LDPC Codes from Semipartial Geometries

James Rhys Harwood Hutton
Doctor of Philosophy in Mathematics
University of Sussex

May 2011

Contents

Chapter 1

Introduction

1.1 General background

A binary low-density parity-check (LDPC) code is a linear block code that is defined by a sparse parity-check matrix H, that is H has a low density of 1's. LDPC codes were originally presented by Gallager in his doctoral dissertation [9], but largely overlooked for the next 35 years. A notable exception was [29], in which Tanner introduced a graphical representation for LDPC codes, now known as Tanner graphs. However, interest in these codes has greatly increased since 1996 with the publication of [22] and other papers, since it has been realised that LDPC codes are capable of achieving near-optimal performance when decoded using iterative decoding algorithms.

LDPC codes can be constructed randomly by using a computer algorithm to generate a suitable matrix H. However, it is also possible to construct LDPC codes explicitly using various incidence structures in discrete mathematics. For example, LDPC codes can be constructed based on the points and lines of finite geometries: there are many examples in the literature (see for example [18, 28]). These constructed codes can possess certain advantages over randomly-generated codes. For example they may provide more efficient encoding algorithms than randomly-generated codes. Furthermore it can be easier to understand and determine the properties of such codes because of the underlying structure.

LDPC codes have been constructed based on incidence structures known as partial geometries [16]. The aim of this research is to provide examples of new codes constructed based on structures known as semipartial geometries (SPGs), which are generalisations of partial geometries.

Since the commencement of this thesis [19] was published, which showed that codes could be constructed from semipartial geometries and provided some examples and basic results. By necessity this thesis contains a number of results from that paper. However, it should be noted that the scope of [19] is fairly limited and that the overlap between the current thesis and [19] is consequently small. [19] also contains a number of errors, some of which have been noted and corrected in this thesis.

The next two sections provide a brief review of LDPC codes and semipartial geometries, introducing the notation and results that will be used in this thesis.

1.2 LDPC codes

A binary low-density parity-check code is a linear block code for which the parity-check matrix H has a low density of 1's. LDPC codes can be generalised to other alphabets, but non-binary LDPC codes will not be considered in this thesis.

Let $H = [h_{ij}]$ be an $m \times n$ parity-check matrix of an LDPC code $\mathcal{C}$. If H contains exactly γ 1's in each column and exactly $\rho = \gamma n/m$ 1's in each row, then we say that $\mathcal{C}$ is *regular*, or more specifically (γ, ρ)-*regular*.[1] H will have a low density of 1's and therefore define an LDPC code provided that $\gamma \ll m$, or equivalently that $\rho \ll n$. If H has full rank over $GF(2)$ then the rate R of $\mathcal{C}$ is given by $R = 1 - \gamma/\rho$.

If the number of 1's is not constant in either the rows or the columns, we say that $\mathcal{C}$ is irregular.

We shall denote the density of a matrix H by H_d, where H_d is the total number of 1's in H divided by mn.

An LDPC code can be represented by a bipartite graph known as a Tanner graph [29]. The nodes of the graph are referred to as bit nodes and check (or parity) nodes, and the nodes are connected so that bit node j is connected to check node i if h_{ij} is 1. If $\mathcal{C}$ is regular then the bit nodes have uniform degree γ and the check nodes have uniform degree ρ.

The girth of a Tanner graph is its minimum cycle length. The girth is always even and must be greater than or equal to 4. Note that a 4-cycle in

[1] This notation is not used consistently in the literature. I have used the notation as it is used in [17], [15] and elsewhere. However some authors reverse the order of the row and column weights (e.g. [27] .)

the Tanner graph corresponds to a submatrix of H whose corner entries are all 1.

1.3 Semipartial geometries

1.3.1 Definition and basic properties

In this section we define semipartial geometries, and give some notation and basic results that will be used throughout this thesis.

We begin with the definition.

Definition 1.1. *A finite semipartial geometry is an incidence structure* $\mathcal{S} = (\mathcal{P}, \mathcal{B}, \mathrm{I})$ *in which* $\mathcal{P}$ *and* $\mathcal{B}$ *are disjoint non-empty sets of objects called points and lines respectively, and for which* I *is a symmetric point-line incidence relation satisfying the following axioms:*

(i) *each point is incident with* $t+1$ $(t \geq 1)$ *lines and two distinct points are incident with at most one line;*

(ii) *each line is incident with* $s+1$ $(s \geq 1)$ *points and two distinct lines are incident with at most one point;*

(iii) *if a point* P *and line* l *are not incident, then there are either* 0 *or* $\alpha \geq 1$ *points that are collinear with* P *and incident with* l*;*

(iv) *if two points are not collinear then there are* $\mu > 0$ *points collinear with both.*

We denote a semipartial geometry by $\mathrm{spg}(s, t, \alpha, \mu)$, where s, t, α and μ are defined as above.

Given $P, P' \in \mathcal{P}$, we write $P \sim P'$ and say that P and P' are collinear if there is some line l for which P I l I P'. Here $P \nsim P'$ means that P and P' are not collinear. Dually, for $l, l' \in \mathcal{B}$, we write $l \sim l'$ or $l \nsim l'$ according as l and l' are concurrent or non-concurrent.

Let $|\mathcal{P}| = v$ and $|\mathcal{B}| = b$. Then (see [14]) it can be shown that

$$\begin{aligned} v(t+1) &= b(s+1) \\ v &= 1 + (t+1)s(1 + t(s - \alpha + 1)/\mu). \end{aligned}$$

The point graph of a semipartial geometry is strongly regular with parameters $(v, s(t+1), s-1+t(\alpha-1), \mu)$. This means that each of the v vertices of the point graph is connected to exactly $s(t+1)$ other vertices, any two connected vertices are both connected to exactly $s-1+t(\alpha-1)$ other vertices, and any two unconnected vertices are both connected to exactly μ other vertices.

1.3.2 Special cases of semipartial geometries

Semipartial geometries were introduced in [6] as a generalisation of partial geometries: a semipartial geometry is a partial geometry if and only if the zero in axiom (iii) of Definition 1.1 does not occur. From this it is clear that a partial geometry is a semipartial geometry with $\mu = \alpha(t+1)$, and conversely. We call a semipartial geometry that is not a partial geometry a *proper* semipartial geometry.

Semipartial geometries may also be viewed as generalisations of partial quadrangles: a partial quadrangle is a semipartial geometry with $\alpha = 1$. Partial quadrangles were introduced in [5] as a generalisation of generalised quadrangles. We recall that a generalised quadrangle is a partial geometry with $\alpha = 1$.

It follows from these remarks that generalised quadrangles, partial quadrangles and partial geometries are all special cases of semipartial geometries and can be defined by restricting axiom (iii) of Definition 1.1 as follows:

- Generalised quadrangles: $\alpha = 1$ and the zero does not occur;
- Partial quadrangles: $\alpha = 1$;
- Partial geometries: the zero does not occur.

Since generalised quadrangles, partial quadrangles and partial geometries are all special cases of semipartial geometries we shall for the most part use the notation $\mathrm{spg}(s, t, \alpha, \mu)$ in all cases. A semipartial geometry is a partial geometry if and only if $\mu = (t+1)\alpha$, for which reason the value of μ is not required in order to specify a partial geometry. We shall also sometimes use the notation $\mathrm{pg}(s, t, \alpha)$ to denote a partial geometry.

As observed in [14] we have the following scheme, in which '$\rightarrow$' stands for 'generalises to':

$$\begin{array}{ccc} \text{generalised quadrangle} & \longrightarrow & \text{partial geometry} \\ \downarrow & & \downarrow \\ \text{partial quadrangle} & \longrightarrow & \text{semipartial geometry.} \end{array}$$

1.4 LDPC codes from semipartial geometries

A semipartial geometry can be used to define a regular binary LDPC code by using an incidence matrix of the geometry as the parity-check matrix H for the code. We shall call an LDPC code that is defined using the incidence matrix of a semipartial geometry an *SPG code*.

The following subsections provide some general results and observations that will be used later in this thesis. These results are largely based on results contained in [16].

1.4.1 The matrix H

We shall follow [3, Chapter 9] and label the columns of H with lines and the rows with points. Thus H has v rows and b columns. The row-weight is $\rho = t + 1$ and the column weight is $\gamma = s + 1$. The matrix will have a low density of 1's and therefore define an LDPC code provided that $s + 1 \ll v$, or equivalently that $t + 1 \ll b$.

Note that for a proper semipartial geometry we have that $b \geq v$ ([14]), so in this case H has at least as many columns as rows.

1.4.2 Girth

In general, we are particularly interested in constructing codes that have Tanner graphs with large girth, that is they are free of short cycles. This is because short cycles in the Tanner graph degrade the performance of the decoding algorithm.

The girth will always be even since the graph is bipartite, and the girth will always be at least six since it is not possible in a semipartial geometry for two distinct points to lie on two distinct lines.

If $\alpha = 1$ then the Tanner graph is free of 6-cycles and has girth at least 8. Therefore we expect that semipartial geometries that are generalised quadrangles or partial quadrangles could yield SPG codes that perform well under sum-product decoding.

If $\alpha > 1$ then the Tanner graph will contain 6-cycles. In this case it is useful to have a measure of how many 6-cycles the graph contains. We have the following lemma, which is based on a result from [16]. The proof uses the fact that a 6-cycle in the Tanner graph corresponds to a triangle in the corresponding semipartial geometry.

Lemma 1.1. *The number of* 6*-cycles* N_6 *in the Tanner graph of an SPG code derived from a semipartial geometry* $\mathrm{spg}(s,t,\alpha,\mu)$ *with* $|\mathcal{P}| = v$ *and* $|\mathcal{B}| = b$ *satisfies*

$$N_6 = \frac{bt(\alpha-1)}{3}\binom{s+1}{2}.$$

Proof. Let P and P' be two distinct points both incident with a line l. P is incident with t lines other than l, none of which is incident with P'. Since $P \sim P'$ we have by axiom (iii) of Definition 1.1 that P' is collinear with α of the points on each of these lines. Therefore P and P' are contained in $t(\alpha-1)$ triangles. There are $\binom{s+1}{2}$ pairs of points on l, and therefore there are $t(\alpha-1)\binom{s+1}{2}$ triangles containing a pair of points on l. There are b lines in total, but a given triangle contains three pairs of points, from which the result follows. □

1.4.3 Minimum distance - lower bounds

In [30] Tanner presented the *bit-oriented* and *parity-oriented* bounds for the minimum distance d_{min} of a (γ,ρ)-regular code of length n defined by a parity-matrix H. Let $\mu_1 > \mu_2 > \ldots > \mu_s$ be the ordered distinct eigenvalues of HH^T where H is interpreted as a real matrix of zeros and ones. Then the bit-oriented bound ([30] Theorem 3.1) is

$$d_{min} \geq \frac{n(2\gamma-\mu_2)}{(\gamma\rho-\mu_2)}$$

and the parity-oriented bound ([30] Theorem 4.1) is

$$d_{min} \geq \frac{2n(2\gamma+\rho-2-\mu_2)}{\rho(\gamma\rho-\mu_2)}.$$

In [16] these bounds are used to obtain lower bounds for d_{min} in terms of α, s and t, where H is the incidence matrix of a partial geometry. The following lemma is derived.

Lemma 1.2 ([16]). *The minimum distance d_{min} of an LDPC code $\mathcal{C}$ derived from a partial geometry* $\mathrm{pg}(s,t,\alpha)$ *satisfies*

$$d_{min} \geq \max\left\{ \frac{(t+1)(s+1-t+\alpha)}{\alpha}, \frac{2(s+\alpha)}{\alpha} \right\}.$$

Proof. See [16]. □

As noted in [16], for particular classes of partial geometries the bound of Lemma 1.2 is weak. In particular, for partial geometries that are Steiner 2-designs ($\alpha = s+1$), nets ($\alpha = t$) or dual nets ($\alpha = s$) the following Lemma provides a better bound, which is based on a result contained in [23] and referred to as the Massey bound.

Lemma 1.3. *The minimum distance d_{min} of a (γ,ρ)-*regular *LDPC code $\mathcal{C}$ satisfies*

$$d_{min} \geq \gamma + 1.$$

Proof. See [23]. □

We use Lemma 1.3 for LDPC codes derived from Steiner 2-designs, nets and dual nets; for codes derived from other partial geometries we usually use Lemma 1.2.

Using similar arguments to Lemma 1.2 bounds are derived in [19] for codes derived from semipartial geometries. The bounds obtained in [19] contain some errors; a corrected version is given in the following lemma.

