Functions Manual

Prepared for Dr. Kharaghani and his students.

Written by: Thomas Pender

Summary

1 Configuration 1
- 1.1 Main Set-Up 1
- 1.2 Required Packages 2

2 Bgw Constructions 2
- 2.1 ω-Circulant Matrices 2
- 2.2 Generalized Conference Matrices 4
- 2.3 Generalized Hadamard Matrices 5
- 2.4 Bgw-Oa Construction 6

3 Generalized Simplex Codes 7

4 Operations on Symmetric Block Designs 8
- 4.1 Number of Collineations 8
- 4.2 Binary Rank 9

5 Sagemath's Interface with Cliquer 9
- 5.1 The Clique Polynomial 10
- 5.2 Maximum Cliques 10

6 Cyclic Matrix Group Generator 11
- 6.1 Circulant and Negacyclic Generators 12
- 6.2 Block Circulant and Negacyclic Generators 12

7 Note on the Formatting of Matrix Files 13

1 Configuration

1.1 Main Set-Up

You should have already dowloaded the archive file `hadi_funcs.tar.gz`. Navigate to your downloads folder using `cd ~/Downloads/`, where your copy of `hadi_funcs.tar.gz` should now be located. Then move the archive file to wherever you want to unpack it. For example, if you have a folder called `myfuncs/` in your home directory, then move the file viz. `mv ~/Downloads/hadi_funcs.tar.gz ~/myfuncs/`.

Once you have the archive file placed where you want it, unpack it with the command `tar -zxf hadi_funcs.tar.gz`. This will unpack the folder `sagefuncs/`, where all of the functions are located. Navigate into this new directory with `cd sagefuncs/`.

Once inside of `sagefuncs/`, you need to make the required files executable and construct their soft links into your systems `PATH`. A file has been prepared for you to do this. Execute this file with `sudo bash mklinks.bash`. You will be required to enter the root user's password in order to complete this command.

After you have run the above bash script, you should be ready to go if the requirements of the next subsection are met. Additionally, if you ever need to remove the links, then run `sudo bash rmlinks.bash`.

1.2 Required Packages

In order for these functions to work, it is assumed that you have sagemath installed together with the GAP packages DESIGNS and GRAPE.

To install sagemath, run the command `sudo apt install sagemath` (or `yum` if your system requires). The GRAPE package can be found here, and the DESIGN package can be found here.

Once you have downloaded the packages, navigate again to your downloads folder. Move the packages to the directory `/usr/lib/gap/pkg`. Navigate to this directory and unpack each file using `tar -zxf` *filename*`.tar.gz`.

The GRAPE package requires a little more work to get up and running. Move into the `grape/` folder you have just unpacked, and run `./configure`. This will create a make file which you will run with `make`. Once finished this step, you should be done. In case the install instructions have changed, you can double check the first chapter of the GRAPE manual found here.

2 Bgw Constructions

The `bgw` command offers several constructions which are covered in the next few subsections.

The command requires one or two parameters predicated upon which options have been invoked. If one of `acH` are invoked, then two parameters are necessary. The first is always a prime power q, and the second is always a positive integer d. If `C` is used, then only one parameter, namely, q, is needed.

The output of the commands are always written to some output file. To store the matrix in a file named `mat.txt`, you must use the option `o` which takes `mat.txt` as a parameter.

2.1 ω-Circulant Matrices

Using the `c` option, the classical parameter, ω-circulant BGWs over $GF(q)^*$ are constructed. The matrices stored in the output file are integral matrices where 0 corresponds with the zero element of the field, and where the non-zero integers are the logarithms of the primitive element ω.

