcs_64_nbE13:
$\text{(ORBITER)} -v 3 \$
$\text{define } F \text{ -finite_field -q 64 -end } \$
$\text{define } P \text{ -projective_space -n 2 -field } F -v 0 \text{ -end } \$
$\text{with } P \text{ -do } \$
$\text{-projective_space_activity } \$
6180 ▷▷ -control_six_arcs -problem_label sixarcs_q64 -end \\
6181 ▷▷ -six_arcs_not_on_conic -filter_by_nb_Eckardt_points 13 -end 
6182
6183
6184 #User time: 0:7
6185 # 9 arcs: ago: 4, 8, 24^5, 48^2
6186
6187
6188
6189 six_arcs_128_nbE13:
6190 ▷ $(ORBITER) -v 3 \\
6191 ▷▷ -define F -finite_field -q 128 -end \\
6192 ▷▷ -define P -projective_space -n 2 -field F -v 0 -end \\
6193 ▷▷ -with P -do \\
6194 ▷▷ -projective_space_activity \\
6195 ▷▷ -control_six_arcs -problem_label sixarcs_q128 -end \\
6196 ▷▷ -six_arcs_not_on_conic -filter_by_nb_Eckardt_points 13 -end 
6197
6198 # 1 min 39 sec
6199 # 12 arcs, ago: 4^3, 24^9
6200
6201 six_arcs_256_nbE13:
6202 ▷ $(ORBITER) -v 3 \\
6203 ▷▷ -define F -finite_field -q 256 -end \\
6204 ▷▷ -define P -projective_space -n 2 -field F -v 0 -end \\
6205 ▷▷ -with P -do \\
6206 ▷▷ -projective_space_activity \\
6207 ▷▷ -control_six_arcs -problem_label sixarcs_q256 -end \\
6208 ▷▷ -six_arcs_not_on_conic -filter_by_nb_Eckardt_points 13 -end 
6209
6210 # 27 minutes on ripoff
6211 #User time: 29:11 on ripoff 7/30/21
6212
6213
6214
6215
6216 five_arcs_q13:
6217 ▷ $(ORBITER) -v 4 \\
6218 ▷▷ -define F -finite_field -q 13 -end \\
6219 ▷▷ -define P -projective_space -n 2 -field F -v 0 -end \\
6220 ▷▷ -with P -do \\
6221 ▷▷ -projective_space_activity \\
6222 ▷▷ ▷ -classify_arcs \\
6223 ▷▷ ▷▷ -poset_classification_control \\
6224 ▷▷ ▷▷ ▷ -problem_label five_arcs_q13 -W -depth 5 \\
6225 ▷▷ ▷▷ ▷ -report -end \\
6226 ▷▷ ▷▷ -end \\

641
SECTION_CUBIC_CURVES:

cubic_curves_PG_2.4:

\begin{verbatim}
cubic_curves_PG_2.4:  
\texttt{\$}\hspace{1pt}(\texttt{ORBITER}) -v 3 \texttt{\$} \end{verbatim}

\begin{verbatim}
\texttt{\$}\hspace{1pt}(\texttt{ORBITER}) -v 3 \texttt{\$} \end{verbatim}
cubic_curves_PG_2_8:

```latex
\$(\text{ORBITER}) \text{-v} 3 \text{-define G} \\
\text{-define F -finite_field -q 8 -end} \\
\text{-define P -projective_space -n 2 -field F -v 0 -end} \\
\text{-with P -do} \\
\text{-projective_space_activity} \\
\text{-classify_cubic_curves -q 8 -target_size 9 -n 3 -d 3} \\
\text{-poset_classification_control} \\
\text{-problem_label cc_8 -W -depth 9} \\
\text{-draw_options -radius 200 -embedded -end} \\
\text{-recognize "0,1,2,3,35,28"} \\
\text{-recognize "1,2,3,51,28,61,46,71,40"} \\
\text{-draw_poset} \\
\text{-Kramer_Mesner_matrix 6 9} \\
\text{-end} \\
\text{-end}
```

```
pdflatex cc_4_poset_lvl_9_draw.tex
open cc_4_poset_lvl_9_draw.pdf
```

```
\$\text{ORBITER} \text{-v} 2 \text{-draw_matrix} \\
\text{-input_csv_file cc_8_KM_6_9.csv} \\
\text{-box_width 50 -bit_depth 8 -end}
```

```
pdflatex Cubic_curves_q8.tex
open Cubic_curves_q8.pdf
```

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

```
cubic_curves_PG_2_8.draw:
```

```
\$(\text{ORBITER}) \text{-v} 3 \\
\text{-draw_layered_graph} \\
\text{cc_8_poset_lvl_9.layered_graph} \\
\text{-radius 2 -embedded -line_width 0.01} \\
\text{-y_stretch 1.3 -scale 0.5} \\
\text{-paths_in_between 6 7 9 1} \\
\text{-end}
```

```
pdflatex cc_8_poset_lvl_9_draw.tex
open cc_8_poset_lvl_9_draw.pdf
```

```
#cc_8_poset_lvl_9.layered_graph
```

```
#cc_8_poset_detailed_lvl_9.layered_graph
```

643
# Chapter 7 - Cubic Surfaces

# Section 7.1: Cubic Surfaces Creation

SECTION_CUBIC_SURFACES_CREATION:

surface_4_0:

$\$(ORBITER) -v 3 \$

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

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

$\$(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" \$

$\$( -end \$

$\$(with S -do \$

$\$( -cubic_surface_activity \$

$\$( -export something "Hesse.planes" \$

$\$( -end \$

$\$\#pdflatex surface_catalogue_q4_iso0_report.tex \$

$\$\#open surface_catalogue_q4_iso0_report.pdf \$

$\$pdflatex surface_catalogue_q4_iso0_with_group.tex \$

$\$open surface_catalogue_q4_iso0_with_group.pdf \$
HIRSCHFELD
SURFACE
POINTS
OFF="15,16,17,18,19,20,21,22,24,25,28,29,32,33,36,37,40,41,43,44,45,46,49,50,55,56,57,58,63,64,65,66,71,72,73,74,77,78,83,84"
HIRSCHFELD
SURFACE
HESSE
PLANES="7,8,11,13,14,16,17,19,28,29,32,34,35,37,38,40,42,43,44,45,47,48,52,54,56,57,60,61,63,64,65,66,68,69,73,75,77,78,81,82"
Hirschfeld
surface
get incidence matrix 40 40:
\$(ORBITER) -v 3 \
-define points -vector -dense $(HIRSCHFELD
SURFACE
POINTS
OFF) -end \\
-define planes -vector -dense $(HIRSCHFELD
SURFACE
HESSE
PLANES) -end \\
-define F -finite_field -q 4 -end \\
-define P -projective
space -n 3 -field F -v 0 -end \\
-with P -do \\
-projective
space
activity \\
-restricted_incidence_matrix 1 3 points planes "H_incma_40_40" \\
-\end \\

Hirschfeld_surface_incma_40_40_c:
\$(ORBITER) -v 10 \\
-draw_incidence_structure
description \\
-width 60 -with
10 6 -end \\
-define C -combinatorial
objects \\
-file_of_incidence_geometries H_incma_40_40.inc 40 40 480 \ 
-\end \\
-with C -do \\
-combinatorial
object
activity \\
-canonica
l_form \\
-classification_prefix H_incma_40_40 \\
-label H_incma_40_40 \\
-save Ago \\
-save_transversal \\
-\end \\
-report \\

645
\texttt{$\text{ORBITER}$ -v 2 -draw\_matrix \\
-\text{input\_csv\_file } H_{\text{incma}}_{40,40}\_\text{object0\_TDA}\_\text{flag\_orbits.csv} \\
-\text{secondary\_input\_csv\_file } H_{\text{incma}}_{40,40}\_\text{object0\_TDA}.csv \\
-box\_width 32 -bit\_depth 24 \\
-end \}

\texttt{pdflatex H_{\text{incma}}_{40,40}\_\text{classification.tex} \\
open H_{\text{incma}}_{40,40}\_\text{classification.pdf} \\
}

\texttt{surface\_7.0: \\
$\text{ORBITER}$ -v 3 \\
-define F -finite\_field -q 7 -end \\
-define P -projective\_space -n 3 -field F -v 0 -end \\
-define S -cubic\_surface -space P -catalogue 0 -end \\
-with S -do \\
-cubic\_surface\_activity \\
-report \\
-report\_with\_group \\
-all\_quartic\_curves \\
-end \\
pdflatex surface\_catalogue\_q7\_iso0\_report.tex \\
open surface\_catalogue\_q7\_iso0\_report.pdf \\
pdflatex surface\_catalogue\_q7\_iso0\_with\_group.tex \\
open surface\_catalogue\_q7\_iso0\_with\_group.pdf \\
}

\texttt{Family\_general\_F7: \\
$\text{ORBITER}$ -v 3 \\
-define F -finite\_field -q 7 -end \\
-define P -projective\_space -n 3 -field F -v 0 -end \\
-define S7\_abcd\_2\_3\_3\_4 -cubic\_surface -space P -family\_general\_abcd\_2\_3\_3\_4 -end \\
-with S7\_abcd\_2\_3\_3\_4 -do \\
-cubic\_surface\_activity \\
-report \\
-end \\
pdflatex surface\_family\_general\_abcd\_q7\_a2\_b3\_c3\_d4\_report.tex \\
open surface\_family\_general\_abcd\_q7\_a2\_b3\_c3\_d4\_report.pdf}
# Fermat with 18 Eckardt points
# no automorphism group, so no -report_with_group and no -all_quartic_curves

# Joel:
eckardt_13_4_12:

```bash
$ (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
```

```bash
pdflatex surface_family_Eckardt_q13_a4_b12_with_group.tex
open surface_family_Eckardt_q13_a4_b12_with_group.pdf
```

```
surface_8_0_catalogue:
```

```bash
$ (ORBITER) -v 3 \
  define F -finite_field -q 8 -end \
  define P -projective_space -n 3 -field F -v 0 -end \
  define S8_0 -cubic_surface -space P -catalogue 0 -end \
  with S8_0 -do \
  -cubic_surface_activity \
  report \  
  report_with_group \  
  -end
```

```bash
pdflatex surface_catalogue_q8_iso0_report.tex
open surface_catalogue_q8_iso0_report.pdf
```

```
surface_8_0_clean:
```

```bash
$ (ORBITER) -v 3 \
  define F -finite_field -q 8 -end \
  define P -projective_space -n 3 -field F -v 0 -end \
```

```bash
dflatex surface_catalogue_q8_iso0_report.tex
open surface_catalogue_q8_iso0_report.pdf
```

```
surface_8_0_clean:
```

```bash
$ (ORBITER) -v 3 \
  define F -finite_field -q 8 -end \
  define P -projective_space -n 3 -field F -v 0 -end \
```
# clean equation for Tekirdag-1:

```plaintext
surface_{8.0b}:
$$(ORBITER) -v 3 \$
```

```plaintext
$\text{define F -finite_field -q 8 -end} \$
```

```plaintext
$\text{define P -projective_space -n 3 -field F -v 0 -end} \$
```

```plaintext
$\text{define S8.0 -cubic_surface -space P -catalogue 0} \$
```

```plaintext
$\text{-select_double_six "15,11,21,19,24,5,16,10,23,20,25,4"} \$
```

```plaintext
$\text{-select_double_six "3,2,1,0,5,4,9,8,7,6,11,10"} \$
```

```plaintext
$\text{-transform_inverse "1,4,4,0,6,0,0,0,6,2,0,1,7,0,4,0,0"} \$
```

```plaintext
$\text{-transform "4,4,0,0,0,0,1,0,1,0,0,0,0,0,1,0"} \$
```

```plaintext
$\text{-transform_inverse "2,0,0,0,0,2,0,0,0,2,0,1,1,2,3,0"} \$
```

```plaintext
$\text{-end} \$
```

```plaintext
$\text{-with S8.0 -do} \$
```

```plaintext
$\text{-cubic_surface_activity} \$
```

```plaintext
$\text{-report} \$
```

```plaintext
$\text{-report_with_group} \$
```

```plaintext
$\text{-end} \$
```

```plaintext
$\text{pdflatex surface_catalogue_{q8_iso0_report.tex}}$
```

```plaintext
$\text{open surface_catalogue_{q8_iso0_report.pdf}}$
```

6418

```plaintext
$\text{# writes tangents.txt}$
```

```plaintext
$\text{# 13.0 has 4 Eckardt points}$
```

```plaintext
$\text{# 13.1 has 6 Eckardt points}$
```

```plaintext
$\text{# 13.2 has 9 Eckardt points}$
```

```plaintext
$\text{# 13.3 has 18 Eckardt points}$
```

6448
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 \n$report_with.group \n$end

\texttt{pdflatex surface\_family\_Eckardt.q13.a3.b1\_with\_group.tex}
\texttt{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$with S13_0 -do \n$-cubic_surface_activity \n$report_with.group \n$end

\texttt{pdflatex surface\_catalogue.q13.iso0\_report.tex}
\texttt{open surface\_catalogue.q13.iso0\_report.pdf}

# clean equation for Tekirdag-2:

surface_16_0:
$\$(ORBITER) -v 3 \n$define F -finite_field -q 16 -end \n$define P -projective_space -n 3 -field F -v 0 -end \n$define S16_0 -cubic_surface -space P -catalogue 0 \n$transform "1,0,0,0,0,1,0,12,0,0,0,1,12,0,0,0,0,1,0" \n$transform "15,11,4,0,0,0,12,0,0,12,0,0,0,0,0,1,3" \n$with S16_0 -do \n$-cubic_surface_activity \n
\begin{verbatim}
# rank of lines: (66591, 26737, 4093, 69904, 28376, 26470, 70160, 69855, 26208, 5847, 369, 32230, 529, 30293, 70068, 2178, 29964, 29364, 21501, 2656, 22193, 25209, 22193, 49862, 274)

# Rank of points on Klein quadric: (29181, 4677, 29950, 33, 62496, 429, 1, 9205, 37, 29964, 29364, 21501, 4656, 54735, 5425, 30105, 754, 6680, 13354, 758, 30106, 0, 29209, 48736, 25595, 33780, 4657)

# ai: 29181, 4677, 29950, 33, 62496, 429
# bi: 1, 9205, 37, 29964, 29364, 21501

# Tekirdag-1:

G13.8:
\$ (ORBITER) -v 3 \\
-define F -finite_field -q 8 -end \\
-define P -projective_space -n 3 -field F -v 0 -end \\
-define T1 -cubic_surface -space P -family_G13 2 -end \\
-with T1 -do \\
-cubic_surface_activity \\
-report \\
-report_with_group \\
-end \\
pdflatex surface_family_G13_q16_a2_with_group.tex \\
open surface_family_G13_q16_a2_with_group.pdf

F13.8:
\$ (ORBITER) -v 3 \\
-define F -finite_field -q 8 -end \\
-define P -projective_space -n 3 -field F -v 0 -end \\
-define T1 -cubic_surface -space P -family_F13 2 -end \\
-with T1 -do \\
-cubic_surface_activity \\
-report \\
\end{verbatim}
# Tekirdag-2:

F13.16:

$\text{ORBITER} -v 3 \$

-define F -finite_field -q 16 -end \\
-define P -projective_space -n 3 -field F -v 0 -end \\
-define T2 -cubic_surface -space P -family_F13 2 -end \\
-with T2 -do \\
-cubic_surface_activity \\
-report \\
-report_with_group \\
-end \\

open surface_family_F13_q16_a2_with_group.pdf

# 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 \\

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 \\

open surface_family_F13_q64_a2_with_group.pdf
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 \    -transform_inverse "1,0,0,0,0,1,0,96,0,0,1,96,0,0,0,1,0" \    -end \    -with CO-1 -do \    -cubic_surface_activity \    -report \    -report_with_group \    -end

# recognize the arcs from Colorado-1,2,3:

Colorado2:

$\text{ORBITER} -v 3 \    -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 "1,0,0,0,0,1,0,32,0,0,1,32,0,0,0,1,0" \    -end \    -with CO-2 -do \

652
-cubic_surface_activity \
-report \ 
-report_with_group \ 
-end \ 

Colorado3:

$(ORBITER) -v 3 \
.define F -finite_field -q 128 -end \ 
.define P -projective_space -n 3 -field F -v 0 -end \ 
.define CO-3 -cubic_surface -space P -catalogue 928 \ 
.transform_inverse "1,0,0,0,0,1,0,59,0,0,1,59,0,0,0,1,0" \ 
.end \ 
.with CO-3 -do \ 
-cubic_surface_activity \ 
-report \ 
-report_with_group \ 
-end \ 

# Colorado-1:

F13_128a:

$(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 -family_F13 2 -end \ 
.with CO-1 -do \ 
-cubic_surface_activity \ 
-report \ 
-report_with_group \ 
-end \ 

# Colorado-2:

F13_128b:

$(ORBITER) -v 3 \
.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 -family_F13 6 -end \ 
.with CO-2 -do \ 
-cubic_surface_activity \ 
-report \ 
-report_with_group \ 
-end \ 

# Colorado-3:
6781  F13.128c:
6782  $(ORBITER) -v 3 \$
6783  $define F -finite_field -q 128 -end \$
6784  $define P -projective_space -n 3 -field F -v 0 -end \$
6785  $define CO-3 -cubic_surface -space P -family F13 14 -end \$
6786  $with CO-3 -do \$
6787  $cubic_surface_activity \$
6788  $report \$
6789  $report_with_group \$
6790  $end
6791  
6792  move_two_lines:
6793  $(ORBITER) -v 5 \$
6794  $define F -finite_field -q 8 -end \$
6795  $with F -do -finite_field_activity \$
6796  $move_two_lines_in_hyperplane_stabilizer \$
6797  $end
6798  F
6799  $alpha$ $beta$ $gamma$ $delta$:
6800  $(ORBITER) -v 3 \$
6801  $define F -finite_field -q 7 -end \$
6802  $with F -do -finite_field_activity \$
6803  $parse_and_evaluate \$
6804  "$F\alpha\beta\gamma\delta$" "x0,x1,x2,x3" \$
6805  $(F\alpha\beta\gamma\delta) \$
6806  "$\alpha=2,\beta=3,\gamma=4,\delta=5$" \$
6807  $end$
6808  $dot -Tpng F\alpha\beta\gamma\delta.gv >F\alpha\beta\gamma\delta.png
6809  F
6810  abcd
6811  Eckardt_q31:
6812  $(ORBITER) -v 3 \$
6813  $define F -finite_field -q 31 -end \$
6814  $define P -projective_space -n 3 -field F -v 0 -end \$
6815  $define F\abcd -cubic_surface -space P \$
6816  $by_equation "F\abcd" \$
6817  "$\{a,b,c,d\}X0,X1,X2,X3$" \$
6818  $(F\abcd_eqn) \$
6819  "$a=2,b=30,c=30,d=2$" \$
6820  "$\{a=2,b=30,c=30,d=2\}$" \$
6821  $end \$
}

654
\begin{verbatim}
  \$ (ORBITER) -v 3 \$
  \$ define F -finite_field -q 7 -end \$
  \$ define P -projective_space -n 3 -field F -v 0 -end \$
  \$ define F -cubic_surface -space P \$
  \$ by_equation "F_abcd" \$
  \$ \$ (F_abcd_eqn) "a=2,b=3,c=4,d=5" \$
  \$ \$ end \$

  \$ $\text{dot} -Tpng \text{F_alpha_beta_gamma_delta}\text{.gv}$ \$
  \$ >F_alpha_beta_gamma_delta\text{.png} \$

  \$ $\text{F_abcd}\text{.sweep}\_4\_27\_q7: \$
  \$ \$ (ORBITER) -v 3 \$
  \$ \$ define F -finite_field -q 7 -end \$
  \$ \$ define P -projective_space -n 3 -field F -v 0 -end \$
  \$ \$ define F -cubic_surface -space P \$
  \$ \$ by_equation "F_abcd" \$
  \$ \$ (F_abcd_eqn) \$
  \$ \$ end \$

  \$ $\text{F_alpha_beta_gamma_delta}\text{.q7}\text{.override}\text{.group:} \$
  \$ \$ (ORBITER) -v 3 \$
  \$ \$ define F -finite_field -q 7 -end \$
  \$ \$ define P -projective_space -n 3 -field F -v 0 -end \$
  \$ \$ define F -cubic_surface -space P \$
  \$ \$ by_equation "F_alpha_beta_gamma_delta" \$
\end{verbatim}
# cubic surfaces with 15 lines:

```bash
F_alpha_beta_gamma_delta_sweep_4_q3:

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

-define F -finite_field -q 3 -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_sweep_4_q3" -q 3

"x0,x1,x2,x3" 

$(F_ALPHA_BETA_GAMMA_DELTA) 

"alpha=2,beta=3,gamma=4,delta=5" 

"D\alpha=2,\beta=3,\gamma=4,\delta=5\D" 

-end 

-end
```

# cubic surfaces with 15 lines:

```bash
surface_15lines_q7_1:

$\text{(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" 
```

656
\[
\DF{\alpha, \beta, \gamma, \delta} \ 
\]

x0, x1, x2, x3

\[
\alphadef{6, \beta=4, \gamma=2, \delta=2} 
\]

\[
\alphadef{2, \beta=3, \gamma=4, \delta=5} 
\]
with Surf -do 
-classification_of_cubic_surfaces_with_double_sixes_activity 
-recognize 
-q 49 
-by_equation "F_alpha_beta_gamma_delta" 
"\D_{\{alpha, beta, gamma, \delta\}} "x0, x1, x2, x3" 
$(F_ALPHA_BETA_GAMMA_DELTA) 
"alpha=2, beta=1, gamma=1, \delta=2" 
"\D{alpha=2, beta=1, gamma=1, \delta=2}D" 
-end 
-end 
-end 

surf49_recognize:
$(ORBITER) -v 3 
-define F -finite_field -q 49 -end 
-define P -projective_space -n 3 -field F -v 0 -end 
-define S -cubic_surface -space P -by_coefficients $(SURFACE_MCKEAN_15_LINES)
-end 
#pdflatex surface_by_coefficients_q7_report.tex
#open surface_by_coefficients_q7_report.pdf

# 2 Eckardt points

F_4.4.3.3_q7:

$\{(\text{ORBITER}) -v 3 \ \}
\begin{quote}
$-\text{define } F -\text{finite\_field } -q 7 -\text{end }$
$-\text{define } P -\text{projective\_space } -n 3 -\text{field } F -v 0 -\text{end }$
$-\text{define } S -\text{cubic\_surface } -\text{space } P$
$-\text{by\_equation }$
$"F_{alpha\_beta\_gamma\_delta}"
\\
$"x0,x1,x2,x3"$
$\}$
$\}$
\end{quote}

$\}$
\begin{quote}
$-\text{define } P -\text{projective\_space } -n 2 -\text{field } F -v 0 -\text{end }$
$-\text{define } C -\text{quartic\_curve } -\text{space } P -\text{catalogue } 0 -\text{end }$
$-\text{with } S -\text{do }$
$-\text{cubic\_surface\_activity }$
$-\text{report }$
$-\text{end}$
\end{quote}

#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:

$\{(\text{ORBITER}) -v 3 \ \}$
\begin{quote}
$-\text{define } F -\text{finite\_field } -q 9 -\text{end }$
$-\text{define } P -\text{projective\_space } -n 2 -\text{field } F -v 0 -\text{end }$
$-\text{define } C -\text{quartic\_curve } -\text{space } P -\text{catalogue } 0 -\text{end }$
$-\text{with } C -\text{do }$
$-\text{quartic\_curve\_activity }$
$-\text{report }$
$-\text{end}$
quartic_curve_13_0_report:

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

$\$(\text{ORBITER}) -v 2 \$

PG_2_{13} \text{rank_lines}:

PG_2_{13} \text{orbits on lines}:
7109 ▶ ▶ ▶ -report -end \\ 7110 ▶ ▶ ▶ -draw_poset \\ 7111 ▶ ▶ ▶ -draw_options -radius 200 -end \\ 7112 ▶ ▶ ▶ -recognize "170, 111, 140, 2" \\ 7113 ▶ ▶ ▶ -recognize "0,23,24,47" \\ 7114 ▶ ▶ -end \\ 7115 ▶ ▶ -define G -linear_group -PGL 3 13 -end \\ 7116 ▶ ▶ -define G_on_lines -modified_group -from G \\ 7117 ▶ ▶ -on_k_subspaces 2 \\ 7118 ▶ ▶ -end \\ 7119 ▶ ▶ -define Orb -orbits -group G_on_lines \\ 7120 ▶ ▶ -on_subsets 4 Control \\ 7121 ▶ ▶ -end \\ 7122 ▶ #pdflatex PGL_3_13.poset.tex \\ 7123 ▶ #open PGL_3_13.poset.pdf \\ 7124 7125 # stabilizer of \{0,23,24,47\} \\ 7126 #1,0,0,7,9,0,9,5,3, \\ 7127 #1,3,0,1,12,0,10,9,2, \\ 7128 #1,1,11,7,9,6,10,5,2, \\ 7129 #1,4,11,12,1,8,10,4,2, \\ 7130 7131 quartic_curve_13_1_report: \\ 7132 ▶ $(ORBITER) -v 3 \\ 7133 ▶ ▶ -define F -finite_field -q 13 -end \\ 7134 ▶ ▶ -define P -projective_space -n 2 -field F -v 0 -end \\ 7135 ▶ ▶ -define C -quartic_curve -space P -catalogue 1 -end \\ 7136 ▶ ▶ -with C -do \\ 7137 ▶ ▶ ▶ quartic_curve_activity \\ 7138 ▶ ▶ ▶ -report \\ 7139 ▶ ▶ ▶ -end \\ 7140 ▶ pdflatex quartic_curve_catalogue_q13_iso1_report.tex \\ 7141 ▶ open quartic_curve_catalogue_q13_iso1_report.pdf \\ 7142 7143 7144 NB_QUARTIC_CURVES_Q19=14 \\ 7145 7146 quartic_curves_19_report: \\ 7147 ▶ $(ORBITER) -v 3 \\ 7148 ▶ ▶ -define F -finite_field -q 19 -end \\ 7149 ▶ ▶ -define P -projective_space -n 2 -field F -v 0 -end \\ 7150 ▶ ▶ -loop L 0 $(NB_QUARTIC_CURVES_Q19) 1 \\ 7151 ▶ ▶ -define C -quartic_curve -space P -catalogue %L -end \\ 7152 ▶ ▶ -with C -do \\ 7153 ▶ ▶ ▶ -quartic_curve_activity \\ 7154 ▶ ▶ ▶ -report \\ 7155 ▶ ▶ ▶ -end \\
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
-report \n
-report_with_group \n
-all_quartic_curves \n
-end \n
pdflatex surface_catalogue_q4_iso0_report.tex \n
open surface_catalogue_q4_iso0_report.pdf \n
pdflatex surface_catalogue_q4_iso0_with_group.tex \n
open surface_catalogue_q4_iso0_with_group.pdf \n
#pdflatex surface_catalogue_q4_iso0_quartics.tex \n
#open surface_catalogue_q4_iso0_quartics.pdf

NB_CUBIC_SURFACES_Q7=1

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

7203 -loop L 0 $(NB_CUBIC_SURFACES_Q7) 1
7204 -define S_%L -cubic_surface -space P -catalogue %L -end
7205 -end_loop
7206 -print_symbols
7207 -loop L 0 $(NB_CUBIC_SURFACES_Q7) 1
7208 -with S_%L -do
7209 -cubic_surface_activity
7210 -export_all_quartic_curves
7211 -end
7212 -end_loop
7213 -print_symbols
7214
7215
7216
7217 NB_CUBIC_SURFACES_Q13=4
7218
7219 quartic_curves_q13:
7220 -list_arguments
7221 -draw_options
7222 -define F -finite_field -q 13
7223 -define P -projective_space -n 3 -field F
7224 -loop L 0 $(NB_CUBIC_SURFACES_Q13) 1
7225 -define S_%L -cubic_surface -space P -catalogue %L
7226 -end_loop
7227 -print_symbols
7228 -loop L 0 $(NB_CUBIC_SURFACES_Q13) 1
7229 -with S_%L -do
7230 -cubic_surface_activity
7231 -export_all_quartic_curves
7232 -end
7233 -end_loop
7234 -print_symbols
7235 quartic_curves_q13_combine:
7236 quartic_curves_q13_classify:
7244 quartic_curves_q13_classify:
quartic curves q19:

$\text{(ORBITER\_PATH)orbiter.out -v 3 \}
-\text{-list.arguments} \ \
-\text{-draw_options -end} \ \
-\text{-define F -finite_field -q 19 -end} \ 
-\text{-define P -projective_space -n 3 -field F -end} \ 
-\text{-loop L 0 $(NB\_CUBIC\_SURFACES\_Q19) 1 \}
-\text{-define S\_L -cubic_surface -space P -catalogue S\_L -end} \ 
-\text{-end_loop} \ 
-\text{-print_symbols} \ 
-\text{-loop L 0 $(NB\_CUBIC\_SURFACES\_Q19) 1 \}
-\text{-with S\_L -do} \ 
-\text{-cubic_surface_activity} \ 
-\text{-export_all_quartic_curves} \ 
-\text{-end} \ 
-\text{-end_loop} \ 
-\text{-print_symbols} \ 

quartic_curves_q19_combine:

$\text{(ORBITER\_PATH)orbiter.out -v 3 \}
-\text{-csv_file_concatenate_from_mask $(NB\_CUBIC\_SURFACES\_Q19) \}
-\text{surface_catalogue_q19_iso\_ld_quartics.csv \}
-\text{quartics_q19.csv} \ 

quartic_curves_q19_classify:

$\text{(ORBITER) -v 3 \}
-\text{-list_arguments} \ 
-\text{-define F -finite_field -q 19 -end} \ 
-\text{-define P -projective_space -n 2 -field F -v 0 -end} \ 
-\text{-with P -do} \ 
-\text{-projective_space_activity} \ 
-\text{-classify_quartic_curves_with_substructure} \ 
-\text{quartics_q19.csv \}
SECTION 7.3: Classification of Cubic Surfaces with 27 lines:

surface_classify_q4:

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

# time: 0:00

surface_classify_q4_arc_lifting_two_lines:

> $(ORBITER) -v 10 \\n> -define F -finite_field -q 4 -end \\n> -define P -projective_space -n 3 -field F -v 0 -end \\n> -with P -do \\n> -projective_space_activity \\n> -control_six_arcs -problem_label sixarcs_q4 -end \\n> -classify_surfaces_through_arcs_and_two_lines \\n> -end
surface_classify_q7:
$\$(\text{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:
$\$(\text{ORBITER}) \ -v \ 5 \ \$

> \ 
> \define F \ -finite_field \ -q \ 9 \ -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_q9.tex
open Surfaces_q9.pdf

surface_classify_q13:
$\$(\text{ORBITER}) \ -v \ 5 \ \$

> \ 
> \define F \ -finite_field \ -q \ 13 \ -end \ 
> \define P \ -projective_space \ -n \ 3 \ -field \ F \ -v \ 0 \ -end \ 
> \with P \ -do \ 
> \ -projective_space_activity \
7391 \triangleright \triangleright \triangleright -classify_surfaces_with_double_sixes C -W -end \\
7392 \triangleright \triangleright -end \\
7393 \triangleright \triangleright -with C -do \\
7394 \triangleright \triangleright -classification_of_cubic_surfaces_with_double_sixes_activity \\
7395 \triangleright \triangleright \triangleright -report -end \\
7396 \triangleright \triangleright -end \\
7397 \triangleright \triangleright -print_symbols \\
7398 \triangleright pdfflatex Surfaces_q13.tex \\
7399 \triangleright open Surfaces_q13.pdf \\
7400 \\
7401 \\
7402 \\
7403 ###########################################################################
7404 # Section 7.4: Cubic Surfaces - Isomorphism Testing and Recognition 
7405 
7406 
7407 SECTION_CUBIC_SURFACES_ISOMORPHISM_TESTING_AND_RECOGNITION: 
7408 
7409 
7410 surface_recognize_q7_abcd_2_3_3_4: 
7411 \triangleright $(ORBITER) \ -v 3 \ \\
7412 \triangleright \triangleright -define F -finite_field -q 7 -end \ \\
7413 \triangleright \triangleright -define P -projective_space -n 3 -field F -v 0 -end \ \\
7414 \triangleright \triangleright -with P -do \ \\
7415 \triangleright \triangleright -projective_space_activity \ \\
7416 \triangleright \triangleright \triangleright -classify_surfaces_with_double_sixes Surf -W -end \ \\
7417 \triangleright \triangleright -end \ \\
7418 \triangleright \triangleright -with Surf -do \ \\
7419 \triangleright \triangleright \triangleright -classification_of_cubic_surfaces_with_double_sixes_activity \ \\
7420 \triangleright \triangleright \triangleright -recognize \ \\
7421 \triangleright \triangleright \triangleright \triangleright -q 7 \ \\
7422 \triangleright \triangleright \triangleright \triangleright -family_general_abcd 2 3 3 4 \ \\
7423 \triangleright \triangleright \triangleright \triangleright -end \ \\
7424 \triangleright \triangleright \triangleright -end \ \\
7425 \triangleright \triangleright -end \\
7426 
7427 
7428 surface_isomorph_16: 
7429 \triangleright $(ORBITER) \ -v 3 \ \\
7430 \triangleright \triangleright -define F -finite_field -q 16 -end \ \\
7431 \triangleright \triangleright -define P -projective_space -n 3 -field F -v 0 -end \ \\
7432 \triangleright \triangleright -with P -do \ \\
7433 \triangleright \triangleright -projective_space_activity \ \\
7434 \triangleright \triangleright \triangleright -classify_surfaces_with_double_sixes Surf27 -W -end \ \\
7435 \triangleright \triangleright -end \ \\
7436 \triangleright \triangleright -with Surf27 -do \ \\
7437 \triangleright \triangleright \triangleright -classification_of_cubic_surfaces_with_double_sixes_activity \ 
667
# 1 min 8 sec on Mac from scratch (with all data files removed)

```
surface_recognize_B:
$(ORBITER) -v 3 \\
define F -finite_field -q 8 -end \\
define P -projective_space -n 3 -field F -v 0 -end \\
with P -do \\
projective_space_activity \\
classify_surfaces_with_double_sixes Surf27 -W -end \\
-end \\
-with Surf27 -do \\
-classification_of_cubic_surfaces_with_double_sixes_activity \\
-recognize \\
-q 8 \\
-end \\
-end \\
-print_symbols
```

```
surface_recognize_F13_q4:
$(ORBITER) -v 3 \\
define F -finite_field -q 4 -end \\
define P -projective_space -n 3 -field F -v 0 -end \\
with P -do \\
projective_space_activity \\
classify_surfaces_with_double_sixes Surf27 -W -end \\
-end \\
-with Surf27 -do \\
-classification_of_cubic_surfaces_with_double_sixes_activity \\
-identify_F13 \\
-end \\
-print_symbols
```

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

F_sweep_15.q7:
\$ \text{(ORBITER)} -v 20 \$
\$ \text{-define F -finite_field -q 7 -end} \$
\$ \text{-define P -projective_space -n 3 -field F -v 0 -end} \$
\$ \text{-with P -do} \$
\$ \text{-projective_space_activity} \$
\$ \text{-sweep_4_15_lines sweep_4_15_lines_q7 -q 7} \$
\$ \text{-by_equation "F_alpha_beta_gamma_delta"} \$
\$ \text{-equation "DF\{\alpha,\beta,\gamma,\delta}\D" "x0,x1,x2,x3"} \$
\$ \text{-F_ALPHA_BETA_GAMMA_DELTA} \$
\$ \text{-alpha=2,\beta=1,\gamma=2,\delta=3} \$
\$ \text{-end} \$

#0:29

# Section 7.5: Cubic Surfaces of Dickson type
orbits_cubic_surfaces_q3:

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

$\$(define G \text{ -linear_group -PGL 4 3 -end }$

$\$(define Orb \text{ -orbits -group G }$

$\$(define Orb \text{ -orbits -group G }$

$\$(define Orb \text{ -orbits -group G }$

$\$(define Orb \text{ -orbits -group G }$

$\$(define Orb \text{ -orbits -group G }$

$\$(define Orb \text{ -orbits -group G }$

# this takes 3 days and about 150 GB memory on ripoff

orbits_cubic_curves_q2_again:

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

$\$(define G \text{ -linear_group -PGL 3 2 -end }$

$\$(define Orb \text{ -orbits -group G }$

$\$(define Orb \text{ -orbits -group G }$

$\$(define Orb \text{ -orbits -group G }$

$\$(define Orb \text{ -orbits -group G }$

$\$(define Orb \text{ -orbits -group G }$

# compute and analyze properties over F2

poly_orbits_d3_n3_q2_F2.csv: poly_orbits_d3_n3_q2.csv

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

$\$(define F \text{ -finite_field -q 2 -end }$

$\$(define P \text{ -projective_space -n 3 -field F -v 0 -end }$

$\$(with P \text{ -do }$

# this takes 3 days and about 150 GB memory on ripoff
Dickson_q2_analyze: poly_orbits_d3_n3_q2_F2.csv

define F -finite_field -q 4 -end
define P -projective_space -n 3 -field F -v 0 -end
with P -do

Dickson_q4_analyze: poly_orbits_d3_n3_q2_F4.csv

define F -finite_field -q 4 -end
define P -projective_space -n 3 -field F -v 0 -end
with P -do

# compute and analyze properties over F4

poly_orbits_d3_n3_q2_F4.csv: poly_orbits_d3_n3_q2.csv

define F -finite_field -q 4 -end
define P -projective_space -n 3 -field F -v 0 -end
with P -do

# compute and analyze properties over F8
Dickson_q8.analyze: poly_orbits_d3_n3_q2_F8.csv

Dickson_q16.analyze: poly_orbits_d3_n3_q2_F16.csv

# compute and analyze properties over F16

poly_orbits_d3_n3_q2_F16.csv: poly_orbits_d3_n3_q2.csv
# 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_table_latex_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_SURFACES)

▷ $(ORBITER) -v 3 -define F -finite_field -q 7 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

▷ $(ORBITER) -v 3 -define F -finite_field -q 8 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

▷ $(ORBITER) -v 3 -define F -finite_field -q 9 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

▷ $(ORBITER) -v 3 -define F -finite_field -q 11 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

▷ $(ORBITER) -v 3 -define F -finite_field -q 13 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

▷ $(ORBITER) -v 3 -define F -finite_field -q 16 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

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

cubic_surfaces_tables_19_37:
▷ $(ORBITER) -v 3 -define F -finite_field -q 19 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)
$\$(ORBITER) -v 3 -define F -finite_field -q 23 -end $\$(MAKE_TABLE_OF_CUBIC_SURFACES)$

$\$(ORBITER) -v 3 -define F -finite_field -q 25 -end $\$(MAKE_TABLE_OF_CUBIC_SURFACES)$

$\$(ORBITER) -v 3 -define F -finite_field -q 27 -end $\$(MAKE_TABLE_OF_CUBIC_SURFACES)$

$\$(ORBITER) -v 3 -define F -finite_field -q 29 -end $\$(MAKE_TABLE_OF_CUBIC_SURFACES)$

$\$(ORBITER) -v 3 -define F -finite_field -q 31 -end $\$(MAKE_TABLE_OF_CUBIC_SURFACES)$

$\$(ORBITER) -v 3 -define F -finite_field -q 32 -end $\$(MAKE_TABLE_OF_CUBIC_SURFACES)$

$\$(ORBITER) -v 3 -define F -finite_field -q 37 -end $\$(MAKE_TABLE_OF_CUBIC_SURFACES)$

$\$(ORBITER) -v 3 -define F -finite_field -q 41 -end $\$(MAKE_TABLE_OF_CUBIC_SURFACES)$

$\$(ORBITER) -v 3 -define F -finite_field -q 43 -end $\$(MAKE_TABLE_OF_CUBIC_SURFACES)$

$\$(ORBITER) -v 3 -define F -finite_field -q 47 -end $\$(MAKE_TABLE_OF_CUBIC_SURFACES)$

$\$(ORBITER) -v 3 -define F -finite_field -q 49 -end $\$(MAKE_TABLE_OF_CUBIC_SURFACES)$

$\$(ORBITER) -v 3 -define F -finite_field -q 53 -end $\$(MAKE_TABLE_OF_CUBIC_SURFACES)$

$\$(ORBITER) -v 3 -define F -finite_field -q 59 -end $\$(MAKE_TABLE_OF_CUBIC_SURFACES)$

$\$(ORBITER) -v 3 -define F -finite_field -q 61 -end $\$(MAKE_TABLE_OF_CUBIC_SURFACES)$

$\$(ORBITER) -v 3 -define F -finite_field -q 64 -end $\$(MAKE_TABLE_OF_CUBIC_SURFACES)$

$\$(ORBITER) -v 3 -define F -finite_field -q 67 -end $\$(MAKE_TABLE_OF_CUBIC_SURFACES)$

$\$(ORBITER) -v 3 -define F -finite_field -q 71 -end $\$(MAKE_TABLE_OF_CUBIC_SURFACES)$

$\$(ORBITER) -v 3 -define F -finite_field -q 73 -end $\$(MAKE_TABLE_OF_CUBIC_SURFACES)$

$\$(ORBITER) -v 3 -define F -finite_field -q 79 -end $\$(MAKE_TABLE_OF_CUBIC_SURFACES)$

$\$(ORBITER) -v 3 -define F -finite_field -q 81 -end $\$(MAKE_TABLE_OF_CUBIC_SURFACES)$

$\$(ORBITER) -v 3 -define F -finite_field -q 83 -end $\$(MAKE_TABLE_OF_CUBIC_SURFACES)$

$\$(ORBITER) -v 3 -define F -finite_field -q 89 -end $\$(MAKE_TABLE_OF_CUBIC_SURFACES)$

cubic_surfaces_tables_41_and_up:

$\$(ORBITER) -v 3 -define F -finite_field -q 41 -end $\$(MAKE_TABLE_OF_CUBIC_SURFACES)$

$\$(ORBITER) -v 3 -define F -finite_field -q 43 -end $\$(MAKE_TABLE_OF_CUBIC_SURFACES)$

$\$(ORBITER) -v 3 -define F -finite_field -q 47 -end $\$(MAKE_TABLE_OF_CUBIC_SURFACES)$

$\$(ORBITER) -v 3 -define F -finite_field -q 49 -end $\$(MAKE_TABLE_OF_CUBIC_SURFACES)$

$\$(ORBITER) -v 3 -define F -finite_field -q 53 -end $\$(MAKE_TABLE_OF_CUBIC_SURFACES)$

$\$(ORBITER) -v 3 -define F -finite_field -q 59 -end $\$(MAKE_TABLE_OF_CUBIC_SURFACES)$

$\$(ORBITER) -v 3 -define F -finite_field -q 61 -end $\$(MAKE_TABLE_OF_CUBIC_SURFACES)$

$\$(ORBITER) -v 3 -define F -finite_field -q 64 -end $\$(MAKE_TABLE_OF_CUBIC_SURFACES)$

$\$(ORBITER) -v 3 -define F -finite_field -q 67 -end $\$(MAKE_TABLE_OF_CUBIC_SURFACES)$

$\$(ORBITER) -v 3 -define F -finite_field -q 71 -end $\$(MAKE_TABLE_OF_CUBIC_SURFACES)$

$\$(ORBITER) -v 3 -define F -finite_field -q 73 -end $\$(MAKE_TABLE_OF_CUBIC_SURFACES)$

$\$(ORBITER) -v 3 -define F -finite_field -q 79 -end $\$(MAKE_TABLE_OF_CUBIC_SURFACES)$

$\$(ORBITER) -v 3 -define F -finite_field -q 81 -end $\$(MAKE_TABLE_OF_CUBIC_SURFACES)$

$\$(ORBITER) -v 3 -define F -finite_field -q 83 -end $\$(MAKE_TABLE_OF_CUBIC_SURFACES)$