Lemma 1.4 ([19]). *The minimum distance d_{min} of an LDPC code $\mathcal{C}$ of length $n = b = |\mathcal{B}|$ derived from a semipartial geometry* $\mathrm{spg}(s,t,\alpha,\mu)$ *satisfies*

$$\begin{aligned} d_{min} \quad \geq \quad \max \Bigg\{ & \frac{n(3s+3+\mu-t\alpha-t-\sqrt{\Delta})}{2st+s+\mu+t+1-t\alpha-\sqrt{\Delta}}, \\ & \frac{2n(3s+1+\mu+t-t\alpha-\sqrt{\Delta})}{(t+1)(2st+s+\mu+t+1-t\alpha-\sqrt{\Delta})} \Bigg\} \end{aligned}$$

where

$$\Delta = (s+t\alpha-t-1-\mu)^2 + 4(st+s-\mu).$$

Proof. See [19]. The errors are for the most part confined to the the statement of the result itself and therefore do not affect the proof significantly. □

If a semipartial geometry is also a partial geometry then we have $n = b = (t+1)(st+\alpha)/\alpha$ and $\mu = (t+1)\alpha$. In this case, as can be readily verified, the bounds of Lemma 1.4 simplify to give the bounds of Lemma 1.2. We shall therefore only use Lemma 1.4 for proper semipartial geometries, although in some cases Lemma 1.3 provides a better lower bound.

1.4.4 Geometric interpretation of codewords

Let $\mathcal{C}$ be a binary SPG code and let $c = (c_1, \ldots, c_n) \in \mathcal{C}$. Since $cH^T = 0$ it follows that $supp(c) = \{i \in \mathbb{N} : c_i \neq 0\}$ defines a set L of lines of $\mathcal{B}$ such that every point of $\mathcal{P}$ lies on an even number of the lines of L. Conversely, if L is a set of lines with this property then L defines a codeword c of weight $|L|$.

It follows that if for a given semipartial geometry we can find a subset L of the lines with the property that every point lies on an even number of the lines of L then we have found a codeword of weight $|L|$ of the SPG code derived from the geometry.

1.4.5 2-rank of H

As discussed in [16] it can be advantageous for the parity-check matrix of an LDPC code to have some linearly dependent rows as this provides extra parity-checking constraints for the decoding algorithm. It is therefore useful to have a measure of the redundancy of the parity-check matrix H of an SPG code.

As we are dealing with binary codes we are interested in $\text{rank}_2(H)$, the rank of H over $GF(2)$.

For codes derived from partial geometries the following two lemmas provide upper and lower bounds on $\text{rank}_2(H)$ (see [16]).

Lemma 1.5. *Let H be the incidence matrix of an SPG code derived from a partial geometry* $\text{pg}(s, t, \alpha)$*. Then*

$$\text{rank}_2(H) \leq \frac{st(s+1)(t+1)}{\alpha(s+t+1-\alpha)} + 1.$$

Proof. An upper bound for $\text{rank}_2(H)$ is the number of non-zero eigenvalues of HH^T. If H is the incidence matrix of a partial geometry then HH^T has two non-zero eigenvalues $(s+1)(t+1)$ and $s+t+1-\alpha$ with multiplicities 1 and $st(s+1)(t+1)/\alpha(s+t+1-\alpha)$ respectively. □

Lemma 1.6. *Let H be the incidence matrix of an SPG code derived from a partial geometry* $\mathrm{pg}(s,t,\alpha)$ *for which* $s+t+1-\alpha \equiv 1 \bmod 2$. *Then*

$$\mathrm{rank}_2(H) \geq \frac{st(s+1)(t+1)}{\alpha(s+t+1-\alpha)}.$$

Proof. See [16]. □

Corollary 1.1. *Let H be the incidence matrix of an SPG code derived from a partial geometry* $\mathrm{pg}(s,t,\alpha)$ *for which* $s+t+1-\alpha \equiv 1 \bmod 2$. *Then*

$$\mathrm{rank}_2(H) = \frac{st(s+1)(t+1)}{\alpha(s+t+1-\alpha)}$$

or

$$\mathrm{rank}_2(H) = \frac{st(s+1)(t+1)}{\alpha(s+t+1-\alpha)} + 1.$$

For proper semipartial geometries, it is not possible to derive a useful upper bound for $\mathrm{rank}_2(H)$ as we did above in Lemma 1.5 for partial geometries. This is because a proper semipartial geometry HH^T has three non-zero eigenvalues

$$(s+1)(t+1), \frac{s+t\alpha+t+1-\mu+\sqrt{\Delta}}{2}, \frac{s+t\alpha+t+1-\mu-\sqrt{\Delta}}{2},$$

with respective multiplicities

$$1, \quad \frac{s(t+1)(st+t+\mu-t\alpha)(\sqrt{\Delta}-\epsilon)-\zeta}{2\mu\sqrt{\Delta}}, \quad \frac{s(t+1)(st+t+\mu-t\alpha)(\sqrt{\Delta}+\epsilon)+\zeta}{2\mu\sqrt{\Delta}},$$

where $\Delta = (s+t\alpha-t-1-\mu)^2+4(st+s-\mu)$, $\epsilon = s+t\alpha-t-1-\mu$ and $\zeta = 2s\mu(t+1)$. In this case the multiplicities sum to $\frac{s(t+1)(st+t+\mu-t\alpha)}{\mu}+1 = v$, giving $\mathrm{rank}_2(H) \leq v$, which is self-evident since H is a $v \times b$ matrix with $v \leq b$.

The following lemma provides a lower bound for $\mathrm{rank}_2(H)$ where H is the incidence matrix of a proper semipartial geometry. This bound was derived in [19], although the statement of the result given there is incorrect.

Lemma 1.7. *Let H be the incidence matrix of an SPG code derived from a proper semipartial geometry* $\mathrm{spg}(s,t,\alpha,\mu)$ *for which* $s+t\alpha+t+1-\mu \equiv 1 \bmod 2$. *Let* $\Delta = (s+t\alpha-t-1-\mu)^2+4(st+s-\mu)$.

If $s+t\alpha+t+1-\mu-\sqrt{\Delta} \equiv 2 \bmod 4$ *then*

$$\mathrm{rank}_2(H) \geq \frac{s(t+1)(st+t+\mu-t\alpha)(\sqrt{\Delta}+s+t\alpha-t-1-\mu)+2s\mu(t+1)}{2\mu\sqrt{\Delta}}.$$

If $s+t\alpha+t+1-\mu+\sqrt{\Delta} \equiv 2 \bmod 4$ *then*

$$\mathrm{rank}_2(H) \geq \frac{s(t+1)(st+t+\mu-t\alpha)(\sqrt{\Delta}-s-t\alpha+t+1+\mu)-2s\mu(t+1)}{2\mu\sqrt{\Delta}}.$$

Proof. See [19]. (Although the result is stated incorrectly in [19] the proof is generally correct. As with Lemma 1.6, the proof relies on a result concerning the p-rank of strongly regular graphs given in [2].) □

Note that Lemma 1.7 only holds for incidence matrices derived from proper semipartial geometries, as the proof requires $\mu \neq (t+1)\alpha$. (This is not made clear in [19].) We use Lemma 1.6 if the semipartial geometry is a partial geometry.

1.4.6 Code dimension

The lower bounds of Lemmas 1.6 or 1.7, if they exist, provide an upper bound for the dimension k of an SPG code, since $k = n - \mathrm{rank}_2(H)$.

Similarly the upper bounds of Lemma 1.5 provide a lower bound for k for an SPG code derived from a partial geometry.

Chapter 2

SPG-reguli

It is possible to construct semipartial geometries using SPG-reguli. These were introduced in [31] where they are defined as follows.

Definition 2.1. *An SPG-regulus is a set $\mathcal{R}$ of m-dimensional subspaces $PG^{(1)}(m,q),\ldots,PG^{(r)}(m,q)$, $r>1$, of $PG(n,q)$ with the following properties.*

(a) *$PG^{(i)}(m,q)\cap PG^{(j)}(m,q)=\emptyset$ for all $i\neq j$.*

(b) *If $PG(m+1,q)$ contains $PG^{(i)}(m,q)$, then it has a point in common with 0 or $\alpha>0$ spaces in $\mathcal{R}\backslash\{PG^{(i)}(m,q)\}$. If $PG(m+1,q)$ has no point in common with $PG^{(j)}(m,q)$ for all $i\neq j$ then it is called a tangent $(m+1)$-space of $\mathcal{R}$ at $PG^{(i)}(m,q)$.*

(c) *If the point $x\in PG(n,q)$ is not contained in an element of $\mathcal{R}$ then it is contained in a constant number θ of tangent $(m+1)$-spaces of $\mathcal{R}$.*

An SPG-regulus $\mathcal{R}$ gives rise to an incidence structure $\mathcal{S}=(\mathcal{P},\mathcal{B},\mathrm{I})$ as follows. $\mathcal{P}$ is the set of points of $\mathrm{PG}(n+1,q)\backslash\mathrm{PG}(n,q)$, so $v=q^{n+1}$. The set $\mathcal{L}$ is the set of $(m+1)$-dimensional subspaces of $\mathrm{PG}(n+1,q)$ that contain an element of $\mathcal{R}$ but are not contained in $\mathrm{PG}(n,q)$. The incidence relation I is that of $\mathrm{PG}(n+1,q)$.

In the particular case $m=0$ we call this incidence structure the *linear representation* of $\mathcal{R}$ and denote it by $T_n^*(\mathcal{R})$. For $m>0$ we call it the *generalised* linear representation of $\mathcal{R}$. In this thesis we shall consider only the case $m=0$.

We shall frequently use the notation $\Sigma=\mathrm{PG}(n+1,q)$ and $\Sigma_0=\mathrm{PG}(n,q)$ when discussing SPG-reguli and their incidence structures.

Theorem 2.1 ([31]). *The incidence structure $\mathcal{S}$ arising from the SPG-regulus $\mathcal{R}$ is a semipartial geometry* $\mathrm{spg}(q^{m+1}-1, r-1, \alpha, (r-\theta)\alpha)$.

Proof. See [31]. □

2.1 Linear representations

As discussed above, an SPG-regulus that is a set of points $\mathcal{R}$ in $\mathrm{PG}(n,q)$ has a linear representation $T_n^*(\mathcal{R})$ that is a semipartial geometry. The various known examples of linear representations arising from SPG-reguli that are sets of points are introduced below.

Following the discussion in 1.3.2 the examples are grouped into three types: partial geometries, proper partial quadrangles and proper semipartial geometries for which $\alpha > 1$.

2.1.1 Partial geometries

Let $\mathcal{K}_{d,q}$ denote a maximal arc of degree d in the projective plane $\Sigma_0 = \mathrm{PG}(2,q)$. Every line of $\mathrm{PG}(2,q)$ is either a d-secant or an external line of $\mathcal{K}_{d,q}$. From [13] Theorem 12.7 we know that either $d|q$ or $d = q+1$. $d = q$ and $d = q+1$ give trivial maximal arcs: $\mathcal{K}_{q,q}$ is the point set of $\mathrm{AG}(2,q) = \mathrm{PG}(2,q)\backslash l$ for some line l, and $\mathcal{K}_{q+1,q}$ is the point set of $\mathrm{PG}(2,q)$.

Non-trivial maximal arcs are those for which $d|q$ and $2 \leq d < q$, and non-trivial maximal arcs exist only in planes of even order $q = 2^h$ (see [1]). In fact there exists a non-trivial maximal arc of degree 2^m for every positive $m < h$.

$\mathcal{K}_{d,q}$ gives rise to a linear representation $T_2^*(\mathcal{K}_{d,q})$, and $T_2^*(\mathcal{K}_{d,q})$ is a partial geometry with parameters $s = q-1$, $t = (q+1)(d-1)$, $\alpha = d-1$.

Note that $T_2^*(\mathcal{K}_{d,q})$ is a generalised quadrangle if and only if q is even and $d = 2$, i.e. if and only if the arc is a hyperoval. The order of the generalised quadrangle $T_2^*(\mathcal{K}_{2,2^h})$ is $(2^h-1,\, 2^h+1)$, that is there are 2^h points on every line, and 2^h+2 lines through every point.

2.1.2 Proper partial quadrangles

Partial quadrangles that are not generalised quadrangles are known as *proper* partial quadrangles.