For example, the command `bgw 3 3 -co mat.txt` stores the array

$$
\begin{array}{ccccccccccccc}
0 & 0 & 1 & 0 & 1 & 2 & 1 & 1 & 2 & 0 & 1 & 1 & 1 \\
2 & 0 & 0 & 1 & 0 & 1 & 2 & 1 & 1 & 2 & 0 & 1 & 1 \\
2 & 2 & 0 & 0 & 1 & 0 & 1 & 2 & 1 & 1 & 2 & 0 & 1 \\
2 & 2 & 2 & 0 & 0 & 1 & 0 & 1 & 2 & 1 & 1 & 2 & 0 \\
0 & 2 & 2 & 2 & 0 & 0 & 1 & 0 & 1 & 2 & 1 & 1 & 2 \\
1 & 0 & 2 & 2 & 2 & 0 & 0 & 1 & 0 & 1 & 2 & 1 & 1 \\
2 & 1 & 0 & 2 & 2 & 2 & 0 & 0 & 1 & 0 & 1 & 2 & 1 \\
2 & 2 & 1 & 0 & 2 & 2 & 2 & 0 & 0 & 1 & 0 & 1 & 2 \\
1 & 2 & 2 & 1 & 0 & 2 & 2 & 2 & 0 & 0 & 1 & 0 & 1 \\
2 & 1 & 2 & 2 & 1 & 0 & 2 & 2 & 2 & 0 & 0 & 1 & 0 \\
0 & 2 & 1 & 2 & 2 & 1 & 0 & 2 & 2 & 2 & 0 & 0 & 1 \\
2 & 0 & 2 & 1 & 2 & 2 & 1 & 0 & 2 & 2 & 2 & 0 & 0 \\
0 & 2 & 0 & 2 & 1 & 2 & 2 & 1 & 0 & 2 & 2 & 2 & 0
\end{array}
$$

in the file `mat.txt`.

More generally, the command `bgw q d -co mat.txt` will construct an ω-circulant $\mathrm{BGW}(q^{q-1}+q^{d-2}+\cdots+1, q^{d-1}, q^{d-1}-q^{d-2}; GF(q)^*)$ and store the logarithm matrix in the file `mat.txt`.

If q is odd, the matrix can be transformed into a negacyclic balanced weighing matrix by using the `w` option. Then `bgw 3 3 -cwo mat.txt` stores the array

$$
\begin{array}{ccccccccccccc}
0 & 0 & -1 & 0 & -1 & 1 & -1 & -1 & 1 & 0 & -1 & -1 & -1 \\
1 & 0 & 0 & -1 & 0 & -1 & 1 & -1 & -1 & 1 & 0 & -1 & -1 \\
1 & 1 & 0 & 0 & -1 & 0 & -1 & 1 & -1 & -1 & 1 & 0 & -1 \\
1 & 1 & 1 & 0 & 0 & -1 & 0 & -1 & 1 & -1 & -1 & 1 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 & -1 & 0 & -1 & 1 & -1 & -1 & 1 \\
-1 & 0 & 1 & 1 & 1 & 0 & 0 & -1 & 0 & -1 & 1 & -1 & -1 \\
1 & -1 & 0 & 1 & 1 & 1 & 0 & 0 & -1 & 0 & -1 & 1 & -1 \\
1 & 1 & -1 & 0 & 1 & 1 & 1 & 0 & 0 & -1 & 0 & -1 & 1 \\
-1 & 1 & 1 & -1 & 0 & 1 & 1 & 1 & 0 & 0 & -1 & 0 & -1 \\
1 & -1 & 1 & 1 & -1 & 0 & 1 & 1 & 1 & 0 & 0 & -1 & 0 \\
0 & 1 & -1 & 1 & 1 & -1 & 0 & 1 & 1 & 1 & 0 & 0 & -1 \\
1 & 0 & 1 & -1 & 1 & 1 & -1 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & -1 & 1 & 1 & -1 & 0 & 1 & 1 & 1 & 0
\end{array}
$$

in the given file.

The preceding command essentially does the substitution $\omega^k \mapsto (-1)^k$ for the non-zero entries. More generally, given a cyclic matrix group whose order divides $q-1$, we can substitute this group into the BGW with the `g` option together with an input file containing the matrix group generator. If the generator is stored in `gen.txt`, then use `-i gen.txt` to make the file available for the construction. Additionally, the option `g` requires the order of the group as a parameter.

If $\left[\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right]$ is stored in `gen.txt`, then `bgw 3 3 -cg 2 -o mat.txt -i gen.txt` constructs

$$
\begin{array}{cccccccccccccccccccccccccc}
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0
\end{array}.
$$

2.2 Generalized Conference Matrices

The `C` option is used to construct the skew-symmetric conference matrices with type II core. The use is much the same as before, save the exponent parameter is not required.