$\$(ORBITER) -v 3 -define F -finite_field -q 89 -end $\$(MAKE_TABLE_OF_CUBIC_SURFACES)$
$(ORBITER) -v 3 -define F -finite_field -q 97 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$(ORBITER) -v 3 -define F -finite_field -q 101 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$(ORBITER) -v 3 -define F -finite_field -q 103 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$(ORBITER) -v 3 -define F -finite_field -q 107 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$(ORBITER) -v 3 -define F -finite_field -q 109 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$(ORBITER) -v 3 -define F -finite_field -q 113 -end $(MAKE_TABLE_OF_CUBIC_SURFACES)

$(ORBITER) -v 3 -csv_file_latex 1 test.csv

$(ORBITER) -v 3 -csv_file_latex 0 table_of_cubic_surfaces_q4_info.csv

$(ORBITER) -v 3 -csv_file_latex 0 table_of_cubic_surfaces_q7_info.csv

$(ORBITER) -v 3 -csv_file_latex 0 table_of_cubic_surfaces_q8_info.csv

$(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 0 table_of_cubic_surfaces_q23_info.csv

$(ORBITER) -v 3 -csv_file_latex 0 table_of_cubic_surfaces_q25_info.csv

$(ORBITER) -v 3 -csv_file_latex 0 table_of_cubic_surfaces_q27_info.csv

$(ORBITER) -v 3 -csv_file_latex 0 table_of_cubic_surfaces_q29_info.csv

$(ORBITER) -v 3 -csv_file_latex 0 table_of_cubic_surfaces_q31_info.csv

$(ORBITER) -v 3 -csv_file_latex 0 table_of_cubic_surfaces_q32_info.csv

$(ORBITER) -v 3 -csv_file_latex 0 table_of_cubic_surfaces_q37_info.csv

$(ORBITER) -v 3 -csv_file_latex 0 table_of_cubic_surfaces_q41_info.csv

$(ORBITER) -v 3 -csv_file_latex 0 table_of_cubic_surfaces_q43_info.csv

#pdflatex quartic_curves_q9_info.tex

#~/bin/tth 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:

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

\$\text{\ptl surfaces_report.tex}$

\$\text{\open surfaces_report.pdf}$

surface_atlas:

\$\text{\$(ORBITER) -v 3 -create_surface_atlas 97}$

\$\text{/bin/tth surface_atlas.tex}$

surface_reports:

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

\$\text{-orbiter_path $(ORBITER\_PATH) -create_surface_reports 4,7,8,9,11}$

quartic_curve_tables\

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

\$\text{-define F -finite_field -q 19 -end \}\}$

\$\text{-define P -projective_space -n 2 -field F -v 0 -end \}\}$

\$\text{-with P -do \}\}$

\$\text{-projective_space_activity \}\}$

\$\text{-table_of_quartic_curves \}\}$

\$\text{-end}$

quartic_curve_tables_19:

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

\$\text{-define F -finite_field -q 23 -end \}\}$

\$\text{-define P -projective_space -n 2 -field F -v 0 -end \}\}$

\$\text{-with P -do \}\}$

\$\text{-projective_space_activity \}\}$

\$\text{-table_of_quartic_curves \}\}$

\$\text{-end}$

quartic_curve_tables_23:

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

\$\text{-define F -finite_field -q 23 -end \}\}$

\$\text{-define P -projective_space -n 2 -field F -v 0 -end \}\}$

\$\text{-with P -do \}\}$

\$\text{-projective_space_activity \}\}$

\$\text{-table_of_quartic_curves \}\}$

\$\text{-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

$\text{ORBITER} -v 3$
-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

$\text{ORBITER} -v 3$
-define F -finite_field -q 25 -end
-define P -projective_space -n 2 -field F -v 0 -end
-with P -do
-projective_space_activity
-table_of_quartic_curves
-end

$\text{ORBITER} -v 3$
-define F -finite_field -q 27 -end
-define P -projective_space -n 2 -field F -v 0 -end
-with P -do
-projective_space_activity

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

-define F -finite_field -q 29 -end

-define P -projective_space -n 2 -field F -v 0 -end

-with P -do

-projective_space_activity

-table_of_quartic_curves

-end

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

-define F -finite_field -q 31 -end

-define P -projective_space -n 2 -field F -v 0 -end

-with P -do

-projective_space_activity

-table_of_quartic_curves

-end

quartic_curve_tables_latex:

#$\text{(ORBITER)} -v 3 -csv_file_latex 1 test.csv$

#$\text{(ORBITER)} -v 3 -csv_file_latex 0 quartic_curves_q9_info.csv$

#$\text{(ORBITER)} -v 3 -csv_file_latex 0 quartic_curves_q13_info.csv$

#$\text{(ORBITER)} -v 3 -csv_file_latex 0 quartic_curves_q17_info.csv$

#$\text{(ORBITER)} -v 3 -csv_file_latex 0 quartic_curves_q19_info.csv$

#$\text{(ORBITER)} -v 3 -csv_file_latex 0 quartic_curves_q23_info.csv$

#$\text{(ORBITER)} -v 3 -csv_file_latex 0 quartic_curves_q25_info.csv$

#$\text{(ORBITER)} -v 3 -csv_file_latex 0 quartic_curves_q27_info.csv$

#$\text{(ORBITER)} -v 3 -csv_file_latex 0 quartic_curves_q29_info.csv$

#$\text{(ORBITER)} -v 3 -csv_file_latex 0 quartic_curves_q31_info.csv$

#$\text{(ORBITER)} -v 3 -csv_file_latex 1 quartic_curves_q9_info.csv$

#pdflatex quartic_curves_q13_info.tex

#open quartic_curves_q13_info.pdf

#~/bin/tth quartic_curves_q13_info.tex

#open quartic_curves_q13_info.html

# 9#0 has K=63 and ago=12096

# 23#40 has K=21 and ago=168
# Chapter 8 - Ring Theory

# Section 8.1: Polynomials over Finite Fields

SECTION_POLYNOMIALS:

# check which polynomials are irreducible and which are primitive:

sift_polynomials_deg3_q2:

$\texttt{(ORBITER) -v 2 \%}$

-define F -finite_field -q 2 -end

-with F -do

-finite_field_activity -sift_polynomials 8 16 -end

sift_polynomials_deg4_q2:

$\texttt{(ORBITER) -v 2 \%}$

-define F -finite_field -q 2 -end

-with F -do

-finite_field_activity -sift_polynomials 16 32 -end

poly_division:

$\texttt{(ORBITER) -v 2 \%}$

-define F -finite_field -q 2 -end

-with F -do

-finite_field_activity

-polynomial_division "1,0,0,0,0,0,0,0,0,1" "1,0,1,1" -end

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

poly_mult_mod1:
\$(ORBITER) -v 2 \
\$define F -finite_field -q 7 -end 
\$with F -do \
\$finite_field_activity \
\$polynomial_mult_mod "1,2,3" "3,4,5" "6,0,0,1" -end

poly_mult_mod2:
\$(ORBITER) -v 2 \
\$define F -finite_field -q 7 -end 
\$with F -do \
\$finite_field_activity \
\$polynomial_mult_mod "3,1,2" "5,3,4" "6,0,0,1" -end

poly_mult_mod_F4:
\$(ORBITER) -v 2 \
\$define F -finite_field -q 2 -end 
\$with F -do \
\$finite_field_activity \
\$polynomial_mult_mod "1,1" "1,1" "1,1,1" -end

poly_mult_mod_mod:
\$(ORBITER) -v 2 \
\$define F -finite_field -q 2 -end 
\$with F -do \
\$finite_field_activity \
\$polynomial_mult_mod "0,1" "1,1" "1,1,1" -end

\$(ORBITER) -v 2 \
\$define F -finite_field -q 2 -end 
\$with F -do \
\$finite_field_activity \
\$polynomial_mult_mod "0,1" "1,1" "1,1,1" -end
-polynomial_mult_mod "0,1" "0,1" "1,1,1" -end

mult_polynomials_2_5_7:
$$(ORBITER) -v 2 \$
-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 \$
-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:
$$(ORBITER) -v 2 \$
-define F -finite_field -q 2 -end \n-with F -do \n-finite_field_activity -mult_polynomials 8 15 -end
pdflatex polynomial_mult_8_15.tex
open polynomial_mult_8_15.pdf

polynomial_division_ranked_2_120_25:
$$(ORBITER) -v 2 \$
-define F -finite_field -q 2 -end \n-with F -do \n-finite_field_activity \n-polynomial_division_ranked 120 25 \n-end
pdflatex polynomial_division_120_25.tex
open polynomial_division_120_25.pdf

# the answer is 5
mult_polynomials_2.7_7:
-define F -finite_field -q 2 -end
-with F -do
-finite_field_activity
-mult_polynomials 7 7 -end
pdflatex polynomial_mult_2_7_7.tex
open polynomial_mult_2_7_7.pdf

mult_polynomials_2.4_6:
-define F -finite_field -q 2 -end
-with F -do
-finite_field_activity
-mult_polynomials 4 6 -end
pdflatex polynomial_mult_2_4_6.tex
open polynomial_mult_2_4_6.pdf

polynomial_division_ranked_2.24_13:
-define F -finite_field -q 2 -end
-with F -do
-finite_field_activity
-polynomial_division_ranked 24 13 -end
pdflatex polynomial_division_24_13.tex
open polynomial_division_24_13.pdf

mult_polynomials_1.024.999_997:
-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

polynomial_division_ranked_2.349147_1033:
-define F -finite_field -q 2 -end
mult_polynomials_1024_999_997_check:

$\text{ORBITER} -v 3$

$\text{define } F \text{ -finite field } -q 1024 \text{ -end}$

$\text{with } F \text{ -do}$

$\text{finite field activity -parse and evaluate}$

"test" ""a*b" "a=999, b=997" -end

# evaluates to 61

mult_polynomials_17_12:

$\text{ORBITER} -v 2$

$\text{define } F \text{ -finite field } -q 2 \text{ -end}$

$\text{with } F \text{ -do}$

$\text{finite field activity}$

$\text{mult polynomials 17 12 -end}$

$\text{pdflatex polynomial_mult_17_12.tex}$

$\text{open polynomial_mult_17_12.pdf}$

# gives 204

polynomial_division_ranked_2_204_37:

$\text{ORBITER} -v 2$

$\text{define } F \text{ -finite field } -q 2 \text{ -end}$

$\text{with } F \text{ -do}$

$\text{finite field activity}$

$\text{-polynomial division ranked 204 37 -end}$

$\text{pdflatex polynomial_division_204_37.tex}$

$\text{open polynomial_division_204_37.pdf}$

# answer is 18
test_crc32:
  $(ORBITER) -v 3 -crc32 "123456789"

Berlekamp_matrix_crc32:
  $(ORBITER) -v 2
  -define F -finite_field -q 2 -end
  -define v -vector -field F -sparse 33 $(CRC32_SPARSE) -end
  -with F -do
  -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
  -Berlekamp_matrix v -end

# N = 2^32-1 = 3 * 5 * 17 * 257 * 65537
# N / 3 = 1431655765
# N / 5 = 858993459
# N / 17 = 252645135
# N / 257 = 16711935
# N / 65537 = 65535

TWO_TO_THE_32_MINUS_2=4294967294

power_mod_inverse:
  $(ORBITER) -v 2
  -define F -finite_field -q 2 -end
  -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

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

mult_mod_to_get_one:
  $(ORBITER) -v 2
  -define F -finite_field -q 2 -end
  -define M -vector -field F -sparse 33 $(CRC32_SPARSE) -end

684
```plaintext
8197  ▶ ▶ -define A -vector -field F -sparse 2 "1,1" -end \n8198  ▶ ▶ -define B -vector -field F -sparse 33 $(INVERSE SPARSE) -end \n8199  ▶ ▶ -with F -do \n8200  ▶ ▶  -finite_field_activity \n8201  ▶ ▶  -polynomial_mult_mod A B M \n8202  ▶ ▶  -end
8203
8204  #C(X)=1
8205
8206
8207
8208
8209
8210
8211
8212  Berlekamp_matrix_2_3:
8213  ▶ $(ORBITER) -v 2 \n8214  ▶ ▶ -define F -finite_field -q 2 -end \n8215  ▶ ▶ -define v -vector -field F -dense "1,1,0,1" -end \n8216  ▶ ▶ -with F -do \n8217  ▶ ▶  -finite_field_activity \n8218  ▶ ▶  -Berlekamp_matrix v -end
8219
8220  # the polynomial X^3+X+1 is irreducible over GF(2) because the rank of the Berlekamp matrix is 2.
8221
8222  Berlekamp_matrix_2_4:
8223  ▶ $(ORBITER) -v 2 \n8224  ▶ ▶ -define F -finite_field -q 2 -end \n8225  ▶ ▶ -define v -vector -field F -dense "1,1,0,1" -end \n8226  ▶ ▶ -with F -do \n8227  ▶ ▶  -finite_field_activity \n8228  ▶ ▶  -Berlekamp_matrix v -end
8229
8230  # the polynomial X^4+X+1 is irreducible over GF(2) because the rank of the Berlekamp matrix is 3.
8231
8232
8233  Berlekamp_matrix_4_3a:
8234  ▶ $(ORBITER) -v 2 \n8235  ▶ ▶ -define F -finite_field -q 4 -end \n8236  ▶ ▶ -define v -vector -field F -dense "1,3,0,1" -end \n8237  ▶ ▶ -with F -do \n8238  ▶ ▶  -finite_field_activity \n8239  ▶ ▶  -Berlekamp_matrix v -end
8240
8241  Berlekamp_matrix_4_3b:
```
find roots a:

find roots b:

find roots c:

find roots d:

find roots e:
roots_over_F2:
$\text{(ORBITER)} -v 2$
$\text{define } F \text{-finite_field -q 2 -end}$
$\text{define } v \text{-vector -field } F \text{-dense } "0,1,0,1,1,1" \text{-end}$
$\text{with } F -do$
$\text{-finite_field_activity}$
$\text{-polynomial_find_roots } v \text{-end}$

roots_over_F8:
$\text{(ORBITER)} -v 2$
$\text{define } F \text{-finite_field -q 8 -override_polynomial 11 -end}$
$\text{define } v \text{-vector -field } F \text{-dense } "0,1,0,1,1,1" \text{-end}$
$\text{with } F -do$
$\text{-finite_field_activity}$
$\text{-polynomial_find_roots } v \text{-end}$

# degree and then order of the field of coefficients:

irred_3_2:
$\text{(ORBITER)} -v 3$
$\text{define } F \text{-finite_field -q 2 -end}$
$\text{with } F -do$
$\text{-finite_field_activity}$
$\text{-make_table_of_irreducible_polynomials 3 -end}$
$\text{pdflatex Irred_q2_d3.tex}$
$\text{open Irred_q2_d3.pdf}$

irred_4_2:
$\text{(ORBITER)} -v 3$
$\text{define } F \text{-finite_field -q 2 -end}$
$\text{with } F -do$
$\text{-finite_field_activity}$
$\text{-make_table_of_irreducible_polynomials 4 -end}$
$\text{pdflatex Irred_q2_d4.tex}$
$\text{open Irred_q2_d4.pdf}$

# 3 polys
irred_5_2:
$\text{ORBITER} -v 3 \$
-define F -finite_field -q 2 -end \n-with F -do \n-finite_field_activity \n-make_table_of_irreducible_polynomials 5 -end
pdflatex Irred_q2_d5.tex
open Irred_q2_d5.pdf
# 6 polys

irred_6_2:
$\text{ORBITER} -v 3 \$
-define F -finite_field -q 2 -end \n-with F -do \n-finite_field_activity \n-make_table_of_irreducible_polynomials 6 -end
pdflatex Irred_q2_d6.tex
open Irred_q2_d6.pdf
# 9 polys

irred_7_2:
$\text{ORBITER} -v 3 \$
-define F -finite_field -q 2 -end \n-with F -do \n-finite_field_activity \n-make_table_of_irreducible_polynomials 7 -end
pdflatex Irred_q2_d7.tex
open Irred_q2_d7.pdf
# 18 polys

irred_8_2:
$\text{ORBITER} -v 3 \$
-define F -finite_field -q 2 -end \n-with F -do \n-finite_field_activity \n-make_table_of_irreducible_polynomials 8 -end
pdflatex Irred_q2_d8.tex
open Irred_q2_d8.pdf
# 30 polys

irred_9_2:
$\text{ORBITER} -v 3 \$
define F -finite_field -q 2 -end \
with F -do \
FINITE_field_activity \
finite_field -activity \
make_table_of_irreducible_polynomials 9 -end
pdflatex Irred_q2_d9.tex
open Irred_q2_d9.pdf

# 56 polys

irred_10_2:
$\$(ORBITER) -v 3 \
define F -finite_field -q 2 -end \
with F -do \
finite_field_activity \
make_table_of_irreducible_polynomials 10 -end
pdflatex Irred_q2_d10.tex
open Irred_q2_d10.pdf

# 99 polys

irred_2_4:
$\$(ORBITER) -v 3 \
define F -finite_field -q 4 -end \
with F -do \
finite_field_activity \
make_table_of_irreducible_polynomials 2 -end
pdflatex Irred_q4_d2.tex
open Irred_q4_d2.pdf

# 6 polys

irred_3_4:
$\$(ORBITER) -v 6 \
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

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

# stuck in factoring 2^61-1 (which is prime)

search_primitive_poly_3:
 $$(ORBITER) -v 6 \$
 $$(ORBITER) -v 6 \$
 -search_for_primitive_polynomial_in_range 3 3 2 60

search_primitive_poly_4:
 $$(ORBITER) -v 6 \$
 $$(ORBITER) -v 6 \$
 -search_for_primitive_polynomial_in_range 4 4 2 30

searchPrimitivePoly_5:
 $$(ORBITER) -v 6 \$
 $$(ORBITER) -v 6 \$
 -search_for_primitive_polynomial_in_range 5 5 2 30

search_primitive_poly_7:
 $$(ORBITER) -v 6 \$
 $$(ORBITER) -v 6 \$
 -search_for_primitive_polynomial_in_range 7 7 2 20

search_primitive_poly_8:
 $$(ORBITER) -v 6 \$
 $$(ORBITER) -v 6 \$
 -search_for_primitive_polynomial_in_range 8 8 2 20

search_primitive_poly_9:
 $$(ORBITER) -v 6 \$
 $$(ORBITER) -v 6 \$
 -search_for_primitive_polynomial_in_range 9 9 2 15

search_primitive_poly_11:
 $$(ORBITER) -v 6 \$
 $$(ORBITER) -v 6 \$
 -search_for_primitive_polynomial_in_range 11 11 2 15

search_primitive_poly_13:
 $$(ORBITER) -v 6 \$
 $$(ORBITER) -v 6 \$
 -search_for_primitive_polynomial_in_range 13 13 2 15

search_primitive_poly_degree_16:
 $$(ORBITER) -v 6 \$
 $$(ORBITER) -v 6 \$
 -search_for_primitive_polynomial_in_range 2 2 16 16

search Primitive_poly_32:
ORBITER -v 6 \
-define F -finite_field -q 13 -end \
-define P -projective_space -n 2 -field F -v 0 -end \
-define R -polynomial_ring \
-define F -field F \
-define Y0 -formula "y0" "y_0" "y0,y_1,y_2" \n-define Y1 -formula "y1" "y_1" "y1,y_2" \
$(CREMONA_MAP_Y0) \n$(CREMONA_MAP_Y1) \n$(CREMONA_MAP_Y2) \n$(CREMONA_MAP_Y3)
\begin{verbatim}
8524  \>$y1$  $y_1$  "$y0,y1,y2" \$
8525  \>$\$(CREMONA\_MAP\_Y1) \$
8526  \>define Y2 -formula \$
8527  \>$y2$  "$y_2$  "$y0,y1,y2" \$
8528  \>$\$(CREMONA\_MAP\_Y2) \$
8529  \>define Cremona -collection "Y0,Y1,Y2" \$
8530  \>with P -do \$
8531  \>projective\_space\_activity \$
8532  \>map R Cremona \$
8533  \>end
8534
8535
8536
8537
8538  arcs_5_2_q11:
8539  > $(ORBITER) -v 4 \$
8540  > define F -finite\_field -q 11 -end \$
8541  > define P -projective\_space -n 2 -field F -v 0 -end \$
8542  > with P -do \$
8543  > projective\_space\_activity \$
8544  > classify\_arcs \$
8545  > classify\_arcs\_classification\_control \$
8546  > problem\_label arcs_5_2_q11 \$
8547  > W -depth 5 \$
8548  > report -end \$
8549  > end \$
8550  > target\_size 5 \$
8551  > d 2 \$
8552  > end \$
8553  > end
8554  > pdflatex arcs_5_2_q11\_poset.tex
8555  > open arcs_5_2_q11\_poset.pdf
8556
8557
8558  # 2 orbits:
8559  # 0 1 2 3 37
8560  # 0 1 2 3 49
8561
8562
8563  arcs_5_2_q11\_ideal:
8564  > $(ORBITER) -v 2 \$
8565  > define F -finite\_field -q 11 -end \$
8566  > define R -polynomial\_ring \$
8567  > field F \$
8568  > number\_of\_variables 3 \$
8569  > homogeneous\_of\_degree 2 \$
8570  > monomial\_ordering\_lex \$
\end{verbatim}
-variables "x0,x1,x2" "x_0,x_1,x_2"
-end \ 

define C -combinatorial_objects \ 
-file_of_points arcs_5.2_q11_lvl_5 \ 
-end \ 
-with C -do \ 
-combinatorial_object_activity \ 
-ideal R \ 
-end \ 

#( 0, 1, 2, 3, 37 )
generator 0 / 1 is 7*x0*x1 + 5*x0*x2 + 10*x1*x2
We found 12 points on the generator of the ideal
They are : ( 0, 1, 2, 3, 37, 54, 74, 80, 93, 105, 121, 128 )

#( 0, 1, 2, 3, 49 )
generator 0 / 1 is 4*x0*x1 + 8*x0*x2 + 10*x1*x2
looping over all generators of the ideal:
generator 0 / 1 is ( 0, 4, 8, 0, 10, 0 ):
We found 12 points on the generator of the ideal
They are : ( 0, 1, 2, 3, 41, 49, 58, 77, 83, 95, 109, 130 )

PTS_OF_SURFACE_ORBIT211_Q3_L9_E4="\ 
0,1,2,5,7,8,10,14,9,12, \ 
15,3,16,37,31,34,20,19,17,32,36,33"

surface_9lines_4E_ideal:

$(ORBTER) -v 2 \ 
(define Pts -vector -dense \ 
$(PTS_OF_SURFACE_ORBIT211_Q3_L9_E4) \ 
-end \ 
(define F -finite_field -q 3 -end \ 
(define R -polynomial_ring \ 
-field F \ 
-number_of_variables 4 \ 
-homogeneous_of_degree 3 \ 
-monomial_ordering_lex \ 
-variables "x0,x1,x2,x3" "x_0,x_1,x_2,x_3" \ 
-end \ 
-with R -do \ 
-ring_theoretic_activity \ 
-ideal "surf_eqn" "surf\_eqn" Pts \ 
-end \ 

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 2 2 0
#x0*x0*x1 + 2*x0*x1*x1 + 2*x0*x1*x3
#2*x2*x2*x3 + 2*x2*x3*x3

SURFACE F_9="x0*x0*x1 - x0*x1*x1 -x0*x1*x3 -x2*x2*x3 - x2*x3*x3"

F_9.q7:

$\text{ORBITER} -v 3$
$\text{-define F -finite_field -q 7 -end}$
$\text{-define P -projective_space -n 3 -field F -v 0 -end}$
$\text{-define F_9 -cubic_surface -space P}$
$\text{-by_equation "F_9"}$
$\text{"DF_9\"} "x0,x1,x2,x3"$
$\text{\$(SURFACE_F_9) \"D\$}$
$\text{\"Dno parameters\"}$
$\text{-end}$
$\text{-with F_9 -do}$
$\text{-cubic_surface_activity}$
$\text{-report}$
$\text{-end}$

pdflatex surface_equation_F_9.q7_report.tex
open surface_equation_F_9.q7_report.pdf

# we create 20 5-subsets of PG(2,11) at random. Note that PG(2,11) has 133 points
random_k_subsets_PG_2_11:

$\text{ORBITER} -v 4$
$\text{-create_random_k_subsets 133 5 20}$

#random_k_subsets_n133_k5_nb20.csv

# We compute the line intersections:

line_type_in_PG_2_11:

$\text{ORBITER} -v 3$
8664 ▷ ▷ -orbiter_path $(ORBITER_PATH) \ 
8665 ▷ ▷ -define F -finite_field -q 11 -end \ 
8666 ▷ ▷ -define P -projective_space -n 2 -field F -v 0 -end \ 
8667 ▷ ▷ -define C -combinatorial_objects \ 
8668 ▷ ▷ ▷ -file_of_points random_k_subsets_n133_k5_nb20.csv \ 
8669 ▷ ▷ ▷ -end \ 
8670 ▷ ▷ ▷ -with C -do \ 
8671 ▷ ▷ ▷ -combinatorial_object_activity \ 
8672 ▷ ▷ ▷ ▷ -line_type P random_sets \ 
8673  
8674  
8675 # the second one is an arc: 3,33,40,83,102  
8676  
8677 # we compute the ideal:  
8678  
8679  
8680 random_arc_5_2_q11_ideal:  
8681 ▷ $(ORBITER) -v 2 \ 
8682 ▷ ▷ -define F -finite_field -q 11 -end \ 
8683 ▷ ▷ -define R -polynomial_ring \ 
8684 ▷ ▷ ▷ -field F \ 
8685 ▷ ▷ ▷ ▷ -number_of_variables 3 \ 
8686 ▷ ▷ ▷ ▷ -homogeneous_of_degree 2 \ 
8687 ▷ ▷ ▷ ▷ -monomial_ordering_lex \ 
8688 ▷ ▷ ▷ ▷ -variables "x0,x1,x2" "x_0,x_1,x_2" \ 
8689 ▷ ▷ ▷ ▷ -end \ 
8690 ▷ ▷ ▷ ▷ -define C -combinatorial_objects \ 
8691 ▷ ▷ ▷ ▷ ▷ -set_of_points "3,33,40,83,102" \ 
8692 ▷ ▷ ▷ ▷ ▷ -end \ 
8693 ▷ ▷ ▷ ▷ -with C -do \ 
8694 ▷ ▷ ▷ ▷ -combinatorial_object_activity \ 
8695 ▷ ▷ ▷ ▷ ▷ -ideal R \ 
8696 ▷ ▷ ▷ ▷ -end \ 
8697  
8698 #generator 0 / 1 is 10*x0*x0 + 3*x0*x1 + 8*x0*x2 + 2*x1*x1 + 10*x2*x2  
8699 #We found 12 points on the generator of the ideal  
8700 #They are : ( 3, 15, 19, 33, 40, 42, 46, 50, 83, 88, 102, 108 )  
8701  
8702  
8703  
8704  
8705 Endrass_F7.txt:  
8706 ▷ $(ORBITER) -v 2 \ 
8707 ▷ ▷ -define F -finite_field -q 7 -end \ 
8708 ▷ ▷ -define R -polynomial_ring -field F \ 
8709 ▷ ▷ ▷ -number_of_variables 4 \ 
8710 ▷ ▷ ▷ -homogeneous_of_degree 8 \  
695
# we created a set of 33 points, called Endrass_F7.txt

octic_prepare:

#Found 165 solutions with 210 backtrack steps

# 165=binomial(11,3)
# Section 9.1: Number Theory

SECTION_NUMBER_THEORY:

inverse_mod_26_99:

 inverse_mod_26_99:

 inverse_mod_a:

 inverse_mod_a:

 jacobi_35_41:

 jacobi_33_41:

 jacobi_a:

 jacobi_5_19:

 sqrt_mod_7817:

 sqrt_mod_7817:

 # Section 9.2: Representation Theory

SECTION_REPRESENTATION_THEORY:
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} \$

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{"0,1,0,0,0,1,1,0,0,0,1,2,0,0,2,0"} \$
$\text{end} \$
$\text{with G -do} \$
$\text{group_theoretic_activity} \$
$\text{representation_on_polynomials 3} \$
$\text{end} \$
$\text{loop L 0 2 1 -draw_matrix} \$
$\text{input_csv_file GL_3_3_Subgroup_tetra_12_rep_3_%L.csv} \$
$\text{box.width 40 -bit_depth 24 -partition 3 10 10 -end -end_loop} \$
$\text{open GL_3_3_Subgroup_tetra_12_rep_3_0.draw.bmp} \$
$\text{open GL_3_3_Subgroup_tetra_12_rep_3_1.draw.bmp} \$
$\text{write GL_3_3_Subgroup_tetra_12_rep_3_0.csv} \$

# write GL_3_3_Subgroup_tetra_12_rep_3_0.csv

# Section 9.3: Cryptography
SECTION_CRYPTOGRAPHY:

EC_add:
$\text{(ORBITER)} -v 2 \$

698
-define F -finite_field -q 11 -end \
-define F -finite_field -q 11 -end \n
-EC.add 1 3 "1,4" "1,4" -end

EC_cyclic_subgroup:

$(ORBITER) -v 2 \
$(ORBITER) -v 2 -draw_matrix \n
-EC.cyclic_subgroup 1 3 "1,4" -end

EC_points.13:

$(ORBITER) -v 2 \
-EC.points "EC.2.5.q13" 2 5 -end

EC_points.199:

$(ORBITER) -v 2 \
-EC.points "EC.5.7.q199" 5 7 -end

EC_Koblitz_encoding:

$(ORBITER) -v 6 -seed 17 \
$(ORBITER) -v 6 -seed 17 -end

EC_bsgs:
$\text{(ORBITER)} -v 2 \\$
-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

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

NTRU_N=7
NTRU_P=3
NTRU_Q=41
NTRU_D=2
NTRU_XN1="-1,0,0,0,0,0,0,1,"
# D + 1 plus ones and D minus ones
ALICE_PRIVATE_F="-1,0,1,1,-1,0,1"
# D plus ones and D minus ones
ALICE_PRIVATE_G="0,-1,-1,0,1,0,1"
NTRU_Alice1:
$\text{(ORBITER)} -v 2 \\$
-define F -finite_field -q $\text{NTRU_Q} -$end \\
-with F -do \\
-finite_field_activity \\
-extended_gcd_for_polynomials \

#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"

#F
p(x) = x^6 + 2x^5 + x^3 + x^2 + x + 1
ALICE_PUBLIC_KEY="1,1,1,0,2,1"

BOB_MESSAGE="1,-1,1,0,-1"
BOB_ONE_TIME_KEY="-1,1,0,0,0,-1,1"

BOB_ENCRYPT= "25,3,40,2,4,19,31"
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
#C(X)=X^{6} + 10X^{5} + 33X^{4} + 40X^{3} + 40X^{2} + X + 40
ALICE_C1="40,1,40,40,33,10,1"
```

NTRU decrypt2:

```
$ (ORBITER) -v 2 
$define F -finite_field -q $(NTRU_Q) -end 
$with F -do 
-finite_field_activity 
-polynomial_center_lift $(ALICE_C1) -end
#A(X)=X^{6} + 10X^{5} - 8X^{4} - X^{3} - X^{2} + X - 1
ALICE_C2="-1,1,-1,-1,-8,10,1"
```

NTRU decrypt3:

```
$ (ORBITER) -v 2 
$define F -finite_field -q $(NTRU_P) -end 
$with F -do 
-finite_field_activity 
-polynomial_reduce_mod_p $(ALICE_C2) -end
#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 \\
\( ORBITER \) -define F finite_field -q \( \text{NTRU}_P \) -end \\
\( ORBITER \) -with F -do \\
\( ORBITER \) -finite_field_activity \\
\( ORBITER \) -polynomial_center_lift \( \text{ALICE}_C^4 \) -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 \\
\( ORBITER \) -RSA 59 10403 2 "1921,1605,1804,2116,0518"

RSA_d:

\( ORBITER \) -v 2 \\
\( ORBITER \) -RSA 2939 10403 2 "902,3509,9833,3548,5181"

im1:

\( ORBITER \) -v 2 -inverse_mod 869 1843488

the inverse of 869 mod 1843488 is 386093

# FUNFACTOR:

RSA_e1:

\( ORBITER \) -v 2 \\
\( ORBITER \) -RSA 386093 1846303 3 "62114,60103,201518"

RSA_d1:

\( ORBITER \) -v 2 \\
\( ORBITER \) -RSA 869 1846303 3 "1248407,345776,317846"
im1061:

\[(\text{ORBITER}) \ -v 2 \ -inverse \mod 1061 25320204\]

# the inverse of 1061 mod 25320204 is 2076209

RSA

e2:

\[(\text{ORBITER}) \ -v 2 \ -RSA \ \text{encrypt} \ \text{text} 2076209 25330309 3 \ \text{creamcheese}\]

\#-RSA \ \text{encrypt} \ \text{text} 386093 1846303 \ \text{creamcheese}

#408918,1735142,239809,654636

RSA

d2:

\[(\text{ORBITER}) \ -v 2 \ -RSA 1061 25330309 3 "19019931,1619805,740498,2671344"\]

#7253*8171 = 59264263

# 7252*8170 = 59248840

im3:

\[(\text{ORBITER}) \ -v 2 \ -inverse \mod 2909 59248840\]

#the inverse of 2909 mod 59248840 is 4358629

RSA

e3:

\[(\text{ORBITER}) \ -v 2 \ -RSA \ 2909 59264263 3 \ \text{encrypted}\]

RSA

d3:

\[(\text{ORBITER}) \ -v 2 \ -RSA 4358629 59264263 3 "35270141,9642524,49091707"\]
#51403, 182516, 200504 = encrypted

###

# 7879 * 7901 = 62251979
# 7878 * 7900 = 62236200

e =

im4:

\[ (ORBITER) -v 2 -inverse \mod 583 62236200 \]

# the inverse of 583 mod 62236200 is 32559247

RSA
e4:

\[ (ORBITER) -v 2 \-RSA \] encrypt text 583 62251979 venividivici

#-RSA encrypt text 583 62251979 venividivici
#40513610, 53979973, 56449676, 35068535

RSA
d4:

\[ (ORBITER) -v 2 \-RSA \] encrypt text 38543669 52518863 3 fascinating

#31526751, 8962078, 51045732, 51894467

 RSA
e5:

\[ (ORBITER) -v 2 \-RSA \] encrypt text 38543669 52518863 fascinating

#31526751, 8962078, 51045732, 51894467

RSA
d5:

\[ (ORBITER) -v 2 \]
9180  ▶  ▶  -RSA 173 52518863 "31526751,8962078,51045732,51894467"
9181
9182
9183  RSA_d6:
9184  ▶  ▶  $(ORBITER) -v 2 \
9185  ▶  ▶  -RSA 47177497 55040413 "28702119,48926559"
9186
9187
9188  smooth:
9189  ▶  ▶  $(ORBITER) -v 2 \
9190  ▶  ▶  -sift_smooth 100000 100 "2,3,5,7,11,13,17,19"
9191
9192
9193  ▶  ▶
9194  ######
9195  # 1999 * 7907 = 15806093
9196  # 1998 * 7906 = 15796188
9197
9198  im7:
9199  ▶  ▶  $(ORBITER) -v 2 -inverse_mod 3221 15796188
9200
9201  #the inverse of 3221 mod 15796188 is 10048553
9202
9203
9204  RSA_e7:
9205  ▶  ▶  $(ORBITER) -v 2 \
9206  ▶  ▶  -RSA_encrypt_text 10048553 15806093 3 beachandfun
9207  ▶  ▶
9208  #
9209  # 7853 * 7673 = 60256069
9210  # 7852 * 7672 = 60240544
9211
9212  im8:
9213  ▶  ▶  $(ORBITER) -v 2 -inverse_mod 9017 60240544
9214
9215
9216  #the inverse of 9017 mod 60240544 is 14430473
9217
9218  RSA_e8:
9219  ▶  ▶  $(ORBITER) -v 2 \
9220  ▶  ▶  -RSA_encrypt_text 9017 60256069 3 strawberry
9221
9222
9223  sqrt_big:
9224  ▶  ▶  $(ORBITER) -v 2 -square_root 1002001
9225  ▶
9226  sqrt_mod_33_41:
\$\text{ORBITER} -v 2 -\text{square}\_\text{root\_mod} 33 \ 41$

\$\text{ORBITER} -v 5 -\text{quadratic}\_\text{sieve} 31 \ 500 \ 1$

```
# 4460190157
```

```
# product is 284625399616057168619
```

```
# 63814633367
```

```
 pseudoprime3:
```

```
 pseudoprime10:
```

```
 pseudoprime11:
```

```
 pseudoprime20:
```

```
\texttt{PR10:}

\texttt{pseudo prime51:}

\texttt{pseudo prime30:}

\texttt{pseudo prime31:}
pseudoprime33:
$\text{(ORBITER)} -v 5 \ \\
-\text{seed 2531011 -find pseudoprime 33 5 5 5}
\text{pdflatex pseudoprime}_{33}.\text{tex}
\text{open pseudoprime}_{33}.\text{pdf}

#371674199498295345543363004459891

pseudoprime34:
$\text{(ORBITER)} -v 5 \ \\
-\text{seed 2531011 -find pseudoprime 34 5 5 5}
\text{pdflatex pseudoprime}_{34}.\text{tex}
\text{open pseudoprime}_{34}.\text{pdf}

#371674199498295345543363004459891

pseudoprime35:
$\text{(ORBITER)} -v 5 \ \\
-\text{seed 2531011 -find pseudoprime 35 5 5 5}
\text{pdflatex pseudoprime}_{35}.\text{tex}
\text{open pseudoprime}_{35}.\text{pdf}

#81329557792505271120435930267680203

pseudoprime36:
$\text{(ORBITER)} -v 5 \ \\
-\text{seed 2531011 -find pseudoprime 36 5 5 5}
\text{pdflatex pseudoprime}_{36}.\text{tex}
\text{open pseudoprime}_{36}.\text{pdf}

#162624680891993404333363207561599139
MATH360 hw2:

```bash
$ (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
$ (ORBITER) -v 3
  -define F -finite_field -q 16 -end
  -with F -do -finite_field_activity
  -parse_and_evaluate "test" "" "a*a+b*b" "a=5,b=7" -end
```

F_256_Rijndahl:

```bash
$ (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:

```bash
$ (ORBITER) -v 3 -all.square_roots_mod_n 1075922 1549411
```

power_mod_211:

```bash
$ (ORBITER) -v 3 -power_mod_n 2 211
```

710
SECTION CODING THEORY INTRODUCTION:

Allen Gates noise 1 percent:

Hamming space 4 2 distance matrix:
Hamming space 4, 2 distance matrix draw:

```
$(ORBITER) -v 2 -draw_matrix \
  -input_csv_file Hamming_n4_q2.csv \
  -box_width 20 -bit_depth 24 \
  -partition 4 16 16 \n  -end
open Hamming_n4_q2.draw.bmp
```

Hamming code macwilliams:

```
$(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:
$(ORBITER) -v 2 \
  -define F -finite_field -q 2 -end \n  -with F -do -coding_theoretic_activity \n  -code_diagram "code_5_2_3" \n  -metric_balls 5 \n  -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 \n  -end
```

Hamming 5, 2 graph:
```
$(ORBITER) -v 2 \
  -define G -graph -Hamming 5 2 -end \
  -with G -do \n  -graph_theoretic_activity -export_csv -end \n  -with G -do \n  -graph_theoretic_activity -export_graphviz -end \n  -with G -do \n  -graph_theoretic_activity -save -end
$(ORBITER) -v 2 -draw_matrix \
```
Hamming_5_2_with_5_2_3_code:

```
$\text{ORBITER} -v 2 \ \
define G -graph -Hamming 5 2 \ 
define F -finite_field -q 2 -end \ 
with F -do -coding_theoretic_activity \ 
geneneral_code_binary 6 "0,60,50,41,14,21,27,39" \ 
-automorphism_group -end \ 
pdflatex Hamming_5_2.code_5_2_3.report.tex \ 
open Hamming_5_2.code_5_2_3_report.pdf
```

group has order 32

```
code_6: \ 
$\text{ORBITER} -v 2 \ 
define F -finite_field -q 2 -end \ 
with F -do -coding_theoretic_activity \ 
general_code_binary 6 "0,60,50,41,14,21,27,39" \ 
-end \ 
$\text{ORBITER} -v 2 -draw_matrix \ 
-input_csv_file code_matrix_8_8.csv \ 
-box_width 20 -bit_depth 24 \ 
-partition 2 "1,1,1,1,1,1,1,1" "1,1,1,1,1,1,1,1" \ 
-end \ 
pdflatex code_6_8.tex \ 
open code_6_8.pdf \ 
open code_matrix_8_8.draw.bmp
```

linear code with generator matrix

```
# group has order 32

# linear code with generator matrix

111100

110010

101001

713
Section 10.2: Linear codes

SECTION CODING THEORY LINEAR CODES:

RM_3_1:

\>$\langle ORBITER \rangle -v 2$
\>$\langle ORBITER \rangle -define F -finite_field -q 2 -end$
\>$\langle ORBITER \rangle -define C -code -field F$
\>$\langle ORBITER \rangle -define C -first_order_Reed_Muller 3$
\>$\langle ORBITER \rangle -with C -and F -do -coding_theoretic_activity$
\>$\langle ORBITER \rangle -export_magma RM_3_1.magma$
\>$\langle ORBITER \rangle -end$

simplex_code:

\>$\langle ORBITER \rangle -define F -finite_field -q 2 -end$
\>$\langle ORBITER \rangle -define v -vector -field F -format 3$
\>$\langle ORBITER \rangle -dense $\langle SIMPLEX CODE GENERATOR \rangle$
\>$\langle ORBITER \rangle -end$
\>$\langle ORBITER \rangle -define C -code -field F$
\>$\langle ORBITER \rangle -linear_code_through_generator_matrix v$
\>$\langle ORBITER \rangle -end$

Hamming_generator:

\>$\langle ORBITER \rangle -define F -finite_field -q 2 -end$
\>$\langle ORBITER \rangle -define v -vector -field F -format 3$
\>$\langle ORBITER \rangle -dense $\langle SIMPLEX_CODE_GENERATOR \rangle$
\>$\langle ORBITER \rangle -end$
\>$\langle ORBITER \rangle -with F -do$
\>$\langle ORBITER \rangle -finite_field_activity$
\>$\langle ORBITER \rangle -nullspace v$
\>$\langle ORBITER \rangle -end$
\>$\langle ORBITER \rangle -pdflatex nullspace_3_7.tex$
\>$\langle ORBITER \rangle -open nullspace_3_7.pdf$
# basis in binary:
# 67,37,22,15
#-normalize from the right

Hamming_code:
$\text{(ORBITER)} \quad \text{-v} \quad 2$
\begin{verbatim}
  \text{-define F -finite_field -q 2 -end \}
\end{verbatim}
\begin{verbatim}
  \text{-define v -vector -field F -format 3 \}
\end{verbatim}
\begin{verbatim}
  \text{-dense $(SIMPLEX_CODE_GENERATOR) \}
\end{verbatim}
\begin{verbatim}
  \text{-end \}
\end{verbatim}
\begin{verbatim}
  \text{-define C -code -field F \}
\end{verbatim}
\begin{verbatim}
  \text{-linear_code_through_generator_matrix v \}
\end{verbatim}
\begin{verbatim}
  \text{-dual \}
\end{verbatim}
\begin{verbatim}
  \text{-with C -do -coding_theoretic_activity \}
\end{verbatim}
\begin{verbatim}
  \text{-export_magma Hamming.magma \}
\end{verbatim}
\begin{verbatim}
  \text{-end \}
\end{verbatim}

# writes Hamming.magma

RM_3_1_and_codewords:
$\text{(ORBITER)} \quad \text{-v} \quad 2$
\begin{verbatim}
  \text{-define F -finite_field -q 2 -end \}
\end{verbatim}
\begin{verbatim}
  \text{-define C -code -field F -first_order_Reed_Muller 3 -end \}
\end{verbatim}
\begin{verbatim}
  \text{-with C -and F -do -coding_theoretic_activity \}
\end{verbatim}
\begin{verbatim}
  \text{-export_magma RM_3_1.magma \}
\end{verbatim}
\begin{verbatim}
  \text{-end \}
\end{verbatim}
\begin{verbatim}
  \text{-with C -and F -do -coding_theoretic_activity \}
\end{verbatim}
\begin{verbatim}
  \text{-export_codewords RM_3_1_codewords.csv \}
\end{verbatim}
\begin{verbatim}
  \text{-end \}
\end{verbatim}
\begin{verbatim}
  \text{-with C -and F -do -coding_theoretic_activity \}
\end{verbatim}
\begin{verbatim}
  \text{-export_genma RM_3_1_genma.csv \}
\end{verbatim}
\begin{verbatim}
  \text{-end \}
\end{verbatim}

RM_3_1_from_generator_matrix:
$\text{(ORBITER)} \quad \text{-v} \quad 2$
\begin{verbatim}
  \text{-define F -finite_field -q 2 -end \}
\end{verbatim}
\begin{verbatim}
  \text{-define genma -vector -field F -format 8 \}
\end{verbatim}
\begin{verbatim}
  \text{compact $(CODE_RM_3_1_GENMA) \}
\end{verbatim}
\begin{verbatim}
  \text{-end \}
\end{verbatim}
to define C -code -field F \n-linear_code_through_generator_matrix genma \n-end

#pdf_latex code_n8_k4_q2.tex
#open code_n8_k4_q2.pdf

#Codewords: (0,255,170,85,204,51,102,153,240,15,90,165,60,195,150,105)

RM_4_1_and_codewords:
$\{$ORBITER\} -v 2 \n-define F -finite_field -q 2 -end \n-define C -code -field F -first_order_Reed_Muller 4 -end \n-with C -and F -do -coding_theoretic_activity \n-export_magma RM_4_1.magma \n-end \n-with C -and F -do -coding_theoretic_activity \n-export_codewords RM_4_1_codewords.csv \n-end \n-with C -and F -do -coding_theoretic_activity \n-export_genma RM_4_1_genma.csv \n-end

RM_5_1_and_codewords:
$\{$ORBITER\} -v 2 \n-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:
$\{$ORBITER\} -v 2 \n-define v -vector -dense $(HAMMING_CODE_ROWS_IN_BINARY_RANKS) -end \n-linear_code_through_basis 7 v
pdflatex code_n7_k4_q2.tex
open code_n7_k4_q2.pdf
Hamming weight enumerator:

```
9700 ▶ $(ORBITER) -v 2 
9701 ▶ ▶ -define F -finite_field -q 2 -end 
9702 ▶ ▶ -define v -vector -field F -format 4 
9703 ▶ ▶ ▶ -dense $(HAMMING_CODE_GENERATOR) 
9704 ▶ ▶ -end 
9705 ▶ ▶ -define C -code -field F 
9706 ▶ ▶ ▶ -linear_code_through_generator_matrix v 
9707 ▶ ▶ -end 
9708 ▶ ▶ -with C -do 
9709 ▶ ▶ ▶ -coding_theoretic_activity 
9710 ▶ ▶ ▶ ▶ -weightEnumerator 
9711 ▶ ▶ ▶ -end
```

Hamming minimum distance:

```
9713 ▶ $(ORBITER) -v 2 
9714 ▶ ▶ -define F -finite_field -q 2 -end 
9715 ▶ ▶ -define v -vector -field F -format 4 
9716 ▶ ▶ ▶ -dense $(HAMMING_CODE_GENERATOR) 
9717 ▶ ▶ ▶ -end 
9718 ▶ ▶ -with F -do 
9719 ▶ ▶ ▶ -coding_theoretic_activity 
9720 ▶ ▶ ▶ ▶ -minimum_distance v 
9721 ▶ ▶ ▶ ▶ -end
```

Golay23 minimum distance:

```
9724 ▶ $(ORBITER) -v 2 
9725 ▶ ▶ -define F -finite_field -q 2 -end 
9726 ▶ ▶ -define v -vector -field F -format 12 
9727 ▶ ▶ ▶ -dense $(GOLAY23_CODE_GENERATOR) 
9728 ▶ ▶ ▶ -end 
9729 ▶ ▶ ▶ -with F -do 
9730 ▶ ▶ ▶ -coding_theoretic_activity 
9731 ▶ ▶ ▶ ▶ -minimum_distance v 
9732 ▶ ▶ ▶ ▶ -end
```

Hamming code diagram:

```
9733 ▶ - $(ORBITER) -v 2 
9734 ▶ ▶ -define F -finite_field -q 2 -end 
9735 ▶ ▶ -with F -do -coding_theoretic_activity 
9736 ▶ ▶ ▶ -code_diagram "Hamming_7_4" 
9737
```

#d=7 in 0 sec
$(\text{HAMMING_CODE_CODEWORDS}) \ 7$

$-\text{metric}\text{\_balls} \ 1$

$-\text{end}$

$(\text{ORBITER}) \ -v \ 2$

$-\text{draw\_matrix}$

$-\text{input\_csv\_file Hamming\_7\_4\_diagram\_01\_7\_16.csv}$

$-\text{box\_width} \ 25 \ -\text{bit\_depth} \ 24$

$-\text{partition} \ 4 \ 16 \ 8$

$(\text{ORBITER}) \ -v \ 2$

$-\text{draw\_matrix}$

$-\text{input\_csv\_file Hamming\_7\_4\_diagram\_7\_16.csv}$

$-\text{box\_width} \ 25 \ -\text{bit\_depth} \ 14$

$-\text{partition} \ 4 \ 16 \ 8$

$(\text{ORBITER}) \ -v \ 2$

$-\text{define F} \ -\text{finite\_field} \ -q \ 2 \ -\text{with F} \ -\text{do} \ -\text{coding\_theoretic}$

$-\text{activity} \ -\text{code\_diagram} \ "\text{Hamming\_7\_4\_word\_0}" \ "0" \ 7 \ -\text{metric\_balls} \ 1 \ -\text{end}$

$(\text{ORBITER}) \ -v \ 2$

$-\text{define F} \ -\text{finite\_field} \ -q \ 2 \ -\text{with F} \ -\text{do} \ -\text{coding\_theoretic}$

$-\text{activity} \ -\text{code\_diagram} \ "\text{Hamming\_7\_4\_word\_1}" \ "67" \ 7 \ -\text{metric\_balls} \ 1 \ -\text{end}$

$(\text{ORBITER}) \ -v \ 2$

$-\text{define F} \ -\text{finite\_field} \ -q \ 2 \ -\text{with F} \ -\text{do} \ -\text{coding\_theoretic}$

$-\text{activity} \ -\text{code\_diagram} \ "\text{Hamming\_7\_4\_word\_2}" \ "37" \ 7 \ -\text{metric\_balls} \ 1 \ -\text{end}$

$(\text{ORBITER}) \ -v \ 2$

$-\text{define F} \ -\text{finite\_field} \ -q \ 2 \ -\text{with F} \ -\text{do} \ -\text{coding\_theoretic}$

$-\text{activity} \ -\text{code\_diagram} \ "\text{Hamming\_7\_4\_word\_3}" \ "102" \ 7 \ -\text{metric\_balls} \ 1 \ -\text{end}$

$(\text{ORBITER}) \ -v \ 2$

$-\text{define F} \ -\text{finite\_field} \ -q \ 2 \ -\text{with F} \ -\text{do} \ -\text{coding\_theoretic}$

$-\text{activity} \ -\text{code\_diagram} \ "\text{Hamming\_7\_4\_word\_4}" \ "22" \ 7 \ -\text{metric\_balls} \ 1 \ -\text{end}$

$(\text{ORBITER}) \ -v \ 2$

$-\text{define F} \ -\text{finite\_field} \ -q \ 2 \ -\text{with F} \ -\text{do} \ -\text{coding\_theoretic}$

$-\text{activity} \ -\text{code\_diagram} \ "\text{Hamming\_7\_4\_word\_5}" \ "85" \ 7 \ -\text{metric\_balls} \ 1 \ -\text{end}$

$(\text{ORBITER}) \ -v \ 2$

$-\text{define F} \ -\text{finite\_field} \ -q \ 2 \ -\text{with F} \ -\text{do} \ -\text{coding\_theoretic}$

$-\text{activity} \ -\text{code\_diagram} \ "\text{Hamming\_7\_4\_word\_6}" \ "51" \ 7 \ -\text{metric\_balls} \ 1 \ -\text{end}$

$(\text{ORBITER}) \ -v \ 2$

$-\text{define F} \ -\text{finite\_field} \ -q \ 2 \ -\text{with F} \ -\text{do} \ -\text{coding\_theoretic}$

$-\text{activity} \ -\text{code\_diagram} \ "\text{Hamming\_7\_4\_word\_7}" \ "112" \ 7 \ -\text{metric\_balls} \ 1 \ -\text{end}$

$(\text{ORBITER}) \ -v \ 2$

$-\text{define F} \ -\text{finite\_field} \ -q \ 2 \ -\text{with F} \ -\text{do} \ -\text{coding\_theoretic}$

$-\text{activity} \ -\text{code\_diagram} \ "\text{Hamming\_7\_4\_word\_8}" \ "15" \ 7 \ -\text{metric\_balls} \ 1 \ -\text{end}$

$(\text{ORBITER}) \ -v \ 2$

$-\text{define F} \ -\text{finite\_field} \ -q \ 2 \ -\text{with F} \ -\text{do} \ -\text{coding\_theoretic}$

$-\text{activity} \ -\text{code\_diagram} \ "\text{Hamming\_7\_4\_word\_9}" \ "76" \ 7 \ -\text{metric\_balls} \ 1 \ -\text{end}$

$(\text{ORBITER}) \ -v \ 2$

$-\text{define F} \ -\text{finite\_field} \ -q \ 2 \ -\text{with F} \ -\text{do} \ -\text{coding\_theoretic}$

$-\text{activity} \ -\text{code\_diagram} \ "\text{Hamming\_7\_4\_word\_10}" \ "42" \ 7 \ -\text{metric\_balls} \ 1 \ -\text{end}$

$(\text{ORBITER}) \ -v \ 2$

$-\text{define F} \ -\text{finite\_field} \ -q \ 2 \ -\text{with F} \ -\text{do} \ -\text{coding\_theoretic}$

$-\text{activity} \ -\text{code\_diagram} \ "\text{Hamming\_7\_4\_word\_11}" \ "105" \ 7 \ -\text{metric\_balls} \ 1 \ -\text{end}$

$(\text{ORBITER}) \ -v \ 2$

$-\text{define F} \ -\text{finite\_field} \ -q \ 2 \ -\text{with F} \ -\text{do} \ -\text{coding\_theoretic}$

$-\text{activity} \ -\text{code\_diagram} \ "\text{Hamming\_7\_4\_word\_12}" \ "25" \ 7 \ -\text{metric\_balls} \ 1 \ -\text{end}$

$(\text{ORBITER}) \ -v \ 2$

$-\text{define F} \ -\text{finite\_field} \ -q \ 2 \ -\text{with F} \ -\text{do} \ -\text{coding\_theoretic}$

$-\text{activity} \ -\text{code\_diagram} \ "\text{Hamming\_7\_4\_word\_13}" \ "90" \ 7 \ -\text{metric\_balls} \ 1 \ -\text{end}$

$(\text{ORBITER}) \ -v \ 2$

$-\text{define F} \ -\text{finite\_field} \ -q \ 2 \ -\text{with F} \ -\text{do} \ -\text{coding\_theoretic}$

$-\text{activity} \ -\text{code\_diagram} \ "\text{Hamming\_7\_4\_word\_14}" \ "60" \ 7 \ -\text{metric\_balls} \ 1 \ -\text{end}$

$(\text{ORBITER}) \ -v \ 2$

$-\text{define F} \ -\text{finite\_field} \ -q \ 2 \ -\text{with F} \ -\text{do} \ -\text{coding\_theoretic}$

$-\text{activity} \ -\text{code\_diagram} \ "\text{Hamming\_7\_4\_word\_15}" \ "127" \ 7 \ -\text{metric\_balls} \ 1 \ -\text{end}$
code_Hamming_systematic:

$\text{Hamming} \text{nullspace}$:
$(ORBITER) -v 2
$define F2 -finite_field -q 2 -end
$define v -vector -format 4 -field F2
$define $(HAMMING_CODE_GENERATOR)
$end
$with F2 -do
$finite_field_activity
$nullspace v
$normalize_from_the_right
$end
pdflatex nullspace.4_7.tex
open nullspace.4_7.pdf

#check equations of the Hamming code:
# a4+a5+a6+a7 =1+0+1+0=0 mod2 OK.
# a2+a3+a6+a7 =0+1+1+0=0 mod2 OK.
# a1+a3+a5+a7 =1+1+0+0=0 mod2 OK.
#1010101
#0110011
#0001111

Hamming_long:
$(ORBITER) -v 2 -long_code 7 4
"0,5,6"
"1,4,6"
"2,4,5"
"3,4,5,6"
$(ORBITER) -v 2 -loop L 0 16 1 -draw_matrix
"input_csv_file long_code_genma_n7_k4_codeword_%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 \
define on_points 
end
#pdflatex PGL_3_2_Singer_3_2.1_report.tex
#open PGL_3_2_Singer_3_2.1_report.pdf

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

#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 
-with F -do -finite_field_activity 
-nullspace v 
-end

pdflatex nullspace_3_7.tex
open nullspace_3_7.pdf

Hamming_cyclic_long:
$(ORBITER) -v 2 -long_code 7 4 
"0,4,6" 
"1,4,5,6" 
"2,4,5" 
"3,5,6"
$(ORBITER) -v 2 -loop L 0 16 1 -draw_matrix 
-input_csv_file long_code_genma_n7_k4_codeword_%L.csv 
-box_width 25 -bit_depth 8 -partition 3 4 2 -end 
-end_loop

Hamming_cyclic:
$(ORBITER) -v 2 
-define v -vector -dense "69,39,22,11" -end 

Hamming_cyclic_clean_ns:

```bash
$(ORBITER) -v 2 -draw_matrix \
-input_csv_file code_matrix_16_8.csv \
-box_width 25 -bit_depth 8 -partition 2 16 8 -end
open code_matrix_16_8_draw.bmp
pdflatex code_7_16.tex
open code_7_16.pdf
```

Hamming_cyclic_clean:

```bash
$(ORBITER) -v 2 \
-input_csv_file code_matrix_16_8.csv \
-box_width 25 -bit_depth 8 -partition 2 16 8 -end
open code_matrix_16_8_draw.bmp
pdflatex code_7_16.tex
open code_7_16.pdf
```

Hamming_cyclic_clean_long:

```bash
$(ORBITER) -v 2 -long_code 7 4 \
-input_csv_file code_matrix_genma_n7_k4_codeword_%L.csv \
-box_width 25 -bit_depth 8 -partition 3 4 2 -end \
```
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

Golay23_code_diagram:
10009 \$$(ORBITER) -v 2 \$
10010 \$\$ -define F -finite_field -q 2 -end \$
10011 \$\$ -with F -do \$
10012 \$\$ -coding_theoretic_activity \$
10013 \$\$ -code_diagram_from_file "Golay_23" \$
10014 \$\$ codewords_n23_k12_q2.csv 23 \$
10015 \$\$ -enhance 4 \$
10016 \$\$ -end
10017 \$\$
10018 \$\$ -metric_balls 3
10019
10020
10021 $Golay23\_code\_diagram\_draw:$
10022 \$$(ORBITER) -v 2 \$
10023 \$\$ -draw_matrix \$
10024 \$\$ -input_csv_file Golay_23\_diagram_01_23_4096.csv \$
10025 \$\$ -box_width 4 -bit_depth 8 \$
10026 \$\$ -partition 20 4096 2048 \$
10027 \$\$ -end
10028
10029
10030
10031
10032
10033 # Section 10.4: Coding Theory - CRC codes
10034
10035 10036 SECTION CODING THEORY_CRC_CODES:
10037
10038
10039
10040
10041
10042 encode_text_5bits:
10043 \$$(ORBITER) -encode_text_5bits \$
10044 \$\$ "Hithere" "text.csv" \$
10045 \$$(ORBITER) -v 2 \$
10046 \$\$ -define F -finite_field -q 2 -end \$
10047 \$\$ -with F -do \$
10048 \$\$ -coding_theoretic_activity \$
10049 \$\$ -polynomial_division_from_file \$
10050 \$\$ -text.csv 13 -end
10051 \$\$ pdflatex polynomial_division_file_13.tex
10052 \$\$ open polynomial_division_file_13.pdf
10053
10054
10055 encode_text_5bits_check:
encode_text_5bits_1error:

encode_text_5bits:

CRC_3_128_10:

crc32_test:

crc32_test_hexdata:
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  

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:  
$(ORBITER) -v 2 \  
$define F -finite_field -q 256 -end \  
$define C -code -field F \  
-BCH 771 2 \  
-end \  
-vertex 771 \  
-end \  

# degree of polynomial = 3  
#-dense "214,167,1"  
#-sparse "214,0,167,1,1,2"  

CRC_F256_BCH_code_d2.old:  
$(ORBITER) -v 3 \  
$define F -finite_field -q 256 -end \  
$with F -do -coding_theoretic_activity \  
-make_BCH_code 771 2 \  
-end \  

# pdflatex BCH_codes_q256_n771_d2.tex  

# open BCH_codes_q256_n771_d2.pdf  

decomponent = 3  

#pdflatex BCH_codes_q256_n771_d2.tex  

726
10150  #> open BCH_codes_q256_n771_d2.pdf
10151
10152
10153  CRC_POLY_Q256_DEG2_DENSE="214,167,1"
10154
10155
10156  CRC_F256_BCH_write_code_for_division_d2:
10157  $(ORBITER) -v 2 \
10158  -define F -finite_field -q 256 -end \
10159  -define A -vector -field F -sparse 772 "1,771,1,0" -end \
10160  -define B -vector -field F -dense $(CRC_POLY_Q256_DEG2_DENSE) -end \
10161  -with F -do \
10162  -coding_theoretic_activity \
10163  -write_code_for_division \
10164  alfa A B \
10165  -end
10166  g++ crcalfa.cpp -o crcalfa.out
10167  ./crcalfa.out
10168
10169
10170
10171
10172  # bravo:
10173
10174  # degree of polynomial = 4
10175  #-dense "1,23,27,213,1"
10176  #-sparse "1,0,23,1,27,2,213,3,1,4"
10177
10178  CRC_F256_BCH_code_d3:
10179  $(ORBITER) -v 2 \
10180  -define F -finite_field -q 256 -end \
10181  -define C -code -field F \
10182  -BCH 771 3 \
10183  -end \
10184  -with C -and F -do -coding_theoretic_activity \
10185  -export_magma BCH_lq8_n771_d3.magma \
10186  -end
10187  pdflatex BCH_codes_q256_n771_d3.tex
10188  open BCH_codes_q256_n771_d3.pdf
10189
10190
10191  CRC_POLY_BRAVO_DENSE="1,23,27,213,1"
10192
10193
10194  CRC_F256_BCH_write_code_for_division_Bravo:
10195  $(ORBITER) -v 2 \
10196  -define F -finite_field -q 256 -end \
10197

727
define A -vector -field F -sparse 772 "1,771,1,0" -end \
define B -vector -field F -dense $(CRC_POLY_BRAVO_DENSE) -end \
with F -do \
coding.theoretic.activity \\
write_code_for_division \\
bravo A B \\
end \\
g++ crc_bravo.cpp -o crc_bravo.out \\
./crc_bravo.out \\

# Charlie 

CRC_F256_BCH_code.d7: 
$(ORBITER) -v 2 \\
-define F -finite_field -q 256 -end \\
-define C -code -field F \\
-BCH 771 7 \\
-end \\
-with C -and F -do -coding_theoretic_activity \\
-export_magma BCH_lq8_n771_d7.magma \\
-end \\
pdflatex BCH_codes_q256_n771_d7.tex \\
open BCH_codes_q256_n771_d7.pdf \\

# 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_F256_BCH_write_code_for_division_Charlie: 
$(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 \\

728
F256_BCH_code_d16:

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

#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, \\
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"

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 $(POLY_Q256_DEG30_DENSE)$ -end}$ \\
$\text{-with F -do}$ \\
$\text{-coding_theoretic_activity}$ \\
$\text{-write_code_for_division}$ \\
$\text{-end}$ \\
g++ check_q256_n771_r30.cpp A B \\
$\text{-end}$ \\
g++ check_q256_n771_r30.cpp -o check_q256_n771_r30.out \\
./check_q256_n771_r30.out
10287
10288
10289 F256_BCH_code_d16_division:
10290 | $(ORBITER) -v 2 \n10291 | -define F -finite_field -q 256 -end \n10292 | -define A -vector -field F -sparse 772 "1,771,1,0" -end \n10293 | -define B -vector -field F -dense $(POLY_Q256_DEG30_DENSE) -end \n10294 | -with F -do \n10295 | -finite_field_activity \n10296 | -polynomial_division A B -end
10297
10298
10299
10300 F256_BCH_code_d16_error:
10301 | $(ORBITER) -v 2 \n10302 | -define F -finite_field -q 256 -end \n10303 | -define A -vector -field F -sparse 771 "2,30,3,31,55,770" -end \n10304 | -define B -vector -field F -dense $(POLY_Q256_DEG30_DENSE) -end \n10305 | -with F -do \n10306 | -finite_field_activity \n10307 | -polynomial_division A B -end
10308
10309
10310
10311
10312 #CRC_FILE=allen_Gates
10313 CRC_FILE=javad-allahyari-Fs1E2JXM3Gc-unsplash
10314
10315 CRC_FILE_EXTENSION=bmp
10316
10317
10318 crc_encode_16:
10319 | $(ORBITER) -v 3 \n10320 | -define F -finite_field -q 2 -end \n10321 | -with F -do \n10322 | -coding_theoretic_activity \n10323 | -crc_decode_file_based $(CRC_FILE).$(CRC_FILE_EXTENSION) $(CRC_FILE)_crc16.bin crc16 771 \n10324 | -end
10325
10326 #-rw-r--r-- 1 betten staff 646576 Aug 24 14:35 allen_Gates_crc16.bin
10327 #-rw-r--r-- 1 betten staff 21656232 Aug 24 15:35 javad-allahyari-Fs1E2JXM3Gc-unsplash_crc16.bin
10328
10329
10330
10331 crc_encode_32:
10332 $\text{(ORBITER)} -v 3$
10333 $\text{-define } F \text{-finite_field } -q 2 \text{-end}$
10334 $\text{-with } F \text{-do}$
10335 $\text{-coding_theoretic_activity}$
10336 $\text{-crc_encode_file_based } $(\text{CRC_FILE}).$(\text{CRC_FILE_EXTENSION}) $(\text{CRC_FILE})_{crc3}$
10337 $\text{-end}$
10338
10339
10340 $\text{#-rw-r--r-- 1 betten staff 648262 Aug 24 14:34 allen_Gates_crc32.bin}$
10341
10342
10343
10344 $\text{#crc_encode_new:}$
10345 $\text{#} $(\text{ORBITER}) -v 3$
10346 $\text{#-define } F \text{-finite_field } -q 256 \text{-end}$
10347 $\text{-with } F \text{-do}$
10348 $\text{-coding_theoretic_activity}$
10349 $\text{-crc_new_file_based } $(\text{CRC_FILE}).$(\text{CRC_FILE_EXTENSION})$
10350 $\text{-end}$
10351
10352
10353 $\text{introduce_errors_{16,500000:}}$
10354 $\text{#} $(\text{ORBITER}) -v 3$
10355 $\text{#-introduce_errors}$
10356 $\text{#-input } $(\text{CRC_FILE})_{crc16.bin}$
10357 $\text{#-output } $(\text{CRC_FILE})_{crc16.e.bin}$
10358 $\text{#-block_based_error_generator}$
10359 $\text{#-block_length 771}$
10360 $\text{#-threshold 500000}$
10361 $\text{#-file_based_error_generator 500000}$
10362 $\text{#-nb_repeats 30}$
10363 $\text{-end}$
10364
10365
10366 $\text{introduce_errors_{32,100000:}}$
10367 $\text{#} $(\text{ORBITER}) -v 3$
10368 $\text{#-introduce_errors}$
10369 $\text{#-input } $(\text{CRC_FILE})_{crc32.bin}$
10370 $\text{#-output } $(\text{CRC_FILE})_{crc32.e.bin}$
10371 $\text{#-block_based_error_generator}$
10372 $\text{#-block_length 771}$
10373 $\text{#-threshold 100000}$
10374 $\text{#-file_based_error_generator 100000}$
10375 $\text{#-nb_repeats 30}$
10376 $\text{-end}$
10377
check_errors_16:

$ (ORBITER) -v 3 \
  -check_errors \
  -input $(CRC_FILE)_crc16_e.bin \
  -output $(CRC_FILE)_recovered.$(CRC_FILE_EXTENSION) \
  -crc_type crc16 \
  -error_log $(CRC_FILE)_crc16_e_pattern.csv \
  -block_length 771 \
  -end

check_errors_32:

$ (ORBITER) -v 3 \
  -check_errors \
  -input $(CRC_FILE)_crc32_e.bin \
  -output $(CRC_FILE)_recovered.$(CRC_FILE_EXTENSION) \
  -crc_type crc32 \
  -error_log $(CRC_FILE)_crc32_e_pattern.csv \
  -block_length 771 \
  -end

extract_block:

$ (ORBITER) -v 3 \
  -extract_block \
  -input $(CRC_FILE)_crc32_e.bin \
  -output $(CRC_FILE)_recovered.$(CRC_FILE_EXTENSION) \
  -error_log $(CRC_FILE)_crc32_e_pattern.csv \
  -block_length 771 \
  -selected_block 813731 \
  -end

# Section 10.5: Coding Theory - Reed-Muller codes

SECTION CODING THEORY REED-MULLER CODES:
$\text{ORBITER} -v 2$

-define F -finite_field -q 2 -end

-coding.theoretic_activity

-code_diagram "RM_3.1"

$(\text{REED MULLER}_3.1 \text{CODEWORDS}) 8$

-metric_balls 1

-end

# creates

# RM_3.1_8_16.tex

# RM_3.1_diagram_8_16.csv

# RM_3.1_diagram_01_8_16.csv

# RM_3.1_holes_8_16.csv

RM_3.1_split:

$\text{ORBITER} -split_by_values RM_3.1\_holes_8\_16.csv$

RM_3.1_holes_draw:

$\text{ORBITER} -v 2$

-loop L 0 3 1

-draw_matrix

-input_csv_file RM_3.1_holes_8_16_value%L.csv

-box_width 25 -bit_depth 8 -partition 5 16 16

open RM_3.1_diagram_8_16_draw.bmp

RM_3.1_draw:

$\text{ORBITER} -v 2$

-draw_matrix

-input_csv_file RM_3.1_holes_8_16.csv

-box_width 25 -bit_depth 8

-partition 4 16 16

-end

$\text{ORBITER} -v 2$

-draw_matrix

-input_csv_file RM_3.1_diagram_01_8_16.csv

-box_width 25 -bit_depth 8

-partition 4 16 16

-end

$\text{ORBITER} -v 2$

-draw_matrix

-input_csv_file RM_3.1_diagram_8_16.csv

-box_width 25 -bit_depth 8

-partition 4 16 16

-end

open RM_3.1_diagram_8_16_draw.bmp
10472 $\triangleright$ $\triangleright$ $\triangleright$ -end \ 
10473 $\triangleright$ $\triangleright$ $\triangleright$ -end_loop 
10474 
10475 RM_3.1_hole0: 
10476 $\triangleright$ $(ORBITER) -v 3 $ \ 
10477 $\triangleright$ $\triangleright$ -define F -finite_field -q 2 -end $ \ 
10478 $\triangleright$ $\triangleright$ -with F -do -finite_field_activity $ \ 
10479 $\triangleright$ $\triangleright$ $\triangleright$ -algebraic_normal_form $ \ 
10480 $\triangleright$ $\triangleright$ $\triangleright$ RM_3.1_holes_8.16_value0.csv 8 $ \ 
10481 $\triangleright$ $\triangleright$ $\triangleright$ -end $ \ 
10482 
10483 # E_0 + E_1 + E_2 + E_3 + E_4 $ 
10484 
10485 
10486 RM_3.1_hole1: 
10487 $\triangleright$ $(ORBITER) -v 3 $ \ 
10488 $\triangleright$ $\triangleright$ -define F -finite_field -q 2 -end $ \ 
10489 $\triangleright$ $\triangleright$ -with F -do -finite_field_activity $ \ 
10490 $\triangleright$ $\triangleright$ $\triangleright$ -algebraic_normal_form $ \ 
10491 $\triangleright$ $\triangleright$ $\triangleright$ RM_3.1_holes_8.16_value1.csv 8 $ \ 
10492 $\triangleright$ $\triangleright$ $\triangleright$ -end $ \ 
10493 
10494 #E_1 = X_0X^8X^7 + X_1X^8X^7 + X_2X^8X^7 + X_3X^8X^7 + X_4X^8X^7 + X_5X^8X^7 + X_6X^8X^7 
+ X_7X^8X^7 $ 
10495 
10496 RM_3.1_hole2: 
10497 $\triangleright$ $(ORBITER) -v 3 $ \ 
10498 $\triangleright$ $\triangleright$ -define F -finite_field -q 2 -end $ \ 
10499 $\triangleright$ $\triangleright$ -with F -do -finite_field_activity $ \ 
10500 $\triangleright$ $\triangleright$ $\triangleright$ -algebraic_normal_form $ \ 
10501 $\triangleright$ $\triangleright$ $\triangleright$ RM_3.1_holes_8.16_value2.csv 8 $ \ 
10502 $\triangleright$ $\triangleright$ $\triangleright$ -end $ \ 
10503 
10504 #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*X 
8^5 + 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 + 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 + 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 + X3*X4*X5*X8^5 + X3*X4*X6*X8 
^5 + X
3*X4*X7*X8^5 + X3*X5*X6*X8^5 + X3*X5*X7*X8^5 + X3*X6*X7*X8^5 + X4*X5*X6*X8^5 + X4 *
X5*X7*X8^5 + X4*X6*X7*X8^5 + X5*X6*X7*X8^5 + 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*X4*X8^4 + X0*X1*X3*X7*X8^4 + X0*X1*X4*X6*X8^4 + X0*X1*X4*X7*X8^4 + X0*X1*X
5*X6*X8^4 + X0*X1*X6*X7*X8^4 + X0*X2*X3*X4*X8^4 + X0*X2*X3*X5*X8^4 + X0*X2*X3*X6*
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*X6*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*X4*X5*X6*X8^4 + X0*X4*X5*X7*X8^4 + X0*X4*X6*X7*X8^4 + X0*X5*X6 *
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*X
2*X6*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*X4*X5*X6*X8^4 + X1*X4*X5*X7*
X8^4 + X1*X4*X6*X7*X8^4 + X1*X4*X6*X8^4 + X1*X5*X6*X7*X8^4 + X1*X5*X6*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*X4*X5*X6*X8^4 + X2*X4*X5*
X7*X8^4 + X2*X4*X6*X7*X8^4 + X2*X4*X6*X8^4 + X2*X5*X6*X7*X8^4 + X2*X5*X6*X8^4 +
X2*X5*X7*X8^4 + X2*X5*X8^4 + X3*X4*X5*X6*X8^4 + X3*X4*X5*X7*X8^4 + X3*X4*X5*X8*
X8^4 + X3*X4*X5*X8^4 + X3*X4*X6*X7*X8^4 + X3*X4*X6*X8^4 + X3*X4*X7*X8^4 + X3*X
4*X5*X6*X8^4 + X3*X4*X5*X7*X8^4 + X3*X4*X5*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*X5*X8^4 + X3*X6*X7*X8^4 +
X3*X6*X8^4 + X3*X7*X8^4 + X3*X8^4 + X3*X8^4 + X4*X5*X6*X8^4 + X4*X5*X7*X8^4 +
X4*X5*X8^4 + X4*X6*X7*X8^4 + X4*X6*X8^4 + X4*X7*X8^4 + X4*X8^4 + X5*X6*X7*X8^4 +
X5*X6*X8^4 + X5*X7*X8^4 + X5*X8^4 + X6*X7*X8^4 + X6*X8^4 + X7*X8^4

10505
10506 # E_2 + E_3 + E_4
10507
10508
10509
10510 RM_4_1:
10511 $ (ORBITER) -v 2 \n10512 $ define F -finite_field -q 2 -end \n10513 $ define C -code -field F -first_order_Reed_Muller 4 -end \n10514 $ with C -and F -do -coding_theoretic_activity \n10515 $ export_magma RM_4_1.magma \n10516 $ export codewords RM_4_1.codewords.csv \n10517 $ export codewords RM_4_1.old: \n10518 $ (ORBITER) -v 2 \n10519 $ define F -finite_field -q 2 -end \n10520 $ with F -do \n10521 $ coding_theoretic_activity \n10522 $ linear_code_through_columns_of_parity_check 5 \n10523 $ (REED_MULLER_4_1_COLUMNS_OF_PARITY_CHECK) \n10524 $ export code n16 k5 q2.tex \n10525 $ open code n16 k5 q2.pdf
10526
10527
10528
10529
10530
10531 pdflatex code n16 k5 q2.tex \n10532 open code n16 k5 q2.pdf
10533
10534
10535 # codewords n16 k5 q2.csv
10536

735
RM_4_1_diagram:

RM_4_1_diagram.draw:

RM_4_1_split:

RM_4_1_diagram_draw.holes:

RM_4_1_diagram_metric.balls:
# Section 10.6: Coding Theory - BCH Codes

draw_cyclotomic_mod_21_q8:

draw_diagram_from_file "RM_4.1"

codewords_n16_k5_q2.csv

$\text{metric\_balls\_3}$

$\text{end}$

$\text{draw\_matrix}$

csv_file RM\_4.1\_diagram\_16\_32.csv

$\text{line\_width\_25 \ -bit\_depth\_8}$

$\text{partition\_10 256 256}$

$\text{end}$

$\text{RM\_4.1\_hole0:}$

$\text{define\ F \ -finite\_field\ -q\ 2 \ -end}$

$\text{with\ F \ -do \ -finite\_field\_activity}$

$\text{algebraic\_normal\_form}$

$\text{RM\_4.1\_holes\_16\_32\_value0.csv}$

$\text{end}$

$\text{draw\_options}$

$\text{radius\_100}$

$\text{line\_width\_1.0 \ -embedded}$

$\text{end}$

$\text{draw\_mod\_n \ -n\ 21 \ -file\ mod\_21\_cyclotomic}$

$\text{pdflatex mod\_21\_cyclotomic\_draw.tex}$

$\text{open mod\_21\_cyclotomic\_draw.pdf}$
10631
10632
10633
10634
10635
10636
10637
10638
10639
10640
10641
10642
10643
10644
10645
10646
10647
10648
10649
10650
10651
10652
10653
10654
10655
10656
10657
10658
10659
10660
10661
10662
10663
10664
10665
10666
10667
10668
10669
10670
10671
10672
10673
10674
10675
10676

F 8 BCH code d3:
. $(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
#generator polynomial is X^{4} + 4X^{3} + 4X^{2} + 3X + 4
F 8 BCH code d4:
. $(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
#generator polynomial is X^{5} + 6X^{4} + 7X^{3} + 2X + 3

F 8 BCH code d5:
. $(ORBITER) -v 3 \
. . -define F -finite field -q 8 -override polynomial 11 -end \
. . -with F -do \
. . . -coding theoretic activity \
. . . . -make BCH code 21 5 \
. . -end
. pdflatex BCH codes q8 n21 d5.tex
. open BCH codes q8 n21 d5.pdf
#-override polynomial 11
#generator polynomial is X^{7} + 3X^{6} + 3X^{5} + 2X^{4} + X^{3} + 2X^{2} + X +
2

#CODE BCH F8 N21 D5 GENMA
CODE BCH F8 N21 D5 GENMA OVERRIDE POLYNOMIAL11="\
2,1,2,1,2,3,3,1,0,0,0,0,0,0,0,0,0,0,0,0,0,\
0,2,1,2,1,2,3,3,1,0,0,0,0,0,0,0,0,0,0,0,0,\
0,0,2,1,2,1,2,3,3,1,0,0,0,0,0,0,0,0,0,0,0,\
0,0,0,2,1,2,1,2,3,3,1,0,0,0,0,0,0,0,0,0,0,\
0,0,0,0,2,1,2,1,2,3,3,1,0,0,0,0,0,0,0,0,0,\
0,0,0,0,0,2,1,2,1,2,3,3,1,0,0,0,0,0,0,0,0,\

738


F8 BCH code d5 minimum distance:

F8 BCH code d7:
F8_BCH_code_n63_d43:

$\text{ORBITER} -v 3 \ \ \\
\text{-define F -finite_field -q 8 -override_polynomial 11 -end} \ \\
\text{-with F -do} \ \\
\text{-coding_theoretic_activity} \ \\
\text{-make_BCH_code 63 43} \ \\
\text{-end} \ \\
\text{pdflatex BCH_codes_q8_n63_d43.tex} \ \\
\text{open BCH_codes_q8_n63_d43.pdf} \ \\
\text{CODE_BCH_F8_N63_K9_D43_GENMA=\"}

\begin{verbatim}
 4,5,2,2,4,5,6,2,4,2,6,0,0,7,1,5,3,7,1,0,0,5,6,4,7,6,7,7,1,2,6,3,1,6,0,0,6,6,6,4,7
 ,7,0,0,3,4,7,5,6,2,5,1,4,4,1,0,0,0,0,0,0,0, \\\n 0,4,5,2,2,4,5,6,2,4,2,6,0,0,7,1,5,3,7,1,0,0,5,6,4,7,6,7,7,1,2,6,3,1,6,0,0,6,6,6,4
 ,7,7,0,0,3,4,7,5,6,2,5,1,4,4,1,0,0,0,0,0,0,0, \\\n 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 0,0,0,4,5,2,2,4,5,6,2,4,2,6,0,0,7,1,5,3,7,1,0,0,5,6,4,7,6,7,7,1,2,6,3,1,6,0,0,6,6
 ,6,6,7,7,0,0,3,4,7,5,6,2,5,1,4,4,1,0,0,0,0,0, \\\n 0,0,0,0,4,5,2,2,4,5,6,2,4,2,6,0,0,7,1,5,3,7,1,0,0,5,6,4,7,6,7,7,1,2,6,3,1,6,0,0,6
 ,6,6,6,7,7,0,0,3,4,7,5,6,2,5,1,4,4,1,0,0,0,0, \\\n 0,0,0,0,0,4,5,2,2,4,5,6,2,4,2,6,0,0,7,1,5,3,7,1,0,0,5,6,4,7,6,7,7,1,2,6,3,1,6,0,0,6
 ,6,6,6,6,7,7,0,0,3,4,7,5,6,2,5,1,4,4,1,0,0,0, \\\n 0,0,0,0,0,0,4,5,2,2,4,5,6,2,4,2,6,0,0,7,1,5,3,7,1,0,0,5,6,4,7,6,7,7,1,2,6,3,1,6,0,0,6
 ,6,6,6,6,6,7,7,0,0,3,4,7,5,6,2,5,1,4,4,1,0,0, \\\n 0,0,0,0,0,0,0,4,5,2,2,4,5,6,2,4,2,6,0,0,7,1,5,3,7,1,0,0,5,6,4,7,6,7,7,1,2,6,3,1,6,0,0,6
 ,6,6,6,6,6,6,7,7,0,0,3,4,7,5,6,2,5,1,4,4,1,0,0, \\\n 0,0,0,0,0,0,0,0,4,5,2,2,4,5,6,2,4,2,6,0,0,7,1,5,3,7,1,0,0,5,6,4,7,6,7,7,1,2,6,3,1,6,0,0,6
 ,6,6,6,6,6,6,6,7,7,0,0,3,4,7,5,6,2,5,1,4,4,1,0,0, \\\n 0,0,0,0,0,0,0,0,0,4,5,2,2,4,5,6,2,4,2,6,0,0
\end{verbatim}

F8_BCH_code_n63_d43.minimum.distance:

$\text{ORBITER} -v 2 \ \ \\
\text{-define F -finite_field -q 8 -override_polynomial 11 -end} \ \\
\text{-define v -vector -format 9 -field F} \ \\
\text{-compact $(\text{CODE_BCH_F8_N63_K9_D43_GENMA})} \ \\
\text{-end} \ \\
\text{-with F -do} \ \\

740
The minimum distance is $d = 45$, computed in 0 days, 0 hours, 1 minutes, 32 seconds

#coding_theory_domain::do_minimum_distance

F2_BCH_code_n21:

F7_RS_code_n6:

F_64_again:

BCH_255_5_evaluate_elementary_symmetric_functions_1:
define e3 -formula "e3" "e3" "" $(ELEMENTARY_SYMMETRIC_8_3) \\
define e4 -formula "e4" "e4" "" $(ELEMENTARY_SYMMETRIC_8_4) \\
define e5 -formula "e5" "e5" "" $(ELEMENTARY_SYMMETRIC_8_5) \\
define e6 -formula "e6" "e6" "" $(ELEMENTARY_SYMMETRIC_8_6) \\
define e7 -formula "e7" "e7" "" $(ELEMENTARY_SYMMETRIC_8_7) \\
define e8 -formula "e8" "e8" "" $(ELEMENTARY_SYMMETRIC_8_8) \\
define E8 -collection "e1,e2,e3,e4,e5,e6,e7,e8" \\
with F -do -finite_field_activity \\
> -evaluate E8 "x0=2,x1=4,x2=16,x3=29,x4=76,x5=157,x6=95,x7=133" -end

BCH_255_5_evaluate_elementary_symmetric_functions_2:
$($(ORBITER) -v 3 -define F -finite_field -q 256 -end \\
define e1 -formula "e1" "e1" "" $(ELEMENTARY_SYMMETRIC_8_1) \\
define e2 -formula "e2" "e2" "" $(ELEMENTARY_SYMMETRIC_8_2) \\
define e3 -formula "e3" "e3" "" $(ELEMENTARY_SYMMETRIC_8_3) \\
define e4 -formula "e4" "e4" "" $(ELEMENTARY_SYMMETRIC_8_4) \\
define e5 -formula "e5" "e5" "" $(ELEMENTARY_SYMMETRIC_8_5) \\
define e6 -formula "e6" "e6" "" $(ELEMENTARY_SYMMETRIC_8_6) \\
define e7 -formula "e7" "e7" "" $(ELEMENTARY_SYMMETRIC_8_7) \\
define e8 -formula "e8" "e8" "" $(ELEMENTARY_SYMMETRIC_8_8) \\
define E8 -collection "e1,e2,e3,e4,e5,e6,e7,e8" \\
> -with F -do -finite_field_activity \\
> -evaluate E8 "x0=8,x1=64,x2=205,x3=143,x4=70,x5=217,x6=130,x7=23" -end

BCH15:
$($(ORBITER) -BCH 15 2 3 \\
$($(ORBITER) \\
define F -finite_field -q 2 -end \\
> -with F -do \\
> -coding_theoretic_activity \\
> -BCH 15 2 5 \\
> -end

> -BCH 15 2 7 \\
> -BCH 15 2 9

draw_mod_31:
$($(ORBITER) -v 2 \\
> -draw_options -embedded -end \\
> -file mod_31 \\
> -draw_mod_n_power_cycle 2 \\

742
PR127: $(ORBITER) -v 5 -primitive_root 127

PR127: $(ORBITER) -v 2 -draw_options -scale 0.4 -embedded -end -draw_mod_n 127 mod_127 -draw_mod_n_power_cycle 3

PR127: pdflatex mod_127_drawing.tex

PR127: open mod_127_drawing.pdf

PR127: draw_mod_127_power:

PR127: $(ORBITER) -v 2 -draw_options -nodes_empty -radius 10 -embedded -end -draw_mod_n 251 mod_251

PR127: pdflatex mod_251_drawing.tex

PR127: open mod_251_drawing.pdf

PR127: # -draw_mod_n_inverse

PR127: draw_mod_255_cyclotomic_1:

PR127: $(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

PR127: cyclotomic_sets 2 "1" -end

PR127: pdflatex mod_255_cyclotomic_1_drawing.tex

PR127: open mod_255_cyclotomic_1_drawing.pdf

PR127: draw_mod_255_cyclotomic_3:

PR127: $(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

PR127: cyclotomic_sets 2 "3" -end

PR127: pdflatex mod_255_cyclotomic_3_drawing.tex

PR127: open mod_255_cyclotomic_3_drawing.pdf

PR127: draw_mod_255_cyclotomic_1_and_3:

PR127: $(ORBITER) -v 2 -draw_options -nodes_empty -radius 10
BCH elementry symmetric functions 3:

```bash
$ (ORBITER) -make_elementary_symmetric_functions 3 3
```

BCH_63_4_evaluate_elementary_symmetric_functions_1:

```bash
$ (ORBITER) -v 3 -define F -finite_field -q 64 -end 
```

# The values of the formulae are:

```
#0 : 57
#1 : 0
#2 : 1
```

# poly: 1,0,2,1
10949
10950 BCH_63_4.evaluate_elementary_symmetric_functions_2:
10951  defiance $(ORBITER) -v 3 -define F -finite_field -q 64 -end \n10952  defiance -define e1 -formula "e1" "e1" "$(ELEMENTARY_SYMMETRIC_3.1) \n10953  defiance -define e2 -formula "e2" "e2" "$(ELEMENTARY_SYMMETRIC_3.2) \n10954  defiance -define e3 -formula "e3" "e3" "$(ELEMENTARY_SYMMETRIC_3.3) \n10955  defiance -define E3 -collection "e1,e2,e3" \n10956  defiance -with F -do -finite_field_activity \n10957  defiance -evaluate E3 "x0=33,x1=45,x2=52" -end  
10958
10959  #The values of the formulae are:
10960  #0 : 56
10961  #1 : 0
10962  #2 : 1
10963
10964  # poly: 1,0,3,1
10965
10966
10967
10968
10969 BCH_21.poly_mult_mod_F4:
10970  defiance $(ORBITER) -v 2 \n10971  defiance -define F -finite_field -q 4 -end \n10972  defiance -with F -do \n10973  defiance -finite_field_activity \n10974  defiance -polynomial_mult_mod "1,0,2,1" "1,0,3,1" \n10975  defiance -polynomial "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" \n10976  defiance -end \n10977
10978  #C(X)=X^{6} + X^{5} + X^{4} + X^{2} + 1
10979
10980  # poly 1,0,1,0,1,1,1
10981
10982
10983 BCH_21.poly_division_a:
10984  defiance $(ORBITER) -v 2 \n10985  defiance -define F -finite_field -q 4 -end \n10986  defiance -with F -do \n10987  defiance -finite_field_activity \n10988  defiance -polynomial_division \n10989  defiance -polynomial "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" \n10990  defiance -polynomial "1,0,2,1" \n10991  defiance -end \n10992
10993 BCH_21.poly_division_b:
10994  defiance $(ORBITER) -v 2 \n10995  defiance -define F -finite_field -q 4 -end \n
745
10996 \> \> \> -with F -do \>
10997 \> \> \> \> \> -finite_field_activity \>
10998 \> \> \> -polynomial_division \>
10999 \> \> \> \> \> "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" \>
11000 \> \> \> \> \> "1,0,3,1" \>
11001 \> \> \> -end
11002
11003
11004 BCH_21.poly_division_ab:
11005 \> $(ORBITER) -v 2 \>
11006 \> \> -define F -finite_field -q 4 -end \>
11007 \> \> -with F -do \>
11008 \> \> -finite_field_activity \>
11009 \> \> -polynomial_division \>
11010 \> \> \> "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" \>
11011 \> \> \> "1,0,1,0,1,1,1" \>
11012 \> \> -end
11013
11014 BCH_21.generator_matrix:
11015 \> $(ORBITER) -v 2 \>
11016 \> \> -define F -finite_field -q 4 -end \>
11017 \> \> -with F -do \>
11018 \> \> -coding_theoretic_activity \>
11019 \> \> \> -generator_matrix_cyclic_code \>
11020 \> \> \> \> 21 "1,0,1,0,1,1,1" \>
11021 \> \> \> -end
11022
11023
11024
11025
11026 BCH_21.15.weight Enumerator:
11027 \> $(ORBITER) -v 2 \>
11028 \> \> -define F -finite_field -q 4 -end \>
11029 \> \> -define v -vector -field F -format 15 \>
11030 \> \> \> -dense $(BCH_21.15_GENERATOR_MATRIX) \>
11031 \> \> \> -end \>
11032 \> \> -define C -code -field F \>
11033 \> \> \> -linear_code_through_generator_matrix v \>
11034 \> \> \> -end \>
11035 \> \> -with C -do \>
11036 \> \> -coding_theoretic_activity \>
11037 \> \> \> -weight Enumerator \>
11038 \> \> \> -end
11039
11040 # too slow!
11041
11042 BCH_21.15.dual:
\begin{verbatim}
11043 \% $(ORBITER) -v 2 \\
11044 \% -define F -finite_field -q 4 -end \\
11045 \% -define v -vector -field F -format 15 \\
11046 \% -dense $(BCH_21_15_GENERATOR_MATRIX) -end \\
11047 \% -with F -do -finite_field_activity \\
11048 \% -nullspace v \\
11049 \% -normalize_from_the_right \\
11050 \% -end \\
11051 \\
11052 \\
11053 BCH_21_6_weight Enumerator:
11054 \% $(ORBITER) -v 2 \\
11055 \% -define F -finite_field -q 4 -end \\
11056 \% -define v -vector -format 6 -field F \\
11057 \% -dense $(BCH_21_6_GENERATOR_MATRIX) \\
11058 \% -end \\
11059 \% -define C -code -field F \\
11060 \% -linear_code_through_generator_matrix v \\
11061 \% -end \\
11062 \% -with F -do \\
11063 \% -coding_theoretic_activity \\
11064 \% -weight Enumerator \\
11065 \% -end \\
11066 \\
11067 # 1y^{21} + 63x^8y^{13} + 294x^{12}y^9 + 756x^{14}y^7 + 1890x^{16}y^5 + 1092x^{18}y^3
11068 \\
11069 #( 1, 0, 0, 84, 252, 1575, 10080, 58032, 319662, 1411116, 5133744, 15282792, 37951620, 79336530, 135622080, 190615824, 213273081, 188911548, 125744304, 59721732, 17767512, 2580255 )
11070 \\
11071 \\
11072 \\
11073 BCH_21_6_4_macwilliams:
11074 \% $(ORBITER) -v 2 \\
11075 \% -make_macwilliams_system 21 6 4 \\
11076 \% pdflatex MacWilliams_n21_k6_q4.tex \\
11077 \% open MacWilliams_n21_k6_q4.pdf \\
11078 \\
11079 \\
11080 \\
11081 \# w := [1, 0, 0, 84, 252, 1575, 10080, 58032, 319662, 1411116, 5133744, 15282792, 37951620, 79336530, 135622080, 190615824, 213273081, 188911548, 125744304, 59721732, 17767512, 2580255]
11082 \\
11083 \\
11084 BCH_21_15_4_field_reduction:
11085 \% $(ORBITER) -v 2 \\
11086 \% -define F -finite_field -q 4 -end \\
\end{verbatim}
11087 \ verify -with \ F -do \\
11088 \ verify -finite_field_activity \\
11089 \ verify -field_reduction "BCH\_21\_15\_4" 2 15 21 $(BCH\_21\_15) \\
11090 \ verify -end \\
11091 \ $(ORBITER) -v 2 \\
11092 \ verify -draw_matrix \\
11093 \ verify -input\_csv\_file BCH\_21\_15\_4.csv \\
11094 \ verify -box\_width 20 -bit\_depth 24 \\
11095 \ verify -partition 4 "30" "42" \\
11096 \ verify -end \\
11097 \ pdflatex field_reduction\_Q4\_q2\_15\_21.tex \\
11098 \ open BCH\_21\_15\_4.draw.bmp \\
11099 \ #open field_reduction\_Q4\_q2\_15\_21.pdf \\
11100 11101 \ # poly of degree 12: 1,0,1,0,1,0,0,0,1,0,0,0,1 \\
11102 11103 \ BCH\_21\_poly\_division\_c: \\
11104 \ $(ORBITER) -v 2 \\
11105 \ verify -define F -finite_field -q 2 -end \\
11106 \ verify -with F -do \\
11107 \ verify -finite_field_activity \\
11108 \ verify -polynomial\_division \\
11109 \ verify - "1,0,0,0,0,0,0,0,0,0,0,0,0,1" \\
11110 \ verify - "1,0,1,0,1,0,0,1,0,0,0,1" \\
11111 \ verify -end \\
11112 11113 11114 \ F16\_roots\_5: \\
11115 \ $(ORBITER) -v 3 \\
11116 \ verify -define F -finite_field -q 2 -end \\
11117 \ verify -with F -do -coding\_theoretic\_activity \\
11118 \ verify -nth\_roots 5 \\
11119 \ verify -end \\
11120 \ pdflatex Nth\_roots\_q2\_n5.tex \\
11121 \ open Nth\_roots\_q2\_n5.pdf \\
11122 11123 11124 11125 \ F64\_roots\_21: \\
11126 \ $(ORBITER) -v 3 \\
11127 \ verify -define F -finite_field -q 2 -end \\
11128 \ verify -with F -do -coding\_theoretic\_activity \\
11129 \ verify -nth\_roots 21 \\
11130 \ verify -end \\
11131 \ pdflatex Nth\_roots\_q2\_n21.tex \\
11132 \ open Nth\_roots\_q2\_n21.pdf 

748
BCH_F256_roots_771:

```
$($ORBITER) -v 3 \ 
$define F -finite_field -q 256 -end \ 
-with F -do -coding_theoretic_activity \ 
-nth_roots 771 \ 
-end
```

BCH_F256_BCH_code_d16:

```
$($ORBITER) -v 3 \ 
$define F -finite_field -q 256 -end \ 
-with F -do -coding_theoretic_activity \ 
-make_BCH_code 771 16 \ 
-end
```

dpdflatex BCH_codes_q256_n771_d16.tex
dopen 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

# Section 10.7: Coding Theory - Reed Solomon codes

SECTION CODING THEORY REED SOLOMON CODES:

ToDo:

F_7_BCH_code_n6:

```
$($ORBITER) -v 3 \ 
$define F -finite_field -q 7 -end \ 
-with F -do -finite_field_activity \ 
-coding_theoretic_activity 7 3 \ 
```
11176 ▷ ▷ -end

11177

11178

11179

11180 RREF_RS_6_4_7_weight_enumerator:

11181 ▷ $(ORBITER) -v 2 \n
11182 ▷ ▷ -define F -finite_field -q 7 -end \n
11183 ▷ ▷ -define v -vector -format 4 -field F \n
11184 ▷ ▷ ▷ -compact $(CODE_RS_6_4_7) \n
11185 ▷ ▷ -end \n
11186 ▷ ▷ -define C -code -field F \n
11187 ▷ ▷ ▷ -linear_code_through_generator_matrix v \n
11188 ▷ ▷ -end \n
11189 ▷ ▷ -with C -do \n
11190 ▷ ▷ -coding_theoretic_activity \n
11191 ▷ ▷ ▷ -weight Enumerator \n
11192 ▷ ▷ ▷ -end

11193

11194

11195 #1y^6 + 120x^3y^3 + 360x^4y^2 + 972x^5y + 948x^6

11196 #weight Enumerator: 

11197 #( 1, 0, 0, 120, 360, 972, 948 )

11198

11199

11200

11201

11202

11203 Code_RS_11:

11204 ▷ $(ORBITER) -v 2 \n
11205 ▷ ▷ -define F -finite_field -q 11 -end \n
11206 ▷ ▷ -define v -vector -format 8 -field F \n
11207 ▷ ▷ ▷ -compact $(CODE_RS_10_8_11) \n
11208 ▷ ▷ ▷ -end \n
11209 ▷ ▷ ▷ -with F -do \n
11210 ▷ ▷ ▷ -finite_field_activity -RREF v -end

11211 ▷ pdflatex RREF_example_q11_8_10.tex

11212 ▷ #gs -sDEVICE=png16 -dFIXEDMEDIA -dDEVICEWIDTHPOINTS=500 -dDEVICEHEIGHTPOINTS=45 0 \n
11213 ▷ #> -r240 -oRREF_example_q11_8_10_page%02d.png \

11214 ▷ #> RREF_example_q11_8_10.pdf

11215 ▷ open RREF_example_q11_8_10.pdf

11216

11217

11218 Code_RS_11_weight_enumerator:

11219 ▷ $(ORBITER) -v 2 \n
11220 ▷ ▷ -define F -finite_field -q 11 -end \n
11221 ▷ ▷ -define v -vector -format 8 -field F \n
750
-compact $(\text{CODE}_{\text{RS}_{11_{RREF}}})$

-define C -code -field F

-linear_code_through_generator_matrix v

-end

-with C -do

-coding_theoretic_activity

-weight Enumerator

-end

-coding_theoretic_activity

-weight Enumerator

-end

# the group cannot be computed

RS\_8\_weight Enumerator:

$(\text{ORBITER}) -v 2$

-define F -finite_field -q 8 -end

-define v -vector -format 5 -field F

-compact $(\text{CODE}_{\text{RS}_{8}})$

-end

-define C -code -field F

-linear_code_through_generator_matrix v

-end

-with C -do

-coding_theoretic_activity

-weight Enumerator

-end

-end

-field reduction "RS\_8\_red\_2"

-2 5 7 $(\text{CODE}_{\text{RS}_{8}})$

-end

$(\text{ORBITER}) -v 2$

-draw_matrix -input_csv_file RS\_8\_red\_2.csv

-box_width 40 -bit_depth 24

-partition 4 "3,3,3,3,3,3,3,3,3,3,3,3,3" -end

-pdflatex field_reduction\_Q8\_q2\_5\_7.tex

-open field_reduction\_Q8\_q2\_5\_7.pdf

751
11268
11269 RREF_RS_8_reduced_weight Enumerator:
11270
11271 \$\text{\texttt{(ORBITER)}} -v 2 \\
11272 \text{define F -finite_field -q 2 -end} \\
11273 \text{define v -vector -format 15 -field F} \\
11274 \text{compact \$\text{\texttt{RS}}_8\text{\texttt{reduced}}} \\
11275 \text{end} \\
11276 \text{define C -code -field F} \\
11277 \text{linear_code_through_generator_matrix v} \\
11278 \text{end} \\
11279 \text{with C -do} \\
11280 \text{coding_theoretic_activity} \\
11281 \text{weight Enumerator} \\
11282 \text{end}
11283
11284
11285 CODE_21_15_4 store:
11286 \$\text{\texttt{(ORBITER)}} -v 2 \\
11287 \text{store_as_csv_file "code_21_15_4.csv"} \\
11288 \text{15 21 \$\text{\texttt{(CODE}}_21_15_4\text{\texttt{)}}} \\
11289 \text{\texttt{(ORBITER)}} -v 2 -draw_matrix \\
11290 \text{-input_csv_file code_21_15_4.csv} \\
11291 \text{-box_width 40 -bit_depth 24} \\
11292 \text{-partition 4 "15" "21"} \\
11293 \text{end}
11294
11295 CODE_21_15_4 weight Enumerator:
11296 \$\text{\texttt{(ORBITER)}} -v 2 \\
11297 \text{define F -finite_field -q 2 -end} \\
11298 \text{define v -vector -format 15 -field F} \\
11299 \text{compact \$\text{\texttt{CODE}}_21_15_4\text{\texttt{)}}} \\
11300 \text{end} \\
11301 \text{define C -code -field F} \\
11302 \text{linear_code_through_generator_matrix v} \\
11303 \text{end} \\
11304 \text{with C -do} \\
11305 \text{coding_theoretic_activity} \\
11306 \text{weight Enumerator} \\
11307 \text{end}
11308
11309 CODE_21_15_4 minimum distance:
11310 \$\text{\texttt{(ORBITER)}} -v 2 \\
11311 \text{define F -finite_field -q 2 -end} \\
11312 \text{define v -vector -format 15 -field F} \\
11313 \text{compact \$\text{\texttt{CODE}}_21_15_4\text{\texttt{)}}} \\
11314 \text{end}
$\text{bounds for } d \text{ given } n_6, k_4, q_7$: 
$\text{define } F \text{ -finite field -q 8 -end }$
$\text{make bounds for } d \text{ given } n \text{ and } k \text{ and } q_6 4 7$

$\text{bounds for } d \text{ given } n_15, k_6, q_2$: 
$\text{define } F \text{ -finite field -q 2 -end }$
$\text{make bounds for } d \text{ given } n \text{ and } k \text{ and } q_15 6 2$

# produces table of bounds n20_q2.csv

$\text{coding theory bounds q2:}$
$\text{define } F \text{ -finite field -q 2 -end }$

$\text{coding theory bounds q8:}$

$\text{GV_n15_k6_d5:}$

753
-coding_theoretic_activity \n-define F -finite_field -q 2 -end \n-define v -vector -format 6 -field F \n-compact $(CODE_GV_N15_K6) \n-end \n-define C -code -field F \n-linear_code_through_generator_matrix v \n-end \n-with C -do \n-coding_theoretic_activity \n-weight Enumerator \n-end

# [15,6] code created

bounds_for_d_given_n12_k4_q13:

$(ORBITER) -v 2 \n-make_bounds_for_d_given_n_and_k_and_q 12 4 13

GV N15 K6 D5 weight enumerator:

$(ORBITER) -v 2 \n-define F -finite_field -q 2 -end \n-define v -vector -format 6 -field F \n-compact $(CODE_GV_N15_K6) \n-end \n-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

#1y^{15} + 27x^6y^9 + 24x^8y^7 + 9x^{10}y^5 + 3x^{12}y^3

surprise: d = 6
#1y^{15} + 27x^6y^9 + 24x^8y^7 + 9x^{10}y^5 + 3x^{12}y^3

# weight enumerator
#1y^{15} + 28x^6y^9 + 21x^8y^7 + 12x^{10}y^5 + 2x^{12}y^3

code
n15
k6
d6
RREF:

⊿ $(ORBITER)$ -v 2 \
⊿ ⊿ -define F -finite_field -q 2 -end \
⊿ ⊿ -define v -vector -format 6 -field F \
⊿ ⊿ ⊿ -compact $(CODE\_GV\_N15\_K6)$ \
⊿ ⊿ ⊿ -end \
⊿ ⊿ ⊿ -with F -do -finite_field_activity \
⊿ ⊿ ⊿ -RREF v -normalize_from_the_right \
⊿ ⊿ ⊿ -end

⊿ pdflatex RREF\_example\_q2\_6\_15.tex
⊿ open RREF\_example\_q2\_6\_15.pdf

code
n15
k6
d6
check

⊿ $(ORBITER)$ -v 2 \
⊿ ⊿ -define F -finite_field -q 2 -end \
⊿ ⊿ -define v -vector -format 9 -field F \
⊿ ⊿ ⊿ -compact $(CODE\_GV\_N15\_K6\_CHECK)$ \
⊿ ⊿ ⊿ -end \
⊿ ⊿ ⊿ -with F -do -finite_field_activity \
⊿ ⊿ ⊿ -RREF v -normalize_from_the_right \
⊿ ⊿ ⊿ -end

⊿ pdflatex RREF\_example\_q2\_9\_15.tex
⊿ open RREF\_example\_q2\_9\_15.pdf

# Section 10.9: Coding Theory - Classification

SECTION CODING THEORY CLASSIFICATION:

# code classification:

codes_8_4_4:
11456 > $(ORBITER) -v 6 \
11457 > -orbiter_path $(ORBITER_PATH) \
11458 > -define G \
11459 > -linear_group -PGL 4 2 -end \
11460 > -with G -do \
11461 > -group_theoretic_activity \
11462 > -poset_classification_control \
11463 > -problem_label codes_8_4_4 \
11464 > -draw_poset \
11465 > -draw_options -embedded -radius 250 \
11466 > -line_width 1.0 -spanning_tree -end \
11467 > -report -end \
11468 > -end \
11469 > -linear_codes 3 8 \
11470 > -end \
11471 > pdflatex codes_8_4_4_poset_lvl_8.tex 
11472 > open codes_8_4_4_poset_lvl_8.pdf 
11473 > pdflatex codes_8_4_4_poset.tex 
11474 > open codes_8_4_4_poset.pdf 
11475 
11476 codes_8_4_4_draw:
11477 > $(ORBITER) -v 3 \
11478 > -draw_layered_graph \
11479 > -codes_8_4_4_poset_lvl_8.layered_graph \
11480 > -radius 250 -embedded -line_width 1.0 \
11481 > -y_stretch 1.0 -scale 0.5 \
11482 > -end \
11483 > pdflatex codes_8_4_4_poset_lvl_8_draw.tex 
11484 > open codes_8_4_4_poset_lvl_8_draw.pdf 
11485 
11486 
11487 
11488 codes_14_4_9_3:
11489 > $(ORBITER) -v 6 \
11490 > -define G \
11491 > -linear_group -PGL 10 3 -end \
11492 > -with G -do \
11493 > -group_theoretic_activity \
11494 > -poset_classification_control \
11495 > -problem_label codes_14_4_9_3 \
11496 > -draw_poset \
11497 > -draw_options \
11498 > -embedded -radius 250 \
11499 > -end \
11500 > -end \
11501 > -linear_codes 9 14 \
11502 > -end
11503 ▷ pdflatex codes_14_4_9_3_poset_lvl_13.tex
11504 ▷ open codes_14_4_9_3_poset_lvl_13.pdf
11505
11506
11507 codes_15_6_6_2:
11508 ▷ $(ORBITER) -v 6 \$
11509 ▷ ▷ -define G \$
11510 ▷ ▷ -linear_group -PGL 9 2 -end \$
11511 ▷ ▷ -with G -do \$
11512 ▷ ▷ -group_theoretic_activity \$
11513 ▷ ▷ -poset_classification_control \$
11514 ▷ ▷ ▷ -problem_label codes_15_6_6_2 \$
11515 ▷ ▷ ▷ -draw_poset \$
11516 ▷ ▷ ▷ -draw_options \$
11517 ▷ ▷ ▷ ▷ -embedded -radius 250 \$
11518 ▷ ▷ ▷ ▷ -end \$
11519 ▷ ▷ ▷ -end \$
11520 ▷ ▷ -linear_codes 6 15 \$
11521 ▷ ▷ -end
11522 ▷ pdflatex codes_15_6_6_2_poset_lvl_15.tex
11523 ▷ open codes_15_6_6_2_poset_lvl_15.pdf
11524
11525
11526
11527 # ToDo
11528
11529 codes_16_5_9_3:
11530 ▷ $(ORBITER) -v 6 \$
11531 ▷ ▷ -codes_classify -n 16 -k 5 -q 3 -d 9 -w -W -lex \$
11532 ▷ ▷ -draw_poset \$
11533 ▷ ▷ -end
11534 ▷ ▷
11535
11536 # 5/31/2020: 28 min 22 sec on Mac
11537
11538 #0 : 1 orbits
11539 #1 : 1 orbits
11540 #2 : 1 orbits
11541 #3 : 1 orbits
11542 #4 : 1 orbits
11543 #5 : 1 orbits
11544 #6 : 1 orbits
11545 #7 : 1 orbits
11546 #8 : 1 orbits
11547 #9 : 2 orbits
11548 #10 : 3 orbits
11549 #11 : 4 orbits
11550 #12 : 5 orbits
11551 #13 : 5 orbits
11552 #14 : 4 orbits
11553 #15 : 3 orbits
11554 #16 : 1 orbits
11555 #total: 36
11556
11557
11558 codes_d4:
11559 \$((ORBITER) -v 3 \\n11560 \$define G -linear_group -PGL 4 2 -end \n11561 \$with G -do \\
11562 \$group_theoretic_activity \\
11563 \$poset_classification_control -W \\
11564 \$problem_label codes_r4_d4 -draw_poset \\
11565 \$embedded -end -linear_codes 4 100 \\
11566 \$ -end \\
11567 \$ -end
11568
11569
11570 codes_24_12_8:
11571 \$((ORBITER) -v 6 \\
11572 \$orbiter_path $(ORBITER_PATH) \\
11573 \$define G \\
11574 \$linear_group -PGL 12 2 -end \\
11575 \$with G -do \\
11576 \$group_theoretic_activity \\
11577 \$poset_classification_control \\
11578 \$problem_label codes_24_12_8 \\
11579 \$draw_poset \\
11580 \$draw_options -embedded -radius 250 \\
11581 \$line_width 1.0 -spanning_tree -end \\
11582 \$report -end \\
11583 \$ -end \\
11584 \$linear_codes 8 24 \\
11585 \$ -end
11586 \$pdflatex codes_24_12_8_poset.tex
11587 \$open codes_24_12_8_poset.pdf
11588
11589 #codes_24_12_8_poset_lvl_24.layered_graph
11590
11591 codes_24_12_8_draw:
11592 \$((ORBITER) -v 3 \\
11593 \$draw_layered_graph \\
11594 \$codes_24_12_8_poset_lvl_24.layered_graph \\
11595 \$radius 100 -spanning_tree -embedded \\
11596 \$line_width 0.5 -x_stretch 1.4
glynn arc:

```latex
\$(\text{ORBITER})\ -v 5 \$
```

five points in general:

```latex
\$(\text{ORBITER})\ -v 5 \$
```

codes_q13_12_4:
$$(\text{ORBITER}) -v 6 \$
$$(\text{ORBITER}) -v 2 -\text{conjugacy_classes_Sym}_n 4 \$
$$(\text{ORBITER}) -v 2 -\text{conjugacy_classes_Sym}_n 10 \$
$$(\text{ORBITER}) -v 2 -\text{conjugacy_classes_Sym}_n 15 \$
$$(\text{ORBITER}) -v 2 -\text{character_table_symmetric_group}_4 \$

# Chapter 11 - Combinatorics

# Section 11.1: Combinatorics

SECTION COMBINATORICS:

Sym_4_conj_classes:
$$(\text{ORBITER}) -v 2 -\text{conjugacy_classes_Sym}_n 4 \$

Sym_10_conj_classes:
$$(\text{ORBITER}) -v 2 -\text{conjugacy_classes_Sym}_n 10 \$

Sym_15_conj_classes:
$$(\text{ORBITER}) -v 2 -\text{conjugacy_classes_Sym}_n 15 \$

Char_Sym_4:
$$(\text{ORBITER}) -v 2 -\text{character_table_symmetric_group}_4 \$
Char_Sym_5:
$\text{\textbackslash ORBITER} -v 2 -character_table_symmetric_group 5$

Char_Sym_6:
$\text{\textbackslash ORBITER} -v 2 -character_table_symmetric_group 6$

all_subsets_10_3:
$\text{\textbackslash ORBITER} -v 2 -tree_of_all_k_subsets 10 3$

random_k_subsets:
$\text{\textbackslash ORBITER} -v 4$

create_random_k_subsets 10 5 20

rank_k_subsets_test:
$\text{\textbackslash ORBITER} -v 2$

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

Walsh_matrix_4:
$\text{\textbackslash ORBITER} -v 3$

Walsh_matrix 4

Walsh activity

Walsh_01_4.csv

box_width 10 -bit_depth 24 -partition 3 16 16

Dedekind_10_10:
$\text{\textbackslash ORBITER} -v 3$

Dedekind_numbers 2 10 2 10

Dedekind_30_2:
$\text{\textbackslash ORBITER} -v 3$

Dedekind_numbers 2 30 2 2

Dedekind_100_2:
$\text{\textbackslash ORBITER} -v 3$

Dedekind_numbers 2 100 2 2

elementary_symmetric_functions_4:
$\text{\textbackslash ORBITER} -make_elementary_symmetric_functions 4 4$
elementary_symmetric_functions_8:
$(ORBITER) -make_elementary_symmetric_functions 8 8

LARGE_SET_S0="0,1,2,3,4,5,6,7,8,9,10,11,12"
# identity

LARGE_SET_S1="6,8,9,2,7,10,1,11,0,3,5,4,12"
$(0, 6, 1, 8)(2, 9, 3)(4, 7, 11)(5, 10),\

LARGE_SET_S2="2,0,1,6,3,4,11,5,7,8,10,9,12"
$(0, 2, 1)(3, 6, 11, 9, 8, 7, 5, 4),\

LARGE_SET_S3="12,5,6,11,3,7,10,8,9,1,4,2,0"
$(0, 12)(1, 5, 7, 8, 9)(2, 6, 10, 4, 3, 11),\

LARGE_SET_S4="5,8,10,3,11,0,2,1,12,4,6,7,9"
$(0, 5)(1, 8, 12, 9, 4, 11, 7)(2, 10, 6),\

LARGE_SET_S5="10,11,0,7,12,2,3,1,4,5,8,6,9"
$(0, 10, 8, 4, 12, 9, 5, 2)(1, 11, 6, 3, 7),\

LARGE_SET_S6="3,4,1,9,5,6,8,2,7,11,12,10,0"
$(0, 3, 9, 11, 10, 12)(1, 4, 5, 6, 8, 7, 2),\

LARGE_SET_S7="9,11,0,6,1,3,5,4,2,12,8,7,10"
$(0, 9, 12, 10, 8, 2)(1, 11, 7, 4)(3, 6, 5),\

LARGE_SET_S8="10,2,12,8,0,3,4,1,5,6,9,7,11"
$(0, 10, 9, 6, 4)(1, 2, 12, 11, 7)(3, 8, 5),\

LARGE_SET_S9="1,3,4,10,5,6,9,7,8,11,0,12,2"
11785 #(0, 1, 3, 10)(2, 4, 5, 6, 9, 11, 12), &
11786
11787
11788 LARGE_SET_S10="7, 12, 1, 6, 0, 4, 5, 2, 3, 10, 9, 8, 11"
11789 #(0, 7, 2, 1, 12, 11, 8, 3, 6, 5, 4)(9, 10).
11790
11791
11792 file_S:
11793 > echo ROW,C0"n0,"$(LARGE_SET_S0)"n1,"$(LARGE_SET_S1)"n2,"$(LARGE_SET_S2)"n3,"$(LARGE_SET_S3)"n4,"$(LARGE_SET_S4)"n5,"$(LARGE_SET_S5)"n6,"$(LARGE_SET_S6)"n7,"$(LARGE_SET_S7)"n8,"$(LARGE_SET_S8)"n9,"$(LARGE_SET_S9)"n10,"$(LARGE_SET_S10)"nEND"n >S.csv
11794
11795 Large_set_H5:
11796 > $(ORBITER) -v 10 \\  
11797 > -define G -permutation_group -symmetric_group 13 \\  
11798 > -subgroup_by_generators H5 5 1 $(GENERATORS_H5) -end \\
11799 > -with G -do \\
11800 > -group_theoretic_activity \\
11801 > -report \\
11802 > -end \\
11803 > -with G -do \\
11804 > -group_theoretic_activity \\
11805 > -save_elements_csv "H5_elts.csv" \\
11806 > -end \\
11807 > pdflatex Perm13_Subgroup_H5_5_report.tex \\
11808 > open Perm13_Subgroup_H5_5_report.pdf \\
11809
11810 Large_set_C13:
11811 > $(ORBITER) -v 10 \\  
11812 > -define G -permutation_group -symmetric_group 13 \\  
11813 > -subgroup_by_generators C13 13 1 $(GENERATORS_C13) -end \\
11814 > -with G -do \\
11815 > -group_theoretic_activity \\
11816 > -export_orbiter \\
11817 > -end \\
11818 > -with G -do \\
11819 > -group_theoretic_activity \\
11820 > -report \\
11821 > -end \\
11822 > -with G -do \\
11823 > -group_theoretic_activity \\
11824 > -save_elements_csv "C13_elts.csv" \\
11825 > -end \\
11826 > pdflatex Perm13_Subgroup_C13_13_report.tex \\
11827 > open Perm13_Subgroup_C13_13_report.pdf \\
11828
763
## the following lines were created using -export_orbiter:

```bash
GENERATORS_Perm13_Subgroup_C13_13= \n"11,0,10,12,5,3,7,4,2,8,6,9,1"

Perm13_Subgroup_C13_13:

```(ORBITER) -v 2 \n-subgroup_by_generators Perm13_Subgroup_C13_13 1 \n$(GENERATORS_Perm13_Subgroup_C13_13) \n-end

###

Large_set_mult_C13xS:

```bash
$(ORBITER) -v 10 \n-define G -permutation_group -symmetric_group 13 \n-with G -do \n-group_theoretic_activity \n-multiply_elements_csv_column_major_ordering \n-C13_elts.csv S.csv C13xS.csv \n-end

Large_set_mult_C13xSxH5:

```bash
$(ORBITER) -v 10 \n-define G -permutation_group -symmetric_group 13 -end \n-with G -do \n-group_theoretic_activity \n-multiply_elements_csv_column_major_ordering \n-C13xS.csv H5_elts.csv C13xSxH5.csv \n-end

Large_set_mult_C13xSxH5_apply:

```bash
$(ORBITER) -v 10 \n-define G -permutation_group -symmetric_group 13 -end \n-with G -do \n-group_theoretic_activity \n-apply_elements_csv_to_set \n-C13xSxH5.csv C13xSxH5.images.csv "0,1,2,3" \n-end

domino_portrait:

```bash
$(ORBITER) -v 3 -domino_portrait 7 4 anton_28x32 -end
```
# Section 11.2: Diophantine Systems

**SECTION_DIOPHANT:**

part10:

```
$\text{ORBITER} -v 4 \\
-def A -vector -dense "10,9,8,7,6,5,4,3,2,1" -end \\
-def D -diophant \\
-label part10 \\
-coefficient matrix A \\
-RHS "10,10,1" \\
-x_min_global 0 -x_max_global 10 \\
-end \\
-with D -do \\
-diophant_activity -solve_mckay \\
-end
```

# Finds 42 solutions with 67 backtrack steps

octic_monomials:

```
$\text{ORBITER} -v 4 \\
-def A -vector -dense "1,1,1" -end \\
-def D -diophant \\
-label octic_monomials \
```
11923 \> \> \> -coefficient_matrix A \n11924 \> \> \> -RHS "8,8,1" \n11925 \> \> \> -x_min_global 0 -x_max_global 8 \n11926 \> \> \> -end \n11927 \> \> -with D -do \n11928 \> \> \> -diophant_activity -solve_mckay \n11929 \> \> -end \n11930 \> sort -r octic_monomials.sol >octic_monomials_sorted.txt \n11931 \n11932 #Found 165 solutions with 210 backtrack steps \n11933 # 165=binomial(11,3) \n11934 \n11935 \n11936 solve_test_system: \n11937 \> $(ORBITER) -v 4 \n11938 \> \> -define A -vector -format 7 -dense $(TEST_SYSTEM) -end \n11939 \> \> -define D -diophant \n11940 \> \> \> -label test_system \n11941 \> \> \> -coefficient_matrix A \n11942 \> \> \> -RHS $(TEST_RHS) \n11943 \> \> \> -x_min_global 0 -x_max_global 1 \n11944 \> \> -end \n11945 \n11946 \n11947 McKay_test: \n11948 \> $(ORBITER) -v 4 \n11949 \> \> -define A -vector -format 7 -dense $(TEST_SYSTEM) -end \n11950 \> \> -define D -diophant \n11951 \> \> \> -label test_system \n11952 \> \> \> -coefficient_matrix A \n11953 \> \> \> -RHS $(TEST_RHS) \n11954 \> \> \> -x_min_global 0 -x_max_global 1 \n11955 \> \> -end \n11956 \> \> -with D -do \n11957 \> \> \> -diophant_activity -solve_mckay \n11958 \> \> -end \n11959 \n11960 DLX_test: \n11961 \> $(ORBITER) -v 4 \n11962 \> \> -define A -vector -format 7 -dense $(TEST_SYSTEM) -end \n11963 \> \> -define D \n11964 \> \> \> -diophant -label test_system \n11965 \> \> \> -coefficient_matrix A \n11966 \> \> \> -RHS $(TEST_RHS) \n11967 \> \> \> -x_min_global 0 -x_max_global 1 \n11968 \> \> -end \n11969 \> \> -with D -do \n
766
\[\text{ORBITER} -v 4 \]

linsp6:

\[\text{define A -vector -format 1 -dense "15,10,6,3,1" -end} \]

\[\text{-define D -diophant -label linsp6} \]

\[\text{-coefficient \ matrix A} \]

\[\text{-RHS "15,15,1"} \]

\[\text{-x\_min\_global 0} \]

\[\text{-x\_max\_global 15} \]

\[\text{-end} \]

\[\text{-with D -do} \]

\[\text{-diophant\_activity -solve\_mckay} \]

\[\text{-end} \]

# Found 15 solutions with 22 backtrack steps

linsp7:

\[\text{define A -vector -format 1 -dense "21,15,10,6,3,1" -end} \]

\[\text{-define D -diophant -label linsp7} \]

\[\text{-coefficient\_matrix A} \]

\[\text{-RHS "21,21,1"} \]

\[\text{-x\_min\_global 0} \]

\[\text{-x\_max\_global 21} \]

\[\text{-end} \]

\[\text{-with D -do} \]

\[\text{-diophant\_activity -solve\_mckay} \]

\[\text{-end} \]
12017 # 32 solutions in 45 backtrack steps
12018
12019
12020
12021
12022
12023
12024
12025 linsp30_pt_types:
12026 > $(ORBITER) -v 4 \
12027 > -define A -vector -format 1 -dense "6,4,3" -end \
12028 > -define D -diophant \
12029 > > -label linsp30_pt_types \
12030 > > -coefficient_matrix A \
12031 > > -RHS "29,29,1" -x_bounds "0,1,0,27,0,24" \
12032 > > -end \
12033 > > -with D -do \
12034 > > > -diophant_activity -solve_mckay \
12035 > > > -end
12036
12037 linsp30_pt_distribution:
12038 > $(ORBITER) -v 4 \
12039 > -define A -vector -format 6 -dense \
12040 > > "1,1,1,1,1,0,0,5,2,5,2,1,5,3,7,10,1,10,1,0,10,3,21" \
12041 > > -end \
12042 > > -define D -diophant \
12043 > > > -label linsp30_pt_distribution \
12044 > > > -coefficient_matrix A \
12045 > > > -RHS "30,30,1,7,7,1,135,135,1,96,96,1,0,351,2,0,276,2" \
12046 > > > -x_min_global 0 -x_max_global 30 \
12047 > > > -end \
12048 > > > -with D -do \
12049 > > > > -diophant_activity -solve_mckay \
12050 > > > > -end \
12051 > > > > -with D -do \
12052 > > > > > -diophant_activity -draw_as_bitmap 20 8 \
12053 > > > > > -end
12054
12055
12056
12057 ###################################################################
12058 # Section 11.4: Combinatorial Linear Spaces
12059
12060 ###
12061 # geometry builder:
12062 ###
12063
768
geo_pasz:
    $(ORBITER) -v 8 \
    -define Test_lines -set -loop 1 7 1 -end \
    -define Geo -geometry_builder \
    -V 6 -B 4 -TDO 2 -fuse 1 \
    -fname_GEO pasch \ 
    -test Test_lines \ 
    -end

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

geo_7_3:
    $(ORBITER) -v 2 \
    -define Test_lines -set -loop 3 8 1 -end \
    -define Geo -geometry_builder \
    -V 7 -B 7 -TDO 3 \
    -fuse 1 -fname_GEO 7_3 \
    -test Test_lines \ 
    -end

geo_7_3_no_square.test:
    $(ORBITER) -v 2 \
    -define Test_lines -set -loop 3 8 1 -end \
    -define Geo -geometry_builder \
    -V 7 -B 7 -TDO 3 \
    -fuse 1 -fname_GEO 7_3_nst \
    -test Test_lines \ 
    -no_square_test \ 
    -end

g_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 \ 
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 \ 
xout 1000000 -yout 1000000 \ 
nodes_empty \ 
scale 0.5 -line_width 0.3 \ 
-end \ 
tree_draw -file 7_3_tree.txt -end \ 
pdflatex 7_3_tree_drawer.tex \ 
open 7_3_tree_drawer.pdf \ 
geo_7_3_orderly_mem_debug: \ 
$(ORBITER) -v 20 \ 
-memory_debug 2 \ 
define Test_lines -set -loop 3 8 1 -end \ 
define Geo -geometry_builder \ 
-V 7 -B 7 -TDO 3 \ 
fuse 1 -fname_GEO 7_3 \ 
test Test_lines \ 
-search_tree \ 
-orderly \ 
-end \ 
geo_8_3: \ 
$(ORBITER) -v 2 \ 

define Test_lines -set -loop 3 9 1 -end 
define Geo -geometry_builder 
define Test_lines -set -loop 3 8 8 -TDO 3 
-fuse 1 -fname_GEO_8_3 
test Test_lines 
-end
-print at line 4
 geo: 0 11 18 29 30 38 44 54
 ago=48
go
geo 9
$(ORBITER) -v 2 
define Test_lines -set -loop 3 10 1 -end 
define Geo -geometry_builder 
-V 9 -B 9 -TDO 3 
fuse 1 -fname_GEO_9_3 
test Test_lines 
-end
geo 10
$(ORBITER) -v 10 
draw incidence structure description 
-width 60 -with 10 6 -end 
define C -combinatorial_objects 
-file_of_incidence_geometries 

# 10 geos
# 8/26/2021: 0 sec on Mac
12205  \begin{verbatim}
12206  10.3.inc 10 10 30 \
12207  -end \ 
12208  -with C -do \ 
12209  -combinatorial_object_activity \ 
12210  10.3.inc \ 
12211  -end
12212  pdflatex 10.3_inc_incma.tex
12213  open 10.3_inc_incma.pdf
12214
12215
12216 geo_10.3_orderly:
12217  $(ORBITER) -v 20 \ 
12218  define Test_lines -set -loop 4 11 1 -end \ 
12219  define Geo -geometry_builder \ 
12220  V 10 -B 10 -TDO 3 -fuse 1 \ 
12221  fname GEO 10_3 \ 
12222  test Test_lines \ 
12223  orderly \ 
12224  -end
12225
12226 geo_10.3_orderly_mem_debug:
12227  $(ORBITER) -v 2 \ 
12228  memory_debug 2 \ 
12229  define Test_lines -set -loop 4 11 1 -end \ 
12230  define Geo -geometry_builder \ 
12231  V 10 -B 10 -TDO 3 -fuse 1 \ 
12232  fname GEO 10_3 \ 
12233  test Test_lines \ 
12234  orderly \ 
12235  -end
12236
12237
12238 geo_10.3_tree:
12239  $(ORBITER) -v 20 \ 
12240  define Test_lines -set -loop 0 11 1 -end \ 
12241  define GEO -geometry_builder \ 
12242  V 10 -B 10 -TDO 3 -fuse 1 \ 
12243  fname GEO 10_3 \ 
12244  search_tree \ 
12245  test Test_lines \ 
12246  -end
12247  $(ORBITER) -v 20 \ 
12248  draw_options -embedded -radius 40 \ 
12249  paperheight 220 \ 
12250  paperwidth 330 \ 
12251  -xin 10000 -yin 10000 \ 

772
geo.10_3_tree_path:
$(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 \
-search_tree \
-test Test_lines \
-end \
$(ORBITER) -v 20 \
-draw_options -embedded -radius 20 \
-paperheight 220 \
-paperwidth 330 \
-xin 10000 -yin 10000 \
-xout 1000000 -yout 500000 \
-scale 2 -line_width 0.3 \
-end \
-tree_draw \
-restrict 2 \
-file 10_3_tree.txt \
-select_path "0,0,15,26,46,56,72,80,93,106,119" \
-end \
pdflatex 10_3_tree_draw.tex
open 10_3_tree_draw.pdf