Let $\mathcal{R}$ be a point set in $\mathrm{PG}(n,q)$. Then $T_n^*(\mathcal{R})$ is a proper partial quadrangle $\mathrm{spg}(q-1,t,1,\mu)$ if and only if $\mathcal{R}$ is a $(t+1)$-cap in $\mathrm{PG}(n,q)$ with the property that each point not in $\mathcal{R}$ lies on $t+1-\mu$ tangents, where $\mu < t+1$. From this it can be deduced that the following cases of linear representations that are proper partial quadrangles can occur (see [3]):

1. $T_3^*(\mathcal{O})$ with $\mathcal{O}$ an ovoid in $\Sigma_0 = \mathrm{PG}(3,q)$. This is a proper partial quadrangle $\mathrm{spg}(q-1,q^2,1,q(q-1))$ and was first constructed in [5].

   The classical example of an ovoid in $\mathrm{PG}(3,q)$ is the elliptic quadric, denoted by $Q^-(3,q)$. A non-classical example is the Tits ovoid, which exists for any $q = 2^{2e+1}, e \geq 1$. There are no other known examples of (q^2+1)-caps in $\mathrm{PG}(3,q)$ other than the elliptic quadric and Tits ovoid.

2. If $q = 3$ then $\mathcal{R}$ is either an ovoid in $\mathrm{PG}(3,3)$, an 11-cap in $\mathrm{PG}(4,3)$ or a 56-cap in $\mathrm{PG}(5,3)$. The linear representation of the 11-cap is an spg(2, 10, 1, 2) and of the 56-cap is an spg(2, 55, 1, 20). The 11-cap and 56-cap are both projectively unique.

3. If $q = 4$ then $\mathcal{R}$ is either an ovoid in $\mathrm{PG}(3,4)$, a 78-cap in $\mathrm{PG}(5,4)$ such that each external point is on 7 secants or a 430-cap in $\mathrm{PG}(6,4)$ such that each external point is on 55 secants. At least one example of such a 78-cap exists, first constructed in [12]. Its linear representation is an spg(3, 77, 1, 14). There is no known example of such a 430-cap; however if one were to exist then its linear representation would be an spg(3, 429, 1, 110).

4. If $q \geq 5$ then the partial quadrangle has to be $T_3^*(\mathcal{O})$ with $\mathcal{O}$ an ovoid in $\mathrm{PG}(3,q)$.

2.1.3 Proper semipartial geometries with $\alpha > 1$

There are two known cases:

1. Let U be a unital in $\Sigma_0 = \mathrm{PG}(2,q^2)$. Then we have that $T_2^*(U)$ is an $\mathrm{spg}(q^2-1,q^3,q,q^2(q^2-1))$. The classical example is a non-singular Hermitian curve U_2; there are several non-classical examples, including Ree unitals and Buekenhout-Metz unitals. See [3].

2. Let B be a Baer-subgeometry of $\Sigma_0 = \mathrm{PG}(n,q^2)$. Then $T_n^*(B)$ is an $\mathrm{spg}(q^2-1,(q^{n+1}-1)/(q-1)-1,q,q(q+1))$.

Chapter 3

SPG codes from linear representations: general results

In this chapter we give some general results concerning the properties of SPG codes derived from linear representations.

We have the following Lemma, which provides lower and upper bounds on the minimum distance of a code derived from any linear representation $T_n^*(\mathcal{R})$ for which $|\mathcal{R}| \geq 2$.

Lemma 3.1. *The minimum distance of a code derived from a linear representation $T_n^*(\mathcal{R})$, $|\mathcal{R}| \geq 2$, satisfies $q+1 \leq d_{min} \leq 2q$.*

Proof. The lower bound follows from Lemma 1.3 since the column weight of H is q.

For the upper bound let $P, P' \in \mathcal{R} \subset \Sigma_0$, and let $\Pi \neq \Sigma_0$ be any plane in Σ that contains P and P'. Consider the set $L \subset \mathcal{L}$ consisting of all lines of Π through P or P'. Every point of $\mathcal{P}$ lies on either 0 or 2 lines of L, and hence (see Section 1.4.4) the code contains a codeword of weight $|L| = 2q$. □

The next result gives a lower bound for the 2-rank of an incidence matrix H of an SPG code.

Result 3.1. *Let H be the incidence matrix of an SPG code derived from a linear representation $T_n^*(\mathcal{R})$. Then* $\text{rank}_2(H) \geq q^n$.

Proof. Consider a point $P \in \mathcal{R}$. There are q^n lines of $\mathcal{L}$ through P. These q^n lines are parallel in $\mathcal{S}$, which means that the corresponding columns of H are linearly independent and therefore that $\text{rank}_2(H) \geq q^n$. □

The following theorem gives us the order of a subgroup of the automorphism group of an SPG code derived from a linear representation.

Theorem 3.1. *Let $\mathcal{C}$ be an SPG code derived from a linear representation $T_n^*(\mathcal{R})$ and let H be the stabiliser of $\mathcal{R}$ in* $\mathrm{PGL}(n+1,q)$ *(i.e. the projective group of $\mathcal{R}$). Then the automorphism group of the code $\mathcal{C}$ contains a subgroup of order $q^{n+1}(q-1) \times |H|$.*

Proof. The stabiliser of $\mathrm{PG}(n,q)$ in $\mathrm{PGL}(n+2,q)$ is $\mathrm{AGL}(n+1,q)$, and the kernel G of the action of $\mathrm{AGL}(n+1,q)$ on $\mathrm{PG}(n,q)$ is the group of translations and dilations of $\mathrm{AG}(n+1,q)$, which has order $q^{n+1}(q-1)$. The extension of G by H is a subgroup of the automorphism group of $\mathcal{C}$ since both G and H fix $\mathcal{L}$. This extension has order $q^{n+1}(q-1) \times |H|$, as required. □

Chapter 4

SPG codes from linear representations: examples

Referring to Section 2.1, we have the following known examples of linear representations of SPG-reguli[1]:

1. The partial geometries $T_2^*(\mathcal{K}_{q+1,q})$. We shall denote the codes arising from their linear representations by $\mathcal{C}_{K_{q+1,q}}$.

2. The partial geometries $T_2^*(\mathcal{K}_{q,q})$. We shall denote the codes arising from their linear representations by $\mathcal{C}_{K_{q,q}}$.

3. The partial geometries $T_2^*(\mathcal{K}_{2^m,2^h})$, $1 < m < h$. We shall denote the codes arising from their linear representations by $\mathcal{C}_{K_{2^m,2^h}}$.

4. The generalised quadrangles $T_2^*(\mathcal{K}_{2,2^h})$. We shall denote the codes arising from their linear representations by $\mathcal{C}_{K_{2,2^h}}$.

5. The proper partial quadrangles $T_3^*(\mathcal{O})$ with $\mathcal{O}$ an ovoid in $\Sigma_0 = \mathrm{PG}(3,q)$. The known ovoids in $\mathrm{PG}(3,q)$ are the elliptic quadric $Q^-(3,q)$ and the Tits ovoid. We shall denote the codes arising from their linear representations by $\mathcal{C}_{Q_q^-}$ and $\mathcal{C}_{TO_q}$ respectively.

6. A proper partial quadrangle $T_4^*(C_{11})$ arising from the projectively unique 11-cap C_{11} in $\Sigma_0 = \mathrm{PG}(4,3)$. $T_4^*(C_{11})$ is an spg(2, 10, 1, 2). We shall denote the code arising from the linear representation by $\mathcal{C}_{C_{11}}$.

[1]The spg(3, 429, 1, 110) arising from a 430-cap has been omitted as it is not known to exist.

7. A proper partial quadrangle $T_5^*(C_{56})$ arising from the projectively unique 56-cap C_{56} in $\Sigma_0 = \mathrm{PG}(5,3)$. $T_5^*(C_{56})$ is an spg(2, 55, 1, 20). We shall denote the code arising from the linear representation by $\mathcal{C}_{C_{56}}$.

8. A proper partial quadrangle $T_5^*(C_{78})$ arising from a 78-cap in $\Sigma_0 = \mathrm{PG}(5,4)$. $T_5^*(C_{78})$ is an spg(3, 77, 1, 14). We shall denote the code arising from the linear representation by $\mathcal{C}_{C_{78}}$.

9. The semipartial geometries $T_2^*(U)$ arising from a unital in $\Sigma_0 = \mathrm{PG}(2,q^2)$. The classical example is a non-singular Hermitian curve U_{2,q^2}: we shall denote the code arising from the linear representation of a non-singular Hermitian curve U_{2,q^2} in $\mathrm{PG}(2,q^2)$ by $\mathcal{C}_{U_{2,q^2}}$. We shall not consider the codes arising from the linear representations of non-classical unitals in this thesis.

10. The semipartial geometries $T_n^*(B)$ arising from a Baer-subgeometry of $\Sigma_0 = \mathrm{PG}(n,q^2)$. We shall denote the code arising from the linear representation of a Baer-subgeometry of $\mathrm{PG}(n,q^2)$ by $\mathcal{C}_{B_{n,q^2}}$.

The SPG codes arising from the linear representations listed above are discussed in the following sections.

4.1 The codes $\mathcal{C}_{K_{q+1,q}}$

In this case $\mathcal{K}$ is the point set of $\mathrm{PG}(2,q)$ and the point set $\mathcal{P}$ of $T_2^*(\mathcal{K})$ is simply the point set of $\mathrm{AG}(3,q) = \mathrm{PG}(3,q)\backslash\mathrm{PG}(2,q)$. $\mathcal{L}$ is the set of lines of $\mathrm{PG}(3,q)$ that contain a point of $\mathrm{PG}(2,q)$ but are not contained in $\mathrm{PG}(2,q)$. We have $s = q-1$, $t = q(q+1)$ and $\alpha = q$. Since $\alpha = s+1$ the partial geometry $T_2^*(\mathcal{K})$ is a 2-$(q^3, q, 1)$ design.

Let $\mathcal{S} = (\mathcal{P}, \mathcal{B}, \mathrm{I})$ be the incidence structure defined by $T_2^*(\mathcal{K})$.

4.1.1 The matrix H

For these codes we have $v = q^3$, $b = n = q^2(q^2+q+1)$, $\gamma = q$ and $\rho = q^2+q+1$. The matrix density is $H_d = \gamma/v = \rho/b = 1/q^2$.

4.1.2 Girth

Since $\alpha = d-1 = q > 1$ the girth of the Tanner graph is 6.

From Lemma 1.1 we have $N_6 = \frac{bt(\alpha-1)}{3}\binom{s+1}{2} = (q^9 - q^7 - q^6 + q^4)/6$.

4.1.3 Minimum distance

From Lemma 3.1 we know that $q+1 \leq d_{min} \leq 2q$.

Using Magma, the minimum distance was calculated for $q = 2, 3, 4$. Unfortunately for larger values of q Magma was not able to determine the minimum distance. The results are displayed in Table 4.1.

Table 4.1: Minimum distance of $\mathcal{C}_{K_{q+1,q}}$

q	d_{min}
2	3
3	6
4	5

We note that for even values of q the lower bound of Lemma 3.1 is attained and for $q = 3$ the upper bound is attained. However it did not prove possible to prove a general result concerning the minimum distance of the codes $\mathcal{C}_{K_{q+1,q}}$.

4.1.4 2-rank of H

Lemma 1.5 gives $\text{rank}_2(H) \leq q^3$. Unfortunately Lemma 1.6 does not apply since $s + t + 1 - \alpha = q^2 + q \equiv 0 \bmod 2$ for both even and odd q.

Based on Magma calculations (see Table 4.2) we can make the following conjecture for odd values of q.

Conjecture 4.1. *For q odd the 2-rank of H is q^3, that is H has full 2-rank.*

4.1.5 Code dimension

Using Magma the dimension of $\mathcal{C}_{K_{q+1,q}}$ was found for small values of q. The results are displayed in Table 4.2.

Table 4.2: Dimension of $\mathcal{C}_{K_{q+1,q}}$ and 2-rank of H

q	length n	dimension k	2-rank of H
2	28	21	7
3	117	90	27
4	336	285	51
5	775	650	125
7	2793	2450	343
8	4672	4299	373
9	7371	6642	729

From these results we can make the following conjecture for odd values of q.

Conjecture 4.2. *For q odd $\mathcal{C}_{K_{q+1,q}}$ has dimension $q^2(q^2+1)$.*

This conjecture is true if and only if all the rows of H are linearly independent over $GF(2)$, which is true if and only if it is not possible to find a non-empty proper subset M of $\mathcal{P}$ such that every line of $\mathcal{B}$ contains an even number of the points in M.

4.1.6 The automorphism group of $\mathcal{C}_{K_{q+1,q}}$

The projective group of the set $\mathcal{K} = \text{PG}(2,q)$ is $\text{PGL}(3,q)$. Hence using Theorem 3.1 we find that the automorphism group of the code $\mathcal{C}_{K_{q+1,q}}$ has a subgroup of order $q^3(q-1) \times |\text{PGL}(3,q)| = q^6(q^3-1)(q^2-1)(q-1)$.