As an example, the command `bgw 11 -Co mat.txt` constructs the BGW$(12, 11, 10; GF(11)^*)$ given by

$$
\begin{array}{cccccccccccc}
0 & 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 \\
5 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
5 & 6 & 0 & 1 & 9 & 8 & 3 & 7 & 4 & 10 & 2 & 5 \\
5 & 7 & 6 & 0 & 2 & 10 & 9 & 4 & 8 & 5 & 1 & 3 \\
5 & 8 & 4 & 7 & 0 & 3 & 1 & 10 & 5 & 9 & 6 & 2 \\
5 & 9 & 3 & 5 & 8 & 0 & 4 & 2 & 1 & 6 & 10 & 7 \\
5 & 10 & 8 & 4 & 6 & 9 & 0 & 5 & 3 & 2 & 7 & 1 \\
5 & 1 & 2 & 9 & 5 & 7 & 10 & 0 & 6 & 4 & 3 & 8 \\
5 & 2 & 9 & 3 & 10 & 6 & 8 & 1 & 0 & 7 & 5 & 4 \\
5 & 3 & 5 & 10 & 4 & 1 & 7 & 9 & 2 & 0 & 8 & 6 \\
5 & 4 & 7 & 6 & 1 & 5 & 2 & 8 & 10 & 3 & 0 & 9 \\
5 & 5 & 10 & 8 & 7 & 2 & 6 & 3 & 9 & 1 & 4 & 0
\end{array}.
$$

This can be transformed into a weighing matrix viz. `bgw 11 -Cwo mat.txt` as

$$
\begin{matrix}
0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
-1 & 0 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\
-1 & 1 & 0 & -1 & -1 & 1 & -1 & -1 & 1 & 1 & 1 & -1 \\
-1 & -1 & 1 & 0 & 1 & 1 & -1 & 1 & 1 & -1 & -1 & -1 \\
-1 & 1 & 1 & -1 & 0 & -1 & -1 & 1 & -1 & -1 & 1 & 1 \\
-1 & -1 & -1 & -1 & 1 & 0 & 1 & 1 & -1 & 1 & 1 & -1 \\
-1 & 1 & 1 & 1 & 1 & -1 & 0 & -1 & -1 & 1 & -1 & -1 \\
-1 & -1 & 1 & -1 & -1 & -1 & 1 & 0 & 1 & 1 & -1 & 1 \\
-1 & 1 & -1 & -1 & 1 & 1 & 1 & -1 & 0 & -1 & -1 & 1 \\
-1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 & 0 & 1 & 1 \\
-1 & 1 & -1 & 1 & -1 & -1 & 1 & 1 & 1 & -1 & 0 & -1 \\
-1 & -1 & 1 & 1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 & 0
\end{matrix}
.
$$

As before, we can substitute a matrix group. Using the same `gen.txt` as before, `bgw 11 -Cg 2 -o mat.txt -i gen.txt` yields

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

2.3 Generalized Hadamard Matrices

We can construct classical $\mathrm{GH}(q, q^{d-1})$s over $\mathrm{EA}(q)$ using the option `H`. We again require both a prime power and an exponent. Using `bgw q d -Ho mat.txt` constructs the $\mathrm{GH}(q, q^{d-1})$. Note that the elements of $\mathrm{EA}(q)$ are given in lexocographic order.

To be explicit, `bgw 3 2 -Ho mat.txt` constructs a $\mathrm{GH}(3, 3)$ over $\mathrm{EA}(3)$

given by

$$
\begin{array}{ccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 \\
0 & 2 & 1 & 0 & 2 & 1 & 0 & 2 & 1 \\
0 & 0 & 0 & 1 & 1 & 1 & 2 & 2 & 2 \\
0 & 1 & 2 & 1 & 2 & 0 & 2 & 0 & 1 \\
0 & 2 & 1 & 1 & 0 & 2 & 2 & 1 & 0 \\
0 & 0 & 0 & 2 & 2 & 2 & 1 & 1 & 1 \\
0 & 1 & 2 & 2 & 0 & 1 & 1 & 2 & 0 \\
0 & 2 & 1 & 2 & 1 & 0 & 1 & 0 & 2
\end{array}
$$