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 \n-with C -do \n-combinatorial_object_activity \n-draw_incidence_matrices \nDesargues_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 0\n2 112 119\n3 89 112 119\n4 118 89 82\n5 106 114 69 107 111\n6 85 105 112 99 83 61\n7 94 105 113 85 35 83 6\n8 26 119 55 105 92 79 74 48\n9 119 93 106 15 26 79 55 73 47\n10 0 119 93 106 15 26 79 55 73 47\n-1"

edesargues_path_can_draw:

echo $(DESARGUES_PATH_CANONICAL_ANCESTOR) >Desargues_path_can_anc.inc
$ORBITER -v 10 
-draw_incidence_structure_description 
-width 60 -with_10_6 -end 
-combinatorial_objects 
-draw_incidence_matrices 
Desargues_path_can_anc 
-end 
pdflatex Desargues_path_can_anc_incma.tex 
open Desargues_path_can_anc_incma.pdf 

geo_11_3:
$ORBITER -v 2 
-define Test_lines -set -loop 4 12 1 -end 
define Geo -geometry_builder 
-V 11 -B 11 -TDO 3 
-fuse 1 -fname_GEO 11_3 
test Test_lines 
-end 

# 31 geos 
# 8/26/2021: 0 sec on Mac
geo_12.3:
  $(ORBITER) -v 2 \n  -define Test_lines -set -loop 4 13 1 -end \n  -define Geo -geometry_builder \n  -V 12 -B 12 -TDO 3 \n  -fuse 1 -fname_GEO 12.3 \n  -test Test_lines \n  -output_to_sage_file \n  -end

# 229 geos
#User time: 0.45 of a second, dt=45 tps = 100
#nb_calls_to_densenauty=24586

geo_12.3_orderly:
  $(ORBITER) -v 2 \n  -define Test_lines -set -loop 4 13 1 -end \n  -define Geo -geometry_builder \n  -V 12 -B 12 -TDO 3 \n  -fuse 1 -fname_GEO 12.3 \n  -test Test_lines \n  -end

geo_13.3:
  $(ORBITER) -v 2 \n  -define Test_lines -set -loop 4 14 1 -end \n  -define Geo -geometry_builder \n  -V 13 -B 13 -TDO 3 \n  -fuse 1 -fname_GEO 13.3 \n  -test Test_lines \n  -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 \n  -define Test_lines -set -loop 4 14 1 -end \n  -define Geo -geometry_builder \n  -V 13 -B 13 -TDO 3 \n  -fuse 1 -fname_GEO 13.3 \n
12389 ▶ ▶ ▶ -test Test_lines \n12390 ▶ ▶ ▶ -f_orderly \n12391 ▶ ▶ -end
12392
12393
12394
12395 geo_14_3:
12396 ▶ $(ORBITER) -v 2 \n12397 ▶ ▶ -define Test_lines -set -loop 4 15 1 -end \n12398 ▶ ▶ -define Geo -geometry_builder \n12399 ▶ ▶ ▶ -V 14 -B 14 -TDO 3 \n12400 ▶ ▶ ▶ -fuse 1 -fname_GEO 14_3 \n12401 ▶ ▶ ▶ -test Test_lines \n12402 ▶ ▶ ▶ -end
12403
12404 # 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
12405 #User time: 0:55
12406 #nb_calls_to_densenauty=2089344
12407
12408
12409 geo_14_3_orderly:
12410 ▶ $(ORBITER) -v 2 \n12411 ▶ ▶ -define Test_lines -set -loop 4 15 1 -end \n12412 ▶ ▶ -define Geo -geometry_builder \n12413 ▶ ▶ ▶ -V 14 -B 14 -TDO 3 \n12414 ▶ ▶ ▶ -fuse 1 -fname_GEO 14_3 \n12415 ▶ ▶ ▶ -test Test_lines \n12416 ▶ ▶ ▶ -f_orderly \n12417 ▶ ▶ ▶ -end
12418
12419 #User time: 0:50
12420
12421
12422 15_3.inc:
12423 ▶ $(ORBITER) -v 2 \n12424 ▶ ▶ -define Test_lines -set -loop 4 16 1 -end \n12425 ▶ ▶ -define Geo -geometry_builder \n12426 ▶ ▶ ▶ -V 15 -B 15 -TDO 3 \n12427 ▶ ▶ ▶ -fuse 1 -fname_GEO 15_3 \n12428 ▶ ▶ ▶ -test Test_lines \n12429 ▶ ▶ ▶ -end
12430
12431 # 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
12432 # 8 min 11 sec on Mac
12433 # running out of memory
12434
12435
g4:
12437    $(ORBITER) -v 2 \n12438      -define Test_lines -set -loop 4 16 1 -end \n12439      -define Geo -geometry_builder \n12440      -V 15 -B 15 -TDO 3 \n12441      -define Test -geometry Builder \n12442      -fuse 1 -fname GEO 15_3_g4 \n12443      -girth 4 \n12444      -search_tree \n12445      -test Test_lines \n12446    -end
12447    $(ORBITER) -v 2 \n12448      -draw_options -embedded -radius 50 \n12449      -nodes_empty \n12450      -scale 0.5 -line_width 0.3 -end \n12451    -tree_draw -file 15_3_g4_tree.txt -end
12452    open 15_3_g4_tree_draw.pdf
12453
12454    # The unique Cremona Richmond configuration with group of order 720
12455    #User time: 0 of a second, dt=0 tps = 100
12456    #nb_calls_to_densenauty=23
12457
12458    #-sideways
12459
12460
12461
12462
12463    geo_15_3_g4:
12464    $(ORBITER) -v 2 \n12465      -memory_debug 2 \n12466      -define Test_lines -set -loop 4 18 1 -end \n12467      -define Geo -geometry_builder \n12468      -V 17 -B 17 -TDO 3 \n12469      -fuse 1 -fname GEO 17_3_g4 \n12470      -girth 4 \n12471      -test Test_lines \n12472      -orderly \n12473    -end
12474
12475    # 1 sol
12476
12477    geo_18_3_g4:
12478    $(ORBITER) -v 2 \n12479      -define Test_lines -set -loop 4 19 1 -end \n12480      -define Geo -geometry_builder \n
777
12481 > > > -V 18 -B 18 -TDO 3 \\
12482 > > > -fuse 1 -fname_GEO 18_3_g4 \\
12483 > > > -girth 4 \\
12484 > > > -search_tree \\
12485 > > > -test Test_lines \\
12486 > > -end
12487
12488 # 4 sol, 1 sec
12489
12490 geo_19_3_g4:
12491 $(ORBITER) -v 2 \\
12492 > > -define Test_lines -set -loop 4 20 1 -end \\
12493 > > -define Geo -geometry_builder \\
12494 > > > -V 19 -B 19 -TDO 3 \\
12495 > > > -fuse 1 -fname_GEO 19_3_g4 \\
12496 > > > -girth 4 \\
12497 > > > -test Test_lines \\
12498 > > -end
12499
12500
12501 # 14 sol, 10 sec on Mac
12502
12503 geo_20_3_g4:
12504 $(ORBITER) -v 2 \\
12505 > > -define Test_lines -set -loop 4 21 1 -end \\
12506 > > -define Geo -geometry_builder \\
12507 > > > -V 20 -B 20 -TDO 3 \\
12508 > > > -fuse 1 -fname_GEO 20_3_g4 \\
12509 > > > -girth 4 \\
12510 > > > -test Test_lines \\
12511 > > -end
12512
12513 # 162 sol, User time: 2:5 on Mac
12514
12515 geo_21_3_g4:
12516 $(ORBITER) -v 2 \\
12517 > > -define Test_lines -set -loop 4 22 1 -end \\
12518 > > -define Geo -geometry_builder \\
12519 > > > -V 21 -B 21 -TDO 3 \\
12520 > > > -fuse 1 -fname_GEO 21_3_g4 \\
12521 > > > -girth 4 \\
12522 > > > -test Test_lines \\
12523 > > -end
12524
12525
12526 geo_15_4:
12527 $(ORBITER) -v 2 \

778
\define Test_lines -set -loop 4 16 1 -end \n\define Geo -geometry builder \n\V 15 -B 15 -TDO 4 \n\fuse 1 -fname GEO 15.4 \n\search_tree \n\test Test_lines \n\end

$(ORBITER) -v 2 \n\draw_options -embedded -radius 50 \n\nodes_empty \n\scale 0.5 -line_width 0.3 -end \n\tree_draw -file 15_4_tree_draw.txt -end

pdflatex 15_4_tree_draw.tex
open 15_4_tree_draw.pdf

geo_16_4_g4:
$(ORBITER) -v 2 \n\define Test_lines -set -loop 4 17 1 -end \n\define Geo -geometry builder \n\V 16 -B 16 -TDO 4 \n\fuse 1 -fname GEO 16_4_g4 \n\girth 4 \n\test Test_lines \n\end

# none

geo_LSQ6:
$(ORBITER) -v 2 \n\define Test_lines -set -loop 1 19 1 -end \n\define Geo -geometry_builder \n\V 6,6,6 -B 1,1,1,36 -TDO \n"1,0,0,6, 0,1,0,6, 0,0,1,6" \n\fuse 3 -fname GEO LSQ6 \n\test Test_lines \n\end
12575
12576 geo_16:
12577 ▷ $(ORBITER) -v 2
12578 ▷ ▷ -define Test_lines -set -loop 3 17 1 -end
12579 ▷ ▷ -define Geo -geometry_builder
12580 ▷ ▷ ▷ -V 16 -B 20 -TDO 5
12581 ▷ ▷ ▷ -fuse 1 -fname_GEO geo_16
12582 ▷ ▷ ▷ -test Test_lines
12583 ▷ ▷ -end
12584
12585
12586 40_4_g4.inc:
12587 ▷ $(ORBITER) -v 5
12588 ▷ ▷ -define Test_lines -set -loop 0 41 1 -end
12589 ▷ ▷ -define Geo -geometry_builder
12590 ▷ ▷ ▷ -V 40 -B 40 -TDO 4
12591 ▷ ▷ ▷ -fuse 1
12592 ▷ ▷ ▷ -fname_GEO 40_4_g4
12593 ▷ ▷ ▷ -girth 4
12594 ▷ ▷ ▷ -search_tree
12595 ▷ ▷ ▷ -special_test_not_orderly
12596 ▷ ▷ ▷ -test Test_lines
12597 ▷ ▷ ▷ -output_to_sage_file
12598 ▷ ▷ ▷ -output_to_inc_file
12599 ▷ ▷ ▷ -end
12600 ▷ $(ORBITER) -v 2
12601 ▷ ▷ -draw_options -embedded -radius 50
12602 ▷ ▷ ▷ -xin 10000 -yin 10000
12603 ▷ ▷ ▷ -xout 1000000 -you 1000000
12604 ▷ ▷ ▷ -nodes_empty
12605 ▷ ▷ ▷ -scale 0.5 -line_width 0.3 -end
12606 ▷ ▷ -tree_draw -file 40_4_g4_tree.txt -end
12607 ▷ ▷ pdflatex 40_4_g4_tree_draw.tex
12608 ▷ ▷ open 40_4_g4_tree_draw.pdf
12609
12610
12611 #-special_test_not_orderly is important for speed purposes
12612 # 2 geos, ago=51840^2
12613 # 40_4_g4.inc
12614
12615
12616 geo_63_3_g6:
12617 ▷ $(ORBITER) -v 2
12618 ▷ ▷ -define Test_lines -set -loop 4 64 1 -end
12619 ▷ ▷ -define Geo -geometry_builder
12620 ▷ ▷ ▷ -V 63 -B 63 -TDO 3
12621 ▷ ▷ ▷ -fuse 1 -fname_GEO 63_3_g6
design_PG_2.3:

design_PG_2.4:

design_PG_2.3_table.create:
12669 # written file PG_2.13_design_table.csv
12670 # 1108800 designs
12671 #User time: 7:30
12672
12673 design_PG_2.3_group_5:
12674    $(ORBITER) -v 2 \n12675    -define F -finite_field -q 3 -end \n12676    -define D -design -field F -family PG_2.q -end \n12677    -define T -design_table D "PG_2.13" Sym13 -end \n12678    -define LSW -large_set_with_symmetry_assumption T \n12679    -H "5" $(GENERATORS_H5) \n12680    -N "1200" $(GENERATORS_N5) \n12681    -prefix "H5" \n12682    -selected_orbit_length 5 \n12683    -end \n12684    -with LSW -do \n12685    -large_set_with_symmetry_assumption_activity \n12686    -normalizer_on_orbits_of_a_given_length 5 \n12687    -end
12688
12689 #H5_N_orbit_reps.csv
12690 # 678 orbits
12691 #User time: 2:39
12692
12693 design_PG_2.3_group_5_sol_0:
12694    $(ORBITER) -v 2 \n12695    -define F -finite_field -q 3 -end \n12696    -define D -design -field F -family PG_2.q -end \n12697    -define T -design_table D "PG_2.13" Sym13 -end \n12698    -define LSW -large_set_with_symmetry_assumption T \n12700    -H "5" $(GENERATORS_H5) \n12701    -N "1200" $(GENERATORS_N5) \n12702    -prefix "H5" \n12703    -selected_orbit_length 5 \n12704    -end \n12705    -with LSW -do \n12706    -large_set_with_symmetry_assumption_activity \n12707    -read_solution_file 5 case_0_sol.txt \n12708    -end
12709
12710 wreath_product_designs_n4_k2_inc.txt:
12711    $(ORBITER) -v 8 \n12712    -define D -design -wreath_product_designs 4 2 -end \n12713    -with D -do \n12714    -design_activity \n12715    -export_inc \n
782
wreath_product.designs_n8_k6_inc.txt:
$(ORBITER) -v 8 \
-define 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 \
-define gens -vector -dense "1,2,3,4,5,6,0" -end \n-define 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-Kramer_Mesner_matrix 2 3 \n-draw_poset \n-draw_options -embedded -sideways -radius 50 \n-scale 0.5 -line_width 0.3 -end \n-end \n-define Orb -orbits -group G \n-on_subsets 3 Control \n-end

$(ORBITER) -v 4 \
-define A -vector -file C7 KM 2 3.csv -end \n-define D -diophant \n-label "C7 KM 2 3_system" \n-coefficient_matrix A \n-RHS_constant "1,1,1" \n-x_min_global 0 -x_max_global 1 \n-end \n-with D -do \n-diophant_activity -solve_mckay \n-end

# to create simple 7-designs on 33 points with block size 8 and lambda = 10 invar
iant under PGGL(2,32):

12763
12764 KM_PGGL_2_32:
12765 $(\text{ORBITER}) -v 3$
12766 \text{define Control -poset\_classification\_control} \\n12767 \text{define} \ W -depth 8 \n12768 \text{define Kramer\_Mesner\_matrix 7 8} \n12769 \text{-draw\_poset} \n12770 \text{-draw\_options -embedded -sideways -radius 50} \n12771 \text{-scale 0.5 -line\_width 0.3 -end} \n12772 \text{-problem label KM PGGL 2 32 -W -depth 10} \n12773 \text{-Kramer\_Mesner\_matrix 8 10} \n12774 \text{-draw\_poset} \n12775 \text{-draw\_options -embedded -sideways -radius 50} \n12776 \text{-scale 0.5 -line\_width 0.3 -end} \n12777 \text{-define G -linear\_group -PGGL 2 32 -end} \n12778 \text{-define Orb -orbits -group G} \n12779 \text{-on\_subsets 8 Control} \n12780 \text{-end} \n12781 $(\text{ORBITER}) -v 4$
12782 \text{define A -vector -file KM_PGGL_2_32_KM_7_8.csv} \n12783 \text{define D -diophant} \n12784 \text{-label "KM_PGGL_2_32_KM_7_8\_system"} \n12785 \text{-coefficient\_matrix A} \n12786 \text{-RHS\_constant "10,10,1"} \n12787 \text{-x\_min\_global 0 -x\_max\_global 1} \n12788 \text{-end} \n12789 \text{-with D -do} \n12790 \text{-diophant\_activity -solve\_mckay} \n12791 \text{-end} \n12792 \text{-end} \n12793 \text{KM_PSL_3_5:} \n12794 \text{define Control -poset\_classification\_control} \n12795 \text{-problem_label KMPSL_3_5 -W -depth 10} \n12796 \text{-Kramer\_Mesner\_matrix 8 10} \n12797 \text{-draw\_poset} \n12798 \text{-draw\_options -embedded -sideways} \n12799 \text{-radius 50 -scale 0.5 -line\_width 0.3 -end} \n12800 \text{-end} \n12801 \text{-define G -linear\_group -PSL 3 5 -end} \n
784
12809 \  \  \  -define Orb -orbits -group G \\
12810 \  \  \  -on_subsets 10 Control \\
12811 \  \  \  -end \\
12812 \  \  \  $(ORBITER) -v 2 -draw_matrix \\
12813 \  \  \  -input_csv_file KM_PSL_3_5_KM_8_10.csv \\
12814 \  \  \  -box_width 10 -bit_depth 8 -partition 3 42 174 -end \\
12815 \  \  \  $(ORBITER) -v 4 \\
12816 \  \  \  -define A -vector -file KM_PSL_3_5_KM_8_10.csv -end \\
12817 \  \  \  -define D -diophant \\
12818 \  \  \  \  \  \  -label "KM_PSL_3_5_KM_8_10_system" \\
12819 \  \  \  \  \  \  -coefficient_matrix A \\
12820 \  \  \  \  \  \  -RHS_constant "93,93,1" \\
12821 \  \  \  \  \  \  -x_min_global 0 -x_max_global 1 \\
12822 \  \  \  \  \  \  -end \\
12823 \  \  \  \  \  \  -with D -do \\
12824 \  \  \  \  \  \  -diophant_activity -solve_mckay \\
12825 \  \  \  \  \  \  -end \\
12854
12855
12859 # Section 11.6: Design Theory - Large Sets
12860
12861 SECTION DESIGN THEORY LARGE SETS:
12862
12863 AG_2.3.inc:
12864 $(ORBITER) -v 2 \\
12865 \  \  \  -define Geo -geometry_builder \\
12866 \  \  \  \  \  \  -V 9 -B 12 \\
12867 \  \  \  \  \  \  -TD0 4 -fuse 1 \\
12868 \  \  \  \  \  \  -fname_GEO AG_2.3 \\
12869 \  \  \  \  \  \  -test 3,4,5,6,7,8,9 \\
12870 \  \  \  \  \  \  -end \\
12871 #9 12 3 \\
12872 #0 13 22 27 35 41 47 53 55 59 71 76 \\
12873 #1 1 \\
12874 #432 \\
12875
785
LS_AG_2_3_design_table_create:
\$(ORBITER) -v 5 \
   -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

# creates AG_2_3_design_table.csv
# and AG_2_3.makefile

#0,0,13,22,27,35,41,47,53,55,59,71,76
# is the first design in AG_2_3_design_table.csv

#poset_orbit_node::init_root_node storing strong generators for a group of order 362880
# stabilizer order 432
# 840 designs

#User time: 0.13 of a second, dt=13 tps = 100

AG_2_3_on_designs:
\$(ORBITER) -v 2 \
   -define gens -vector -file AG_2_3_gens.csv -end \n   -define G -permutation_group \n   -bsgs AG_2_3 "AG_2_3" 840 362880 "0,1,2,3,4,5,6,7" 8 gens -end \n   -define Orb -orbits -group G \n   -on_points \n   -end

#Written file AG_2_3_stab_orb_0.makefile of size 239

# the stabilizer of the first design:

AG_2_3_stab_orb_0:
12902 \>$\$(ORBITER) -v 2 \$
12903 \>$ -define gens -vector -file AG_2_3_stab_orb_0_gens.csv -end \$
12904 \>$ -define G -permutation_group \$
12905 \>$ -bsgs AG_2_3_stab_orb_0 "AG_2_3_stab_orb_0" \$
12906 \>$ \> 840 432 "0,1,2,3,4,5,6,7,8" 5 gens \$
12907 \>$ -end \$
12908 \>$ -define G -permutation_group -from G \$
12909 \>$ \> -restricted_action $(LARGE_SET_AG_2_3_NEIGHBOR_SET) \$
12910 \>$ -end \$
12911 \>$ -with Gr -do \$
12912 \>$ -group_theoretic_activity \$
12913 \>$ \> -export_orbiter \$
12914 \>$ -end \$
12915 \>$ \$
12916
12917 AG_2_3_stab_orb_0_Perms40_res192:
12918 \>$ \$(ORBITER) -v 2 \$
12919 \>$ -define gens -vector -file Perm840_res192_gens.csv -end \$
12920 \>$ -define G -permutation_group \$
12921 \>$ -bsgs Perm840_res192 "Perm840 {\rm rm res192}" \$
12922 \>$ \> 192 432 "0,1,2,3,4,5,6,7,8" 4 gens \$
12923 \>$ -end \$
12924 \>$ -with G -do \$
12925 \>$ -group_theoretic_activity \$
12926 \>$ \> -report \$
12927 \>$ -end \$
12928 \>$ pdfflatex Perm840_res192_report.tex
12929 \>$ open Perm840_res192_report.pdf
12930
12931
12932
12933 LS_AG_2_3_disjoint_sets_graph_and_cliques:
12934 \>$ \$(ORBITER) -v 2 \$
12935 \>$ -define Gamma -graph \$
12936 \>$ \> -disjoint_sets_graph \$
12937 \>$ \> AG_2_3_design_table.csv \$
12938 \>$ -end \$
12939 \>$ -with Gamma -do \$
12940 \>$ -graph_theoretic_activity \$
12941 \>$ \> -save \$
12942 \>$ -end \$
12943 \>$ -with Gamma -do \$
12944 \>$ -graph_theoretic_activity \$
12945 \>$ \> -find_cliques -target_size 7 -end \$
12946 \>$ -end \$
12947 \>$ -print_symbols
12948
LS_AG_2_3_disjoint_sets_split:
$ (ORBITER) -v 4 \n -define Gamma -graph -load \n AG_2_3_design_table_disjoint_sets_colored_graph \n -end \n -with Gamma -do \n -graph_theoretic_activity \n -split_by_clique "0" "0" \n -end

LS_AG_2_3_export_solutions:
$ (ORBITER) -v 20 \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 \n -with D -do \n -design_activity \n -extract_solutions_by_index "AG_2_3" Sym9 \n AG_2_3_design_table_disjoint_sets_sol.csv \n solutions.csv \n -" \n -end

User time: 0.39 of a second, dt=39 tps = 100
solutions.csv
SECTION DESIGN THEORY DELANDTSHEER DOYEN:

DD_PP4:
$\text{(ORBITER)} -v 6$
- Delandtsheer Doyen $(PP4) (PP4 \_GROUP1) (PP4 \_MASK1)$
- end

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

DD_CC:
$\text{(ORBITER)} -v 6$
- Delandtsheer Doyen -search wrt subgroup
- $(DELANDTSHEER DOYEN PROBLEM COLBOURN COLBOURN 7 13)$
- $(DELANDTSHEER DOYEN PROBLEM COLBOURN COLBOURN 7 13 GROUP1)$
- $(DELANDTSHEER DOYEN PROBLEM COLBOURN COLBOURN 7 13 MASK1)$
- end

#target level: 6
#k2: 15
#number of k-orbits at target level: 1774964
# creates DD_CC_7_13_pair_covering.csv

DD_CC_system:
$\text{(ORBITER)} -v 4$
- define D -diophant -label DD_CC_7_13
- problem of Steiner type 45 DD_CC_7_13_pair_covering.csv
- has sum 3
- end
- with D -do
- diophant_activity -solve_mckay
- end
13043 |
13044 |
13045 |
13046 # no solution
13047 |
13048 |
13049 |
13050 # 18603 = 27 * 53 * 13
13051 |
13052 DD_M1_G1:
13053 > $(ORBITER) -v 4 \n13054 > >> -Delandtsheer_Doyen \n13055 > >> >> $(DELANDTSHEER_DOYEN_PROBLEM_27_53) \n13056 > >> >> $(DELANDTSHEER_DOYEN_PROBLEM_27_53_GROUP1) \n13057 > >> >> $(DELANDTSHEER_DOYEN_PROBLEM_27_53_MASK1) \n13058 > >> >> -end
13059 |
13060 DD_M1_G1_S:
13061 > $(ORBITER) -v 4 \n13062 > >> -Delandtsheer_Doyen \n13063 > >> >> $(DELANDTSHEER_DOYEN_PROBLEM_27_53) \n13064 > >> >> $(DELANDTSHEER_DOYEN_PROBLEM_27_53_GROUP1) \n13065 > >> >> $(DELANDTSHEER_DOYEN_PROBLEM_27_53_MASK1) \n13066 > >> >> -singletons \n13067 > >> >> -end
13068 |
13069 |
13070 DD_PG_2_4_M1_G1:
13071 > $(ORBITER) -v 4 \n13072 > >> -Delandtsheer_Doyen \n13073 > >> >> $(DELANDTSHEER_DOYEN_PROBLEM_3_7) \n13074 > >> >> $(DELANDTSHEER_DOYEN_PROBLEM_3_7_GROUP1) \n13075 > >> >> $(DELANDTSHEER_DOYEN_PROBLEM_3_7_MASK1) \n13076 > >> >> -end
13077 |
13078 PG_2_27_special:
13079 > $(ORBITER) -v 2 \n13080 > >> -define F -finite_field -q 27 -override_polynomial 46 -end \n13081 > >> -define P -projective_space -n 2 -field F -v 0 -end \n13082 > >> -with P -do -projective_space_activity \n13083 > >> >> -cheat_sheet \n13084 > >> >> -end
13085 > pdflatex PG_2_27.tex
13086 > open PG_2_27.pdf
13087 |
13088 |
13089 |
# Section 11.8: Tactical Decompositions

SECTION TACTICAL_DECOMPOSITIONS:

max_arc_16_4_start:

max_arc_16_4_convert Stack_tdo:

max_arc_16_4_refine:

max_arc_16_4r_print:

# Chapter 12 - Finite Geometry

SECTION SPREADS:

create_spread_9a:
# desarguesian spread, ago = 5760

create spread 9b:

```bash
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
```

# Hall spread, ago = 1920

create spread 25_7:

```bash
define F -finite_field -q 5 -end 
define G -linear_group -PGL 4 F -end 
define S -spread -kernel_field F 
group G -k 2 -catalogue 7 
end
```

SPREAD_SET 27 RAO RAO="

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

792
create_spread_Rao_Rao_27:

\[
\begin{align*}
\text{define } F & \text{ -finite_field -q 3 } \text{-end } \\
\text{define SS } & \text{-vector -dense $(\text{SPREAD_SET_27_RAO_RAO})$ -end } \\
\text{define G } & \text{-linear_group -PGL 6 F -end } \\
\text{define S } & \text{-spread -kernel_field F } \\
\text{group G } & \text{-k 3 -spread_set SS } \\
\text{-end }
\end{align*}
\]

SPREAD_S27_RAO_RAO="

0, 33879, 5418, 13103, 30556, 22107, 27225, 4045, 24924, 31961, 
3196, 30100, 28081, 25862, 1339, 6696, 8242, 11747, 14000, 14705, 
9784, 17843, 20772, 9271, 19413, 18678, 16109, 23924"

desarguesian_spread_in_PG_3_2:

\[
\begin{align*}
\text{define } FQ & \text{-finite_field -q 4 } \text{-end } \\
\text{define Fq } & \text{-finite_field -q 2 } \text{-end } \\
\text{with FQ -and Fq -do -finite_field_activity } \\
\text{-cheat_sheet_desarguesian_spread 2 -end }
\end{align*}
\]

\text{pdflatex Desarguesian_Spread_3_2.tex}

\text{open Desarguesian_Spread_3_2.pdf}

desarguesian_spread_in_PG_5_2:

\[
\begin{align*}
\text{define } FQ & \text{-finite_field -q 8 } \text{-end } \\
\text{define Fq } & \text{-finite_field -q 2 } \text{-end } \\
\text{with FQ -and Fq -do -finite_field_activity } \\
\text{-cheat_sheet_desarguesian_spread 2 -end }
\end{align*}
\]

\text{pdflatex Desarguesian_Spread_5_2.tex}

\text{open Desarguesian_Spread_5_2.pdf}

desarguesian_spread_in_PG_3_4:

\[
\begin{align*}
\text{define } FQ & \text{-finite_field -q 16 } \text{-end } \\
\text{define Fq } & \text{-finite_field -q 4 } \text{-end } \\
\text{with FQ -and Fq -do -finite_field_activity } \\
\text{-cheat_sheet_desarguesian_spread 2 -end }
\end{align*}
\]

\text{pdflatex Desarguesian_Spread_3_4.tex}

\text{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 \$
```

```
$\text{-define F -finite_field -q 2 -end}$
```

```
$\text{-define P -projective_space -n 3 -field F -v 0 -end}$
```

```
$\text{-define C -spread_classifier}$
```

```
$\text{-projective_space P}$
```

```
$\text{-k 2}$
```

```
$\text{-starter_size 5}$
```

```
$\text{-poset_classification_control}$
```

```
$\text{-draw_options}$
```

```
$\text{-embedded}$
```

```
$\text{-end}$
```

```
$\text{-draw_poset}$
```

```
$\text{-problem_label spreads_2.2}$
```

```
$\text{-end}$
```

```
$\text{-output_prefix "."}$
```

```
$\text{-end}$
```

```
$\text{-with C -do -spread_classify_activity}$
```

```
$\text{-compute_starter}$
```

```
$\text{-problem_label spreads_2.2}$
```

```
$\text{-W -depth 5}$
```

```
$\text{-report -end}$
```

```
$\text{-end}$
```

```
$\text{pdflatex spreads_2.2.posetlvl_5.tex}$
```

```
open spreads_2.2.posetlvl_5.pdf
```

classify spreads 16 4:

```
$\text{ORBITER} -v 4 \$
```

```
$\text{-define F -finite_field -q 4 -end}$
```

```
$\text{-define P -projective_space -n 3 -field F -v 0 -end}$
```

```
$\text{-define C -spread_classifier}$
```

```
$\text{-projective_space P}$
```

```
$\text{-k 2}$
```

```
$\text{-starter_size 17}$
```

```
pdflatex spreads_2.2.posetlvl_5.tex
```

```
open spreads_2.2.posetlvl_5.pdf
```

```
pdflatex spreads_2.2.posetlvl_5.tex
```

```
open spreads_2.2.posetlvl_5.pdf
```

```
classify_spreads_25_starter_lift_case_0:

$(ORBITER) -v 3 \\n-define F -finite_field -q 5 -end \\n-define P -projective_space -n 3 -field F -v 0 -end \\n-define C -spread_classifier \\n-projective_space P \\n-k 2 \\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-embedded \\n-embedded \\n-W \\n-draw_poset \\n-problem_label_spreads_25 \\n-end \\n-output_prefix "" \

13325  ▶  ▶  -end \\
13326  ▶  ▶  -with C -do -spread_classify_activity \\
13327  ▶  ▶  ▶  -compute_starter \\
13328  ▶  ▶  ▶  ▶  -problem_label spreads_25 \\
13329  ▶  ▶  ▶  ▶  -W -depth 5 \\
13330  ▶  ▶  ▶  ▶  -report -end \\
13331  ▶  ▶  ▶  -end \\
13332  ▶  ▶  -end \\
13333  ▶  ▶  -with C -do -spread_classify_activity \\
13334  ▶  ▶  ▶  -prepare_lifting_single_case 0 \\
13335  ▶  ▶  -end \\
13336 \\
13337 
13338  #save_colored_graph fname=spreads_25_graph_0.bin 
13339  #save_colored_graph nb_vertices=225 
13340  #save_colored_graph nb_colors=21 
13341  #save_colored_graph nb_colors_per_vertex=1 
13342  #save_colored_graph done 
13343  #colored_graph::save done 
13344  #Written file spreads_25_graph_0.bin of size 5914 
13345 
13346 
13347  spreads_25_starter_0_cliques: 
13348  ▶  $(ORBITER) -v 2 \\
13349  ▶  ▶  -define G -graph -load spreads_25_graph_0.bin -end \\
13350  ▶  ▶  -with G -do \\
13351  ▶  ▶  -graph_theoretic_activity \\
13352  ▶  ▶  ▶  -findCliques -rainbow -target_size 21 -end \\
13353  ▶  ▶  ▶  -end \\
13354 
13355  #graph_theoretic_activity::perform_activity Gr->label=spreads_25_graph_0 nb_sol = 7680 
13356 
13357 
13358  classify_spreads_25_starter_lift_all_cases: 
13359  ▶  $(ORBITER) -v 3 \\
13360  ▶  ▶  -define F -finite_field -q 5 -end \\
13361  ▶  ▶  -define P -projective_space -n 3 -field F -v 0 -end \\
13362  ▶  ▶  -define C -spread_classifier \\
13363  ▶  ▶  ▶  -projective_space P \\
13364  ▶  ▶  ▶  -k 2 \\
13365  ▶  ▶  ▶  -starter_size 5 \\
13366  ▶  ▶  ▶  -recoordinatize \\
13367  ▶  ▶  ▶  -poset_classification_control \\
13368  ▶  ▶  ▶  ▶  -draw_options \\
13369  ▶  ▶  ▶  ▶  ▶  -radius 20 \\
13370  ▶  ▶  ▶  ▶  ▶  ▶  -nodes_empty \\

796
spreads_25_starter_cliques:

classify_spreads_25_isomorph:

classify_spreads_25_label_spreads_25:

classify_spreads_25_label_spreads_25_depth_5:

classify_spreads_25_label_spreads_25_report:

classify_spreads_25_label_spreads_25_end:

classify_spreads_25_label_spreads_25_with_C:

classify_spreads_25_label_spreads_25_spread_classify_activity:

classify_spreads_25_label_spreads_25_compute_starter:

classify_spreads_25_label_spreads_25_problem_label_spreads_25:

classify_spreads_25_label_spreads_25_problem_label_spreads_25_W:

classify_spreads_25_label_spreads_25_problem_label_spreads_25_report:

classify_spreads_25_label_spreads_25_problem_label_spreads_25_end:

classify_spreads_25_label_spreads_25_spread_classify_activity:

classify_spreads_25_label_spreads_25_prepare_lifting_all_cases:

classify_spreads_25_label_spreads_25_end:

classify_spreads_25_isomorph:

classify_spreads_25_isomorph_v_3:

classify_spreads_25_isomorph_define_F:

classify_spreads_25_isomorph_define_P:

classify_spreads_25_isomorph_define_C:

classify_spreads_25_isomorph_projective_space:

classify_spreads_25_isomorph_k_2:

classify_spreads_25_isomorph_starter_size_5:

classify_spreads_25_isomorph_recoordinatize:

classify_spreads_25_isomorph_poset_classification_control:

classify_spreads_25_isomorph_draw_options:

classify_spreads_25_isomorph_radius_20:

classify_spreads_25_isomorph_nodes_empty:

classify_spreads_25_isomorph_line_width_0.2:

classify_spreads_25_isomorph_embedded:

classify_spreads_25_isomorph_end:
with C -do -spread_classify_activity 
  -compute_starter 
  -problem_label spreads_25 
  -W -depth 5 
  -report -end 
  -end 
  -with C -do -spread_classify_activity 
  -isomorph 
  -prefix_iso "./spreads_25" 
  -use_database_for_starter 
  -build_db 
  -solution_prefix "" 
  -base_fname "" 
  -end 
  -end 
  -with C -do -spread_classify_activity 
  -isomorph 
  -prefix_iso "./spreads_25" 
  -use_database_for_starter 
  -read_solutions 
  -solution_prefix "" 
  -base_fname "spreads_25_graph" 
  -end 
  -end 
  -with C -do -spread_classify_activity 
  -isomorph 
  -prefix_iso "./spreads_25" 
  -use_database_for_starter 
  -compute_orbits 
  -solution_prefix "" 
  -base_fname "spreads_25_graph" 
  -end 
  -end 
  -with C -do -spread_classify_activity 
  -isomorph 
  -prefix_iso "./spreads_25" 
  -use_database_for_starter 
  -isomorph_testing 
  -solution_prefix "" 
  -base_fname "spreads_25_graph" 
  -end 
  -end 
  -with C -do -spread_classify_activity 
  -isomorph 
  -prefix_iso "./spreads_25" 
  -use_database_for_starter 
  -isomorph_testing 
  -solution_prefix "" 
  -base_fname "spreads_25_graph" 
  -end 
  -end 
  -with C -do -spread_classify_activity 
  -isomorph 
  -prefix_iso "./spreads_25" 
  -use_database_for_starter 
  -isomorph_testing 
  -solution_prefix "" 
  -base_fname "spreads_25_graph" 
  -end 
  -end
classify_spreads_27_3_starter:
13484  $\text{(ORBITER)} -v 10$
13485  $\text{-define F -finite_field -q 3 -end}$
13486  $\text{-define P -projective_space -n 5 -field F -v 0 -end}$
13487  $\text{-define C -spread_classifier}$
13488  $\text{-projective_space P}$
13489  $\text{-k 3}$
13490  $\text{-starter_size 5}$
13491  $\text{-recoordinatize}$
13492  $\text{-poset_classification_control}$
13493  $\text{-draw_options}$
13494  $\text{-radius 20}$
13495  $\text{-nodes_empty}$
13496  $\text{-line_width 0.2}$
13497  $\text{-embedded}$
13498  $\text{-end}$
13499  $\text{-draw_poset}$
13500  $\text{-problem_label spreads_27_3}$
13501  $\text{-end}$
13502  $\text{-output_prefix "."}$
13503  $\text{-end}$
13504  $\text{-with C -do -spread_classify_activity}$
13505  $\text{-compute_starter}$
13506  $\text{-problem_label spreads_27_3}$
13507  $\text{-W -depth 5}$
13508  $\text{-report -end}$
13509  $\text{-end}$
13510  $\text{-end}$
13511  $\text{-end}$

# We found 21 isomorphism types

13490  #1:33

13480
13481
13482
13483

13479

13483 classify_spreads_27_3_starter:
13484  $\text{(ORBITER)} -v 10$
13485  $\text{-define F -finite_field -q 3 -end}$
13486  $\text{-define P -projective_space -n 5 -field F -v 0 -end}$
13487  $\text{-define C -spread_classifier}$
13488  $\text{-projective_space P}$
13489  $\text{-k 3}$
13490  $\text{-starter_size 5}$
13491  $\text{-recoordinatize}$
13492  $\text{-poset_classification_control}$
13493  $\text{-draw_options}$
13494  $\text{-radius 20}$
13495  $\text{-nodes_empty}$
13496  $\text{-line_width 0.2}$
13497  $\text{-embedded}$
13498  $\text{-end}$
13499  $\text{-draw_poset}$
13500  $\text{-problem_label spreads_27_3}$
13501  $\text{-end}$
13502  $\text{-output_prefix "."}$
13503  $\text{-end}$
13504  $\text{-with C -do -spread_classify_activity}$
13505  $\text{-compute_starter}$
13506  $\text{-problem_label spreads_27_3}$
13507  $\text{-W -depth 5}$
13508  $\text{-report -end}$
13509  $\text{-end}$
13510  $\text{-end}$
13511  $\text{-end}$

13479 # We found 21 isomorphism types

13480  #1:33

13481
13482
13483

13479

13483 classify_spreads_27_3_starter:
13484  $\text{(ORBITER)} -v 10$
13485  $\text{-define F -finite_field -q 3 -end}$
13486  $\text{-define P -projective_space -n 5 -field F -v 0 -end}$
13487  $\text{-define C -spread_classifier}$
13488  $\text{-projective_space P}$
13489  $\text{-k 3}$
13490  $\text{-starter_size 5}$
13491  $\text{-recoordinatize}$
13492  $\text{-poset_classification_control}$
13493  $\text{-draw_options}$
13494  $\text{-radius 20}$
13495  $\text{-nodes_empty}$
13496  $\text{-line_width 0.2}$
13497  $\text{-embedded}$
13498  $\text{-end}$
13499  $\text{-draw_poset}$
13500  $\text{-problem_label spreads_27_3}$
13501  $\text{-end}$
13502  $\text{-output_prefix "."}$
13503  $\text{-end}$
13504  $\text{-with C -do -spread_classify_activity}$
13505  $\text{-compute_starter}$
13506  $\text{-problem_label spreads_27_3}$
13507  $\text{-W -depth 5}$
13508  $\text{-report -end}$
13509  $\text{-end}$
13510  $\text{-end}$
13511  $\text{-end}$

13479 # We found 21 isomorphism types

13480  #1:33

13481
13482
13483

13479

13483 classify_spreads_27_3_starter:
13484  $\text{(ORBITER)} -v 10$
13485  $\text{-define F -finite_field -q 3 -end}$
13486  $\text{-define P -projective_space -n 5 -field F -v 0 -end}$
13487  $\text{-define C -spread_classifier}$
13488  $\text{-projective_space P}$
13489  $\text{-k 3}$
13490  $\text{-starter_size 5}$
13491  $\text{-recoordinatize}$
13492  $\text{-poset_classification_control}$
13493  $\text{-draw_options}$
13494  $\text{-radius 20}$
13495  $\text{-nodes_empty}$
13496  $\text{-line_width 0.2}$
13497  $\text{-embedded}$
13498  $\text{-end}$
13499  $\text{-draw_poset}$
13500  $\text{-problem_label spreads_27_3}$
13501  $\text{-end}$
13502  $\text{-output_prefix "."}$
13503  $\text{-end}$
13504  $\text{-with C -do -spread_classify_activity}$
13505  $\text{-compute_starter}$
13506  $\text{-problem_label spreads_27_3}$
13507  $\text{-W -depth 5}$
13508  $\text{-report -end}$
13509  $\text{-end}$
13510  $\text{-end}$
13511  $\text{-end}$
# 50 orbits at level 5:

# total: 60

$((39^2, 26^2, 10, 6^2, 5, 3^9, 2^9, 1^2{24}))$ average is $426 / 50$

# time 4:31

classify_spreads_27_starter_lift_all_cases:

```
(OBITTER) -v 3 \
```

```
$\text{-define F -finite_field -q 3 -end}$ \
```

```
$\text{-define P -projective_space -n 5 -field F -v 0 -end}$ \
```

```
$\text{-define C -spread_classifier}$ \
```

```
$\text{-projective_space P}$ \
```

```
$\text{-k 3}$ \
```

```
$\text{-starter_size 5}$ \
```

```
$\text{-recoordinatize}$ \
```

```
$\text{-poset_classification_control}$ \
```

```
$\text{-draw_options}$ \
```

```
$\text{-radius 20}$ \
```

```
$\text{-nodes_empty}$ \
```

```
$\text{-line_width 0.2}$ \
```

```
$\text{-embedded}$ \
```

```
$\text{-end}$ \
```

```
$\text{-W}$ \
```

```
$\text{-draw_poset}$ \
```

```
$\text{-problem_label spreads_27}$ \
```

```
$\text{-end}$ \
```

```
$\text{-output_prefix "$"}$ \
```

```
$\text{-end}$ \
```

```
$\text{-with C -do -spread_classify_activity}$ \
```

```
$\text{-compute_starter}$ \
```

```
$\text{-problem_label spreads_27}$ \
```

```
$\text{-W -depth 5}$ \
```

```
$\text{-report -end}$ \
```

```
$\text{-end}$ \
```

```
$\text{-with C -do -spread_classify_activity}$ \
```

```
$\text{-prepare_lifting_all_cases}$ \
```

```
$\text{-end}$ \
```

800
spreads_27_starter_cliques:

$\begin{align*}
\text{(ORBITER)} & -v 2 \\
\text{loop} & L 0 50 1 \\
\text{define} G & -graph -load spreads_27_graph_{%L}.bin -end \\
\text{with} G & -do \\
\text{graph_theoretic_activity} & \\
\text{find_cliques} & -rainbow -target_size 23 -end \\
\text{end_loop}
\end{align*}$

classify_spreads_27_isomorph_and_recognize:

$\begin{align*}
\text{(ORBITER)} & -v 3 \\
\text{define} F & -finite_field -q 3 -end \\
\text{define} P & -projective_space -n 5 -field F -v 0 -end \\
\text{define} C & -spread_classifier \\
\text{projective_space} & P \\
\text{k} & 3 \\
\text{starter_size} & 5 \\
\text{recoordinatize} & \\
\text{poset_classification_control} & \\
\text{draw_options} & \\
\text{radius} & 20 \\
\text{nodes_empty} & \\
\text{line_width} & 0.2 \\
\text{embedded} & \\
\text{end} & \\
\text{W} & \\
\text{draw_poset} & \\
\text{problem_label_spreads_27} & \\
\text{end} & \\
\text{output_prefix} & "" \\
\text{end} & \\
\text{with} C & -do -spread_classify_activity \\
\text{compute_starter} & \\
\text{problem_label_spreads_27} & \\
\text{W} & -depth 5 \\
\text{report -end} &
\end{align*}$
classify activity

-isomorph

-prefix_iso "/spreads_27"

-use_database_for_starter

-build_db

-solution_prefix ""

-base_fname ""

-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

-with C -do -spread_classify_activity

-isomorph

-prefix_iso "/spreads_27"

-use_database_for_starter

-compute_orbits

-solution_prefix ""

-base_fname "spreads_27_graph"

-end

-with C -do -spread_classify_activity

-isomorph

-prefix_iso "/spreads_27"

-use_database_for_starter

-isomorph_testing

-solution_prefix ""

-base_fname "spreads_27_graph"

-end

-with C -do -spread_classify_activity

-isomorph

-prefix_iso "/spreads_27"

-use_database_for_starter

-isomorph_report

-solution_prefix ""

-base_fname "spreads_27_graph"

-end

-end

-end
# SPREAD_S27_RAO_RAO is isomorphic to spread 0 in the list
# (which is different from the ordering of the Orbiter catalogue)
# the stabilizer of the spread has order 84.