4.2 The codes $\mathcal{C}_{K_{q,q}}$

In this case $\mathcal{K}$ is the point set of $\mathrm{AG}(2,q) = \Sigma_0 \backslash l$ for some line l. We have $s = q-1$, $t = q^2-1$ and $\alpha = q-1$. Since $\alpha = s$ the partial geometry $T_2^*(\mathcal{K})$ is a dual net. Let $\mathcal{S} = (\mathcal{P}, \mathcal{B}, \mathrm{I})$ be the incidence structure defined by $T_2^*(\mathcal{K})$.

4.2.1 The matrix H

For these codes $v = q^3$, $b = n = q^4$, $\gamma = q$ and $\rho = q^2$. The matrix density is $H_d = \gamma/v = \rho/b = 1/q^2$.

4.2.2 Girth

Since $\alpha = q-1 > 1$ for $q > 2$ the girth of the Tanner graph is 6.

From Lemma 1.1 we have

$$N_6 = \frac{bt(\alpha-1)}{3}\binom{s+1}{2} = q^5(q^2-1)(q-1)(q-2)/6.$$

If $q = 2$ then $\mathcal{K}$ is a hyperoval in $\mathrm{PG}(2,q)$. This case is dealt with in Section 4.4 below.

4.2.3 Minimum distance

From Lemma 3.1 we know that $q+1 \leq d_{min} \leq 2q$.

Using Magma the minimum distance was calculated for small values of q and the results are displayed in Table 4.3.

Table 4.3: Minimum distance of $\mathcal{C}_{K_{q,q}}$

q	d_{min}
2	4
3	6
4	6

4.2.4 2-rank of H

Lemma 1.5 gives $\text{rank}_2(H) \leq q^3 - q + 1$.

Lemma 1.6 applies for odd q since $s + t + 1 - \alpha = q^2 \equiv 1 \bmod 2$ for odd q. Hence by Corollary 1.1 we have the following result.

Result 4.1. *For odd q the 2-rank of H for $\mathcal{C}_{K_{q,q}}$ is either $q^3 - q + 1$ or $q^3 - q$.*

Magma computations for the dimension of these codes (see Table 4.4) suggest that in fact for odd q the 2-rank of H is $q^3 - q + 1$.

Comparing Tables 4.2 and 4.4 we can make the following interesting conjecture.

Conjecture 4.3. *For even q, let H_1 be a parity-check matrix for $\mathcal{C}_{K_{q+1,q}}$ and let H_2 be a parity-check matrix for $\mathcal{C}_{K_{q,q}}$. Then* $\text{rank}_2(H_1) = \text{rank}_2(H_2)$.

4.2.5 Code dimension

From Result 4.1 we have the following result.

Result 4.2. *For odd q we have $k = n - \text{rank}_2(H) = q^4 - q^3 + q$ or $q^4 - q^3 + q - 1$.*

Magma computations for small values of q returned the results in Table 4.4.

Table 4.4: Dimension of $\mathcal{C}_{K_{q,q}}$ and 2-rank of H

q	length n	dimension k	2-rank of H
2	16	9	7
3	81	56	25
4	256	205	51
5	625	504	121
7	2401	2064	337
8	4096	3723	373
9	6561	5840	721

These results suggest that for odd values of q the dimension is in fact $q^4 - q^3 + q - 1$.

4.2.6 The automorphism group of $\mathcal{C}_{K_{q,q}}$

The stabiliser of the set $\mathcal{K} = \mathrm{AG}(2,q)$ in $\mathrm{PGL}(3,q)$ is $\mathrm{AGL}(2,q)$. We have $|\mathrm{AGL}(2,q)| = q^2|\mathrm{AGL}(2,q)| = q^2(q-1)|\mathrm{PGL}(2,q)| = q^3(q^2-1)(q-1)$. Hence using Theorem 3.1 we find that the automorphism group of the code $\mathcal{C}_{K_{q,q}}$ has a subgroup of order $q^3(q-1) \times |\mathrm{AGL}(2,q)| = q^6(q^2-1)(q-1)^2$.

4.3 The codes $\mathcal{C}_{K_{2^m,2^h}}, 1 < m < h$

We represent the points of $\Sigma = \mathrm{PG}(3, 2^h)$ by the non-zero left-normalised vectors of $V(4, 2^h)$. Points of $\Sigma_0 = \mathrm{PG}(2, 2^h)$ are represented by the (non-zero left-normalised) vectors $X = (x_0, x_1, x_2, x_3)$ with $x_0 = 0$, and the points of $\Sigma \backslash \Sigma_0$ are represented by the vectors with $x_0 = 1$.

We use the method for constructing maximal arcs for even q given in [8]. We shall call these Denniston maximal arcs to distinguish them from other constructions.

Consider an irreducible homogeneous quadratic polynomial $F(X, Y)$ over $GF(2^h)$, and let H be a subgroup of order 2^m of the additive group of $GF(2^h)$. Then

$$K_{2^m,2^h} = \{(0, 1, x_2, x_3) : F(x_2, x_3) \in H\}$$

is a Denniston maximal arc of degree 2^m in $\mathrm{PG}(2, 2^h)$.

Let $\mathcal{S} = (\mathcal{P}, \mathcal{B}, \mathrm{I})$ be the incidence structure defined by $T_2^*(\mathcal{K}_{d,q}), d = 2^m, q = 2^h, 1 < m < h$.

4.3.1 The matrix H

For these codes $v = q^3$, $b = n = q^2(q(d-1)+d)$, $\gamma = s+1 = q$ and $\rho = t+1 = q(d-1)+d$. The matrix density is $H_d = \gamma/v = \rho/b = 1/q^2$.

4.3.2 Girth

$\alpha = d - 1 > 1$ and therefore the Tanner graph contains 6-cycles.

From Lemma 1.1 we have

$$N_6 = \frac{q^3(q^2-1)(d-1)(d-2)(q(d-1)+d)}{6}.$$

4.3.3 Minimum distance

Using Lemma 3.1 we have $q + 1 \leq d_{min} \leq 2q$.

Applying Lemma 1.2 we find that $d_{min} \geq 2(\frac{q-1}{d-1} + 1)$. Since $d \geq 4$ it follows that for all values of q and d we cannot use Lemma 1.2 to improve the lower bound of $q + 1$.

Unfortunately Magma was unable to compute the minimum distance for the first case for which $1 < m < h$, namely $m = 2$ and $h = 3$ (that is $d = 4$ and $q = 8$).

4.3.4 2-rank of H

Applying Lemma 1.5 gives $\text{rank}_2(H) \leq q^3(1-1/d) + q^2 - q(1-1/d)$. Table 4.5 shows that this upper bound is not sharp.

Lemma 1.6 does not apply since $s + t + 1 - \alpha = dq \equiv 0 \bmod 2$.

4.3.5 Code dimension

Using the result in the previous section we have

$$\begin{aligned} k &= n - \text{rank}_2(H) \\ &\geq q^2(q(d-1)+d) - \{q^3(1-1/d) + q^2 - q(1-1/d)\} \\ &= q^3(d-2+1/d) + q^2(d-1) + q(1-1/d). \end{aligned}$$

Note that when $d = 2$ we have $k \geq q(q+1)^2/2$, which is the result obtained in Section 4.4.5.

Magma was only able to find the dimension for the case $m = 2$ and $h = 3$ (that is $d = 4$ and $q = 8$). This result (see Table 4.5) shows that the lower bound for k given above is not sharp.

Table 4.5: Dimension of $\mathcal{C}_{K_{2^m,2^h}}$ and 2-rank of H

q	d	length n	dimension k	2-rank of H
8	4	1792	1436	356

4.3.6 The automorphism group of $\mathcal{C}_{K_{2^m,2^h}}, 1 < m < h$

In this case we do not have a general result that will give us the projective group of a Denniston maximal arc in $\text{PG}(2, 2^h)$. For example (see [10]) there are two Denniston maximal arcs of degree 4 in $\text{PG}(2, 16)$, which have projective groups of different orders.

We can however state the following, which follows directly from results in [10].

Result 4.3. *The projective group of a Denniston maximal arc in* $\text{PG}(2, 2^h)$ *contains a subgroup of order* $2^h + 1$.

It follows from this and the proof of Theorem 3.1 that the automorphism group of the code $\mathcal{C}_{K_{2^m,2^h}}$ has a subgroup of order $2^{3h}(2^h-1)(2^h+1) = 2^{3h}(2^{2h}-1)$.

4.4 The codes $\mathcal{C}_{K_{2,2^h}}$

In this case $\mathcal{K}$ is a hyperoval in Σ_0 and the linear representation is a generalised quadrangle $T_2^*(\mathcal{K}_{2,2^h})$. Let $\mathcal{S} = (\mathcal{P}, \mathcal{B}, \mathrm{I})$ be the incidence structure defined by $T_2^*(\mathcal{K}_{2,2^h})$.

We know that all hyperovals are regular (that is consist of a conic plus its nucleus) for $h \leq 3$, and that regular hyperovals exist for all values of h (see [13]). As we are trying to find general results that hold for all values of q we shall restrict our attention to regular hyperovals in the following sections.

4.4.1 The matrix H

For these codes $v = q^3$, $b = n = q^2(q+2)$, $\gamma = s+1 = q$ and $\rho = t+1 = q+2$. The matrix density is $H_d = \gamma/v = \rho/b = 1/q^2$.

4.4.2 Girth

Result 4.4. *The Tanner graph of the incidence structure defined by $T_2^*(\mathcal{K}_{2,2^h})$ has girth $g = 8$.*

Proof. $\alpha = 1$ and therefore $g \geq 8$. Let P be any point on the conic and let N be the nucleus. Let $l \in \Sigma_0$ be the bisecant defined by P and N. Consider a plane $\Pi \subset \Sigma$ such that $\Pi \cap \Sigma_0 = l$. Let l_1, l_2, l_3, l_4 be distinct lines in Π, $l_i \neq l$, such that l_1 I P I l_2 and l_3 I N I l_4. Then $l_i \sim l_j$ for $i = 1, 2$, $j = 3, 4$. The four points of intersection along with the four lines correspond to a cycle of length 8 in the Tanner graph. Hence $g = 8$. □

4.4.3 Minimum distance

Result 4.5. *The minimum distance of $\mathcal{C}_{K_{2,2^h}}$ is $2q$.*

Proof. Applying Lemma 1.2 we find that $d_{min} \geq 2q$. It follows from Lemma 3.1 that $d_{min} = 2q$.

□

4.4.4 2-rank of H

Applying Lemma 1.5 gives $\mathrm{rank}_2(H) \leq q(q^2 + 2q - 1)/2$. Lemma 1.6 does not apply since $s + t + 1 - \alpha = 2q \equiv 0 \bmod 2$.

For small values of q Table 4.6 shows the 2-rank of H, obtained by Magma computations. These results show that the upper bound given by Lemma 1.5 is sharp for $q = 2$ and $q = 4$, but not for $q = 8$.

4.4.5 Code dimension

Magma computations for small values of q returned the results in Table 4.6.

Table 4.6: Dimension of $\mathcal{C}_{K_{2,2^h}}$ and 2-rank of H

q	length n	dimension k	2-rank of H
2	16	9	7
4	96	50	46
8	640	341	299

From the previous section we have

$$k = n - \text{rank}_2(H) \geq q^2(q+2) - q(q^2+2q-1)/2 = q(q+1)^2/2.$$

Although this bound is sharp for $q = 2$ and $q = 4$ it is not sharp for $q = 8$: see Table 4.6.

4.4.6 The automorphism group of $\mathcal{C}_{K_{2,2^h}}$

The order of the projective group G of $\mathcal{K}_{2,2^h}$ is given by [13] Corollary 8.27. There are three cases: $h = 1, h = 2$ and $h > 2$.

1. For $q = 2, G \cong \mathrm{S}_4$. Hence the automorphism group of the code $\mathcal{C}_{K_{2,2}}$ has a subgroup of order $2^3 \times |\mathrm{S}_4| = 192$.

2. For $q = 4, G \cong \mathrm{A}_6$. Hence the automorphism group of the code $\mathcal{C}_{K_{2,4}}$ has a subgroup of order $4^3 \times 3 \times |\mathrm{A}_6| = 69120$.

3. For $q = 2^h > 4, G \cong \mathrm{PGL}(2,q)$. Hence the automorphism group of the code $\mathcal{C}_{K_{2,2^h}}$ has a subgroup of order $q^3(q-1) \times |\mathrm{PGL}(2,q)| = 2^{4h}(2^h-1)(2^{2h}-1)$.