For ease of use in constructing affine or projective geometries, we can give a representation of EA(3) using permutation matrices viz. `bgw 3 2 -Hpo mat.txt` given by

$$
\begin{array}{ccccccccccccccccccccccccccc}
1&0&0&1&0&0&1&0&0&1&0&0&1&0&0&1&0&0&1&0&0&1&0&0&1&0&0 \\
0&1&0&0&1&0&0&1&0&0&1&0&0&1&0&0&1&0&0&1&0&0&1&0&0&1&0 \\
0&0&1&0&0&1&0&0&1&0&0&1&0&0&1&0&0&1&0&0&1&0&0&1&0&0&1 \\
1&0&0&0&1&0&0&0&1&1&0&0&0&1&0&0&0&1&1&0&0&0&1&0&0&0&1 \\
0&1&0&0&0&1&1&0&0&0&1&0&0&0&1&1&0&0&0&1&0&0&0&1&1&0&0 \\
0&0&1&1&0&0&0&1&0&0&0&1&1&0&0&0&1&0&0&0&1&1&0&0&0&1&0 \\
1&0&0&0&0&1&0&1&0&1&0&0&0&0&1&0&1&0&1&0&0&0&0&1&0&1&0 \\
0&1&0&1&0&0&0&0&1&0&1&0&1&0&0&0&0&1&0&1&0&1&0&0&0&0&1 \\
0&0&1&0&1&0&1&0&0&0&0&1&0&1&0&1&0&0&0&0&1&0&1&0&1&0&0 \\
1&0&0&1&0&0&1&0&0&0&1&0&0&1&0&0&1&0&0&0&1&0&0&1&0&0&1 \\
0&1&0&0&1&0&0&1&0&0&0&1&0&0&1&0&0&1&1&0&0&1&0&0&1&0&0 \\
0&0&1&0&0&1&0&0&1&1&0&0&1&0&0&1&0&0&0&1&0&0&1&0&0&1&0 \\
1&0&0&0&1&0&0&0&1&0&1&0&0&0&1&1&0&0&0&0&1&1&0&0&0&1&0 \\
0&1&0&0&0&1&1&0&0&0&0&1&1&0&0&0&1&0&1&0&0&0&1&0&0&0&1 \\
0&0&1&1&0&0&0&1&0&1&0&0&0&1&0&0&0&1&0&1&0&0&0&1&1&0&0 \\
1&0&0&0&0&1&0&1&0&0&1&0&1&0&0&0&0&1&0&0&1&0&1&0&1&0&0 \\
0&1&0&1&0&0&0&0&1&0&0&1&0&1&0&1&0&0&1&0&0&0&0&1&0&1&0 \\
0&0&1&0&1&0&1&0&0&1&0&0&0&0&1&0&1&0&0&1&0&1&0&0&0&0&1 \\
1&0&0&1&0&0&1&0&0&0&0&1&0&0&1&0&0&1&0&1&0&0&1&0&0&1&0 \\
0&1&0&0&1&0&0&1&0&1&0&0&1&0&0&1&0&0&0&0&1&0&0&1&0&0&1 \\
0&0&1&0&0&1&0&0&1&0&1&0&0&1&0&0&1&0&1&0&0&1&0&0&1&0&0 \\
1&0&0&0&1&0&0&0&1&0&0&1&1&0&0&0&1&0&0&1&0&0&0&1&1&0&0 \\
0&1&0&0&0&1&1&0&0&1&0&0&0&1&0&0&0&1&0&0&1&1&0&0&0&1&0 \\
0&0&1&1&0&0&0&1&0&0&1&0&0&0&1&1&0&0&1&0&0&0&1&0&0&0&1 \\
1&0&0&0&0&1&0&1&0&0&0&1&0&1&0&1&0&0&0&1&0&1&0&0&0&0&1 \\
0&1&0&1&0&0&0&0&1&1&0&0&0&0&1&0&1&0&0&0&1&0&1&0&1&0&0 \\
0&0&1&0&1&0&1&0&0&0&1&0&1&0&0&0&0&1&1&0&0&0&0&1&0&1&0
\end{array}
$$

2.4 Bgw-Oa Construction

Hadi Kharaghani's construction of weighing matrices using BGWs and OAs is available as well. This is employed with the `a` option. A weighing matrix with odd prime power weight needs to be supplied as well. If this matrix has weight q and is supplied in `wmat.txt`, then `bgw q d -ai wmat.txt -o mat.txt` does the $(d-1)$-st iteration of Hadi's construction.