# We found 7 isomorphism types
#0:36

#generators for the stabilizer of the Rao/Rao spread:
#1,2,2,2,0,2,1,1,2,2,2,1,2,1,2,1,1,0,2,2,0,2,0,0,1,1,2,2,0,0,1,1,2,0,1, #1,0,1,  
#0,1,2,0,1,2,2,2,1,1,1,0,1,1,2,0,1,1,2,1,1,0,1,0,2,2,0,2,0,1,1,0, 

create_spread_27_0: 

$ (ORBITER) -v 3 \ 

#substructure_lifting.data::write_hash_and_datref_file id_to_hash tallied: 
#( 1^6076, 2^289, 3^35, 4^5 )

# using 64 bit hash values, based on a modified version of Paul Hsieh's SuperFast Hash
create_spread_27_1:
$$(ORBITER) -v 3 \$
\begin{verbatim}
$define F -finite_field -q 3 -end \$
\end{verbatim}
\begin{verbatim}
$define G -linear_group -PGL 6 F -end \$
\end{verbatim}
\begin{verbatim}
$define S -spread -kernel_field F \$
\end{verbatim}
\begin{verbatim}
$define G -group -k 3 -catalogue 1 \$
\end{verbatim}
\begin{verbatim}
end
\end{verbatim}
\begin{verbatim}
with S -do -spread_activity \$
\end{verbatim}
\begin{verbatim}
end
\end{verbatim}
pdflatex catalogue_q3_k3_1_report.tex
open catalogue_q3_k3_1_report.pdf

create_spread_27_2:
$$(ORBITER) -v 3 \$
\begin{verbatim}
$define F -finite_field -q 3 -end \$
\end{verbatim}
\begin{verbatim}
$define G -linear_group -PGL 6 F -end \$
\end{verbatim}
\begin{verbatim}
$define S -spread -kernel_field F \$
\end{verbatim}
\begin{verbatim}
$define G -group -k 3 -catalogue 2 \$
\end{verbatim}
\begin{verbatim}
end
\end{verbatim}
\begin{verbatim}
with S -do -spread_activity \$
\end{verbatim}
\begin{verbatim}
end
\end{verbatim}
pdflatex catalogue_q3_k3_2_report.tex
open catalogue_q3_k3_2_report.pdf

create_spread_27_3:
$$(ORBITER) -v 3 \$
\begin{verbatim}
$define F -finite_field -q 3 -end \$
\end{verbatim}
\begin{verbatim}
$define G -linear_group -PGL 6 F -end \$
\end{verbatim}
\begin{verbatim}
$define S -spread -kernel_field F \$
\end{verbatim}
\begin{verbatim}
$define G -group -k 3 -catalogue 3 \$
\end{verbatim}
\begin{verbatim}
end
\end{verbatim}
\begin{verbatim}
with S -do -spread_activity \$
\end{verbatim}
\begin{verbatim}
end
\end{verbatim}
pdflatex catalogue_q3_k3_3_report.tex
open catalogue_q3_k3_3_report.pdf

create_spread_27_4:
$$(ORBITER) -v 3 \$
\begin{verbatim}
$define F -finite_field -q 3 -end \$
\end{verbatim}
\begin{verbatim}
$define G -linear_group -PGL 6 F -end \$
\end{verbatim}
\begin{verbatim}
$define S -spread -kernel_field F \$
\end{verbatim}
\begin{verbatim}
$define G -group -k 3 -catalogue 4 \$
\end{verbatim}
\begin{verbatim}
end
\end{verbatim}
\begin{verbatim}
with S -do -spread_activity \$
\end{verbatim}

804
create_spread_27.5:

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

define F -finite_field -q 3 -end \backslash

define G -linear_group -PGL 6 F -end \backslash

define S -spread -kernel_field F \backslash

group G -k 3 -catalogue 5 \backslash

do -spread_activity \backslash

-end \backslash

-pdflatex catalogue_q3_k3_4_report.tex\backslash

-open catalogue_q3_k3_4_report.pdf\backslash

create_spread_27.6:

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

define F -finite_field -q 3 -end \backslash

define G -linear_group -PGL 6 F -end \backslash

define S -spread -kernel_field F \backslash

group G -k 3 -catalogue 6 \backslash

-end \backslash

-pdflatex catalogue_q3_k3_5_report.tex\backslash

-open catalogue_q3_k3_5_report.pdf\backslash

classify_spreads_32_starter:

$$(\text{ORBITER}) -v 5 \backslash$

define F -finite_field -q 2 -end \backslash

define P -projective_space -n 9 -field F -v 0 -end \backslash

define C -spread_classifier \backslash

-projective_space P \backslash

-k 5 \backslash

-starter_size 5 \backslash

-recoordinatize \backslash
poset\_classification\_control

\texttt{-draw\_options}

\texttt{-radius 20}

\texttt{-nodes\_empty}

\texttt{-line\_width 0.2}

\texttt{-embedded}

\texttt{-end}

\texttt{-W -depth 5}

\texttt{-draw\_poset}

\texttt{-problem\_label spreads\_32}

\texttt{-end}

\texttt{-output\_prefix ""}

\texttt{-end}

\texttt{-with C -do -spread\_classify\_activity}

\texttt{-compute\_starter}

\texttt{-problem\_label spreads\_32}

\texttt{-report -end}

\texttt{-end}

\texttt{-end}

\texttt{-with C -do -spread\_classify\_activity}

\texttt{-prepare\_lifting\_single\_case 0}

\texttt{-end}

\texttt{classify\_spreads 49} starter lift all cases:

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

\texttt{-define F -finite\_field -q 7 -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}
13838 ▶ ▶ ▶ -starter_size 5 \n13839 ▶ ▶ ▶ -recoordinatize \n13840 ▶ ▶ ▶ -poset_classification_control \n13841 ▶ ▶ ▶ ▶ -draw_options \n13842 ▶ ▶ ▶ ▶ ▶ -radius 20 \n13843 ▶ ▶ ▶ ▶ ▶ -nodes_empty \n13844 ▶ ▶ ▶ ▶ ▶ -line_width 0.2 \n13845 ▶ ▶ ▶ ▶ ▶ -embedded \n13846 ▶ ▶ ▶ ▶ ▶ -end \n13847 ▶ ▶ ▶ ▶ ▶ -W \n13848 ▶ ▶ ▶ ▶ ▶ -draw_poset \n13849 ▶ ▶ ▶ ▶ ▶ ▶ -problem_label spreads_49 \n13850 ▶ ▶ ▶ ▶ ▶ ▶ ▶ -end \n13851 ▶ ▶ ▶ ▶ ▶ ▶ ▶ -output_prefix "" \n13852 ▶ ▶ ▶ ▶ ▶ ▶ ▶ -end \n13853 ▶ ▶ ▶ ▶ ▶ ▶ ▶ -with C -do -spread_classify_activity \n13854 ▶ ▶ ▶ ▶ ▶ ▶ ▶ -compute_starter \n13855 ▶ ▶ ▶ ▶ ▶ ▶ ▶ -problem_label spreads_49 \n13856 ▶ ▶ ▶ ▶ ▶ ▶ ▶ -W -depth 5 \n13857 ▶ ▶ ▶ ▶ ▶ ▶ ▶ -report -end \n13858 ▶ ▶ ▶ ▶ ▶ ▶ ▶ -end \n13859 ▶ ▶ ▶ ▶ ▶ ▶ ▶ -end \n13860 ▶ ▶ ▶ ▶ ▶ ▶ ▶ -with C -do -spread_classify_activity \n13861 ▶ ▶ ▶ ▶ ▶ ▶ ▶ -prepare_lifting_all_cases \n13862 ▶ ▶ ▶ ▶ ▶ ▶ ▶ -end \n13863

13864 #save_colored_graph fname=spreads_49_graph_125.bin
13865 # 126 cases
13866
13867
13868
13869 spreads_49_starter_cliques_loop:
13870 ▶ $(ORBITER) -v 2 \n13871 ▶ ▶ -loop L 0 126 1 \n13872 ▶ ▶ ▶ -define G -graph -load spreads_49_graph_%L.bin -end \n13873 ▶ ▶ ▶ -with G -do \n13874 ▶ ▶ ▶ -graph_theoretic_activity \n13875 ▶ ▶ ▶ ▶ -find_cliques -rainbow -target_size 45 -end \n13876 ▶ ▶ ▶ ▶ -end \n13877 ▶ ▶ ▶ ▶ -end_loop
13878
13879 spreads_49_starter_cliques_0:
13880 ▶ $(ORBITER) -v 2 \n13881 ▶ ▶ ▶ -define G -graph -load spreads_49_graph_0.bin -end \n13882 ▶ ▶ ▶ -with G -do \n13883 ▶ ▶ ▶ -graph_theoretic_activity \n13884 ▶ ▶ ▶ ▶ -find_cliques -rainbow -target_size 45 -end \n
807
13892 # Section 12.2: Translation planes

13893

13894 SECTION_TRANSLATION_PLANES:

13895 create_translation_plane_9b:

13896 $(ORBITER) -v 3 \n
13897 $(ORBITER) -v 3 \n
13898 $(ORBITER) -v 3 \n
13899 $(ORBITER) -v 3 \n
13900 $(ORBITER) -v 3 \n
13901 $(ORBITER) -v 3 \n
13902 $(ORBITER) -v 3 \n
13903 $(ORBITER) -v 3 \n
13904 $(ORBITER) -v 3 \n
13905 $(ORBITER) -v 3 \n
13906 $(ORBITER) -v 3 \n
13907 $(ORBITER) -v 3 \n
13908 $(ORBITER) -v 3 \n
13909 $(ORBITER) -v 3 \n
13910 $(ORBITER) -v 3 \n
13911 $(ORBITER) -v 3 \n
13912 $(ORBITER) -v 3 \n
13913 $(ORBITER) -v 3 \n
13914 $(ORBITER) -v 3 \n
13915 $(ORBITER) -v 3 \n
13916 $(ORBITER) -v 3 \n
13917 $(ORBITER) -v 3 \n
13918 $(ORBITER) -v 3 \n
13919 $(ORBITER) -v 3 \n
13920 $(ORBITER) -v 3 \n
13921 $(ORBITER) -v 3 \n
13922 $(ORBITER) -v 3 \n
13923 $(ORBITER) -v 3 \n
13924 $(ORBITER) -v 3 \n
13925 $(ORBITER) -v 3 \n
13926 $(ORBITER) -v 3 \n
13927 $(ORBITER) -v 3 \n
13928 $(ORBITER) -v 3 \n
13929 $(ORBITER) -v 3 \n
13930 $(ORBITER) -v 3 \n
13931 $(ORBITER) -v 3 \n
808
13932 \> open group_of_plane_plane_catalogue_q3_k2_1_stab_orb_3_report.pdf
13933 \> \#pdflatex group_of_plane_plane_catalogue_q3_k2_1_stab_orb_3_report.tex
13934 \> \#open group_of_plane_plane_catalogue_q3_k2_1_stab_orb_3_report.pdf
13935
13936
13937 \#> \> -with A -do \>
13938 \#> \> -group_theoretic_activity \>
13939 \#> \> \> -report \>
13940 \#> \> \> -orbits_on_points \>
13941 \#> \> \> -stabilizer_of_orbit_rep 3 \>
13942 \#> \> \> -export_trees \>
13943 \#> \> \> -end
13944
13945
13946
13947 create_translation_plane_16_4_0:
13948 \> \$ (ORBITER) -v 3 \>
13949 \> \> -define F -finite_field -q 4 -end \>
13950 \> \> -define G -linear_group -PGL 4 F -end \>
13951 \> \> -define G1 -linear_group -PGL 5 F -end \>
13952 \> \> -define S -spread -kernel_field F \>
13953 \> \> \> -group G -k 2 -catalogue 0 \>
13954 \> \> \> -end \>
13955 \> \> -define T -translation_plane S G G1 -end
13956 \> \$ (ORBITER) -v 2 \>
13957 \> \> -draw_matrix \>
13958 \> \> \> -input_csv_file plane_catalogue_q4_k2_0_incma.csv \>
13959 \> \> \> -box_width 6 -bit_depth 8 \>
13960 \> \> \> -partition 4 273 273 \>
13961 \> \> \> -end
13962 \> open plane_catalogue_q4_k2_0_incma_draw.bmp
13963
13964
13965 \#0 : "1200", // Hall spread
13966 \#1 : "81600", // Desarguesian spread
13967 \#2 : "576", // Semifield spread
13968
13969
13970 create_translation_plane_16_2_0:
13971 \> \$ (ORBITER) -v 3 \>
13972 \> \> -define F -finite_field -q 2 -end \>
13973 \> \> -define G -linear_group -PGL 8 F -end \>
13974 \> \> -define G1 -linear_group -PGL 9 F -end \>
13975 \> \> -define S -spread -kernel_field F \>
13976 \> \> \> -group G -k 4 -catalogue 0 \>
13977 \> \> \> -end \>
13978 \> \> -define T -translation_plane S G G1 -end
$(ORBITER) -v 2 \n$draw_matrix \n-input_csv_file plane_catalogue_q2_k4_0_incma.csv \n-box_width 6 -bit_depth 8 \n-partition 4 273 273 \n-end \nopen plane_catalogue_q2_k4_0_incma_draw.bmp

#0 : "1008",
#1 : "1008",
#2 : "1728",
#3 : "216",
#4 : "360",
#5 : "288",
#6 : "3600",
#7 : "244800",

RREF plane_16_2_0_rank_of_incma:

#2-rank is 106, so the plane is Lorimer-Rahilly

create_translation_plane_25_14_rank:

define F -finite_field -q 2 -end \n-define v -vector -field F \n-file plane_catalogue_q2_k4_0_incma.csv \n-end \n-with F -do -finite_field_activity \n-RREF v -normalize_from_the_right \n-end

# 2-rank is 106, so the plane is Lorimer-Rahilly

create_translation_plane_25_14_rank:

define F -finite_field -q 5 -end \n-define G -linear_group -PGL 4 F -end \n-define G1 -linear_group -PGL 5 F -end \n-define S -spread -kernel_field F \n-group G -k 2 -catalogue 14 \n-end \n-with T -translation_plane S G G1 -end \n-with T -do -translation_plane_activity \n-export incma \n-end

define F -finite_field -q 2 -end \n-define v -vector -field F \n-file plane_catalogue_q5_k2_14_incma.csv \n-end \n
create_translation_plane_27_Rao_Rao:

define F -finite_field -q 3 -end

$\text{define }\mathbb{F}\text{ -finite field -q 3 -end}$

define SS -vector -dense $(\text{SPREAD SET } 27\text{ RAO RAO})$ -end

define G -linear_group -PGL 6 F -end

define G1 -linear_group -PGL 7 F -end

define S -spread -kernel_field F

define G -linear_group -k 3 -spread_set SS -end

define T -translation_plane S G G1 -end

with T -do -translation_plane_activity

define T -translation_plane S G G1 -end

export incma

end

RREF

Rao_Rao_plane_incma.rank:

define F -finite_field -q 3 -end

define v -vector -field F

$file\text{ plane }\text{incma}.csv$

end

with F -do -finite_field_activity

$\text{define }\mathbb{F}\text{ -finite field -q 3 -end}$

RREF v -normalize from the right

end

3-rank is 271, so the Rao / Rao plane is Moorhouse IV.

create_translation_plane_27_27_p.rank_of_incidence_matrix:

define F -finite_field -q 3 -end

define G -linear_group -PGL 6 F -end

define G1 -linear_group -PGL 7 F -end

define S -spread -kernel_field F

define G -linear_group -k 3 -catalogue 6 -end

define T -translation_plane S G G1 -end

export incma

end
create_translation_plane_27_5_Rao_Rao:

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

-define F -finite_field -q 3 -end
-define G -linear_group -PGL 6 F -end
-define G1 -linear_group -PGL 7 F -end
-define S -spread -kernel_field F
-define T -translation_plane S G G1 -end
-define A -linear_group -import_group_of_plane T -end
-define Orb -orbits -group A
-on_points

-pdflatex group_of_plane.plane_catalogue_q3_k3_5_report.tex
-open_group_of_plane.plane_catalogue_q3_k3_5_report.pdf
-pdflatex group_of_plane.plane_catalogue_q3_k3_5_orbits_report.tex
-open_group_of_plane.plane_catalogue_q3_k3_5_orbits_report.pdf
-pdflatex group_of_plane.plane_catalogue_q3_k3_5_stab_orb_3_report.tex
-open_group_of_plane.plane_catalogue_q3_k3_5_stab_orb_3_report.pdf

-define T -translation_plane activity
-export incma
-end

-so, Rao / Rao is OCN=5
14119 \ #open orbit_PGL_5_3_on_andre_3_draw.pdf
14120 \ #pdflatex group_of_plane_plane_catalogue_q3_k2_1_stab_orb_3_report.tex
14121 \ #open group_of_plane_plane_catalogue_q3_k2_1_stab_orb_3_report.pdf
14122 \ #pdflatex group_of_plane_plane_catalogue_q3_k2_1_stab_orb_3_report.tex
14123 \ #open group_of_plane_plane_catalogue_q3_k2_1_stab_orb_3_report.pdf
14124
14125
14126 \ #\with A -do \ 
14127 \ #\group_theoretic_activity \ 
14128 \ \#\report \ 
14129 \ \#\orbits_on_points \ 
14130 \ \#\stabilizer_of_orbit_rep 3 \ 
14131 \ \#\export_trees \ 
14132 \ \#\end
14133
14134
14135
14136 \ create_translation_plane_27_4_block_stab:
14137 \ $(\text{ORBITER}) -v 3 \ 
14138 \ \#define F -finite_field -q 3 -end \ 
14139 \ \#define G -linear_group -PGL 6 F -end \ 
14140 \ \#define G1 -linear_group -PGL 7 F -end \ 
14141 \ \#define S -spread -kernel_field F \ 
14142 \ \#define G -k 3 -catalogue 4 \ 
14143 \ \#define T -translation_plane S G G1 -end \ 
14144 \ \#with T -do -translation_plane_activity \ 
14145 \ \#export_incma \ 
14146 \ \#end \ 
14147 \ \#with T -do -translation_plane_activity \ 
14148 \ \#report \ 
14149 \ \#end \ 
14150 \ \#end \ 
14151 \ \#define A -linear_group -import_group_of_plane T -end \ 
14152 \ \#define Orb -orbits -group A \ 
14153 \ \#on_points \ 
14154 \ \#end
14155 \ \#pdflatex group_of_plane_plane_catalogue_q3_k3_4_stab_orb_3_report.tex
14156 \ \#open group_of_plane_plane_catalogue_q3_k3_4_stab_orb_3_report.pdf
14157
14158
14159 \ \#\with A -do \ 
14160 \ \#\group_theoretic_activity \ 
14161 \ \#\report \ 
14162 \ \#\orbits_on_points \ 
14163 \ \#\stabilizer_of_orbit_rep 3 \ 
14164 \ \#\export_trees \ 
14165 \ \#\end
create_translation_plane_27_5_block_stab:

$\text{ORBITER} -v 3 \$

-define F -finite_field -q 3 -end \n
-define G -linear_group -PGL 6 F -end \n
-define G1 -linear_group -PGL 7 F -end \n
-define S -spread -kernel_field F \n
-define G -group G -k 3 -catalogue 5 \n
-define T -translation_plane S G G1 -end \n
-with T -do -translation_plane_activity \n
-export Incma \n
-end \n
-with T -do -translation_plane_activity \n
-report \n
-end \n
-define A -linear_group -import_group_of_plane T -end \n
-define Orb -orbits -group A \n
-on_points \n
-end \n
#pdflatex group_of_plane_plane_catalogue_q3_k3_5_stab_orb_3_report.tex

#open group_of_plane_plane_catalogue_q3_k3_5_stab_orb_3_report.pdf

create_translation_plane_27_6_block_stab:

$\text{ORBITER} -v 3 \$

-define F -finite_field -q 3 -end \n
-define G -linear_group -PGL 6 F -end \n
-define G1 -linear_group -PGL 7 F -end \n
-define S -spread -kernel_field F \n
-define G -group G -k 3 -catalogue 6 \n
-end \n
-with T -do -translation_plane_activity \n
-export Incma \n
-end \n
-with T -do -translation_plane_activity \n
-report \n
-end \n
-end \n

-define A -linear_group -import_group_of_plane T -end \n-define Orb -orbits -group A \n-define on_points \n-end \n#pdflatex group_of_plane_plane_catalogue_q3_k3_6_stab_orb_3_report.tex \n#open group_of_plane_plane_catalogue_q3_k3_6_stab_orb_3_report.pdf \n#with A -do \n-group_theoretic_activity \n-report \n-orbits_on_points \n-stabilizer_of_orbit_rep 3 \n-export_trees \n-end \n
# Section 12.3: Packings
SECTION PACKINGS:

spread_table_PG_3_4:
- mdir 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:
- mdir 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
PG(3,5) has 21 isomorphism types of spreads, of which 12 is the index of the regular spread in the classification.

The group has order 124, and its normalizer has order 372.

The group has order 31, and its normalizer has order 372 = 4*3*31.
-define F -finite_field -q 5 -end \\
-define P -projective_space -n 3 -field F -end \\
-define T -spread_table P 2 "12" "SPREAD_TABLES_5_REG/" \\
-define PW -packing_with_symmetry_assumption T \\
-define H "H31" $(PGL_4_5_SUBGROUP_31_ME) -end \\
-define N "N31" $(PGL_4_5_SUBGROUP_31_ME_NORMALIZER) -end \\
-define PW -packing_choose_fixed_points PW 0 -end \\
-define L -packing_long_orbits PW \\
-define T -spread_table P 2 "12" "SPREAD_TABLES_5_REG/" \\
-define PW -packing_choose_fixed_points PW 0 -end \\
pdflatex H31 Reduced Spread Orbits Orbits Report.tex \\
open H31 Reduced Spread Orbits Orbits Report.pdf \\
pdflatex H31 Point Orbits Orbits Report.tex \\
open H31 Point Orbits Orbits Report.pdf \\
#pdflatex H31 Spread Orbits Orbits Report.tex \\
#open H31 Spread Orbits Orbits Report.pdf \\
#H31 Line Orbits Orbits.bin \\
#H31 Line Orbits Orbits Report.tex \\
#H31 Spread Orbits Orbits Types Report.tex \\
#H31 Spread Orbits Orbits.bin \\
#H31 Good Orbits \\
#H31 Spread Types Reduced Orbit Types Report.tex \\
#H31 Reduced Spread Orbits Orbits.bin \\
#H31 FPC0 LO.graph \\
#Section 12.4: BLT-sets \\
SECTION BLT-SETS:

BLT_5_1:
- $(ORBITER) -v 2 \\
-define F -finite_field -q 5 -end \\
-define 0 -orthogonal_space 0 5 F -end \\
-define O -orthogonal_space_activity \\
-define T -spread_table P 2 "12" "SPREAD_TABLES_5_REG/" \\
-define PW -packing_with_symmetry_assumption T \\
-define H "H31" $(PGL_4_5_SUBGROUP_31_ME) -end \\
-define N "N31" $(PGL_4_5_SUBGROUP_31_ME_NORMALIZER) -end \\
-define PW -packing_choose_fixed_points PW 0 -end \\
pdflatex H31 Reduced Spread Orbits Orbits Report.tex \\
open H31 Reduced Spread Orbits Orbits Report.pdf \\
pdflatex H31 Line Orbits Orbits Report.tex \\
open H31 Line Orbits Orbits Report.pdf \\
pdflatex H31 Line Orbits Orbits Report.tex \\
open H31 Line Orbits Orbits Report.pdf \\
pdflatex H31 Line Orbits Orbits Report.tex \\
open H31 Line Orbits Orbits Report.pdf \\
pdflatex H31 Point Orbits Orbits Report.tex \\
open H31 Point Orbits Orbits Report.pdf \\
pdflatex H31 Point Orbits Orbits Report.tex \\
open H31 Point Orbits Orbits Report.pdf \\
pdflatex H31 Point Orbits Orbits Report.tex \\
open H31 Point Orbits Orbits Report.pdf \\
#pdflatex H31 Spread Orbits Orbits Report.tex \\
#open H31 Spread Orbits Orbits Report.pdf \\
#H31 Line Orbits Orbits.bin \\
#H31 Line Orbits Orbits Report.tex \\
#H31 Spread Orbits Orbits Types Report.tex \\
#H31 Spread Orbits Orbits.bin \\
#H31 Good Orbits \\
#H31 Spread Types Reduced Orbit Types Report.tex \\
#H31 Reduced Spread Orbits Orbits.bin \\
#H31 FPC0 LO.graph \\
#Section 12.4: BLT-sets \\
SECTION BLT-SETS:

BLT_5_1:
BLT_5_Linear:
\$\text{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:
\$\text{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
\>

BLT_11_0:
\$\text{ORBITER} -v 2 \$
\>
\>
-define F -finite_field -q 11 -end \
\>
-define 0 -orthogonal_space 0 5 F -end \
\>
-with 0 -do -orthogonal_space_activity \
\>
-create_BLT_set -catalogue 0 -end \
\>
-end
\>
\>
#pdflatex 0_1_6_2_report.tex
\>
#open 0_1_6_2_report.pdf
\>

BLT_11_Fisher:
\$\text{ORBITER} -v 2 \$
\>
\>
-define F -finite_field -q 11 -end \
\>
-define 0 -orthogonal_space 0 5 F -end \
\>
-with 0 -do -orthogonal_space_activity \
\>
-create_BLT_set -family "Fisher" -end \
\>
-end
\>
\>
pdflatex BLT_Fisher_q11.tex
\>
open BLT_Fisher_q11.pdf
\>
818
14401
14402 BLT_11_Mondello:
14403 ▶ $(ORBITER) -v 2 \ 14404 ▶ ▶ -define F -finite_field -q 11 -end \ 14405 ▶ ▶ -define 0 -orthogonal_space 0 5 F -end \ 14406 ▶ ▶ -with 0 -do -orthogonal_space_activity \ 14407 ▶ ▶ ▶ -create_BLT_set -family "Mondello" -end \ 14408 ▶ ▶ -end 14409 ▶ pdflatex BLT_Mondello_q11.tex 14410 ▶ open BLT_Mondello_q11.pdf 14411 14412 14413 BLT_13_FTWK:
14414 ▶ $(ORBITER) -v 2 \ 14415 ▶ ▶ -define F -finite_field -q 11 -end \ 14416 ▶ ▶ -define 0 -orthogonal_space 0 5 F -end \ 14417 ▶ ▶ -with 0 -do -orthogonal_space_activity \ 14418 ▶ ▶ ▶ -create_BLT_set -family "FTWKB" -end \ 14419 ▶ ▶ -end 14420 ▶ pdflatex BLT_FTWK_q11.tex 14421 ▶ open BLT_FTWK_q11.pdf 14422 14423 14424 # for K2, q must be congruent to 2 or 3 mod 5 14425 BLT_13_K2:
14426 ▶ $(ORBITER) -v 2 \ 14427 ▶ ▶ -define F -finite_field -q 13 -end \ 14428 ▶ ▶ -define 0 -orthogonal_space 0 5 F -end \ 14429 ▶ ▶ -with 0 -do -orthogonal_space_activity \ 14430 ▶ ▶ ▶ -create_BLT_set -family "Kantor2" -end \ 14431 ▶ ▶ -end 14432 ▶ pdflatex BLT_K2_q13.tex 14433 ▶ open BLT_K2_q13.pdf 14434 14435 14436 14437 14438 14439 14440 BLT_13_deep_14:
14441 ▶ $(ORBITER) -v 2 \ 14442 ▶ ▶ -define F -finite_field -q 13 -end \ 14443 ▶ ▶ -define 0 -orthogonal_space 0 5 F -end \ 14444 ▶ ▶ -with 0 -do -orthogonal_space_activity \ 14445 ▶ ▶ ▶ -BLT_set_starter 14 \ 14446 ▶ ▶ ▶ -problem_label BLT_q13 -W -depth 14 -end \ 14447 ▶ ▶ -end

819
BLT_11_deep_search:
  $(ORBITER) -v 2 \n  -define F -finite_field -q 11 -end \n  -define O -orthogonal_space 0 5 F -end \n  -define C -BLT_set_classifier 0 -starter_size 12 -end \n  -with C -do -BLT_set_classify_activity \n  -compute_starter \n  -problem_label BLT_q11 \n  -W -depth 12 \n  -report -end \n  -end \n
pdflatex BLT_q11_poset.tex
open BLT_q11_poset.pdf

BLT_13_deep_search:
  $(ORBITER) -v 2 \n  -define F -finite_field -q 13 -end \n  -define O -orthogonal_space 0 5 F -end \n  -define C -BLT_set_classifier 0 -starter_size 14 -end \n  -with C -do -BLT_set_classify_activity \n  -compute_starter \n  -problem_label BLT_q13 \n  -W -depth 14 \n  -report -end \n  -end \n
pdflatex BLT_q13_poset.tex
open BLT_q13_poset.pdf
BLT.13.cliques:
$ORBTER -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

# 3 solutions:
BLT.q13.graph.5_0.sol.txt
BLT.q13.graph.5_0.sol.csv

BLT.13.isomorph_read_DB:
$ORBTER -v 2
-define F -finite_field -q 13
-define O -orthogonal_space 0 5 F
-define C -BLT.set.classifier 0 -starter_size 5
-with C -do -BLT.set.classify.activity
-compute_starter
-problem_label BLT.q13
-W -depth 5
-end
-with C -do -BLT.set.classify.activity
-isomorph
-prefix_iso "/BLT.q13"
-use_database_for_starter
-build.db
-solution_prefix ""
-base_fname ""
-end
-end
14542
14543 BLT.13_isomorph_read_solutions:
14544 ▷ $(ORBITER) -v 2 \n14545 ▷ ▷ -define F -finite_field -q 13 -end \n14546 ▷ ▷ -define O -orthogonal_space 0 5 F -end \n14547 ▷ ▷ -define C -BLT_set_classifier 0 -starter_size 5 -end \n14548 ▷ ▷ -with C -do -BLT_set_classify_activity \n14549 ▷ ▷ ▷ -compute_starter \n14550 ▷ ▷ ▷ ▷ -problem_label BLT.q13 \n14551 ▷ ▷ ▷ ▷ -W -depth 5 \n14552 ▷ ▷ ▷ ▷ -end \n14553 ▷ ▷ ▷ -end \n14554 ▷ ▷ ▷ -with C -do -BLT_set_classify_activity \n14555 ▷ ▷ ▷ ▷ -isomorph \n14556 ▷ ▷ ▷ ▷ ▷ -prefix_iso "/.BLT_q13" \n14557 ▷ ▷ ▷ ▷ ▷ -use_database_for_starter \n14558 ▷ ▷ ▷ ▷ ▷ -read_solutions \n14559 ▷ ▷ ▷ ▷ ▷ -list_of_cases BLT.q13_list_of_cases_5.0.1.csv \n14560 ▷ ▷ ▷ ▷ ▷ -solution_prefix "" \n14561 ▷ ▷ ▷ ▷ ▷ -base_fname "BLT.q13_graph" \n14562 ▷ ▷ ▷ ▷ ▷ -end \n14563 ▷ ▷ ▷ ▷ -end
14564
14565
14566 BLT.13_isomorph_stabilizer_orbits:
14567 ▷ $(ORBITER) -v 2 \n14568 ▷ ▷ -define F -finite_field -q 13 -end \n14569 ▷ ▷ -define O -orthogonal_space 0 5 F -end \n14570 ▷ ▷ -define C -BLT_set_classifier 0 -starter_size 5 -end \n14571 ▷ ▷ -with C -do -BLT_set_classify_activity \n14572 ▷ ▷ ▷ -compute_starter \n14573 ▷ ▷ ▷ ▷ -problem_label BLT.q13 \n14574 ▷ ▷ ▷ ▷ -W -depth 5 \n14575 ▷ ▷ ▷ ▷ -end \n14576 ▷ ▷ ▷ -end \n14577 ▷ ▷ ▷ -with C -do -BLT_set_classify_activity \n14578 ▷ ▷ ▷ ▷ -isomorph \n14579 ▷ ▷ ▷ ▷ ▷ -prefix_iso "/.BLT_q13" \n14580 ▷ ▷ ▷ ▷ ▷ -use_database_for_starter \n14581 ▷ ▷ ▷ ▷ ▷ -compute_orbits \n14582 ▷ ▷ ▷ ▷ ▷ -list_of_cases BLT.q13_list_of_cases_5.0.1.csv \n14583 ▷ ▷ ▷ ▷ ▷ -solution_prefix "" \n14584 ▷ ▷ ▷ ▷ ▷ -base_fname "BLT.q13_graph" \n14585 ▷ ▷ ▷ ▷ ▷ -end \n14586 ▷ ▷ ▷ ▷ -end
14587
14588 BLT.13_isomorph_testing:
$(ORBITER) -v 4 \n-define F -finite_field -q 13 -end \n-define O -orthogonal_space 0 5 F -end \n-define C -BLT_set_classifier 0 -starter_size 5 -end \n-with C -do -BLT_set_classify_activity \n-compute_starter \n-problem_label BLT_q13 \n-W -depth 5 \n-report -end \n-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-include /BLT_q13raph.tex
SECTION_CREATING_GRAPHS:

Cycle_graph_13:
$ (ORBITER) -v 2 
$define Gamma -graph 
$define cycle 13 
$end

make_triangle_graph:
echo $(TRIANGLE_GRAPH) >triangle_graph.csv
$(ORBITER) -v 6 
$define G -graph 
$load_csv_no_border 
triangle_graph.csv 
$end

Chain_232:
$ (ORBITER) -v 2 
$define P1 vector -dense 2,3,2 -end 
$define P2 vector -dense 2,3,2 -end 
$define Gamma -graph 
$chain_graph P1 P2 
$end

Paley_13_graph:
$ (ORBITER) -v 2 
$define Gamma -graph -Paley 13 -end 

triheiral_pair_graph:
$ (ORBITER) -v 2 
$define Gamma 
$graph -triheiral_pair_disjointness_graph 
$end
small_graph:

```bash
$(ORBITER) -v 2 -define G -graph -edges_as_pairs
```

```bash
5 "0,1,0,2,0,3,0,4,1,3,1,4,2,4"
```

```bash
-end
```

petersen:

```bash
$(ORBITER) -v 2 -define G -graph -Johnson 5 2 0 -end
```

Johnson 6 2 0:

```bash
$(ORBITER) -v 2 -define G -graph -Johnson 6 2 0 -end
```

Hamming_graph_3:

```bash
$(ORBITER) -v 2 -define G -graph -Hamming 3 2 -end
```

Hamming_graph_7:

```bash
$(ORBITER) -v 2 -define G -graph -Hamming 7 2 -end
```

HJ_graph:

```bash
$(ORBITER) -v 6 -define G -graph
```

```bash
-load_csv_no_border
```

```bash
halljanko315.csv
```

```bash
-end
```

HJ315_graph:

```bash
$(ORBITER) -v 2
```

# 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).
\$\text{define gens} -\text{vector} -\text{file} ~$
\$\text{define G} -\text{permutation}\text{.group} ~$
\$\text{bsgs halljanko315 } \text{"File\\_halljanko315"} ~$
\$315 1209600 "0,1,2" 6 \text{ gens} ~$
\$\text{define Gamma} -\text{graph} ~$
\$\text{orbital}\text{.graph G }3 ~$
\$HJd2\text{.graph:} ~$
\$\text{define G} -\text{graph} ~$
\$\text{load}\text{.csv} \text{no.border} ~$
\$\text{halljanko315.csv} ~$
\$\text{distance} 2 ~$
\$\text{Cayley}\text{.Z11}\text{.1mod3:} ~$
\$\text{define S} -\text{vector} -\text{dense} ~$
\$\text{define G} -\text{linear}\text{.group }\text{AGL 1 F} ~$
\$\text{define Gamma} -\text{graph} ~$
\$\text{Cayley}\text{.graph G S} ~$
\$\text{Cayley}\text{.Sym4.coxeter:} ~$
\$\text{define S} -\text{vector} -\text{dense }1,0,2,3, 0,2,1,3, 0,1,3,2 ~$
\$\text{define G} -\text{permutation}\text{.group }\text{symmetric}\text{.group 4} ~$
\$\text{define Gamma} -\text{graph} ~$
\$\text{Cayley}\text{.graph G S} ~$
\$\text{Cayley}\text{.Sym4.star:} ~$
\$\text{define S} -\text{vector} -\text{dense }1,0,2,3, 2,1,0,3, 3,1,2,0 ~$
\$\text{define G} -\text{permutation}\text{.group }\text{symmetric}\text{.group 4} ~$
Section 13.2: Graphs Theoretic Activities

triangle_graph_properties:

echo $(TRIANGLE_GRAPH) >triangle_graph.csv
$(ORBITER) -v 6
>define G -graph
>load_csv_no_border triangle_graph.csv
>end
>with G -do
>graph_theoretic_activity -properties
>end

cycle

$(ORBITER) -v 2
>define Gamma -graph -cycle 13 -end
>with Gamma -do
>graph_theoretic_activity -export_csv -end
>with Gamma -do
>graph_theoretic_activity -export_graphviz -end
$(ORBITER) -v 2 -draw_matrix
>input_csv_file Cycle_13.csv
>box_width 20 -bit_depth 8 -partition 4 13 13 -end
dot -Tpng Cycle_13.gv >Cycle_13.png
#twopi -Tpng Cycle_13.gv >Cycle_13.png
#open Cycle_13.draw.bmp
#pdflatex Cycle_13.report.tex
#open Cycle_13_report.pdf

cycle_9_eigenvalues:
$(ORBITER) -v 2
>define Gamma -graph

827
Paley_13.draw:

Paley_13.eigenvalues:

Cayley_Z11_1mod3.eigenvalues_and_draw:
Cayley_Sym4_coxeter_draw:
\$(ORBITER) -v 2 \n\-draw_options -xin 2000000 -yin 2000000 \n\-radius 20000 -embedded -nodes_empty -end \n\-define S -vector -dense \n"1,0,2,3, 0,2,1,3, 0,1,3,2" -end \n\-define G -permutation_group -symmetric_group 4 \n\-end \n\-define Gamma -graph \n\-Cayley_graph G S \n\-end \n\-with Gamma -do \n\-graph_theoretic_activity -draw -end

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\-define Gamma -graph \n\-Cayley_graph G S \n\-end \n\-with Gamma -do \n\-graph_theoretic_activity -draw -end

Cayley_Sym4_star.eigenvalues_and_draw:
\$(ORBITER) -v 2 \n\-draw_options -xin 1000000 -yin 1000000 -embedded -end \n\-define S -vector -dense "1,0,2,3, 2,1,0,3, 3,1,2,0" -end \n\-define G -permutation_group -symmetric_group 4 \n\-end \n\-define Gamma -graph \n
define G -graph -Johnson 5 2 0 -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 Johnson_5_2_0.csv \
-box_width 40 -bit_depth 24 -partition 4 "10" "10" -end
dot -Tpng Johnson_5_2_0.gv >Johnson_5_2_0.png

Johnson_6_2_0.draw:
$ORBITER -v 2 \
-define G -graph -Johnson 6 2 0 -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 Johnson_6_2_0.csv \
-box_width 40 -bit_depth 24 -partition 4 "10" "10" -end
dot -Tpng Johnson_6_2_0.gv >Johnson_6_2_0.png

Hamming_graph_3.draw:
$ORBITER -v 2 \
-define G -graph -Hamming 3 2 -end \
-with G -do \
-graph_theoretic_activity -export_csv -end \
-with G -do \
-graph_theoretic_activity -export_graphviz -end \
-with G -do \
-graph_theoretic_activity -save -end
$ORBITER -v 2 -draw_matrix \
-input_csv_file Hamming_3_2.csv \
-box_width 40 -bit_depth 24 \
-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 \

-define G -graph -Hamming 7 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 \ 
-input_csv_file Hamming_7_2.csv \ 
-box_width 8 -bit_depth 24 -partition 4 128 128 -end \ 
dot -Tpng Hamming_7_2.gv >Hamming_7_2.png \ 

Chain_232.properties: \ 
$\text{(OPERATOR)} -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 -export_csv \ 
-end \ 
-with Gamma -do \ 
-graph_theoretic_activity -properties \ 
-end \ 

Chain_232.eigen: \ 
$\text{(OPERATOR)} -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 \ndefine G -graph \nload_csv_no_border \nhalljanko315.csv \n-end \n-with G -do \n-graph_theoretic_activity -properties \n-end
```

#Degree type: (10^{-315} )

HJ_d2_properties:

```
$ORBITER -v 6 \ndefine G -graph \nload_csv_no_border \nhalljanko315.csv \ndistance_2 \n-end \n-with G -do \n-graph_theoretic_activity \n-properties \n-end
```

#Degree type: (80^{-315} )

PGO_5.2_collinearity_graph: 0.5.2_incidence_matrix.csv

```
$ORBITER -v 3 \ndefine Inc -vector -file 0.5.2_incidence_matrix.csv -end \ndefine Gamma -graph -collinearity_graph Inc -end \n-with Gamma -do \n-graph_theoretic_activity \n-properties \n-end
```

triheiral_pair_graph_draw:

```
$ORBITER -v 2 -define Gamma \n-graph -triheiral_pair_disjointness_graph -end \n```
# Section 13.3: Graph Theory: Classification

SECTION GRAPH THEORY CLASSIFICATION:

```
graph_classify_5:
  $(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
  -list_graphs_at_level 5 5
  -end
  -with GC -do
  -graph_classification_activity
  -draw_options
  -radius 300 -nodes_empty
  -line_width 1.5
  -scale 0.1
  -end
  -draw_graphs_at_level 5
  -end
  -print_symbols
  pdflatex graphs_v5_level5_reps.tex
  open graphs_v5_level5_reps.pdf
  pdflatex graphs_v5_poset.tex
```
open graphs_v5_poset.pdf

tournament_classify_4:

$(ORBITER) -v 2 \n-define GC -graph_classification \n-n 4 -tournament \n-poset_classification_control \n-problem_label tournament_4 \n-depth 6 -draw_poset \n-draw_options \n-radius 250 -embedded \n-end \n-end \n-with GC -do \n-graph_classification_activity \n-draw_options \n-radius 400 \n-line_width 2 -scale 0.10 \n-end \n-draw_graphs_at_level 6 \n-end \n-print_symbols

pdflatex tournament_4_level_6_reps.tex
open tournament_4_level_6_reps.pdf

graph_classify_8_r3:

$(ORBITER) -v 3 \n-define GC -graph_classification \n-n 8 -regular 3 \n-poset_classification_control \n-problem_label graphs_v8_r3 \n-depth 12 -draw_poset \n-draw_options -radius 250 \n-line_width 0.2 -embedded \n-end \n-end \n-with GC -do \n-graph_classification_activity \n-draw_options \n-radius 400 \n
835
Symmetric $4$ inversion graph recognize:

Symmetric $5$ inversion graph recognize:
\begin{verbatim}
15245 \>
  \>
  \>
  \>
  \->problem_label graphs.v5 \ 
15246 \>
  \>
  \>
  \>
  \->depth 10 -draw_poset \ 
15247 \>
  \>
  \>
  \>
  \->draw_options \ 
15248 \>
  \>
  \>
  \>
  \->radius 250 -embedded \ 
15249 \>
  \>
  \>
  \>
  \->end \ 
15250 \>
  \>
  \>
  \>
  \->report -end \ 
15251 \>
  \>
  \>
  \>
  \->end \ 
15252 \>
  \>
  \>
  \>
  \->with GC -do \ 
15253 \>
  \>
  \>
  \>
  \->graph_classification_activity \ 
15254 \>
  \>
  \>
  \>
  \->recognize_graphs_from_adjacency_matrix.csv Symmetric5.inversion_graphs.csv \ 
15255 \>
  \>
  \>
  \>
  \->end \ 
15256 \>
  \>
  \>
  \>
  \->end \ 
15257 \>
  \>
  \>
  \>
  \->print_symbols
15258 \> #pdflatex graphs.v5 poset.tex
15259 \> #open graphs.v5 poset.pdf
15260
15261 ###########################################################################
15262 # Section 13.4: Graph Theory: Clique finding
15263
15264 15265 SECTION_GRAPH_THEORY_CLIQUE_FINDING:
15266
15267 15268 small_graph_cliques: graph_v5.e7.colored_graph
15269 \>
  $(ORBITER) -v 2 \ 
15270 \>
  \->define G -graph -load graph_v5.e7.colored_graph -end \ 
15271 \>
  \->with G -do \ 
15272 \>
  \->graph_theoretic_activity \ 
15273 \>
  \->find_cliques -target_size 3 \ 
15274 \>
  \->end \ 
15275
15276 \>
  \# nb_sol = 3 \ 
15277
15278
15279
15280 15281 BLT_q13_graph_5_0_cliques_bw:
15282 \>
  $(ORBITER) -v 2 \ 
15283 \>
  \->define G -graph -load BLT_q13_graph_5_0.bin -end \ 
15284 \>
  \->with G -do \ 
15285 \>
  \->graph_theoretic_activity \ 
15286 \>
  \->find_cliques -target_size 9 \ 
15287 \>
  \->end \ 
15288
15289 \>
  \# all_cliques_black_and_white \ 
15290
\end{verbatim}
BLT_q13_graph_5.0_cliques_rainbow:

$ (ORBTER) -v 2 \n
> -define G -graph -load BLT_q13_graph_5.0.bin -end \n> -with G -do \n> -graph_theoretic_activity \n> -find_cliques -rainbow -target_size 9 \n> -end

# all_rainbow_cliques

small_graph_cliques_Sajeeb:

$ (ORBTER) -v 2 \n
> -define G -graph -load graph_v5_e7.colored_graph -end \n> -with G -do \n> -graph_theoretic_activity \n> -find_cliques -Sajeeb -target_size 3 \n> -end

# nb_sol = 3

Paley_13_aut:

$ (ORBTER) -v 2 \n
> -define Gamma -graph -Paley 13 -end \n> -with Gamma -do \n> -graph_theoretic_activity \n> -automorphism_group \n> -end

# writes Paley_13_group.makefile

User time: 0 of a second, dt=0 tps = 100

#nb_calls_to_densenauty=1

Paley_13_cliques_classify:

$ (ORBTER) -v 4 \n
> -define Control -poset_classification_control \n> -W \n
838
-problem_label Paley13_cliques
-define gens -vector -file Paley_13 gens.csv -end
-define G -permutation_group
-bsgs Paley_13 "Paley\_13" 13 78 "0,1" 3 gens -end
-define Gamma -graph -Paley 13 -end
-define Orb -orbits -group G
-on_subsets 5 Control
-end

Paley_13_cliques_all:
-define Gamma -graph -Paley 13 -end
-with Gamma -do
-graph_theoretic_activity
-find_cliques -target_size 3 -end
-end

PG0_5_2_cliques: 0_5_2.incidence_matrix.csv
-define Inc -vector -file 0_5_2.incidence_matrix.csv -end
-define Gamma -graph -collinearity_graph Inc -end
-with Gamma -do
-graph_theoretic_activity
-find_cliques -target_size 3 -end -end

HJ_d2_c5:
-define G -graph
-load_csv_no_border halljanko315.csv
-distance_2
-end
-with G -do
-graph_theoretic_activity

#User time: 0.01 of a second, dt=1 tps = 100
15385 \>> \>> \>> \>> \:-find_cliques \:-target_size 5 \:-end \:
15386 \>> \>> \>> \>> \:-end
15387
15388
15389
15390 #graph_theoretic_activity::perform_activity Gr->label=halljanko315 nb_sol = 26208 0
15391
15392
15393 HJ64_cliques5:
15394 \>> $(ORBITER) \:-v 6 \:
15395 \>> \>> \:-define Gamma \:-graph \:
15396 \>> \>> \>> \:-load \:
15397 \>> \>> \>> \>> \:Group_Permission315.Orbital_3.colored_graph \:
15398 \>> \>> \>> \>> \:-end \:
15399 \>> \>> \>> \:-with Gamma \:-do \:
15400 \>> \>> \>> \:-graph_theoretic_activity \:
15401 \>> \>> \>> \>> \:-find_cliques \:-target_size 5 \:-end \:
15402 \>> \>> \:-end
15403
15404 #graph_theoretic_activity::perform_activity Gr->label=Group_Permission315.Orbital_3 nb_sol = 1008
15405 #Group_Permission315.Orbital_3_sol.csv
15406
15407
15408
15409 HJ64_cliques5.classify:
15410 \>> $(ORBITER) \:-v 6 \:
15411 \>> \>> \:-define Control \:-poset_classification_control \:
15412 \>> \>> \>> \:-W \:
15413 \>> \>> \>> \:-problem_label HJ64_cliques \:
15414 \>> \>> \>> \:-clique_test Gamma \:
15415 \>> \>> \>> \:-depth 5 \:
15416 \>> \>> \:-end \:
15417 \>> \>> \:-define Gamma \:-graph \:
15418 \>> \>> \>> \:-load \:
15419 \>> \>> \>> \>> \:Group_Permission315.Orbital_3.colored_graph \:
15420 \>> \>> \>> \>> \:-end \:
15421 \>> \>> \:-define gens \:-vector \:
15422 \>> \>> \>> \:-file halljanko315_gens.csv \:
15423 \>> \>> \:-end \:
15424 \>> \>> \:-define G \:-permutation_group \:
15425 \>> \>> \:-bsgs halljanko315 "File\halljanko315" \:
15426 \>> \>> \>> \:315 1209600 "0,1,42,95" 6 gens \:-end \:
15427 \>> \>> \:-define Orb \:-orbits \:-group G \:
15428 \>> \>> \>> \:-on_subsets 5 Control \:
15429 \>> \>> \:-end
# 1 orbit
#ROW,REP,AGO,OL
#0,"0,8,31,110,283",1200,1008
#END

### Chapter 14 - Combinatorial Objects

### Section 14.1: Finite Projective Spaces

**SECTION** COMBINATORIAL OBJECTS:

**Hirschfeld** $q_4$ from set:

```bash
> $(ORBITER) -v 4 \
> -define H -set -here \
> $(HIRSCHFELD_SURFACE_Q4_SET_OF_POINTS) \
> -end \
> -define C -combinatorial_objects \
> -set_of_points H \
> -end
```

**Hyperoval** $16$ create:

```bash
> $(ORBITER) -v 2 \
> -define C -combinatorial_objects \
> -set_of_points $(HYPEROVAL_16_16320) \
> -set_of_points $(HYPEROVAL_16_144) \
> -end \
```

**EC_read**: elliptic_curve_b1_c3_q11.txt

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

---

841
-define C -combinatorial_objects \ -file_of_points elliptic_curve_b1_c3_q11.txt \ -end

PG_3.5

assume 31

read:

$(ORBITER) -v 2 \ -define C -combinatorial_objects \ -file_of_packings_through_spread_table \ -file_of_packings H31_packings.csv \ -file_of_packings SPREAD_TABLES_5_REG/spread25_spreads.csv \ -end

LS_AG_2.3

read:

$(ORBITER) -v 2 \ -define C -combinatorial_objects \ -file_of_designs \ -file_of_solutions solutions.csv 9 84 3 12 \ -end

geo_7.3

read:

$(ORBITER) -v 10 \ -draw_incidence_structure_description \ -width 60 -with 10 6 -end \ -define C -combinatorial_objects \ -file_of_incidence_geometries \ 7.3.inc 7 7 21 \ -end

Desargues_path_lex_least

read:

echo $(DESARGUES_PATH_LEX_LEAST) >Desargues_path_lex_least.inc

$(ORBITER) -v 10 \ -draw_incidence_structure_description \ -width 60 -with 10 6 -end \ -define C -combinatorial_objects \ -file_of_incidence_geometries_by_row_ranks \ Desargues_path_lex_least.inc 10 10 3 \ -end
# Section 14.2: File Formats

geo_pasch_read:

```
$ (ORBITER) -v 10 \
define C -combinatorial_objects \
define -file_of_incidence_geometries \
fichierpasch.inc 6 4 12 \
end
```

given:

```
$ (ORBITER) -v 10 \
define C -combinatorial_objects \
define -incidence_geometry \
"0,1,4,6,8,11,13,14,17,19,22,23" \
6 4 12 \
end
```

# Chapter 15 - Canonical Forms with Nauty

# Section 15.1: Overview of Canonical Forms

SECTION_OVERVIEW_CANONICAL_FORMS:

SECTION_OBJECTS_IN_PROJECTIVE_SPACE:

EC_canon: elliptic_curve_b1_c3_q11.txt
$\text{(ORBITER)} -v 3$

$\text{(ORBITER)} -v 2 -draw_matrix$

$\text{(ORBITER)} -v 6$

Hirschfeld stab subgroup:

```bash
$ORBITER -v 9 -orbiter_path "$ORBITER_PATH" -define G -linear_group -PGGL 4 4 -subgroup_by_generators "Hirschfeld_Stab" -51840 6 "$HIRSCHFELD_STAB_GENERATORS" -end -define Gsp -modified_group -from G -create_special_subgroup -end -with Gsp -do -group_theoretic_activity -report -end -define Orb -orbits -group Gsp -on_points -end
```

```
# 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, 0,2,0,0,0,1,0, 1,0,0,0,0,3,0,0,0,0,3,0,1,0,0,1,0,1,0,0,0,0,1,1,0,1,0,0,1,0,1,0,0,1,0,0,0,0,1,0,0,0,0,0,1,0,1,0,0"```

```
845
```
Hirschfeld\_q4\_set\_c: \$\text{ORBITER}\ -v\ 4 \$

> define H -set -here \$

> define C -combinatorial\_objects \$

> set\_of\_points H \$

> end \$

> define F -finite\_field -q\ 4 -end \$

> define P -projective\_space -n\ 3 -field F -v\ 0 -end \$

- with C -do \$

- combinatorial\_object\_activity \$

- canonical\_form PG P \$

- classification\_prefix Hirschfeld\_surface\_q4 \$

- save\_ago \$

- end \$

Dickson\_sets\_stabilizer: \$

$\text{ORBITER}\ -v\ 3 \$

> define C -combinatorial\_objects \$

> set\_of\_points "0,1,2,5,6" \$

> set\_of\_points "0,1,2,3,6" \$

> set\_of\_points "0,1,2,3,4" \$

> set\_of\_points "0,1,2,3,8" \$

> set\_of\_points "0,1,2,5,6,7,8" \$

> set\_of\_points "0,1,2,3,5,6,7" \$

> set\_of\_points "0,1,2,3,5,6,9" \$

> set\_of\_points "0,1,2,3,5,6,10" \$

> set\_of\_points "0,1,2,3,5,6,4" \$

> set\_of\_points "0,1,2,3,5,6,8,11,13" \$

> set\_of\_points "3,6,9,7,10,12,8,11,13,14,4" \$

> set\_of\_points "3,5,6,9,7,10,12,11,13,14,4" \$

> set\_of\_points "0,1,2,3,5,6,9,7,10,12,4" \$

- end \$

> define F -finite\_field -q\ 2 -end \$

> define P -projective\_space -n\ 3 -field F -v\ 0 -end \$

- with C -do \$

- combinatorial\_object\_activity \$

- canonical\_form PG P \$

- classification\_prefix Dickson\_sets \$

- save\_ago \$

846
# group order is 32

```bash
15740 hyperoval_16_canonical_form:
15741 $(ORBITER) -v 2 \
15742 -define C -combinatorial_objects \
15743 -set_of_points $(HYPEROVAL_16_16320) \
15744 -set_of_points $(HYPEROVAL_16_144) \
15745 -end \
15746 -define F -finite_field -q 16 -end \
15747 -define P -projective_space -n 2 -field F -v 0 -end \
15748 -with C -do \
15749 -combinatorial_object_activity \
15750 -canonical_form_PG P \
15751 -classification_prefix hyperoval_q16 \
15752 -label hyperoval_q16 \
15753 -save_ag \
15754 -save_transversal \
15755 -max_TDO_depth 10 \
15756 -end \
```

847
cubic_curves_PG_2_8.canon:

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$

$\text{ORBITER} -v 6$
$\alpha \beta \gamma \delta$
classify

```
$\texttt{ORBITER} -v 6\
\texttt{\ -define C -combinatorial_objects \\
\texttt{\ -file_of_points \ \\
\texttt{\ -define F -finite_field -q 7 -end \\
\texttt{\ -define P -projective_space -n 3 -field F -v 0 -end \\
\texttt{\ -with C -do \\
\texttt{\ -combinatorial_object_activity \\
\texttt{\ -canonical_form_PG P \\
\texttt{\ -classification_prefix surface_15_lines.q7 \\
\texttt{\ -save_ago \\
\texttt{\ -save_transversal \\
\texttt{\ -end \\
\texttt{\ -end \\
\texttt{\ -pdf latex surface_15_lines.q7_classification.tex \\
\texttt{\ -open surface_15_lines.q7_classification.pdf}
```

```
$\texttt{ORBITER} -v 6 \\
\texttt{\ -define C -combinatorial_objects \\
\texttt{\ -file_of_points ovoid_q8.txt \\
\texttt{\ -end \\
\texttt{\ -define F -finite_field -q 8 -end \\
\texttt{\ -define P -projective_space -n 3 -field F -v 0 -end \\
\texttt{\ -with C -do \\
\texttt{\ -combinatorial_object_activity \\
\texttt{\ -canonical_form_PG P \\
\texttt{\ -classification_prefix ovoid \\
\texttt{\ -label ovoid \\
\texttt{\ -save_ago \\
\texttt{\ -max_TDO_depth 4 \\
\texttt{\ -end \\
\texttt{\ -report \\
\texttt{\ -prefix ovoid \\
\texttt{\ -show_TDO \\
```

\begin{verbatim}
15850   \> \> \> \> -show_TDA \\
15851   \> \> \> \> -dont_show_incidence_matrices \\
15852   \> \> \> \> -export_group_GAP \\
15853   \> \> \> -end \\
15854   \> -end
15855   pdflatex ovoid_classification.tex
15856   open ovoid_classification.pdf
15857
15858
15859   \#> \> \> -report \\
15860   \#> \> \> \> -prefix ovoid \\
15861   \#> \> \> \> -export_flag_orbits \\
15862   \#> \> \> \> -show_TDO \\
15863   \#> \> \> \> -show_TDA \\
15864   \#> \> \> \> -dont_show_incidence_matrices \\
15865   \#> \> \> \> -export_group_GAP \\
15866   \#> \> \> -end \\
15867
15868
15869   ovoid_ST_q8_canon: ovoid_ST_q8.txt
15870   \>$\{\text{ORBITER}\}\> \>-v \> 6 \\\n15871   \> \> \>-define C \>-combinatorial_objects \\\n15872   \> \> \> \>-file_of_points ovoid_ST_q8.txt \\\n15873   \> \> \>-end \\\n15874   \> \> \>-define F \>-finite_field \>-q \> 8 \>-end \\\n15875   \> \> \>-define P \>-projective_space \>-n \> 3 \>-field F \>-v \> 0 \>-end \\\n15876   \> \> \>-with C \>-do \\\n15877   \> \> \>-combinatorial_object_activity \\\n15878   \> \> \> \>-canonical_form_PG P \\\n15879   \> \> \> \> \>-classification_prefix ovoid_ST \\\n15880   \> \> \> \> \>-label ovoid_ST \\\n15881   \> \> \> \> \>-save_ago \\\n15882   \> \> \> \> \>-max_TDO_depth 4 \\\n15883   \> \> \> \>-end \\\n15884   \> \> \> -report \\\n15885   \> \> \> \> -prefix ovoid_ST \\\n15886   \> \> \> \> -show_TDO \\\n15887   \> \> \> \> -show_TDA \\\n15888   \> \> \> \> -dont_show_incidence_matrices \\\n15889   \> \> \> \> -export_group_GAP \\\n15890   \> \> \> \> -end \\\n15891   \> \> \>-end
15892   pdflatex ovoid_ST_classification.tex
15893   open ovoid_ST_classification.pdf
15894
15895   \# group order 87360 = 3 \> 29120
15896   SUZUKI_8GENERATORS="\n\end{verbatim}
Suzuki

\$(\text{ORBITER}) -v 6 \$

\$\text{define F -finite_field -q 8 -end} \$

\$\text{define gens -vector -field F} \$

\$\text{compact $(\text{SUZUKI_8_GENERATORS}) -end} \$

\$\text{define G -linear_group -PGGL 4 8} \$

\$\text{subgroup_by_generators "Sz8" "87360" 5 gens} \$

\$\text{end} \$

\$\text{with G -do } \$

\$\text{group_theoretic_activity} \$

\$\text{end}\$

\$\text{pdflatex PGGL_4_8_Subgroup_Sz8_87360_report.tex}\$

\$\text{open PGGL_4_8_Subgroup_Sz8_87360_report.pdf}\$

# Section 15.3: Incidence Geometries

SECTION_INCIDENCE_GEOMETRIES:

geo_7_3_c:

\$\text{define C -combinatorial objects} \$

\$\text{file of incidence geometries 7_3.inc 7 7 21} \$

\$\text{end} \$

\$\text{with C -do } \$

\$\text{combinatorial_object_activity} \$

\$\text{canonical_form} \$

\$\text{classification_prefix 7_3} \$

\$\text{label 7_3} \$

\$\text{save_ago} \$

\$\text{save_transversal} \$

\$\text{-end} \$

851
15944 ▶ ▶ ▶ -report \ 
15945 ▶ ▶ ▶ -prefix 7.3 \ 
15946 ▶ ▶ ▶ -export_flag_orbits \ 
15947 ▶ ▶ ▶ -show_incidence_matrices \ 
15948 ▶ ▶ ▶ -export_group_GAP \ 
15949 ▶ ▶ -end \ 
15950 ▶ -end 
15951 ▶ pdflatex 7.3_classification.tex 
15952 ▶ open 7.3_classification.pdf 
15953 ▶ $(ORBITER) -v 2 -draw_matrix \ 
15954 ▶ ▶ -input_csv_file 7.3_object0_TDA_flag_orbits.csv \ 
15955 ▶ ▶ -secondary_input_csv_file 7.3_object0_TDA.csv \ 
15956 ▶ ▶ -box_width 32 -bit_depth 24 \ 
15957 ▶ ▶ -end 
15958 ▶ $(ORBITER) -v 2 -draw_matrix \ 
15959 ▶ ▶ -input_csv_file 7.3_object0_INP_flag_orbits.csv \ 
15960 ▶ ▶ -secondary_input_csv_file 7.3_object0_INP.csv \ 
15961 ▶ ▶ -box_width 32 -bit_depth 24 \ 
15962 ▶ ▶ -end 
15963 ▶ open 7.3_object0_INP_flag_orbits_draw.bmp 
15964 
15965 
15966 
15967 geo_10.3_c: 
15968 ▶ $(ORBITER) -v 10 \ 
15969 ▶ ▶ -draw_incidence_structure_description \ 
15970 ▶ ▶ ▶ -width 60 -with_10 6 -end \ 
15971 ▶ ▶ -define C -combinatorial_objects \ 
15972 ▶ ▶ ▶ -file_of_incidence_geometries 10.3.inc 10 10 30 \ 
15973 ▶ ▶ ▶ -end \ 
15974 ▶ ▶ -with C -do \ 
15975 ▶ ▶ ▶ -combinatorial_object_activity \ 
15976 ▶ ▶ ▶ ▶ -canonical_form \ 
15977 ▶ ▶ ▶ ▶ ▶ -classification_prefix 10.3 \ 
15978 ▶ ▶ ▶ ▶ ▶ ▶ -label 10.3 \ 
15979 ▶ ▶ ▶ ▶ ▶ ▶ ▶ -save_ago \ 
15980 ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ -save_transversal \ 
15981 ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ -end \ 
15982 ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ -report \ 
15983 ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ -prefix 10.3 \ 
15984 ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ -export_flag_orbits \ 
15985 ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ -show_incidence_matrices \ 
15986 ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ -export_group_GAP \ 
15987 ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ -end \ 
15988 ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ -end \ 
15989 ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ -pdflatex 10.3_classification.tex 
15990 ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ open 10.3_classification.pdf
$(ORBITER) -v 2 -draw_matrix \
$(ORBITER) -v 2 -draw_matrix \
$ORBITER -v 10 -draw_incidence_structure_description \n-width 60 -with 10 6 -end \n-define Test_lines -set -loop 4 11 1 -end \n-define Geo -geometry_builder \n-V 10 -B 10 -TDO 3 -fuse 1 \n-fname_GEO 10_3 \n-test Test_lines \n-end \n-define C -combinatorial_objects \n-file_of_incidence_geometries 10_3.inc 10 10 30 \n-end \n-with C -do \n-combinatorial_object_activity \n-canonicial_form \n-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-end pdflatex 10_3_classification.tex
16038 \$ open 10_3_classification.pdf
16039 \$\$(ORBITER) -v 2 -draw_matrix \\
16040 \$ -input_csv_file 10_3_object7_TDA_flag_orbits.csv \\
16041 \$ -secondary_input_csv_file 10_3_object7_TDA.csv \\
16042 \$ -box_width 16 -bit_depth 24 \\
16043 \$ $ (ORBITER) -v 2 -draw_matrix \\
16044 \$ -input_csv_file 10_3_object7_INP_flag_orbits.csv \\
16045 \$ -secondary_input_csv_file 10_3_object7_INP.csv \\
16046 \$ -box_width 16 -bit_depth 24 \\
16047 \$ -end
16049
16050 #10_3_object7_TDA_flag_orbits.csv
16051
16052 #0 1 2 10 13 14 20 25 26 31 33 35 41 44 46 52 53 56 62 67 68 74 77 79 85 88 89
16053
16054
16055
16056 geo_14_3.c:
16057 $\$(ORBITER) -v 2 \\
16058 \$ -draw_incidence_structure_description \\
16059 \$ \$ -width 60 -with_10 6 -end \\
16060 \$ \$ -define Test_lines -set -loop 4 15 1 -end \\
16061 \$ \$ -define C -combinatorial_objects \\
16062 \$ \$ -file_of_incidence_geometries 14_3.inc 14 14 42 \\
16063 \$ \$ -end \\
16064 \$ \$ -with C -do \\
16065 \$ \$ -combinatorial_object_activity \\
16066 \$ \$ \$ -canonical_form \\
16067 \$ \$ \$ \$ -classification_prefix 14_3 \\
16068 \$ \$ \$ \$ -label 14_3 \\
16069 \$ \$ \$ \$ -save_ago \\
16070 \$ \$ \$ \$ -save_transversal \\
16071 \$ \$ \$ -end \\
16072 \$ \$ -end
16073
16074
16075 \$ \$ \$ -report \\
16076 \$ \$ \$ \$ -prefix 14_3 \\
16077 \$ \$ \$ \$ -export_flag_orbits \\
16078 \$ \$ \$ \$ -show_incidence_matrices \\
16079 \$ \$ \$ \$ -export_group_GAP \\
16080 \$ \$ \$ -end \\
16081
16082
16083 geo_15_3.c:
16084 $\$(ORBITER) -v 2 \\
16085
854
-draw.incidence.structure.description \ 
-width 50 -with 10 5 -end \ 
-define C -combinatorial.objects \ 
-file_of.incidence.geometries 15_3.inc 15 15 45 \ 
-end \ 
-with C -do \ 
-combinatorial.object.activity \ 
-canonlcal.form \ 
-classification_prefix 10_3 \ 
-label 10_3 \ 
-save.ago \ 
-end \ 
 pdflatex 15_3.classification.tex 
 open 15_3.classification.pdf 

16100 TFC_24.3_c: 
16101 echo $(FILE_24.3_TFC_INC) >24_3_TFC.inc 
16102 $(ORBITER) -v 6 \ 
16103 -define C -combinatorial.objects \ 
16104 -file_of.incidence.geometries 24_3_TFC.inc 24 24 72 \ 
16105 -end \ 
16106 -with C -do \ 
16107 -combinatorial.object.activity \ 
16108 -canonical.form \ 
16109 -classification_prefix 24_3_TFC \ 
16110 -label 24_3_TFC \ 
16111 -save.ago \ 
16112 -end \ 
16113 -report \ 
16114 -prefix 24_3_TFC \ 
16115 -export.flag.orbits \ 
16116 -show.TDO \ 
16117 -show.TDA \ 
16118 -show.incidence.matrices \ 
16119 -end \ 
16120 -end \ 
16121 pdflatex 24_3_TFC_classification.tex 
16122 open 24_3_TFC_classification.pdf 
16123 $(ORBITER) -v 2 -draw_matrix \ 
16124 -input.csv_file 24_3_TFC.object2_TDA.flag.orbits.csv \ 
16125 -secondary.input.csv_file 24_3_TFC.object2_TDA.csv \ 
16126 -box_width 40 -bit_depth 24 \ 
16127 -end \ 
16128 open 24_3_TFC.object2_TDA.flag.orbits.draw.bmp 
16129 
16130 
16131 geo_40_4_g4_c:
$(ORBITER) -v 2 \\
-draw_incidence_structure_description \\
-width 50 -with_10 5 -end \\
-define C -combinatorial_objects \\
-file_of_incidence_geometries 40_4_g4.inc 40 40 160 \\
-end \\
-with C -do \\
-combinatorial_object_activity \\
-canononical_form \\
-classification_prefix 40_4_g4 \\
-label 40_4_g4 \\
-save_ago \\
-end \\
-report \\
-prefix 40_4_g4 \\
-export_flag_orbits \\
-show_TDO \\
-show_TDA \\
-show_incidence_matrices \\
-end \\
-end

dflatex 40_4_g4_classification.tex
open 40_4_g4_classification.pdf
geo_17_3_g4.c:
$(ORBITER) -v 2 \\
-draw_incidence_structure_description \\
-width 50 -with_10 5 -end \\
-define C -combinatorial_objects \\
-file_of_incidence_geometries 17_3_g4.inc 17 17 51 \\
-end \\
-with C -do \\
-combinatorial_object_activity \\
-canononical_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

dflatex 17_3_g4_classification.tex
16179 \> open 17_3_g4_classification.pdf
16180
16181
16182 AG_2_3.c: AG_2.3.inc
16183 \> $(ORBITER) -v 2 \ 
16184 \> \> -define C -combinatorial_objects \ 
16185 \> \> \> -file_of_incidence_geometries \ 
16186 \> \> \> \> AG_2.3.inc 9 12 36 \ 
16187 \> \> -end \ 
16188 \> \> -with C -do \ 
16189 \> \> -combinatorial_object_activity \ 
16190 \> \> \> -canonical_form \ 
16191 \> \> \> \> -classification_prefix AG_2.3 \ 
16192 \> \> \> \> \> -label AG_2.3 \ 
16193 \> \> \> \> \> -save_ago \ 
16194 \> \> \> \> \> -max_TDO_depth 10 \ 
16195 \> \> \> \> -end \ 
16196 \> \> \> -report \ 
16197 \> \> \> \> -prefix AG_2.3 \ 
16198 \> \> \> \> \> -export_flag_orbits \ 
16199 \> \> \> \> \> -show_TDO \ 
16200 \> \> \> \> \> -show_TDA \ 
16201 \> \> \> \> \> -show_incidence_matrices \ 
16202 \> \> \> \> -end \ 
16203 \> \> -end
16204 \> pdflatex AG_2.3_classification.tex
16205 \> open AG_2.3_classification.pdf
16206 \> $(ORBITER) -v 2 -draw_matrix \ 
16207 \> \> -input_csv_file AG_2.3_object0_INP_flag_orbits.csv \ 
16208 \> \> \> -secondary_input_csv_file AG_2.3_object0_INP.csv \ 
16209 \> \> \> \> -box_width 40 \> bit_depth 24 \ 
16210 \> \> -end
16211 \> open AG_2.3_object0_INP_flag_orbits_draw.bmp
16212
16213
16214
16215
16216 geo_LSQ6.c:
16217 \> $(ORBITER) -v 10 \ 
16218 \> \> -draw_incidence_structure_description \ 
16219 \> \> \> -width 60 \> -with_10 6 \> -end \ 
16220 \> \> -define C -combinatorial_objects \ 
16221 \> \> \> -file_of_incidence_geometries \ 
16222 \> \> \> \> LSQ6.inc 18 39 126 \ 
16223 \> \> \> -end \ 
16224 \> \> -with C -do \ 
16225 \> \> \> -combinatorial_object_activity \ 

857
quartic_curve_25_0_0_canonical:

> $(ORBITER) -v 3 \
> -define F -finite_field -q 25 -end \
> -define P -projective_space -n 2 -field F -v 0 -end \
> -with P -do \
> -projective_space_activity \
> -canonical_form_PG \
> -input \
> -set_of_points "10,11,59,63,124,135,136,170,206,257,275,284,285,367,378,393,433,619,641,644" \
> -set_of_points "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"


16274 ▷ ▷ -set_of_points "2, 12, 48, 65, 87, 120, 189, 246, 305, 323, 354, 375, 434, 435, 455, 482, 496, 557, 586, 595" 

16275 ▷ ▷ -end 

16276 ▷ ▷ -classification_prefix quartic_25_0_0 

16277 ▷ ▷ -report 

16278 ▷ ▷ -end 

16279 ▷ ▷ -end 

16280 ▷ pdflatex quartic_25_0_0_classification.tex 

16281 ▷ open quartic_25_0_0_classification.pdf 

16282 

16283 

16284 

16285 

16286 geo_16.c: 

16287 ▷ $(ORBITER) -v 10 

16288 ▷ ▷ -draw_incidence_structure_description 

16289 ▷ ▷ ▷ -width 60 -with_10 6 -end 

16290 ▷ ▷ -define C -combinatorial_objects 

16291 ▷ ▷ ▷ -file_of_incidence_geometries geo_16.inc 16 20 80 

16292 ▷ ▷ -end 

16293 ▷ ▷ -with C -do 

16294 ▷ ▷ -combinatorial_object_activity 

16295 ▷ ▷ ▷ -canonical_form 

16296 ▷ ▷ ▷ ▷ -classification_prefix 16 

16297 ▷ ▷ ▷ ▷ -label 16 

16298 ▷ ▷ ▷ ▷ -save_ago 

16299 ▷ ▷ ▷ ▷ -save_transversal 

16300 ▷ ▷ ▷ -end 

16301 ▷ ▷ ▷ -report 

16302 ▷ ▷ ▷ ▷ -prefix 16 

16303 ▷ ▷ ▷ ▷ -export_flag_orbits 

16304 ▷ ▷ ▷ ▷ -show_incidence_matrices 

16305 ▷ ▷ ▷ ▷ -export_group_GAP 

16306 ▷ ▷ ▷ -end 

16307 ▷ ▷ -end 

16308 ▷ pdflatex 16_classification.tex 

16309 ▷ open 16_classification.pdf 

16310 ▷ $(ORBITER) -v 2 -draw_matrix 

16311 ▷ ▷ -input_csv_file 16_object0_TDA_flag_orbits.csv 

859
SECTION OBJECTS FROM DESIGN THEORY:

LS_AG_2_3.solutions.classify:

pdflatex LS_AG_2_3.classification.tex
open LS_AG_2_3.classification.pdf
$(ORBITER) -v 2 -draw_matrix
input_csv_file LS_AG_2_3.object0_INP_flag.orbits.csv
$-secondary_input_csv_file LS_AG_2_3.object0_INP.csv
$-box_width 12 -bit_depth 24 \\
$-end

Section 15.4: Objects from Design Theory
design.27c:

$(ORBITER) -v 4 \n-define F -finite_field -q 27 -override_polynomial 46 -end \n-define P -projective_space -n 2 -field F -v 0 -end \n-with C -do \n-define C -combinatorial_objects \n-set_of_points "2,56,30,112,253,90,440,508" \n-end \n-define F -finite_field -q 27 -override_polynomial 46 -end \n-define P -projective_space -n 2 -field F -v 0 -end \n-with C -do \n-define C -combinatorial_objects \n-set_of_points "2,56,30,112,253,90,440,508" \n-end \n-define F -finite_field -q 27 -override_polynomial 46 -end \n-define P -projective_space -n 2 -field F -v 0 -end \n-with C -do \n-define C -combinatorial_objects \n-set_of_points "2,56,30,112,253,90,440,508" \n-end \n-define F -finite_field -q 27 -override_polynomial 46 -end \n-define D -design -field F -family PG_2,q -end \n-with D -do \n-design_activity \n-export_inc \n-end \n-end

pdflatex design_classification.tex
open design_classification.pdf

design_PG_2.3_canonical:

$(ORBITER) -v 3 \n-define F -finite_field -q 3 -end \n-define D -design -field F -family PG_2,3 -end \n-with D -do \n-design_activity \n-export_inc \n-end

$(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

861
Combinatorial object activity
-categorical form
-classification_prefix PG_2_3
-label PG_2_3
-save_after
-save_transversal
-end
-report
-prefix PG_2_3
-export_flag_orbits
-show_incidence_matrices
-export_group_GAP
-end
-end
-open PG_2_3_classification.pdf
-
-
-wreath_product_designs_n4_k2_c: wreath_product_designs_n4_k2_inc.txt
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
Section 15.5: Linear Codes

SECTION_CANONICAL_FORMS_OF_LINEAR_CODES:

code_3_2.aut:
\[ \text{define \ F -finite_field -q 2 -end} \]
\[ \text{define \ genma -vector -field \ F -format 2} \]
\[ \text{-dense $(\text{CODE}_N3_K2_Q2\text{GENMA})}$ \]
\[ \text{define \ P -projective_space -n 1 -field \ F -v 0 -end} \]
\[ \text{with \ P -do} \]
\[ \text{-projective_space_activity} \]
\[ \text{-canonical_form_of_code} \]
\[ \text{"3_2" \ genma -save_ago -label "3_2"} \]
\[ \text{-classification_prefix "3_2"} \]
\[ \text{-end} \]
\[ \text{pdflatex 3_2.classification.tex} \]
\[ \text{open 3_2.classification.pdf} \]
\[ \text{\$(\text{ORBITER}) -v 2 -draw_matrix} \]
\[ \text{-input_csv_file 3_2.object0_TDA_flag_orbits.csv} \]
\[ \text{-secondary_input_csv_file 3_2.object0_TDA.csv} \]
\[ \text{-box_width 16 -bit_depth 24} \]
\[ \text{open 3_2.object0_TDA_flag_orbits_draw.bmp} \]
\[ \text{\$(\text{ORBITER}) -v 20} \]
\[ \text{-define \ F -finite_field -q 2 -end} \]
\[ \text{-define \ genma -vector -field \ F -format 3} \]
\[ \text{-compact $(\text{CODE}_N6_K3_Q2\text{GENMA})}$ \]
\[ \text{define \ P -projective_space -n 2 -field \ F -v 0 -end} \]
\[ \text{with \ P -do} \]
\[ \text{-projective_space_activity} \]
\[ \text{-canonical_form_of_code} \]
\[ \text{"6_3" \ genma -save_ago -label "6_3"} \]
\[ \text{-classification_prefix "6_3"} \]
\[ \text{-end} \]
\[ \text{pdflatex 6_3.classification.tex} \]
\[ \text{open 6_3.classification.pdf} \]
\[ \text{\$(\text{ORBITER}) -v 2 -draw_matrix} \]
\[ \text{-input_csv_file 6_3.object0_TDA_flag_orbits.csv} \]
\[ \text{-secondary_input_csv_file 6_3.object0_TDA.csv} \]
\[ \text{-box_width 16 -bit_depth 24} \]
\[ \text{open 6_3.object0_TDA_flag_orbits_draw.bmp} \]
# group of order 24

RM_3.1_group:

$\text{GROUP}$ -v 2 \\
-e -define F -finite_field -q 2 -end \\
-e -define genma -vector -field F -format 4 \\
-e -compact $(\text{CODE}_{\text{RM}_3.1_{\text{GENMA}}})$ \\
-e -end \\
-e -define P -projective_space -n 3 -field F -v 0 -end \\
-e -with P -do \\
-e -projective_space_activity \\
-e -canonical_form_of_code \\
-e -"RM_3.1" genma -save_ago -label "RM_3.1" \\
-e -classification_prefix "RM_3.1" \\
-e -end \\
-e -end

pdflatex RM_3.1_classification.tex

open RM_3.1_classification.pdf

# group order 1344

RM_3.1_object0_INP_flag_orbits.csv

RM_3.1_group_and_diagram:

$\text{GROUP}$ -v 2 \\
-e -define F -finite_field -q 2 -end \\
-e -define genma -vector -field F -format 4 \\
-e -compact $(\text{CODE}_{\text{RM}_3.1_{\text{GENMA}}})$ \\
-e -end \\
-e -define P -projective_space -n 3 -field F -v 0 -end \\
-e -with P -do \\
-e -projective_space_activity \\
-e -canonical_form_of_code \\
-e -"RM_3.1" genma -save_ago -label "RM_3.1" \\
-e -classification_prefix "RM_3.1" \\
-e -end \\
-e -end

pdflatex RM_3.1_classification.tex

open RM_3.1_classification.pdf

$\text{GROUP}$ -v 2 -draw_matrix \\
-e -input_csv_file RM_3.1_object0_INP_flag_orbits.csv \\
-e -secondary_input_csv_file RM_3.1_object0_INP.csv \\
-e -box_width 16 -bit_depth 24 \\
-e -end

$\text{GROUP}$ -v 2 -draw_matrix \\
-e -input_csv_file RM_3.1_object0_TDA_flag_orbits.csv \\

865
# group order 322560 = 24*30*28*16

```
16632 # group order 322560 = 24*30*28*16
16633
16634
16635 RS_6_4_7_group:
16636 > $(ORBITER) -v 20 
16637 > > -define F -finite_field -q 7 -end 
16638 > > -define genma -vector -field F -format 4 
16639 > > > -compact $(CODE_RS_6_4_7) 
16640 > > > -end 
```
\begin{verbatim}
GV_n15_k6_d5_group:
$\langle$ORBITER$\rangle$ -v 20
$\langle$define F -finite_field -q 2 -end$\rangle$
$\langle$define genma -vector -field F -format 6$\rangle$
$\langle$compact \text{CODE GV}_n15_k6_d5$\rangle$
$\langle$end$\rangle$
$\langle$define P -projective_space -n 5 -field F -v 0 -end$\rangle$
$\langle$with P -do$\rangle$
$\langle$projective_space_activity$\rangle$
$\langle$canonical_form_of_code$\rangle$
$\langle$"n15_k6_d5\" genma -save_ago -label "GV_n15_k6_d5\"$\rangle$
$\langle$classification_prefix "GV_n15_k6_d5\"$\rangle$
$\langle$end$\rangle$

pdflatex GV_n15_k6_d5_classification.tex
open GV_n15_k6_d5_classification.pdf

#ago=12

GV_n15_k6_d6_a_group:
$\langle$ORBITER$\rangle$ -v 20
$\langle$define F -finite_field -q 2 -end$\rangle$
$\langle$define genma -vector -field F -format 6$\rangle$
$\langle$compact \text{CODE 15}_{6.6.6}\rangle$
$\langle$end$\rangle$
$\langle$define P -projective_space -n 5 -field F -v 0 -end$\rangle$
$\langle$with P -do$\rangle$
$\langle$projective_space_activity$\rangle$
$\langle$canonical_form_of_code$\rangle$
$\langle$"n15_k6_d6_a\" genma -save_ago -label "n15_k6_d6_a\"$\rangle$
$\langle$classification_prefix "n15_k6_d6_a\"$\rangle$
$\langle$end$\rangle$

pdflatex n15_k6_d6_a_classification.tex
open n15_k6_d6_a_classification.pdf
\end{verbatim}
group:

```bash
$ (ORBITER) -v 20 \
-define F -finite_field -q 2 -end \
-define genma -vector -field F -format 6 \
-compact $(CODE_{15,6,6,B}) \
-define P -projective_space -n 5 -field F -v 0 -end \
-define P -projective_space_activity \
-compact $(CODE_{15,6,6,B}) \n-define P -projective_space_activity -canonical_form_of_code \
"n15_k6_d6_b" genma -save -label "n15_k6_d6_b" \n-classification_prefix "n15_k6_d6_b" \n-end \
-end
```

```bash
pdflatex n15_k6_d6_b_classification.tex
open n15_k6_d6_b_classification.pdf
```

# Section 15.6: General Codes

SECTION CANONICAL FORMS OF GENERAL CODES:

Hamming graph 7 with Hamming code:

```bash
$ (ORBITER) -v 2 \
-define G -graph -Hamming 7 2 \
-subset "Hamming_code" "\\with\Hamming\_code" \
$(HAMMING_CODE_CODEWORDS) -end \
-with G -do \
-graph_theoretic_activity -export_csv -end \
-with G -do \
-graph_theoretic_activity -export_graphviz -end \
-with G -do \
-graph_theoretic_activity -save -end \
-automorphism_group -end
```