4.5 The codes $\mathcal{C}_{Q_q^-}$

Let $\mathcal{S} = (\mathcal{P}, \mathcal{B}, \mathrm{I})$ be the incidence structure defined by $T_3^*(\mathcal{O})$, where $\mathcal{O}$ is an elliptic quadric $Q^-(3,q)$.

4.5.1 The matrix H

For these codes $v = q^4$, $b = n = q^3(q^2+1)$, $\gamma = q$ and $\rho = q^2+1$. The matrix density is $H_d = \gamma/v = \rho/b = 1/q^3$.

4.5.2 Girth

Result 4.6. *The Tanner graph of the incidence structure defined by $T_3^*(\mathcal{O})$ has girth $g = 8$.*

Proof. Since $\alpha = 1$ we know that $g \geq 8$. Consider a line $l \in \Sigma_0$ such that $|l \cap Q^-(3,q)| = 2$. Such a line exists since $Q^-(3,q)$ is a cap in Σ_0. Let $l \cap Q^-(3,q) = \{P, P'\}$. Now consider a plane $\Pi \subset \Sigma$ such that $\Pi \cap \Sigma_0 = l$. Let l_1, l_2, l_3, l_4 be distinct lines in Π, $l_i \neq l$, such that l_1 I P I l_2 and l_3 I P' I l_4. Then $l_i \sim l_j$ for $i = 1, 2$, $j = 3, 4$. The four points of intersection along with the four lines correspond to a cycle of length 8 in the Tanner graph. Hence $g = 8$. □

4.5.3 Minimum distance

Result 4.7. *The minimum distance of $\mathcal{C}_{Q_q^-}$ is $2q$.*

Proof. Applying Lemma 1.4 we find that $d_{min} \geq 2q$. It follows from Lemma 3.1 that $d_{min} = 2q$.

□

4.5.4 2-rank of H

From Result 3.1 we have $\mathrm{rank}_2(H) \geq q^3$. This is a better bound than that provided by Lemma 1.7, which applies only for odd q and gives the bound $\mathrm{rank}_2(H) \geq q^3 - q^2 + q - 1$.

For small values of q Magma was able to compute the 2-rank of H and the results are shown in Table 4.7. These results show that the lower bound

given by Result 3.1 is not sharp for odd or even q. However we can make the following conjecture when q is odd.

Conjecture 4.4. *For q odd the 2-rank of H is q^4, that is H has full 2-rank.*

4.5.5 Code dimension

Since $\text{rank}_2(H) \geq q^3$ we have $k = n - \text{rank}_2(H) \leq q^5$. However Table 4.7 shows that this bound is not sharp for both even and odd q.

Table 4.7: Dimension of $\mathcal{C}_{Q_q^-}$ and 2-rank of H

q	length n	dimension k	2-rank of H
2	40	25	15
3	270	189	81
4	1088	861	227
5	3250	2625	625
7	17150	14749	2401
8	33280	29833	3447

We have the following conjecture for odd values of q.

Conjecture 4.5. *For q odd $\mathcal{C}_{Q_q^-}$ has dimension $q^3(q^2 - q + 1)$.*

This conjecture is true if and only if all the rows of H are linearly independent over $GF(2)$, which is true if and only if it is not possible to find a non-empty proper subset M of $\mathcal{P}$ such that every line of $\mathcal{B}$ contains an even number of the points in M.

4.5.6 The automorphism group of $\mathcal{C}_{Q_q^-}$

Using Theorem 3.1 we find that the automorphism group of the code $\mathcal{C}_{Q_q^-}$ has a subgroup of order $q^4(q-1) \times |PGO_-(4,q)|$. Since $|PGO_-(4,q)| = 2q^2(q^4-1)$ (see [13]) it follows that the automorphism group has a subgroup of order $2q^6(q-1)(q^4-1)$.

4.6 The codes $\mathcal{C}_{TO_q}$

Let $\mathcal{S} = (\mathcal{P}, \mathcal{B}, \mathrm{I})$ be the incidence structure defined by $T_3^*(\mathcal{O})$, where $\mathcal{O}$ is a Tits ovoid in PG$(3, q)$, with $q = 2^{2e+1}, e \geq 1$.

The canonical form of a Tits ovoid (see [3]) is

$$\mathcal{O} = \{(0, 1, 0, 0)\} \cup \{(1, z, y, x) : z = xy + x^{\sigma+2} + y^{\sigma}\},$$

where σ is the automorphism $t \mapsto t^{2e+1}$ of $GF(2^{2e+1})$.

4.6.1 The matrix H

The parameters of H are the same as for the codes $\mathcal{C}_{Q_q^-}$. We have $v = q^4$, $b = n = q^3(q^2 + 1)$, $\gamma = q$ and $\rho = q^2 + 1$. The matrix density is $H_d = \gamma/v = \rho/b = 1/q^3$.

4.6.2 Girth

Result 4.8. *The Tanner graph of the incidence structure defined by $T_3^*(\mathcal{O})$ has girth $g = 8$.*

Proof. As for Result 4.6 above. □

4.6.3 Minimum distance

Result 4.9. *The minimum distance of $\mathcal{C}_{TO_q}$ is $2q$.*

Proof. As for Result 4.9. □

4.6.4 2-rank of H

As for the codes $\mathcal{C}_{Q_q^-}$, the 2-rank of H for $\mathcal{C}_{TO_q}$ satisfies $\mathrm{rank}_2(H) \geq q^3$.

Using Magma it was found that for $\mathcal{C}_{TO_8}$ we have

$$\mathrm{rank}_2(H) = n - k = 33280 - 29722 = 3558.$$

4.6.5 Code dimension

As for the codes $\mathcal{C}_{Q_q^-}$ we have $k \leq q^5$.

The first value of q for which a Tits Ovoid exists is $q = 8$. Using Magma the dimension of the code $\mathcal{C}_{TO_8}$ was found to be 29722. Note that this is not the same as the dimension of the code $\mathcal{C}_{Q_q^-}$, which has dimension 29833.

4.6.6 The automorphism group of $\mathcal{C}_{TO_q}$

The stabiliser of $T_3^*(\mathcal{O})$ in PGL$(4,q)$ is the Suzuki group Sz(q) of order $q^2(q^2+1)(q-1)$ (see [7]). It follows from Theorem 3.1 that the automorphism group of the code $\mathcal{C}_{TO_q}$ has a subgroup of order $q^6(q^2+1)(q-1)^2$.

4.7 The code $\mathcal{C}_{C_{11}}$

Let $\mathcal{S} = (\mathcal{P}, \mathcal{B}, \mathrm{I})$ be the incidence structure defined by $T_4^*(C_{11})$, where C_{11} is the projectively unique 11-cap in PG$(4, 3)$.

We construct C_{11} as follows (see [25]). Let π be a plane in PG$(4, 3)$, and let $Q = \{P_1, P_2, P_3, P_4\}$ be a quadrilateral in π. Let $l = \{V_1, V_2, V_3, V_4\}$ be a line in PG$(4, 3)$ that does not meet π. Consider the four lines $P_iV_i, i = 1, 2, 3, 4$. There are eight points on these lines (two on each line) other than P_i and V_i. These eight points and the three diagonal points of Q form the 11-cap C_{11}.

4.7.1 The matrix H

Here $v = 3^5 = 243$, $b = n = 3^4 \times 11 = 891$, $\gamma = 3$ and $\rho = 11$. The matrix density is $H_d = \gamma/v = \rho/b = 1/81$.

4.7.2 Girth

Since $\alpha = 1$ we know that $g \geq 8$. Arguing as in the proof of Theorem 4.6 we can find four coplanar lines in $\mathcal{B}$ that intersect in four points of $\mathcal{P}$, showing that $g = 8$.

4.7.3 Minimum distance

A computation using Magma shows that the minimum distance of the code $\mathcal{C}_{C_{11}}$ is 6.

4.7.4 2-rank of H

Using Magma we find that $\mathrm{rank}_2(H) = 891 - 648 = 243$. Therefore the matrix H has full rank over $GF(2)$.

Note that this means that there does not exist a non-empty proper subset M of $\mathcal{P}$ such that every line of $\mathcal{B}$ contains an even number of the points in M.

4.7.5 Code dimension

Magma gives the code dimension to be 648.

4.7.6 The automorphism group of $\mathcal{C}_{C_{11}}$

The projective group of the set $\mathcal{K} = C_{11}$ is isomorphic to the Mathieu group M_{11} of order $11 \times 10 \times 9 \times 8 = 7920$ (see [26]). Hence using Theorem 3.1 we find that the automorphism group of the code $\mathcal{C}_{C_{11}}$ has a subgroup of order $3^5 \times 2 \times 7920 = 3849120$.

4.8 The code $\mathcal{C}_{C_{56}}$

Let $\mathcal{S} = (\mathcal{P}, \mathcal{B}, \mathrm{I})$ be the incidence structure defined by $T_5^*(C_{56})$, where C_{56} is the projectively unique 56-cap in PG$(5, 3)$. Following [12], the 56-cap is constructed as half of the points on an elliptic quadric in PG$(5, 3)$.

4.8.1 The matrix H

Here $v = 3^6 = 729$, $b = n = 3^5 \times 56 = 13608$, $\gamma = 3$ and $\rho = 56$. The matrix density is $H_d = \gamma/v = \rho/b = 1/243$.

4.8.2 Girth

Since $\alpha = 1$ we know that $g \geq 8$. Arguing as above, we can find four coplanar lines in $\mathcal{B}$ that intersect in four points of $\mathcal{P}$, showing that $g = 8$.

4.8.3 Minimum distance

It was not possible to find the minimum distance using Magma.

4.8.4 2-rank of H

Using Magma we find that $\mathrm{rank}_2(H) = 13608 - 12879 = 729$. Therefore the matrix H has full rank over $GF(2)$.

As we have seen in other examples the full 2-rank of H implies that there does not exist a non-empty proper subset M of $\mathcal{P}$ such that every line of $\mathcal{B}$ contains an even number of the points in M.

4.8.5 Code dimension

Magma gives the code dimension to be 12879.

4.8.6 The automorphism group of $\mathcal{C}_{C_{56}}$

The projective group of the set $\mathcal{K} = C_{56}$ is isomorphic to $[PSL(3, 4)]C_2$, an extension of $PSL(3, 4)$ by a cyclic group of order 2 (see [11]). From [13], $|PSL(3, 4)| = 20160$. Hence using Theorem 3.1 we find that the automorphism group of the code $\mathcal{C}_{C_{56}}$ has a subgroup of order $3^6 \times 2 \times 20160 \times 2 = 58786560$.

4.9 The code $\mathcal{C}_{C_{78}}$

Let $\mathcal{S} = (\mathcal{P}, \mathcal{B}, \mathrm{I})$ be the incidence structure defined by $T_5^*(C_{78})$, where C_{78} is a 78-cap in PG$(5, 4)$. For the 78-cap we use a construction discovered by Hill. (See [12], although the construction is not actually given in that article. A brief description of the construction is given in [4].)

The points of Hill's 78-cap are listed in the Magma code in the Appendix of this thesis.

4.9.1 The matrix H

Here $v = 4^6 = 4096$, $b = n = 4^5 \times 78 = 79872$, $\gamma = 4$ and $\rho = 78$. The matrix density is $H_d = \gamma/v = \rho/b = 1/1024$.

4.9.2 Girth

Since $\alpha = 1$ we know that $g \geq 8$. Arguing as previously, we can find four coplanar lines in $\mathcal{B}$ that intersect in four points of $\mathcal{P}$, showing that $g = 8$.

4.9.3 Minimum distance

It was not possible to find the minimum distance using Magma.

4.9.4 2-rank of H

It was not possible to find the 2-rank of H using Magma.

4.9.5 Code dimension

It was not possible to find the code dimension using Magma.

4.9.6 The automorphism group of $\mathcal{C}_{C_{78}}$

The projective group of the set $\mathcal{K} = C_{78}$ is isomorphic to the direct product of a cyclic group of order 3 with an extension of a cyclic group of order 13 by a cyclic group of order 6 (see [12]). Hence using Theorem 3.1 we find that the automorphism group of the code $\mathcal{C}_{C_{78}}$ has a subgroup of order $4^6 \times 3 \times 3 \times 6 \times 13 = 2875392$.

4.10 The codes $\mathcal{C}_{U_{2,q^2}}$

Let $\mathcal{S} = (\mathcal{P}, \mathcal{B}, \mathrm{I})$ be the incidence structure defined by $T_2^*(U_{2,q^2})$, where U_{2,q^2} is a non-singular Hermitian curve in $\mathrm{PG}(2, q^2)$.