For instance, `bgw 3 -Cwo wmat.txt` stores the $W(4,3)$ given by

$$\begin{array}{rrrr}
0 & 1 & 1 & 1 \\
-1 & 0 & -1 & 1 \\
-1 & 1 & 0 & -1 \\
-1 & -1 & 1 & 0
\end{array}$$

in the file `wmat.txt`. Then `bgw 3 3 -ai wmat.txt -o mat.txt` gives the W$(40, 27)$ given by

$$\begin{array}{*{40}{c}}
0&0&0&0&0&0&0&-&-&-&0&0&0&-&-&-&1&1&1&-&-&-&-&-&-&1&1&1&0&0&0&-&-&-&-&-&-&-&-&- \\
0&1&1&1&0&0&0&0&0&0&-&-&-&0&0&0&-&-&-&1&1&1&-&-&-&-&-&-&1&1&1&0&0&0&-&-&-&-&-&- \\
0&1&1&1&1&1&1&0&0&0&0&0&0&-&-&-&0&0&0&-&-&-&1&1&1&-&-&-&-&-&-&1&1&1&0&0&0&-&-&- \\
0&1&1&1&1&1&1&1&1&1&0&0&0&0&0&0&-&-&-&0&0&0&-&-&-&1&1&1&-&-&-&-&-&-&1&1&1&0&0&0 \\
0&0&0&0&1&1&1&1&1&1&1&1&1&0&0&0&0&0&0&-&-&-&0&0&0&-&-&-&1&1&1&-&-&-&-&-&-&1&1&1 \\
0&-&-&-&0&0&0&1&1&1&1&1&1&1&1&1&0&0&0&0&0&0&-&-&-&0&0&0&-&-&-&1&1&1&-&-&-&-&-&- \\
0&1&1&1&-&-&-&0&0&0&1&1&1&1&1&1&1&1&1&0&0&0&0&0&0&-&-&-&0&0&0&-&-&-&1&1&1&-&-&- \\
0&1&1&1&1&1&1&-&-&-&0&0&0&1&1&1&1&1&1&1&1&1&0&0&0&0&0&0&-&-&-&0&0&0&-&-&-&1&1&1 \\
0&-&-&-&1&1&1&1&1&1&-&-&-&0&0&0&1&1&1&1&1&1&1&1&1&0&0&0&0&0&0&-&-&-&0&0&0&-&-&- \\
0&1&1&1&-&-&-&1&1&1&1&1&1&-&-&-&0&0&0&1&1&1&1&1&1&1&1&1&0&0&0&0&0&0&-&-&-&0&0&0 \\
0&0&0&0&1&1&1&-&-&-&1&1&1&1&1&1&-&-&-&0&0&0&1&1&1&1&1&1&1&1&1&0&0&0&0&0&0&-&-&- \\
0&1&1&1&0&0&0&1&1&1&-&-&-&1&1&1&1&1&1&-&-&-&0&0&0&1&1&1&1&1&1&1&1&1&0&0&0&0&0&0 \\
0&0&0&0&1&1&1&0&0&0&1&1&1&-&-&-&1&1&1&1&1&1&-&-&-&0&0&0&1&1&1&1&1&1&1&1&1&0&0&0 \\
1&-&0&1&-&0&1&1&-&0&0&1&-&-&0&1&1&-&0&0&1&-&1&-&0&0&1&-&-&0&1&0&1&-&-&0&1&1&-&0 \\
1&1&-&0&-&0&1&0&1&-&1&-&0&-&0&1&0&1&-&1&-&0&1&-&0&-&0&1&0&1&-&0&1&-&1&-&0&-&0&1 \\
1&0&1&-&-&0&1&-&0&1&-&0&1&-&0&1&-&0&1&-&0&1&1&-&0&1&-&0&1&-&0&0&1&-&0&1&-&0&1&- \\
1&-&0&1&1&-&0&0&1&-&-&0&1&-&0&1&1&-&0&0&1&-&0&1&-&-&0&1&1&-&0&1&-&0&0&1&-&-&0&1 \\
1&1&-&0&1&-&0&-&0&1&0&1&-&-&0&1&0&1&-&1&-&0&0&1&-&1&-&0&-&0&1&1&-&0&-&0&1&0&1&- \\