```bash
pdflatex Hamming_7_2_Hamming_code_report.tex
open Hamming_7_2_Hamming_code_report.pdf
```

# group of order 2688 = 16 * 168
# Section 15.7: Graphs

SECTION CANONICAL FORMS OF GRAPHS:

Cycle

$\text{aut:}$

\begin{verbatim}
$\text{define Gamma -graph -cycle 13 -end \}
\text{with Gamma -do \}
\text{-graph_theoretic_activity -automorphism_group \}
\text{-end \}
\end{verbatim}

inversion_graph:

\begin{verbatim}
$\text{define G -graph \}
\text{inversion_graph "1,0,2,3" \]
\text{-end \]
\text{with G -do \}
\text{-graph_theoretic_activity -properties \)
\text{-end \}
\text{with G -do \}
\text{-graph_theoretic_activity -automorphism_group \)
\text{-end \}
\end{verbatim}

Chain

$\text{aut:}$

\begin{verbatim}
$\text{define P1 -vector -dense 2,3,2 -end \}
$\text{define P2 -vector -dense 2,3,2 -end \}
$\text{define Gamma -graph \}
\text{chain_graph P1 P2 \]
\text{-end \]
\text{with Gamma -do \)
\text{-graph_theoretic_activity -automorphism_group \)
\text{-end \}
\end{verbatim}
JK_graph_pp16_1:
  $(ORBITER) -v 2 \
  -define Gamma -graph -load_dimacs \n  ../JUNTTILA_KASKI/benchmarks/pp/pp16-1 \n  -end \n  -with Gamma -do \n  -graph_theoretic_activity -save -end \n  -with Gamma -do \n  -graph_theoretic_activity -automorphism_group -end 

# go=34217164800

#nauty_interface_with_group::create_automorphism_group_of_graph_with_partition_an
d labeling: nb_backtrack1 = 6

#nauty_interface_with_group::create_automorphism_group_of_graph_with_partition_an
d labeling: nb_backtrack2 = 134

JK_graph_pp16_2:
  $(ORBITER) -v 2 \
  -define Gamma -graph -load_dimacs \n  ../JUNTTILA_KASKI/benchmarks/pp/pp16-2 \n  -end \n  -with Gamma -do \n  -graph_theoretic_activity -save -end \n  -with Gamma -do \n  -graph_theoretic_activity -automorphism_group -end 

# does not finish

JK_graph_pp16_9:
  $(ORBITER) -v 2 \
  -define Gamma -graph -load_dimacs \n  ../JUNTTILA_KASKI/benchmarks/pp/pp16-9 \n  -end \n  -with Gamma -do \n  -graph_theoretic_activity -save -end \n  -with Gamma -do \n  -graph_theoretic_activity -automorphism_group -end 

JK_graph_grid_3_3:
  $(ORBITER) -v 2 \
  -define Gamma -graph -load_dimacs \n  ../JUNTTILA_KASKI/benchmarks/grid/grid-w-3-3 \n  -end \n  -with Gamma -do \

# does not finish
16827  ▶  ▶  -graph_theoretic_activity -save -end \\
16828  ▶  ▶  -with Gamma -do \\
16829  ▶  ▶  -graph_theoretic_activity -automorphism_group -end \\
16830
16831
16832  #nauty_interface_with_group::create_automorphism_group_of_graph_with_partition_an
d_labeling: nb_backtrack1 = 4
16833  #nauty_interface_with_group::create_automorphism_group_of_graph_with_partition_an
d_labeling: nb_backtrack2 = 9
16834  #Written file grid-w-3-3_group.makefile of size 579
16835  #User time: 0 of a second, dt=0 tps = 100
16836  #nb_calls_to_densenauty=1
16837
16838
16839  JK_graph_sts_13:
16840  ▶  $(ORBITER) -v 2 \\
16841  ▶  ▶  -define Gamma -graph -load_dimacs \\
16842  ▶  ▶  ▶  ../JUNTTILA_KASKI/benchmarks/srg/sts-13 \\
16843  ▶  ▶  -end \\
16844  ▶  ▶  -with Gamma -do \\
16845  ▶  ▶  ▶  -graph_theoretic_activity -save -end \\
16846  ▶  ▶  -with Gamma -do \\
16847  ▶  ▶  ▶  -graph_theoretic_activity -automorphism_group -end \\
16848  ▶  ▶  make ORBITER_PATH=$(ORBITER_PATH) -f sts-13_group.makefile sts-13
16849
16850
16851  #nauty_interface_with_group::create_automorphism_group_of_graph_with_partition_an
d_labeling: nb_backtrack1 = 3
16852  #nauty_interface_with_group::create_automorphism_group_of_graph_with_partition_an
d_labeling: nb_backtrack2 = 24
16853
16854
16855  HJ_aut:
16856  ▶  $(ORBITER) -v 6 \\
16857  ▶  ▶  -define G -graph \\
16858  ▶  ▶  ▶  -load_csv_no_border \\
16859  ▶  ▶  ▶  halljanko315.csv \\
16860  ▶  ▶  ▶  -end \\
16861  ▶  ▶  ▶  -with G -do \\
16862  ▶  ▶  ▶  ▶  -graph_theoretic_activity -automorphism_group \\
16863  ▶  ▶  ▶  ▶  -end \\
16864  ▶  ▶  ▶  ▶  -with G -do \\
16865  ▶  ▶  ▶  ▶  ▶  -graph_theoretic_activity -properties \\
16866  ▶  ▶  ▶  ▶  ▶  -end
16867
16868
16869
HJ_group_and_orbits:

$\$(ORBITER) -v 2 \n
-define Control -poset_classification_control \n-define -w \n-define_problem_label HJ_orbits \n-define_depth 2 \n-end \n
-define gens -vector -file \n-define_halljanko315_gens.csv -end \n-define G -permutation_group \n-define_halljanko315 "File\halljanko315" \n-define_315 1209600 "0,1,2" 6 gens \n-end \n
-define Orb -orbits -group G \n-define_on_subsets 2 Control \n-end \n
#ROW,REP,AGO,OL
#0,"0,1",96,12600
#1,"0,2",48,25200
#2,"0,4",768,1575
#3,"0,8",120,10080
#END

HJ_orbital_graph_3:

$\$(ORBITER) -v 2 \n
-define gens -vector -file \n-define_halljanko315_gens.csv -end \n-define G -permutation_group \n-define_halljanko315 "File\halljanko315" \n-define_315 1209600 "0,1,2" 6 gens \n-end \n
-define Gamma -graph \n-define_orbital_graph G 3 \n-end \n
-with Gamma -do \n-define_graph_theoretic_activity \n-define_properties \n-end \n
-with Gamma -do \n-define_graph_theoretic_activity \n-save \n-end

#Group_Perm315_Orbital_3.colored_graph

#Degree type: (64^{315} )
PGO_5_2_graph_group: 0_5_2_incidence_matrix.csv
$\texttt{(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 \ 
\texttt{-automorphism_group \ 
-end \ 
with Gamma -do \ 
-graph_theoretic_activity \ 
eigenvalues \ 
-end \ 
pdflatex collinearity_graph_eigenvalues.tex \open collinearity_graph_eigenvalues.pdf}

# Section 15.8: Quartic Curves

SECTION_CANONICAL_FORMS_OF_QUARTIC_CURVES:

F_{17}\_edge:
$\texttt{(ORBITER) -v 3 \ \\
define F -finite_field -q 17 -end \ 
with F -do -finite_field_activity \ 
\texttt{-cheat_sheet_GF -end \ 
pdflatex GF_{17}.tex \open GF_{17}.pdf}$
Edge_curve_17.nauty:

\$ (ORBITER) -v 3 \
-define C -combinatorial_objects \
-file_of_points Edge_q17.txt \
-end \
-define F -finite_field -q 17 -end \
-define P -projective_space -n 2 -field F -v 0 -end \
-with C -do \
-combinatorial_object_activity \
-cannotonical_form PG P \
-classification_prefix Edge_curve_q17 \
-label Edge_curve_q17 \
-save Ago \
-save_transversal \
-max_TDO_depth 10 \
-end \
-report \
-prefix Edge_curve_q17 \
-export_flag_orbits \
-show_TDO \
-show_TDA \
-dont_show_incidence_matrices \
-export_group_GAP \
-end \
-end

pdflatex Edge_curve_q17_classification.tex
open Edge_curve_q17_classification.pdf

\$ (ORBITER) -v 2 -draw_matrix \
-input_csv_file Edge_curve_q17_object0_TDA_flag_orbits.csv \
-secondary_input_csv_file Edge_curve_q17_object0_TDA.csv \
-box_width 4 -bit_depth 24 \
-end
open Edge_curve_q17_object0_TDA_flag_orbits.draw.bmp

# 9 backtrack nodes total

# aut = 24

# User time: 0.04 of a second, dt=4 tps = 100

# generators for a group of order 24:
#1,0,0,0,13,0,0,0,4, 
#1,0,0,0,0,16,0,16,0, 
#0,1,16,2,4,4,15,4,4, 

874
Edge_curve_17_group:

$\$\text{(ORBITER)}$ -v 3 \ 
$\$\text{-define } G \text{-linear_group } $-PGL 3 17 \ 
$\$\text{-subgroup_by_generators } "\text{Stab}\_\text{Edge}" "24" 3 \ 
$\$\text{-end}$ \ 
$\$\text{-with } G \text{-do}$ \ 
$\$\text{-group_theoretic_activities}$ \ 
$\$\text{-print_elements_tex}$ \ 
$\$\text{-end}$ \ 
$\$\text{pdflatex } PGL\_3\_17\_Subgroup\_Stab\_Edge\_24\_report.tex$ \ 
$\$\text{open } PGL\_3\_17\_Subgroup\_Stab\_Edge\_24\_report.pdf$ \ 

# Section 16.1: Graphical Output:

F_7.tables:

$\$$\text{(ORBITER)}$ -v 3 \ 
$\$$\text{-define } F \text{-finite_field } $-q 7 -end \ 
$\$$\text{-with } F \text{-do } \text{-finite_field_activity}$ \ 
$\$$\text{-cheat_sheet_GF}$ \ 
$\$$\text{-end}$ \ 
$\$$\text{open } GF\_q7\_addition_table\_draw.bmp$
17058
17059
17060
17061 PG_2_4_cyclic_incma:
17062 \>
17063 \>
17064 \>
17065 \>
17066 \>
17067 \>
17068 \>
17069 \>
17070 \>
17071 \>
17072 \>
17073 \>
17074 \>
17075 \>
17076 \>
17077 \>
17078 \>
17079
17080
17081
17082
17083 PGL_4_2_Wedge_4_0_graphical_output:
17084 \>
17085 \>
17086 \>
17087 \>
17088 \>
17089 \>
17090 \>
17091 \>
17092 \>
17093 \>
17094
17095 \>
17096 schreier_tree_graphical_output:
17097 \>
17098 \>
17099 \>
17100 \>
17101 \>
17102 \>
17103 \>
17104 \>

876
\[
\text{define G -linear group -PGL 4 2 -end} \backslash
\]
\[
\text{define Orb -orbits -group G} \backslash
\]
\[
\text{on polynomials 3 -draw tree 6} \backslash
\]
\[
\text{end} \backslash
\]
\[
\text{pdflatex poly_orbits_d3_n3_q2_orbit_6.tree.tex} \backslash
\]
\[
\text{open poly_orbits_d3_n3_q2_orbit_6.tree.pdf} \backslash
\]
\[
\text{#pdflatex poly_orbits_d3_n3_q2.tex} \backslash
\]
\[
\text{#open poly_orbits_d3_n3_q2.pdf} \backslash
\]
\[
\text{with G -do} \backslash
\]
\[
\text{group_theoretic_activity} \backslash
\]
\[
\text{orbits_on_polynomials 3} \backslash
\]
\[
\text{orbits_on_polynomials draw.tree 6} \backslash
\]
\[
\text{end} \backslash
\]
\[
\text{Queens_graph:} \backslash
\]
\[
\text{$(ORBITER) -v 2} \backslash
\]
\[
\text{define G -graph -non_attacking_queens_graph 8 -end} \backslash
\]
\[
\text{with G -do} \backslash
\]
\[
\text{graph_theoretic_activity -export.csv -end} \backslash
\]
\[
\text{with G -do} \backslash
\]
\[
\text{graph_theoretic_activity -export_graphviz -end} \backslash
\]
\[
\text{with G -do} \backslash
\]
\[
\text{graph_theoretic_activity -save -end} \backslash
\]
\[
\text{with G -do} \backslash
\]
\[
\text{graph_theoretic_activity -automorphism_group -end} \backslash
\]
\[
\text{with G -do} \backslash
\]
\[
\text{graph_theoretic_activity -findCliques} \backslash
\]
\[
\text{target_size 8 -output_file 8queens -end} \backslash
\]
\[
\text{end} \backslash
\]
\[
\text{Section 16.2: The Povray Interface} \backslash
\]
\[
\text{SECTION_POVRAY:} \backslash
\]
\[
\text{cube:} \backslash
\]
\[
\text{$(ORBITER) -v 2 -povray} \backslash
\]
\[
\text{round 0 -nb_frames_default 30} \backslash
\]
\[
\text{output_mask cube_%d_%03d.pov} \backslash
\]
-video_options -W 1024 -H 768 \
-global_picture_scale 0.5 \
-default_angle 75 \
-clipping_radius 2.7 \
-end \
-scene_objects \
-obj_file cube_centered.obj \
-edge "0, 1" \
-edge "0, 2" \
-edge "0, 4" \
-edge "1, 3" \
-edge "1, 5" \
-edge "2, 3" \
-edge "2, 6" \
-edge "3, 7" \
-edge "4, 5" \
-edge "4, 6" \
-edge "5, 7" \
-edge "6, 7" \
-group_of_things_as_interval 0 8 \
-spheres 0 0.3 $(POLISHED_CHROME_WHITE) \
-group_of_things_as_interval 0 6 \
-prisms 1 0.05 $(YELLOW_TRANSPARENT) \
-group_of_things_as_interval 0 12 \
-cylinders 2 0.15 $(COLOR_RED) \
-scene_objects_end \
-povray_end \
-rm -rf POV 

mkdir POV 

mv cube_0_*.pov POV 

mv makefile_animation POV 

math261_test: 

$(ORBITER) -v 2 -povray \
-round 0 -nb_frames_default 30 \
-output_mask math261_%d%03d.pov \
-video_options -W 1024 -H 768 \
-global_picture_scale 0.1 \
-default_angle 75 \
-clipping_radius 2.7 \
-end \
-scene_objects \
-point "0,0,0" \
-point "5,0,0" \
-point "0,5,0" \
-point "0,0,5" \

-point "1,2,3" \\
-edge "0,1" \\
-edge "0,2" \\
-edge "0,3" \\
-edge "0,4" \\
-edge "0,5" \\
-edge "4,6" \\
-edge "5,6" \\
-face "0,4,6,5" \\
-group_of_things_as_interval 0 7 \\
-spheres 0 0.1 $(POLISHED_CHROME_WHITE) \\
-group_of_things_as_interval 0 7 \\
-cylinders 1 0.05 $(COLOR_RED) \\
-prisms 2 0.05 $(YELLOW_TRANSPARENT) \\
-group_of_things_as_interval 0 1 \\
-scene_objects_end \\
-povray_end \\

-rm -rf POV \\
mkdir POV \\
mv math261_0*.pov POV \\
mv makefile_animation POV \\

plane1: \\
$(ORBITER) -v 2 -povray \\
-round 0 -nb_frames_default 30 \\
-output_mask plane1_%d_%03d.pov \\
-video_options -W 1024 -H 768 \\
-global_picture_scale 0.40 \\
-default_angle 75 \\
-clipping_radius 5 -omit_bottom_plane \\
-camera 0 "0,0,1" "5,5,3" "0,0,0" \\
-rotate_about_z_axis \\
-boundary_box \\
-end \\
-scene_objects \\

-line_through_two_points_recentered_from_csv_file coordinate_grid.csv \\
-plane_by_dual_coordinates "0,0,1,0" \\
-plane_by_dual_coordinates "0,1,0,0" \\
-plane_by_dual_coordinates "1,0,0,0" \\
-point "-2,25,0,0" \\
-point "0,-1,8,0" \\
-point "0,0,9" \\
-face "0,1,2,0"
-group_of_things "0"
-group_of_things "1"
-group_of_things "2"
-lines 0 0.15 $(COLOR\_RED\_SHINY)
-lines 1 0.15 $(COLOR\_GREEN\_SHINY)
-lines 2 0.15 $(COLOR\_BLUE\_SHINY)
-group_of_things_as_interval 3 39
-lines 3 0.05 $(COLOR\_BLACK\_SHINY)
-group_of_things "0"
-group_of_things "1"
-group_of_things "2"
-prisms 0 0.05 $(COLOR\_YELLOW\_THICK)
-scene_objects_end
-povray_end
- rm -rf POV
mkdir POV
mv plane1_0_*.pov POV
mv makefile_animation POV

plane2:
$(ORBITER) -v 2 -povray
-round 0 -nb_frames_default 30
-output_mask plane2_%d_03d.pov
-video_options -W 2560 -H 1920
-global_picture_scale 0.40
-default_angle 75
-clipping_radius 5 -omit_bottom_plane
-camera 0 "0,0,1" "6,6,2" "0,0,0"
-rotate_about_z_axis
-boundary_box
-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"
-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)
17293 -lines 2 0.15 $(COLOR_BLUE_SHINY) \n17294 -lines 3 0.05 $(COLOR_BLACK_SHINY) \n17295 -group_of_things "0" \n17296 -planes 4 $(COLOR_BLUE_SEE THROUGH) \n17297 -group_of_things "3" \n17298 -scene_objects_end \n17299 -povray_end
17300 - rm -rf POV
17301 mkdir POV
17302 mv plane2_0.pov POV
17303 mv makefile_animation POV
17304
17305 # -planes 5 "texture{ pigment{ color Yellow transmit 0.5 } finish{ diffuse 0.9 phong 1}}" \n17306
17307
17308 analytic_geo_1:
17309 $(ORBITER) -v 2 -povray \n17310 -round 0 -nb_frames_default 30 \n17311 -output_mask analytic_geo_1_%d_%03d.pov \n17312 -video_options -W 2560 -H 1920 \n17313 -global_picture_scale 0.80 \n17314 -default_angle 75 \n17315 -clipping_radius 5 -omit_bottom_plane \n17316 -camera 0 "0,0,1" "6,6,2" "0,0,0" \n17317 -rotate_about_z_axis \n17318 -boundary_box \n17319 -end \n17320 -scene_objects \n17321 -line_through_two_points_recentered_from_csv_file coordinate_grid.csv \n17322 -plane_by_dual_coordinates "0,0,1,0" \n17323 -plane_by_dual_coordinates "0,1,0,0" \n17324 -plane_by_dual_coordinates "1,0,0,0" \n17325 -group_of_things "0" \n17326 -group_of_things "1" \n17327 -group_of_things "2" \n17328 -group_of_things_as_interval 3 39 \n17329 -lines 0 0.05 $(COLOR_RED_SHINY) \n17330 -lines 1 0.05 $(COLOR_GREEN_SHINY) \n17331 -lines 2 0.05 $(COLOR_BLUE_SHINY) \n17332 -lines 3 0.04 $(COLOR_BLACK_SHINY) \n17333 -group_of_things "0" \n17334 -group_of_things "1" \n17335 -group_of_things "2" \n17336 -planes 4 $(COLOR_BLUE_SEE THROUGH) \n17337 -planes 5 $(COLOR_GREEN_SEE THROUGH) \n17338
$\text{planes 6 (COLOR\_RED\_SEE\_THROUGH) }$

$\text{point } "0,0,0"$

$\text{point } "1,0,0"$

$\text{point } "1,2,0"$

$\text{point } "1,2,3"$

$\text{edge } "84,85"$

$\text{edge } "85,86"$

$\text{edge } "86,87"$

$\text{edge } "84,87"$

$\text{group_of_things } "84,85,86"$

$\text{spheres 7 0.1 (POLISHED\_CHROME\_WHITE) }$

$\text{group_of_things } "87"$

$\text{spheres 8 0.10 (COLOR\_YELLOW\_SHINY) }$

$\text{group_of_things } "0,1,2"$

$\text{cylinders 9 0.075 (POLISHED\_CHROME\_WHITE) }$

$\text{cylinders 10 0.075 (COLOR\_YELLOW\_SHINY) }$

$\text{scene\_objects\_end}$

$\text{povray\_end}$

$\text{rm -rf POV}$

$\text{mkdir POV}$

$\text{mv analytic\_geo\_1\_0\_*.pov POV}$

$\text{mv makefile\_animation POV}$

$\text{analytic\_geo\_1\_video:}$

$\text{rm -r FRAMES}$

$\text{mkdir FRAMES}$

$\text{rm analytic\_geo\_1.mp4}$

$\text{$(ORBITER) }$

$\text{prepare\_frames }$

$\text{i 0 30 PNG/ANALYTIC\_GEO\_1/analytic\_geo\_1\_0\_%03d.png }$

$\text{output\_starts\_at 0 }$

$\text{o FRAMES/frame\_%04d.png }$

$\text{-end}$

$\text{ffmpeg -r 5 -f image2 -i FRAMES/frame\_%04d.png }$

$\text{-f mp4 -q:v 0 -vcodec mpeg4 analytic\_geo\_1.mp4}$

$\text{monkey:}$

$\text{$(ORBITER) -v 2 -povray }$

$\text{round 0 -nb\_frames\_default 30 }$

$\text{output\_mask monkey\_%d\_%03d.pov }$

$\text{video\_options -W 1024 -H 768 }$

$\text{global\_picture\_scale 0.8 }$

$\text{default\_angle 75 }$

$\text{clipping\_radius 0.8 }$

$\text{camera 0 "0,0,1" "1,1,0.5" "0,0,0" }$
-rotate_about_z_axis
-end
-scene_objects
-cubic_lex $(MONKEY_SADDLE_CUBIC)
-plane_by_dual_coordinates "0,0,1,0"
-group_of_things "0"
-group_of_things "0"
-cubics 0 $(COLOR_GOLD)
-planes 1 $(COLOR_BLUE)
-scene_objects_end
-povray_end

mkdir POV
mv monkey.0.*.pov POV
mv makefile_animation POV

Eckardt:
$(ORBITER) -v 2 -povray
-round 0 -nb_frames_default 30
-output_mask Eckardt_%d.03d.pov
-video_options -W 1024 -H 768
-global_picture_scale 0.9
-default_angle 75
-clipping_radius 2.4
-camera 0 "1,1,1" "+3,1,3" "0.12,0.12,0.12"
-scene_objects
-Hilbert_Cohn_Vossen_surface
-group_of_things "0"
-cubics 0 $(SURFACE_COLOR)
-group_of_things_as_interval 0 6
-group_of_things_as_interval 6 6
-group_of_things_as_interval_with_exceptions 12 15
"14,19,23"
-lines 1 0.02 $(COLOR_RED_SHINY)
-lines 2 0.02 $(COLOR_BLUE_SHINY)
-lines 3 0.02 $(COLOR_YELLOW_SHINY)
-label 0 "a1"
-label 2 "a2"
-label 4 "a3"
-label 6 "a4"
-label 8 "a5"
-label 10 "a6"
-label 12 "b1"
Eckardt deform:

```
{-#3,2.333,4" * 1.5 = "-4.5,3.5,6
17464 #M := Matrix([[[-4.5, 3.5, 6], [1, 1, 1]])
17466 #NullSpace(M)
17467 #=0.186080731891197,-0.781539073943026, 0.595458342051830
17468 #> -rotate about custom_axis "0.186080731891197,-0.781539073943026,0.595458342051830"
```

```
17469 #W 1024 -H 768
17470 #W 2560 -H 1920
17471 #W 4096 -H 3072
```

```
17468 #> -rotate about custom_axis "0.186080731891197,-0.781539073943026,0.595458342051830"
```

```
17476 Eckardt deform:
17477 #> $(ORBITER) -v 2 -povray
17478 #> -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: 
$(ORBITER) -v 2 -povray 
-round 0 -nb_frames_default 30 
-output_mask Eckardt_deform_%d_%03d.pov 
-video_options -W 1024 -H 768 
-global_picture_scale 0.9 
-default_angle 75 
-clipping_radius 2.4 
-camera 0 "1,1,1" "-3,1,3" "0.12,0.12,0.12" 
-end 
-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) 

17539  Clebsch:
17540  $(ORBITER) -v 2 -povray \n17541  -round 0 -nb_frames_default 30 \n17542  -output_mask Clebsch.%d%03d.pov \n17543  -video_options -W 1024 -H 768 \n17544  -global_picture_scale 0.9 \n17545  -default_angle 80 \n17546  -clipping_radius 2.4 \n17547  -camera 0 "1,1,1" "-4.5,3.5,6" "0,0,0" \n17548  -end \n17549  -scene_objects \n17550  -Clebsch_surface \n17551  -group_of_things "0" \n17552  -cubics 0 $(SURFACE_COLOR) \n17553  -group_of_things_as_interval 0 6 \n17554  -group_of_things_as_interval 6 6 \n17555  -group_of_things_as_interval 12 15 \n17556  -lines 1 0.02 $(COLOR_RED_SHINY) \n17557  -lines 2 0.02 $(COLOR_BLUE_SHINY) \n17558  -lines 3 0.02 $(COLOR_YELLOW_SHINY) \n17559  -group_of_things_as_interval 0 12 \n17560  -spheres 4 0.08 $(COLOR_TURQUOISE) \n17561  -scene_objects_end \n17562  -povray_end
17563  -rm -rf POV
17564  mkdir POV
17565  mv Clebsch_0_*_.pov POV
17566  mv makefile_animation POV
17567
17568
17569
17570  endrass8:
17571  $(ORBITER) -v 2 -povray \n17572  -round 0 -nb_frames_default 30 \n886
Section 16.3: Creating Animations

dode:

$\$(ORBITER) -v 2 \n
```
    -povray \
17621  -round 0 -nb_frames_default 30 \
17622  -output_mask dode_\%d_\%03d.pov \
17623  -video_options -W 1024 -H 768 \
17624  -global_picture_scale 0.50 \
17625  -default_angle 45 \
17626  -clipping_radius 5 \
17627  -camera 0 "1,1,1" "-2,2,4" "0,0,0" \
17628  -rotate_about_111 \
17629  -end \n17630  -scene_objects \n17631  -dodecahedron \n17632  -group_of_things_as_interval 0 20 \n17633  -spheres 0 0.075 $(POLISHED_CHROME_WHITE) \n17634  -group_of_things_as_interval 0 30\n17635  -cylinders 1 0.05 $(COLOR_RED_SHINY) \n17636  -group_of_things_as_interval 0 12\n17637  -prisms 2 0.02 $(YELLOW_TRANSPARENT) \n17638  -scene_objects_end \n17639  -povray_end
17640  - rm -rf POV
17641  mkdir POV
17642  mv dode_0_*.pov POV
17643  mv makefile_animation POV
17644
17645
dode_video:
17647  - rm -r FRAMES
17648  - mkdir FRAMES
17649  - rm dode.mp4
17650  $(ORBITER) \n17651  -prepare_frames \n17652  -i 0 30 DODE/dode_\%d_\%03d.png \n17653  -output_starts_at 0 \n17654  -o FRAMES/frame\%04d.png \n17655  -end
17656  ffmpeg -r 5 -f image2 -i FRAMES/frame\%04d.png \n17657  -f mp4 -q:v 0 -vcodec mpeg4 dode.mp4
17658
17659
monkey.video:
17661  - rm -r FRAMES
17662  - mkdir FRAMES
17663  - rm monkey.mp4
17664  $(ORBITER) \n17665  -prepare_frames \n17666  -i 0 30 monkey_0_\%03d.png \
```
Eckardt_deform_video:
    - rm -r FRAMES
    - mkdir FRAMES
    - rm Eckardt_deform.mp4
    - $(ORBITER) -prepare_frames
        - i 0 93 Eckardt_deform_0/Eckardt_deform_0_0%03d.png
    - ffmpeg -r 5 -f image2 -i FRAMES/frame%04d.png
    - f mp4 -q:v 0 -vcodec mpeg4 monkey.mp4

Eckardt_surface:
    - $(ORBITER) -v 2 -povray
        - round 0 -nb_frames_default 30
        - output_mask Eckardt_0%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"
    - povray_end
    - scene_objects
        - cubic Goursat "6,3,-15"
        - group_of_things "0"
        - cubics 0 $(SURFACE_COLOR_SEETHROUGH)
    - scene_objects_end
    - rm -rf POV
    - mkdir POV
    - mv Eckardt_0_*.pov POV
    - mv makefile_animation POV

Kummer_surface:
    - $(ORBITER) -v 2 -povray
        - round 0 -nb_frames_default 30
        - output_mask Kummer_0%03d.pov
Beauville_surface:

```bash
17752 $(ORBITER) -v 2 -povray
17753 -round 0 -nb_frames_default 30
17754 -output_mask Beauville.%d.%03d.pov
17755 -video_options -W 1024 -H 768
17756 -global_picture_scale 0.3
17757 -default_angle 75
17758 -clipping_radius 2.4
17759 camera 0 "1,1,1" "-3,1,3" "0.12,0.12,0.12"
17760 -end
```

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.
# The Clebsch surface has
# a = d = 2.618033988 = (3+sqrt(5))/2
# b = c = 1.618033988 = (1+sqrt(5))/2
# to go from the arclifting surface to the defining equation:
#Matrix(4, 4, [[-0.44721360215312733, 1.1708204000530853, 1.1708204000530853, -0.4472135957999158], [-1.1708204000530853, 0.4472136021531272, 1.4472136021531272, 0.4472135957999158], [4.236068044124255, -4.236068044124255, -4.236068044124255, 4.236068044124255, -5, 0.], [1.6180340022062127, -2.6180340022062127, -1.6180340022062127, 1.6180340022062127, 0.]])
#-0.44721360215312733, 1.1708204000530853, 1.1708204000530853, -0.4472135957999158
#-1.1708204000530853, 0.4472136021531272, 1.4472136021531272, 0.4472135957999158
#[4.236068044124255, -4.236068044124255, -4.236068044124255, 4.236068044124255, 5, 0.], [1.6180340022062127, -2.6180340022062127, -1.6180340022062127, 1.6180340022062127, 0.])
CLEBSCH_CUBICS=

push b push b mult push d push c push m mult add mult 
push b push c push d push d push m mult mult add mult 
push a push d push d push m add mult mult add add 
push a push c push m mult mult add mult 
store c001 

push b push d mult 
push b push 1 push m push c mult add mult 
push d push a push 1 push m push m mult add mult add 
push m push a mult add push c add 
push c push m push a mult add 
mult mult 
store c002 
push b 

push d push c push a push m mult add mult \
push c push a push m push 1 mult add mult add mult 
push a push d mult push c push 1 push m mult add mult 
push m mult add 
push a push c push m mult add mult 
store c011 
push b push b push c mult mult 
push 1 push d push m mult add mult 
push a push b mult push c push d push m mult mult add mult 
push m mult add 
push a push d mult push c push d push m mult add mult add 
push a push c push m mult add mult 
store c012 
push m \npush b push d push m mult add mult 
store d001 
push m \npush d push c push m mult add push a push a mult mult 
push c push c mult push d push m mult add push a mult add 
push m push b push c mult mult push c push 1 push m mult add mult add mult 
push b push d push m mult add mult 
store c012 
push b push d push m mult add push a mult push m push b mult push 1 add push c mult add 
push b add push m push d mult add 
push a push c mult mult 
push b push d push m mult add mult 
store d112 
push m \npush b push d push m mult add push c mult push d push b push 1 push m mult add 
push m mult add push a mult push b push c mult push d push 1 push m mult add mult add 
push b push d push m mult add multi add mult mult 
store m002 
push m \
893
push d push c push m mult add push a push a mult mult \
push c push c mult push d push m mult add push a mult add \
push b push c push m mult mult push c push 1 push m mult add mult add \
push b push d push m mult add mult mult 
store m012 
push m\npush c push d mult push d push m mult add push a push a mult mult 
push m push c push c mult push d push m mult add push a push b mult mult 
mult add 
push m push b push d push c push m mult add push c mult mult mult add 
push b push d push m mult add mult mult 
store m022 
push d push 1 push m mult add push a mult 
push m push b mult push 1 add push c mult add 
push b add push m push d mult add 
push a push c mult mult 
push b push d push m mult add mult 
store n002 
push m 
push m push a mult push c add push d mult push c push a push 1 push m mult add mult add 
push b mult 
push m push a push d mult mult push c push 1 push m mult add mult add 
push b push d push m mult add mult 
store n002 
push m 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 mult 
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 
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 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 n112 
push m 
push m push d push m mult add push b mult 
push m push d push c push 1 push m mult add 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 n112 
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 add 
894
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:

$(ORBITER) -v 2 
-smooth_curve "Clebsch_map_of_circle_to_defining_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 
push y1 push y2 push y2 mult mult store y122 
push y1 push y2 push y2 mult mult store y122 
$(CLEBSCH_CUBICS) 
push c001 push y001 mult 
push c002 push y002 mult add 
push c011 push y011 mult add 
push c012 push y012 mult add 
store x0 
push d001 push y001 mult 
push d011 push y011 mult add 
push d012 push y012 mult add 
push d112 push y112 mult add 
store x1 

895
17984 push m002 push y002 mult \n17985 push m012 push y012 mult add \n17986 push m022 push y022 mult add \n17987 push m122 push y122 mult add \n17988 store x2 \n17989 push n002 push y002 mult \n17990 push n012 push y012 mult add \n17991 push n022 push y022 mult add \n17992 push n112 push y112 mult add \n17993 push n122 push y122 mult add \n17994 store x3 \n17995 push x0 push t00 mult \n17996 push x1 push t10 mult add \n17997 push x2 push t20 mult add \n17998 push x3 push t30 mult add \n17999 return \n18000 push x0 push t01 mult \n18001 push x1 push t11 mult add \n18002 push x2 push t21 mult add \n18003 push x3 push t31 mult add \n18004 return \n18005 push x0 push t02 mult \n18006 push x1 push t12 mult add \n18007 push x2 push t22 mult add \n18008 push x3 push t32 mult add \n18009 return \n18010 push x0 push t03 mult \n18011 push x1 push t13 mult add \n18012 push x2 push t23 mult add \n18013 push x3 push t33 mult add \n18014 return \n18015 -code_end
18016
18017
18018 Clebsch_surface:
18019 $(ORBITER) -v 2 -povray \n18020 -round 0 -nb_frames_default 30 \n18021 -output_mask Clebsch_%d_%03d.pov \n18022 -video_options -W 1024 -H 768 \n18023 -global_picture_scale 0.9 \n18024 -default_angle 75 \n18025 -clipping_radius 2.4 \n18026 -camera 0 "1,1,1" "-3,1,3" "0.12,0.12,0.12" \n18027 -end \n18028 -scene_objects \n18029 -cubic_orbiter "0,0,0,0,0,-4.236067972,\n18030 0,0,4.236067972,4.236067972,17.94427188,\n
896
Clebsch_surface_defining_equation:

```
18048  ▷▷▷ $(ORBITER) -v 2 -povray \
18049  ▷▷▷ -round 0 -nb_frames_default 30 \n18050  ▷▷▷ -output_mask Clebsch_%d%03d.pov \n18051  ▷▷▷ -video_options -W 1024 -H 768 \n18052  ▷▷▷ -global_picture_scale 0.6 \n18053  ▷▷▷ -default_angle 75 \n18054  ▷▷▷ -clipping_radius 1.6 \n18055  ▷▷▷ -camera 0 "1,1,1" ":-2,0,2" ":0,0,0" \n18056  ▷▷▷ -end \n18057  ▷▷▷ -scene_objects \n18058  ▷▷▷ ▷▷ -cubic_orbiter "0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,2,2,2,2,2,2" \n18059  ▷▷▷ ▷▷ -group_of_things "0" \n18060  ▷▷▷ ▷▷ -cubics 0 $(SURFACE_COLOR_SEETHROUGH) \n18061  ▷▷▷ ▷▷ -scene_objects_end \n18062  ▷▷▷ -povray_end 
18063  ▷▷ -rm -rf POV
18064  ▷▷ mkdir POV
18065  ▷▷ mv Clebsch_0*.pov POV
18066  ▷▷ mv makefile_animation POV
18067
18068
18069
18070 Clebsch_surface_defining_equation_and_curves:
18071  ▷ $(ORBITER) -v 2 -povray \
18072  ▷ -round 0 -nb_frames_default 30 \n18073  ▷ -output_mask Clebsch_2curves_%d%03d.pov \n18074  ▷ -video_options -W 1024 -H 768 \n18075  ▷ -global_picture_scale 0.6 \n18076  ▷ -default_angle 75 \n18077  ▷ -clipping_radius 1.6 
```
-camera 0 "1,1,1" ":-2,0,2" "0,0,0"
-scene_objects
-cubic_orbiter "0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,2,2,2,2,2" 
-group_of_things "0"
-cubics 0 $(SURFACE_COLOR_SEETHROUGH)
-point_list_from_csv_file
-function_Clebsch_map_of_circle_to_defininig_eqn_N1000_points.csv
-group_of_things_as_interval 0 656
-spheres 1 0.07 $(COLOR_RED)
-point_list_from_csv_file
-function_Clebsch_map_of_circle_to_defininig_eqn_r2_N1000_points.csv
-group_of_things_as_interval 656 1042
-spheres 2 0.07 "texture{ pigment{ color Blue } 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}}"

-ORBITER -v 2 -povray
-round 0 -nb_frames_default 30
-output_mask F7.15_lines_%d%03d.pov
-video_options -W 1024 -H 768
-global_picture_scale 1.5
-default_angle 80
-clipping_radius 4.4
-omit_bottom_plane
-camera 0 "1,1,1" ":-4.5,3.5,6" "0,0,0"
-scene_objects
-cubic_lex "0, 0, 6, 0, 0, -13.39014946, -3.341901346, -6.972931640, 5.827182718, 0, 0, 7.390149464, 7.390149464, 6.972931640, -1.512349728, -8.485281372, 0, 0, 0" 
-group_of_things "0"
-cubics 0 $(SURFACE_COLOR_SEETHROUGH) \
-line_through_point_with_direction "0, 0, 0, 1, 0, 0" \
-line_through_point_with_direction "0, 0, -1, 0, 1, 0" \
-line_through_point_with_direction "0, 0, 0, 0, 0, -1" \
-line_through_point_with_direction "1, 0, 0, 1, 1, 1" \
-line_through_point_with_direction "-1.414213562, 0, 0, 4.146264370, 1.732050808, 1.732050808" \
-line_through_point_with_direction "0, 1.732050808, -1, 2.414213562, -0.317837246, 2.414213562" \
-line_through_point_with_direction "-2.133352390, 0, -1, 1.674708020, 1, 0" \
-line_through_point_with_direction "-2.539058015, 0, 0, 2.211360755, 1, 0" \
-line_through_point_with_direction "0, 1.148188060, 0, 0, -0.9435440612, 1" \
-line_through_point_with_direction "-0.9711971171, 0, 0, 1.162155272, 0, 1" \
-line_through_point_with_direction "2.096037870, 2.096037870, 0, -1.065851905, -1.065851905, 1" \
-line_through_point_with_direction "3.921555783, 2.921555781, 0, -1.722456585, -1.722456585, 1" \
-group_of_things_as_interval 0 12 \n-scene_objects_end \n-povray_end \n-rm -rf POV \n-mkdir POV \n-mv F7_15_lines_0_*.pov POV \n-mv makefile_animation POV

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 \n-ffmpeg -r 5 -f image2 -i FRAMES/frame%04d.png \n-mp4 -q:v 0 -vcodec mpeg4 fifteen_with_lines.mp4
SECTION_CONTINUOUS_FUNCTION_PLOTTER:
lissajous:

```python
⊿ $(ORBITER) -v 2 
⊿ ⊿ -smooth_curve "lissajous" 0.07 2000 15 0 18.85 
⊿ ⊿ -const a 3 b 2 c 1.57 r 7 -const_end 
⊿ ⊿ -var t -var_end 
⊿ ⊿ -code 
⊿ ⊿ ⊿ push t push a mult push c add sin push r mult return 
⊿ ⊿ ⊿ push t push b mult sin push r mult return 
⊿ ⊿ -code_end 
⊿ ⊿ -code
```

`#function_lissajous_N2000_points.csv`

lissajous_plot:

```python
⊿ $(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 
```
lissajous_3d:

```bash
$(ORBITER) -v 2 -smooth_curve "lissajous_3d" 0.07 2000 50 0 18.85 -const a 3 b 2 c 1.57 r 7 -const_end -var t -var_end -code
```

```c
push t push a mult push c add sin push r mult return
push t push b mult sin push r mult return
push t return
-code_end
```

lissajous_3d_plot:

```bash
$(ORBITER) -v 2 -povray -round 0 -nb_frames_default 30 -output_mask lissajous_3d_%d_043d.pov -video_options -W 1024 -H 768 -global_picture_scale 0.40 -default_angle 45 -clipping_radius 5 -omit_bottom_plane -camera 0 "0,0,1" "7,7,5" "0,0,1" -rotate_about_z_axis -end -scene_objects -line_through_two_points_recentered_from_csv_file -coordinate_grid.csv -group_of_things "0" -group_of_things "1" -group_of_things "2" -lines 0 0.09 "texture{ pigment{ color Yellow } }" -lines 1 0.09 "texture{ pigment{ color Yellow } }" -lines 2 0.09 "texture{ pigment{ color Yellow } }" -group_of_things_as_interval 3 39 -lines 3 0.02 "texture{ pigment{ color Black } }" -point_list_from_csv_file -function_lissajous_3d_N20000.points.csv -group_of_things_as_interval 0 6538 -spheres 4 0.1 "texture{ pigment{ color Red } }" finish { diffuse 0.9 phong 1} -plane_by_dual_coordinates "0,0,1,0" -group_of_things "0"
```
18301 ▷ ▷ -scene_objects_end \
18302 ▷ ▷ -povray_end
18303 ▷ - rm -rf POV
18304 ▷ mkdir POV
18305 ▷ mv lissajous_3d_0_*.pov POV
18306 ▷ mv makefile_animation POV
18307
18308 # ▷ ▷ -planes 5 "texture{ pigment{ color Blue*0.5 transmit 0.5 } }" \
18309
18310
18311
18312
18313 # Chapter 17 - Miscellaneous
18314 # Section 17.1: Miscellaneous
18320
18321 SECTION_MISC:ELLANEOUS:
18322
18323
18324
18325
18326 misc_select:
18327 ▷ $(ORBITER) -v 3 \
18328 ▷ ▷ -define F -finite_field -q 7 -end \
18329 ▷ ▷ -with F -do -finite_field_activity -cheat_sheet_GF -end
18330 ▷ $(ORBITER) -v 4 -csv_file_select_rows_and_cols \
18331 ▷ ▷ GF_q7_multiplication_table_reordered.csv \
18332 ▷ ▷ "0,2,4" "0,2,4"
18333
18334
18335 misc_join:
18336 ▷ $(ORBITER) -v 4 \
18337 ▷ ▷ -csv_file_join poly_orbits_d3_n3_q2_select_F2.csv Orbit_idx \
18338 ▷ ▷ -csv_file_join poly_orbits_d3_n3_q2_select_F4.csv Orbit_idx \
18339 ▷ ▷ -csv_file_join poly_orbits_d3_n3_q2_select_F8.csv Orbit_idx \
18340 ▷ ▷ -csv_file_join poly_orbits_d3_n3_q2_select_F16.csv Orbit_idx \
18341 ▷ ▷ -csv_file_join poly_orbits_d3_n3_q2_select_F32.csv Orbit_idx
18342
18343
18344 table_mod_12:
18345 ▷ $(ORBITER) -v 2 \
18346 ▷ ▷ -define M -vector -load_csv_no_border clock_mult_excel.csv -end \
18347 ▷ ▷ -define all_one_r -vector -repeat 1 12 -end \

903
-define all_one_c -vector -repeat 1 12 -end \ 
-draw_matrix \ 
-input_object M \ 
-box_width 50 -bit_depth 24 \ 
-partition 3 \ 
-all_one_r all_one_c \ 
-end

draw
eigenvalue
diag23:

extract:

draw.eigenvalue.diag23:

Section 17.2: Limitations

SECTION LIMITATIONS:

extract:

draw.eigenvalue.diag23:
Bibliography


[13] Dieter Betten. 4-dimensionale Translationsebenen mit 8-dimensionaler Kollineations-


[19] Henri Cohen. *A course in computational algebraic number theory*, volume 138 of *Grad-

Warwick, 2014.


[22] Terry Czerwinski and David Oakden. The translation planes of order twenty-five. *J.

[23] Anne Delandtsheer and Jean Doyen. Most block-transitive $t$-designs are point-primitive.

(1915), 139-157.


manual/make.html.


[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.