4.10.1 The matrix H

Here $v = q^6$, $b = n = q^4(q^3+1)$, $\gamma = q^2$ and $\rho = q^3 + 1$. The matrix density is $H_d = \gamma/v = \rho/b = 1/q^4$.

4.10.2 Girth

$\alpha = q > 1$ and therefore the Tanner graph contains 6-cycles.

From Lemma 1.1 we have

$$N_6 = \frac{q^9(q^3+1)(q^2-1)(q-1)}{6}.$$

4.10.3 Minimum distance

Applying Lemma 1.3 we find that $d_{min} \geq q^2 + 1$. However this bound is not sharp as the following results from [27] show.

Result 4.10. *For even q the minimum distance of the code $\mathcal{C}_{U_{2,q^2}}$ is $q^2 + q$.*

Result 4.11. *For odd q the minimum distance of the code $\mathcal{C}_{U_{2,q^2}}$ is even and satisfies*

$$q^2 + q \leq d_{min} \leq 2q^2.$$

4.10.4 2-rank of H

Applying Lemma 1.7 we find that $s + t\alpha + t + 1 - \mu - \sqrt{\Delta} = 2q^2 \equiv 2 \bmod 4$ for odd q, but $s + t\alpha + t + 1 - \mu + \sqrt{\Delta} = 2(q^3 + q^2) \not\equiv 2 \bmod 4$ for all q. We therefore have the following result.

Result 4.12. *For q odd H satisfies*

$$\mathrm{rank}_2(H) \geq q^5 - q^3 + q^2 - 1.$$

Magma results for the 2-rank for small values of q^2 are given in Table 4.8. For $q^2 = 9$ we have $\text{rank}_2(H) = 2268 - 1539 = 729$. Therefore for this value of q^2 the matrix H has full rank over $GF(2)$.

We conjecture that for odd values of q^2 the matrix H has full rank q^6 over $GF(2)$.

4.10.5 Code dimension

Using Result 4.12 we can derive an upper bound for the dimension for odd q.

Result 4.13. *For odd q the dimension of $\mathcal{C}_{U_{2,q^2}}$ satisfies*

$$k = n - \text{rank}_2(H) \leq q^7 - q^5 + q^4 + q^3 - q^2 + 1.$$

Using Magma the code dimension was computed for small values of q. The results are contained in Table 4.8.

Table 4.8: Dimension of $\mathcal{C}_{U_{2,q^2}}$ and 2-rank of H

q^2	length n	dimension k	2-rank of H
4	144	96	48
9	2268	1539	729

We conjecture that for odd values of q^2 the dimension of the code is $q^4(q^3 - q^2 + 1)$. As in other cases, we note that this conjecture is true if and only if all the rows of H are linearly independent over $GF(2)$, which is true if and only if it is not possible to find a non-empty proper subset M of $\mathcal{P}$ such that every line of $\mathcal{B}$ contains an even number of the points in M.

4.10.6 The automorphism group of $\mathcal{C}_{U_{2,q^2}}$

The projective group of the set $\mathcal{K} = U_{2,q^2}$ is $\text{PGU}(3, q^2)$, which has order $q^3(q^3 + 1)(q^2 - 1)$ (see [13]). Hence using Theorem 3.1 we find that the automorphism group of $\mathcal{C}_{U_{2,q^2}}$ has a subgroup of order $q^9(q^3 + 1)(q^2 - 1)^2$.

4.11 The codes $\mathcal{C}_{B_{2,q^2}}$

We restrict our attention to codes arising from the linear representation of a Baer-subgeometry of PG$(2, q^2)$.

Let $\mathcal{S} = (\mathcal{P}, \mathcal{B}, \mathrm{I})$ be the incidence structure defined by $T_2^*(B)$, where B is a Baer-subgeometry of PG$(2, q^2)$.

4.11.1 The matrix H

In the general case, where B is a Baer-subgeometry of PG(n, q^2), we have $v = q^{2(n+1)}$, $b = q^{2n}(q^{n+1} - 1)/(q - 1)$, $\gamma = q^2$ and $\rho = (q^{n+1} - 1)/(q - 1)$. The matrix density is $H_d = \gamma/v = \rho/b = 1/q^{2n}$.

Hence for $\mathcal{C}_{B_{2,q^2}}$ we have $v = q^6$, $b = q^4(q^2+q+1)$, $\gamma = q^2$ and $\rho = q^2+q+1$. The matrix density is $H_d = 1/q^4$.

4.11.2 Girth

$\alpha = q > 1$ and therefore the Tanner graph contains 6-cycles.

From Lemma 1.1 we have

$$N_6 = \frac{q^7(q^2 - 1)^2(q^2 + q + 1)}{6}.$$

4.11.3 Minimum distance

Applying Lemma 1.3 we find that $d_{min} \geq q^2 + 1$. However this bound is not sharp as the following results from [27] show.

Result 4.14. *For even q the minimum distance of the code $\mathcal{C}_{B_{2,q^2}}$ is $q^2 + q$.*

Result 4.15. *For odd q the minimum distance of the code $\mathcal{C}_{B_{2,q^2}}$ is even and satisfies*

$$q^2 + q \leq d_{min} \leq 2q^2 - 2.$$

4.11.4 2-rank of H

Applying Lemma 1.7 we find that $s + t\alpha + t + 1 - \mu - \sqrt{\Delta} = 2q^2 \equiv 2 \bmod 4$ iff q is odd, but $s + t\alpha + t + 1 - \mu + \sqrt{\Delta} = 2(q^3 + q^2) \not\equiv 2 \bmod 4$ for all q. We therefore have the following result.

Result 4.16. *For odd q, the code $\mathcal{C}_{B_{2,q^2}}$ satisfies* $\mathrm{rank}_2(H) \geq q^6 - q^4 - q^3 + q$.

Magma results for the 2-rank are given in Table 4.9. We conjecture that for odd values of q^2 the matrix H has full rank q^6 over $GF(2)$.

4.11.5 Code dimension

Using Result 4.16 we can derive an upper bound for the dimension for odd values of q.

Result 4.17. *For odd q the dimension of $\mathcal{C}_{B_{2,q^2}}$ satisfies*

$$k = n - \text{rank}_2(H) \leq q^5 + 2q^4 + q^3 - q.$$

Using Magma the code dimension was computed for small values of q. The results are contained in Table 4.9.

Table 4.9: Dimension of $\mathcal{C}_{B_{2,q^2}}$ and 2-rank of H

q^2	length n	dimension k	2-rank of H
4	112	67	45
9	1053	324	729

We conjecture that for odd values of q^2 the code dimension is $q^4(q+1)$. As in other cases, we note that this conjecture is true if and only if all the rows of H are linearly independent over $GF(2)$, which is true if and only if it is not possible to find a non-empty proper subset M of $\mathcal{P}$ such that every line of $\mathcal{B}$ contains an even number of the points in M.

4.11.6 The automorphism group of $\mathcal{C}_{B_{2,q^2}}$

The projective group of a Baer-subgeometry of $\text{PG}(2, q^2)$ is isomorphic to $\text{PGL}(3, q)$ and has order $q^3(q^3-1)(q^2-1)$. Hence using Theorem 3.1 we find that the automorphism group of the code $\mathcal{C}_{B_{2,q^2}}$ has a subgroup of order $q^9(q^3-1)(q^2-1)^2$.

Chapter 5

Code simulation results 1

This section provides some simulation results that test the performance of some of the SPG codes presented in the previous section.

The codes were tested on the additive white gaussian noise (AWGN) channel and decoded using iterative probabilistic decoding. The simulations were performed using source code written by the author of [24]. The source code can be downloaded from http://the-art-of-ecc.com. The decoding used a maximum of ten iterations.

The performance of each code was compared with that of a randomly generated code of the same rate and length. The random codes were constructed following the methods in [22] and [21] and using source code downloaded from http://www.cs.utoronto.ca/∼radford/ldpc.software.html.[1]

The intention was to generate random codes that would perform well and thereby give a good benchmark for assessing the performance of the SPG codes. At the signal-to-noise ratios we are considering randomly generated LDPC codes perform well with a column weight of 3 (as discussed in [15]) and therefore the random codes were constructed so that they would have as many columns of weight 3 as possible. This was achieved using the evencol option in Neal's program. In addition the no4cycle option was used, which eliminates cycles of length four in the Tanner graph, if possible.

A further comparison was provided by an uncoded signal transmitted on the same AWGN channel.

[1]The format of the alist files generated by Neal's program is not entirely compatible with Morelos-Zaragoza's simulation code. This problem was overcome by modifying the downloaded source code.

5.1 The codes $\mathcal{C}_{K_{q+1,q}}$

The codes $\mathcal{C}_{K_{3,2}}$, $\mathcal{C}_{K_{4,3}}$, $\mathcal{C}_{K_{5,4}}$ and $\mathcal{C}_{K_{6,5}}$ were tested.

5.1.1 $\mathcal{C}_{K_{3,2}}$

The code $\mathcal{C}_{K_{3,2}}$ is a (2,7)-regular [28, 21, 3] code of rate 0.75. Its parity-check matrix is an 8×28 matrix with 2-rank $28 - 21 = 7$.

The performance of $\mathcal{C}_{K_{3,2}}$ was compared with a randomly constructed LDPC code that had an 8×28 parity-check matrix with column weight 3 and row weights that ranged between 1 and 25. The 2-rank of this matrix was calculated using Magma and was found to be 7, the same as for $\mathcal{C}_{K_{3,2}}$. The rate and length of the randomly constructed code were therefore the same as for $\mathcal{C}_{K_{3,2}}$. The matrix contained some 4-cycles.

The results are shown in Figure 5.1. The code $\mathcal{C}_{K_{3,2}}$ performed well in comparison with the randomly generated code. This is probably because the Tanner graph of $\mathcal{C}_{K_{3,2}}$ contains no 4-cycles and only 56 6-cycles (see Section 4.1.2), whereas the program that generated the random code was not able to eliminate all 4-cycles from the Tanner graph. It is probable that the presence of 4-cycles combined with a greater number of 6-cycles in the Tanner graph explains the poor performance of the randomly constructed code in comparison to $\mathcal{C}_{K_{3,2}}$.

5.1.2 $\mathcal{C}_{K_{4,3}}$

The code $\mathcal{C}_{K_{4,3}}$ is a (3, 13)-regular [117, 90, 6] code of rate $10/13 \approx 0.769$. Its parity-check matrix is a 27×117 matrix of full 2-rank.

The performance of $\mathcal{C}_{K_{4,3}}$ was compared with a randomly constructed LDPC code of the same rate and length. Its parity-check matrix was a 27×117 matrix with column weight 3 and row weights that ranged between 3 and 53. A Magma computation showed that the matrix had full 2-rank. The matrix contained some 4-cycles.

The results are shown in Figure 5.2. The SPG code performed well in comparison with the randomly generated code. As with $\mathcal{C}_{K_{3,2}}$ this can probably be explained by the fact that the Tanner graph of the randomly generated code contained some 4-cycles.

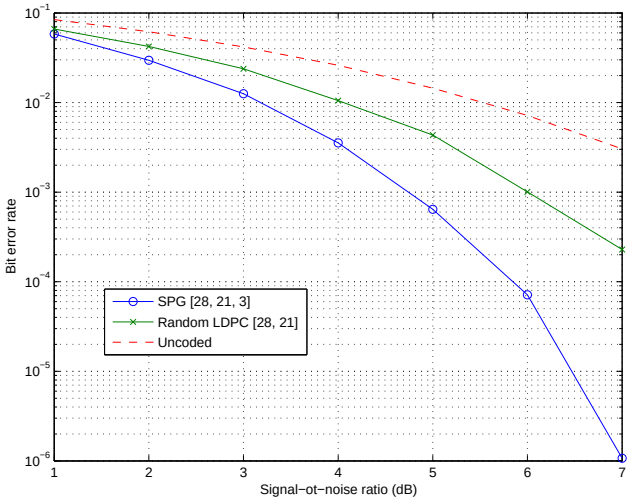

Figure 5.1: The decoding performance of the SPG code $\mathcal{C}_{K_{3,2}}$ on an AWGN channel using iterative probabilistic decoding with a maximum of 10 iterations.