1&0&1&-&1&-&0&1&-&0&1&-&0&-&0&1&-&0&1&-&0&1&0&1&-&0&1&-&0&1&-&1&-&0&1&-&0&1&-&0 \\
1&-&0&1&0&1&-&-&0&1&1&-&0&-&0&1&1&-&0&0&1&-&-&0&1&1&-&0&0&1&-&-&0&1&1&-&0&0&1&- \\
1&1&-&0&0&1&-&1&-&0&-&0&1&-&0&1&0&1&-&1&-&0&-&0&1&0&1&-&1&-&0&-&0&1&0&1&-&1&-&0 \\
1&0&1&-&0&1&-&0&1&-&0&1&-&-&0&1&-&0&1&-&0&1&-&0&1&-&0&1&-&0&1&-&0&1&-&0&1&-&0&1 \\
1&-&0&1&-&0&1&1&-&0&0&1&-&1&-&0&0&1&-&-&0&1&0&1&-&-&0&1&1&-&0&-&0&1&1&-&0&0&1&- \\
1&1&-&0&-&0&1&0&1&-&1&-&0&1&-&0&-&0&1&0&1&-&0&1&-&1&-&0&-&0&1&-&0&1&0&1&-&1&-&0 \\
1&0&1&-&-&0&1&-&0&1&-&0&1&1&-&0&1&-&0&1&-&0&0&1&-&0&1&-&0&1&-&-&0&1&-&0&1&-&0&1 \\
1&-&0&1&1&-&0&0&1&-&-&0&1&1&-&0&0&1&-&-&0&1&-&0&1&1&-&0&0&1&-&0&1&-&-&0&1&1&-&0 \\
1&1&-&0&1&-&0&-&0&1&0&1&-&1&-&0&-&0&1&0&1&-&-&0&1&0&1&-&1&-&0&0&1&-&1&-&0&-&0&1 \\
1&0&1&-&1&-&0&1&-&0&1&-&0&1&-&0&1&-&0&1&-&0&-&0&1&-&0&1&-&0&1&0&1&-&0&1&-&0&1&- \\
1&-&0&1&0&1&-&-&0&1&1&-&0&1&-&0&0&1&-&-&0&1&1&-&0&0&1&-&-&0&1&1&-&0&0&1&-&-&0&1 \\
1&1&-&0&0&1&-&1&-&0&-&0&1&1&-&0&-&0&1&0&1&-&1&-&0&-&0&1&0&1&-&1&-&0&-&0&1&0&1&- \\
1&0&1&-&0&1&-&0&1&-&0&1&-&1&-&0&1&-&0&1&-&0&1&-&0&1&-&0&1&-&0&1&-&0&1&-&0&1&-&0 \\
1&-&0&1&-&0&1&1&-&0&0&1&-&0&1&-&-&0&1&1&-&0&-&0&1&1&-&0&0&1&-&1&-&0&0&1&-&-&0&1 \\
1&1&-&0&-&0&1&0&1&-&1&-&0&0&1&-&1&-&0&-&0&1&-&0&1&0&1&-&1&-&0&1&-&0&-&0&1&0&1&- \\
1&0&1&-&-&0&1&-&0&1&-&0&1&0&1&-&0&1&-&0&1&-&-&0&1&-&0&1&-&0&1&1&-&0&1&-&0&1&-&0 \\
1&-&0&1&1&-&0&0&1&-&-&0&1&0&1&-&-&0&1&1&-&0&1&-&0&0&1&-&-&0&1&-&0&1&1&-&0&0&1&- \\
1&1&-&0&1&-&0&-&0&1&0&1&-&0&1&-&1&-&0&-&0&1&1&-&0&-&0&1&0&1&-&-&0&1&0&1&-&1&-&0 \\
1&0&1&-&1&-&0&1&-&0&1&-&0&0&1&-&0&1&-&0&1&-&1&-&0&1&-&0&1&-&0&-&0&1&-&0&1&-&0&1 \\
1&-&0&1&0&1&-&-&0&1&1&-&0&0&1&-&-&0&1&1&-&0&0&1&-&-&0&1&1&-&0&0&1&-&-&0&1&1&-&0 \\
1&1&-&0&0&1&-&1&-&0&-&0&1&0&1&-&1&-&0&-&0&1&0&1&-&1&-&0&-&0&1&0&1&-&1&-&0&-&0&1 \\
1&0&1&-&0&1&-&0&1&-&0&1&-&0&1&-&0&1&-&0&1&-&0&1&-&0&1&-&0&1&-&0&1&-&0&1&-&0&1&-
\end{array},$$

where -1 has been replaced by $-$.

3 Generalized Simplex Codes

Recall that the generalized simplex codes are equidistant linear $\left(\frac{q^d-1}{q-1}, d, \frac{q^{d-1}-1}{q-1}\right)_q$-codes. These codes are constructed by the `simplex` command. This command requires two parameters. First, a prime power, and second an exponent. Additionally, it requires an output file as a parameter for the `o` option. These can be invoked by `simplex q d -o code.txt`.

As an example, `simplex 3 3 -o code.txt` constructs the 27×13 array given by

1	1	2	0	1	2	0	2	0	1	0	1	2
2	1	0	2	1	0	2	2	1	0	0	2	1
0	1	1	1	1	1	1	2	2	2	0	0	0
1	2	0	1	1	2	0	0	1	2	2	0	1
2	2	1	0	1	0	2	0	2	1	2	1	0
0	2	2	2	1	1	1	0	0	0	2	2	2
1	0	1	2	1	2	0	1	2	0	1	2	0
2	0	2	1	1	0	2	1	0	2	1	0	2
0	0	0	0	1	1	1	1	1	1	1	1	1
1	1	2	0	2	0	1	0	1	2	1	2	0
2	1	0	2	2	1	0	0	2	1	1	0	2
0	1	1	1	2	2	2	0	0	0	1	1	1
1	2	0	1	2	0	1	1	2	0	0	1	2
2	2	1	0	2	1	0	1	0	2	0	2	1
0	2	2	2	2	2	2	1	1	1	0	0	0
1	0	1	2	2	0	1	2	0	1	2	0	1
2	0	2	1	2	1	0	2	1	0	2	1	0
0	0	0	0	2	2	2	2	2	2	2	2	2
1	1	2	0	0	1	2	1	2	0	2	0	1
2	1	0	2	0	2	1	1	0	2	2	1	0
0	1	1	1	0	0	0	1	1	1	2	2	2
1	2	0	1	0	1	2	2	0	1	1	2	0
2	2	1	0	0	2	1	2	1	0	1	0	2
0	2	2	2	0	0	0	2	2	2	1	1	1
1	0	1	2	0	1	2	0	1	2	0	1	2
2	0	2	1	0	2	1	0	2	1	0	2	1

and stores it in the file `code.txt`.

The generalized simplex codes are used in Hadi's BGW-OA construction.

4 Operations on Symmetric Block Designs

The `design` command offers two operations on symmetric block designs. These are finding the number of collineations and calculating the binary rank of the incidence matrix. The command expects one parameter, namely, the file containing the incidence matrix of the design, and one option from `abh`.

4.1 Number of Collineations

To be concrete, we assume the following projective plane of order 2

0	1	1	0	1	0	0
0	0	1	1	0	1	0
0	0	0	1	1	0	1
1	0	0	0	1	1	0
0	1	0	0	0	1	1
1	0	1	0	0	0	1
1	1	0	1	0	0	0

is stored in the file `plane.txt`.

To find the number of collineations, we use the `a` option viz. `design -a plane.txt`. This outputs `Number of collineations: 168`. It should be noted that calculating the number of collineations of a symmetric design is an arduous calculation, hence this can only feasibly be used for small parameter designs. For large designs, the calculation should be passed to a place with larger resources, like Compute Canada.

This proceedure uses calls to both sagemath and GAP. It might be benificial for one to use only GAP resources; however, this has not been done at this point.

4.2 Binary Rank

Continuing with the above example, we can calculate the binary rank quite simply with the command `design -b plane.txt`. The output is `Binary rank: 4`.