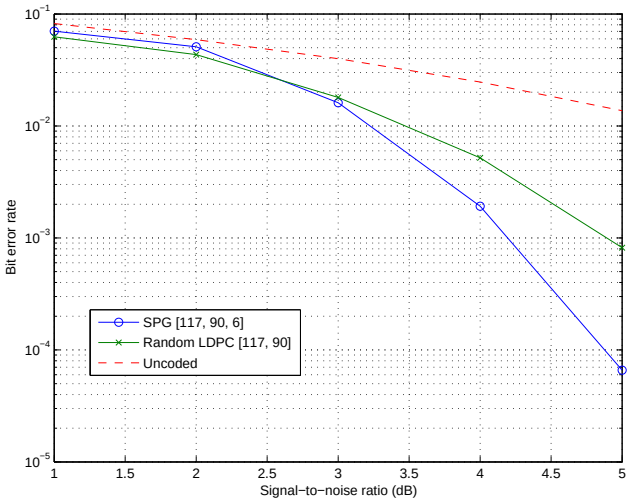

Figure 5.2: The decoding performance of the SPG code $\mathcal{C}_{K_{4,3}}$ on an AWGN channel using iterative probabilistic decoding with a maximum of 10 iterations.

5.1.3 $\mathcal{C}_{K_{5,4}}$

The code $\mathcal{C}_{K_{5,4}}$ is a (4, 21)-regular [336, 285, 5] code of rate $285/336 = 95/112 \approx 0.848$. Its parity-check matrix is a 64×336 matrix of 2-rank $336 - 285 = 51$.

The performance of $\mathcal{C}_{K_{5,4}}$ was compared with a randomly constructed LDPC code of the same rate and length. Its parity-check matrix was a 51×336 matrix with column weight 3 and row weights that ranged between 8 and 79. A Magma computation showed that the matrix had full 2-rank. The matrix contained some 4-cycles.

The results are shown in Figure 5.3. Again the SPG code performed well in comparison with the randomly generated code. One explanation for this is that the Tanner graph of the randomly constructed code contained some 4-cycles. In addition the parity-check matrix of $\mathcal{C}_{K_{5,4}}$ contains linearly dependent rows, which can enhance decoding performance by providing extra checks. These two factors appear to outweigh the fact that the parity-check matrix of the randomly generated code is less dense than the parity-check matrix of $\mathcal{C}_{K_{5,4}}$ (1/17 as opposed to 1/16 for $\mathcal{C}_{K_{5,4}}$).

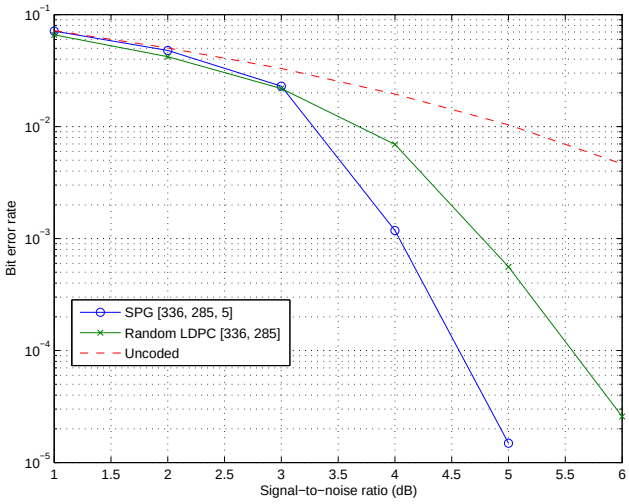

Figure 5.3: The decoding performance of the SPG code $\mathcal{C}_{K_{5,4}}$ on an AWGN channel using iterative probabilistic decoding with a maximum of 10 iterations.

5.1.4 $\mathcal{C}_{K_{6,5}}$

The code $\mathcal{C}_{K_{6,5}}$ is a (5,31)-regular [775, 650] code of rate $650/775 = 26/31 \approx 0.839$. Its parity-check matrix is a 125×775 matrix of full 2-rank.

The performance of $\mathcal{C}_{K_{6,5}}$ was compared with a randomly constructed LDPC code of the same rate and length. Its parity-check matrix was a 125×775 matrix with column weight 3 and row weights that ranged between 9 and 31. A Magma computation showed that the matrix had full 2-rank. The matrix contained some 4-cycles.

Because of the relatively long length of these codes the log-likelihood version of the decoding algorithm was used to perform the simulations.

The results are shown in Figure 5.4. In this case the randomly constructed code performed better than the SPG code. This is slightly surprising since again the parity-check matrix of the randomly constructed code contained some 4-cycles. However in this case the density of the parity-check matrix for $\mathcal{C}_{K_{6,5}}$ is $1/25$ compared to $3/125$ for the randomly constructed code. Therefore the randomly constructed matrix contain $3/5$ the number of 1's in its parity-check matrix. This means that overall the randomly constructed code will have a lower proportion of short cycles in its Tanner graph than the SPG code — specifically it will contain many fewer 6-cycles. This probably explains the better performance of the randomly generated code.

5.2 The codes $\mathcal{C}_{K_{q,q}}$

The codes $\mathcal{C}_{K_{3,3}}$, $\mathcal{C}_{K_{4,4}}$ and $\mathcal{C}_{K_{5,5}}$ were tested.

5.2.1 $\mathcal{C}_{K_{3,3}}$

The code $\mathcal{C}_{K_{3,3}}$ is a (3, 9)-regular [81, 56, 6] code of rate $56/81 \approx 0.691$. Its parity-check matrix is a 27×81 matrix with 2-rank $81 - 56 = 25$.

The performance of $\mathcal{C}_{K_{3,3}}$ was compared with a randomly constructed LDPC code of the same rate and length. Its parity-check matrix was a 25×81 matrix with column weight 3 and row weights that ranged between 3 and 41. A Magma computation showed that the matrix had full 2-rank. The matrix contained some 4-cycles.

The results are shown in Figure 5.5. $\mathcal{C}_{K_{3,3}}$ performed well in comparison with the randomly generated code. This can be attributed to the following factors:

- the Tanner graph of the randomly generated code contains some 4-cycles
- the parity-check matrix of $\mathcal{C}_{K_{3,3}}$ is slightly less dense (1/9 compared to 3/25 for the random code)
- the parity-check matrix of $\mathcal{C}_{K_{3,3}}$ contains linearly dependent rows.

5.2.2 $\mathcal{C}_{K_{4,4}}$

The code $\mathcal{C}_{K_{4,4}}$ is a (4,16)-regular [256, 205, 6] code of rate $205/256 \approx 0.801$. Its parity-check matrix is a 64×256 matrix with 2-rank $256 - 205 = 51$.

The performance of $\mathcal{C}_{K_{4,4}}$ was compared with a randomly constructed LDPC code of the same rate and length. Its parity-check matrix was a 51×256 matrix with column weight 3 and row weights that ranged between 6 and 60. A Magma computation showed that the matrix had full 2-rank. The matrix contained some 4-cycles.

The results are shown in Figure 5.6. Again the SPG code performed well in comparison with the randomly generated code. This can be attributed to the fact that the parity-check matrix of the randomly generated code contains some 4-cycles, as well as the fact that the parity-check matrix of $\mathcal{C}_{K_{4,4}}$ contains some linearly dependent rows.

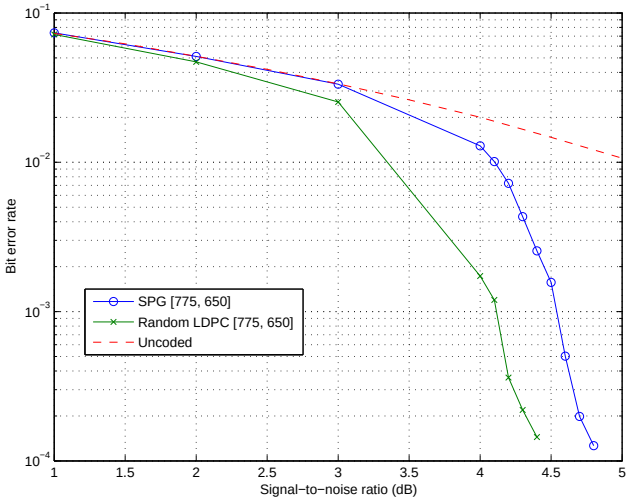

Figure 5.4: The decoding performance of the SPG code $\mathcal{C}_{K_{6,5}}$ on an AWGN channel using log-likelihood iterative probabilistic decoding with a maximum of 10 iterations.

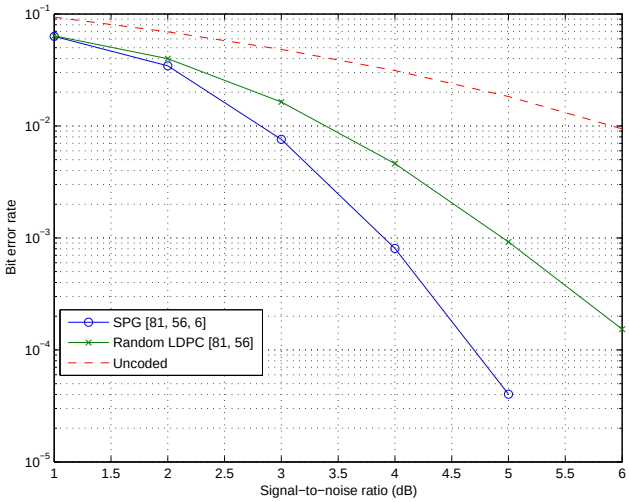

Figure 5.5: The decoding performance of the SPG code $\mathcal{C}_{K_{3,3}}$ on an AWGN channel using iterative probabilistic decoding with a maximum of 10 iterations.

It is interesting that the randomly generated code performed very slightly better than $\mathcal{C}_{K_{4,4}}$ at low signal-to-noise ratios (see Figure 5.6). This is probably because the parity-check matrix of the randomly generated code is slightly less dense than the SPG matrix (1/17 compared to 1/16 for $\mathcal{C}_{K_{4,4}}$), which at low signal-to-noise ratios seems to be a more important factor than the linear dependence of rows in the parity-check matrix of the SPG code.

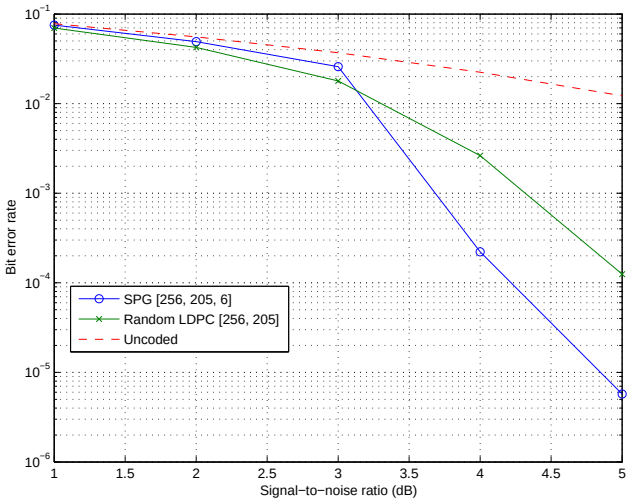

Figure 5.6: The decoding performance of the SPG code $\mathcal{C}_{K_{4,4}}$ on an AWGN channel using iterative probabilistic decoding with a maximum of 10 iterations.

5.2.3 $\mathcal{C}_{K_{5,5}}$

The code $\mathcal{C}_{K_{5,5}}$ is a (5,25)-regular [625,504] code of rate $504/625 \approx 0.806$ and with minimum distance $6 \leq d_{min} \leq 10$. Its parity-check matrix is a 125×625 matrix with 2-rank $625 - 504 = 121$.

The performance of $\mathcal{C}_{K_{5,5}}$ was compared with a randomly constructed LDPC code of the same rate and length. Its parity-check matrix was a 121×625 matrix with column weight 3 and row weights that ranged between 7 and 27. A Magma computation showed that the matrix had full 2-rank. The matrix contained no 4-cycles.

The results are shown in Figure 5.7. In this case the randomly constructed

code performed better than the SPG code. The probable explanation for this is that the parity-check matrix of the randomly generated code is less dense (3/121 compared to 1/25 for $\mathcal{C}_{K_{5,5}}$). Since the random matrix contains no 4-cycles its lower density implies that the Tanner graph will have a lower proportion of 6-cycles in its Tanner graph than the SPG code, which would explain the better performance of the randomly generated code.

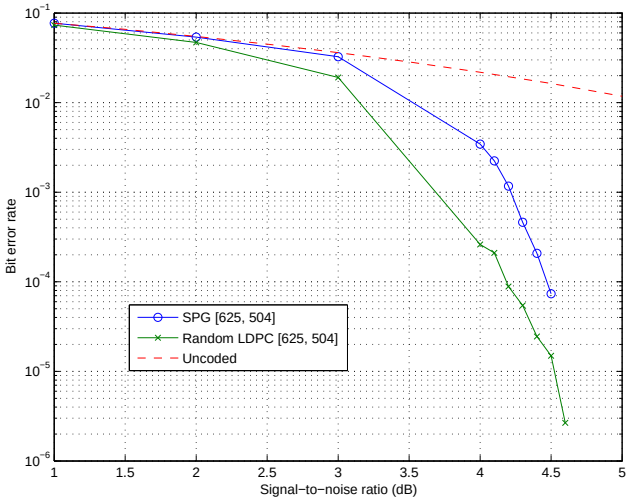

Figure 5.7: The decoding performance of the SPG code $\mathcal{C}_{K_{5,5}}$ on an AWGN channel using iterative probabilistic decoding with a maximum of 10 iterations.