Both this and the previous option for the `design` command, along with all of the commands herein, have the `STDOUT` redirected to `dev/null`. The outputs given by these commands are output to the terminal viz. `STDERR`. They can be redirected to a file with `design [-a | -b] infile 2> outfile`.

5 Sagemath's Interface with Cliquer

Sagemath provides and interface with cliquer, a clique finding routine that offers several functionalities. We have given two in this collection of functions that would seem to be most pertinent. One can use the `cliq` command with one of the required options `pm` to display the clique polynomial and to find and display the maximum cliques of the graph.

5.1 The Clique Polynomial

We will use the following example of an $SRG(40, 12, 2, 4)$

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

stored in the file `srg.txt` Using the command `cliq -p srg.txt`, we obtain the output `40*t^4 + 160*t^3 + 240*t^2 + 40*t + 1`, which is the clique polynomial of the graph.

5.2 Maximum Cliques

Using the same graph of the previous subsection, we can also find and store all of its maximum cliques. From the clique polynomial given above, we expect to find 40 cliques of 4 vertices. Using the command `cliq -m srg.txt -o cliques.txt`, these 40 4-cliques are found and stored in the file `cliques.txt`.

For this example, they are as follows.

0	1	2	3
0	7	19	39
0	11	23	31
0	15	27	35
1	6	22	34
1	10	26	38
1	14	18	30
2	5	20	33
2	9	24	37
2	13	16	29
3	4	17	36
3	8	21	28
3	12	25	32
4	5	6	7
4	9	30	35
4	13	23	26
5	8	18	27
5	12	31	38
6	11	16	25
6	15	28	37
7	10	29	32
7	14	21	24
8	9	10	11
8	13	34	39
9	12	19	22
10	15	17	20
11	14	33	36
12	13	14	15
16	17	18	19
16	21	35	38
17	24	31	34
18	23	32	37
19	26	28	33
20	21	22	23
20	25	30	39
22	27	29	36
24	25	26	27
28	29	30	31
32	33	34	35
36	37	38	39

6 Cyclic Matrix Group Generator

The `circ` command can be used to construct (block) circulant and (block) negacyclic matrices. This command requires one of `CcNn`, where `CN` require an additional input file using `-i infile`. This command expects one parameter, namely, the (block) dimension of the (block) matrix.

6.1 Circulant and Negacyclic Generators

The circulant and negacyclic matrices constructed here are the usual with first row $(0, 1, 0, \ldots, 0)$. To construct these matrices, use `circ [-c | -n] dim -o gen.txt`, where `dim` is the dimension of the matrix.

For instance, `circ -c 8 -o gen.txt` produces

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

while `circ -n 8 -o gen.txt` yields

$$\begin{matrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}.$$

6.2 Block Circulant and Negacyclic Generators

We can also construct block circulant and negacyclic generators as well. To use this functionality, the command requires an input files viz. `-i infile`. If the matrix stored in the input file is A, then we can construct the block circulant or negacyclic matrix with first row $(O, A, O, \ldots, O)$.

If $A = J_3$ is in the file `mat.txt`, then `circ -C 4 -i mat.txt -o gen.txt` constructs

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

The command `circ -N 4 -i mat.txt -o gen.txt` gives

$$\begin{array}{cccccccccccc} 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}.$$

7 Note on the Formatting of Matrix Files

The format of the matrix input files used in the previous command invokations is important. Rows must be seperated by '`\n`', while the seperation of row elements varies by command. The commands `bgw, circ` require that row elements be seperated by '` `'. The commands `cliq, design` require that there be no seperation between the row elements. Further all of the output matrix files are such that row elements are seperated by '` `' with the exception of `bgw -Hp`. It becomes important, then, for the matrix files to be formatted correctly. In what follows, we will assume that the rows are already seperated by a '`\n`'.

If you need to remove the whitespaces between row elements, use

```
sed 's/[[:blank:]]//g' oldfile > newfile
```

To add a space between row elements when none exists, use

```
sed 's/./& /g' oldfile > newfile
```

When using the `ExportMatrix` function in maple, the matrix is stored with '`\t`' seperating row elements. This must be changed to '' or '` `'. The first `sed` line shown above shows the first of these, while the second is accomplished with

```
sed 's/\t/ /g' oldfile > newfile
```