5.3 The codes $\mathcal{C}_{K_{2^m,2^h}}, 1 < m < h$

The shortest code in this class in $\mathcal{C}_{K_{4,8}}$. This is an (8, 28)-regular [1792, 1436, ≥ 5] code of rate $1436/1792 \approx 0.801$. Its parity-check matrix is a 512×1792 matrix with 2-rank $1792 - 1436 = 356$. Other codes in the class were too long to test with the available software.

The performance of $\mathcal{C}_{K_{4,8}}$ was compared with a randomly constructed LDPC code of the same rate and length. Its parity-check matrix was a 356×1792 matrix with column weight 3 and row weights that ranged between 5 and 27. A Magma computation showed that the matrix had full 2-rank. The matrix contained no 4-cycles.

Because of the relatively long length of these codes the log-likelihood version of the decoding algorithm was used to perform the simulations.

The results are shown in Figure 5.8. The randomly constructed code performed significantly better than $\mathcal{C}_{K_{4,8}}$, despite the fact that the parity-check matrix of $\mathcal{C}_{K_{4,8}}$ contains a large number of linearly dependent rows. The probable explanation is that the density of the random matrix is $3/356 \approx 0.00843$ compared to $8/512 \approx 0.0156$ for the SPG code, which means that the random matrix contains nearly half as many 1's. As the random matrix did not contain any 4-cycles its lower density implies that the Tanner graph of the random code must contain a lower proportion of 6-cycles and 8-cycles, which would explain its better performance.

5.4 The codes $\mathcal{C}_{K_{2,2^h}}$

The codes $\mathcal{C}_{K_{2,4}}$ and $\mathcal{C}_{K_{2,8}}$ were tested. Magma was not able to generate the 4096×4608 parity-check matrix for the code $\mathcal{C}_{K_{2,16}}$.

5.4.1 $\mathcal{C}_{K_{2,4}}$

The code $\mathcal{C}_{K_{2,4}}$ is a (4,6)-regular [96, 50, 8] code of rate $50/96 \approx 0.521$. Its parity-check matrix is a 64×96 matrix with 2-rank $96 - 50 = 46$.

The performance of $\mathcal{C}_{K_{2,4}}$ was compared with a randomly constructed LDPC code of the same rate and length. Its parity-check matrix was a 46×96 matrix with column weight 3 and row weights that ranged between 2 and 10. A Magma computation showed that the matrix had full 2-rank. The matrix contained no 4-cycles.

The results are shown in Figure 5.9. $\mathcal{C}_{K_{2,4}}$ performed better than the randomly constructed code. One explanation for this is that the SPG matrix is slightly less dense than the randomly constructed matrix (1/16 compared to 3/46 for the random matrix). In addition the parity-check matrix of $\mathcal{C}_{K_{2,4}}$ contains a number of linearly dependent rows, which can enhance decoding performance.

5.4.2 $\mathcal{C}_{K_{2,8}}$

The code $\mathcal{C}_{K_{2,8}}$ is an (8, 10)-regular [640, 341, 16] code of rate $341/640 \approx 0.533$. Its parity-check matrix is a 512×640 matrix with 2-rank $640 - 341 = 299$.

The performance of $\mathcal{C}_{K_{2,8}}$ was compared with a randomly constructed LDPC code of the same rate and length. Its parity-check matrix was a 299×640 matrix with 634 columns of weight 3 and 6 columns of weight 4. The row weights ranged between 2 and 14. A Magma computation showed that the matrix had full 2-rank.

The results are shown in Figure 5.10. The randomly constructed code performed slightly better for signal-to-noise ratios less than about 1.2dB, but the SPG code performed better for signal-to-noise ratios greater than 1.2dB. It would appear that for values of the signal-to-noise ratio less then 1.2dB the lower density of the random matrix (approximately 0.0101 for the random matrix compared to 0.0156 for the SPG matrix) is the most significant factor,

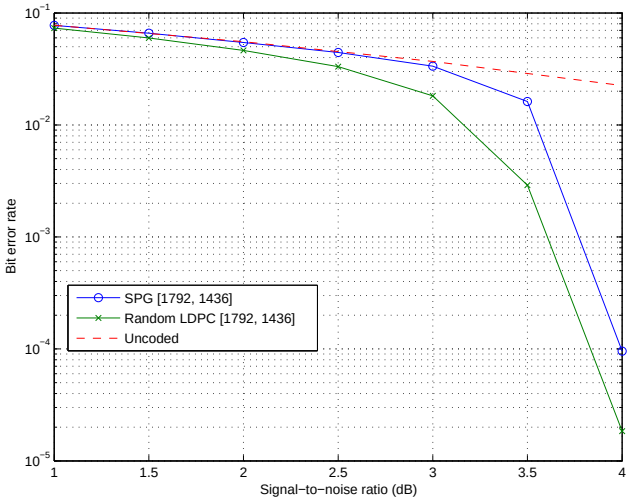

Figure 5.8: The decoding performance of the SPG code $\mathcal{C}_{K_{4,8}}$ on an AWGN channel using log-likelihood iterative probabilistic decoding with a maximum of 10 iterations.

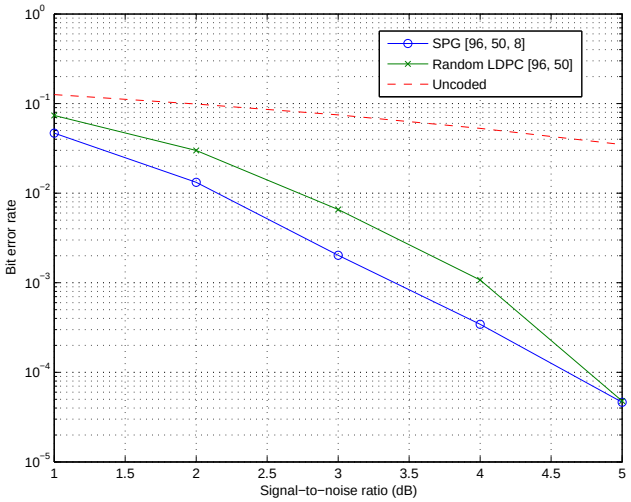

Figure 5.9: The decoding performance of the SPG code $\mathcal{C}_{K_{2,4}}$ on an AWGN channel using iterative probabilistic decoding with a maximum of 10 iterations.

whereas for values greater than 1.2dB the fact that the SPG matrix contains a large number of linearly dependent rows is a more important factor. This is in agreement with the analysis presented above regarding $\mathcal{C}_{K_{4,4}}$.

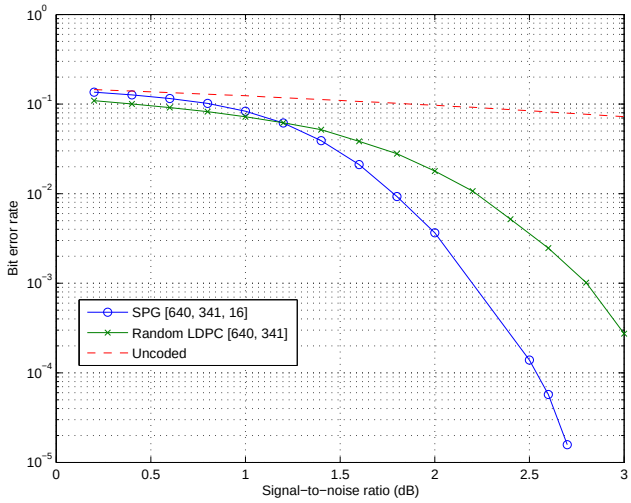

Figure 5.10: The decoding performance of the SPG code $\mathcal{C}_{K_{2,8}}$ on an AWGN channel using iterative probabilistic decoding with a maximum of 10 iterations.

5.5 The codes $\mathcal{C}_{Q_q^-}$

The codes $\mathcal{C}_{Q_2^-}$, $\mathcal{C}_{Q_3^-}$ and $\mathcal{C}_{Q_4^-}$ were tested.

5.5.1 $\mathcal{C}_{Q_2^-}$

The code $\mathcal{C}_{Q_2^-}$ is a (2, 5)-regular [40, 25, 4] code of rate 0.625. Its parity-check matrix is a 16×40 matrix with 2-rank $40 - 25 = 15$.

The performance of $\mathcal{C}_{Q_2^-}$ was compared with a randomly constructed LDPC code of the same rate and length. Its parity-check matrix was a 15×40 matrix with column weight 3 and row weights that ranged between 1 and 26. A Magma computation showed that the matrix had full 2-rank. The matrix contained some 4-cycles.

The results are shown in Figure 5.11. The SPG code performed better than the random code. This can be attributed to the following factors:

- The Tanner graph of the randomly generated code contains some 4-cycles, whereas the Tanner graph of $\mathcal{C}_{Q_2^-}$ has girth 8 (Result 4.6).
- The parity-check matrix of $\mathcal{C}_{Q_2^-}$ is less dense (1/8 compared to 1/5 for the random code).
- The parity-check matrix of $\mathcal{C}_{Q_2^-}$ contains a linearly dependent row.
- The minimum distance of $\mathcal{C}_{Q_2^-}$ is 4, which by Lemma 3.1 is the largest possible value for a code derived from the linear representation of a set of points in PG$(3, 2)$.

The most significant of these reasons is likely to be the discrepancy in the matrix densities and therefore another test was performed in order to compare $\mathcal{C}_{Q_2^-}$ with a randomly constructed code with a matrix density as close to that of $\mathcal{C}_{Q_2^-}$ as possible. In this second simulation the parity-check matrix of the random code was a 15×40 matrix with 38 columns of weight 2 and 2 columns of weight 3. The row weights ranged between 3 and 8. A Magma computation showed that the matrix had full 2-rank. The matrix contained no 4 cycles.

The results are shown in Figure 5.12, on which both the randomly generated codes have been plotted. As can be seen the second random code improves upon the performance of the first, showing that for very short codes

such as these a column weight of 2 is preferable to a column weight of 3. However the second random code still performs less well than $\mathcal{C}_{Q_2^-}$. One reason for this is likely to the linearly dependent row in the parity-check matrix of the SPG code. Another reason is probably the regularity of the row weights of the SPG code.

5.5.2 $\mathcal{C}_{Q_3^-}$

The code $\mathcal{C}_{Q_3^-}$ is a (3, 10)-regular [270, 189, 6] code of rate 0.7. Its parity-check matrix is a 81×270 matrix of full 2-rank.

The performance of $\mathcal{C}_{Q_3^-}$ was compared with a randomly constructed LDPC code whose parity-check matrix was an 81×270 matrix with column weight 3 and row weights that ranged between 5 and 20. A Magma computation showed that the matrix had full 2-rank. The matrix contained no 4-cycles.

The results are shown in Figure 5.13. The code $\mathcal{C}_{Q_3^-}$ performs better than the randomly generated code for all values of the signal-to-noise ratio. One reason for this is that the Tanner graph of $\mathcal{C}_{Q_3^-}$ has girth 8 (Result 4.6) and is therefore free of 4-cycles and 6-cycles, whereas the Tanner graph of the randomly constructed code is likely to contain a large number of 6-cycles. We note also that the minimum distance of $\mathcal{C}_{Q_3^-}$ is 6, which by Lemma 3.1 is the largest possible value for a code derived from the linear representation of a set of points in $\mathrm{PG}(2, 3)$.

5.5.3 $\mathcal{C}_{Q_4^-}$

The code $\mathcal{C}_{Q_4^-}$ is a (4,17)-regular [1088, 861, 8] code of rate $861/1088 \approx 0.791$. Its parity-check matrix is a 256×1088 matrix with 2-rank $1088 - 861 = 227$.

The performance of $\mathcal{C}_{Q_4^-}$ was compared with a randomly constructed LDPC code whose parity-check matrix was an 227×1088 matrix with column weight 3 and row weights that ranged between 6 and 25. A Magma computation showed that the matrix had full 2-rank. The matrix contained no 4-cycles.

Because of the relatively long length of these codes the log-likelihood version of the decoding algorithm was used to perform the simulations.

The results are shown in Figure 5.14. We see that $\mathcal{C}_{Q_4^-}$ performs well compared to the random code. This may be attributed to the following

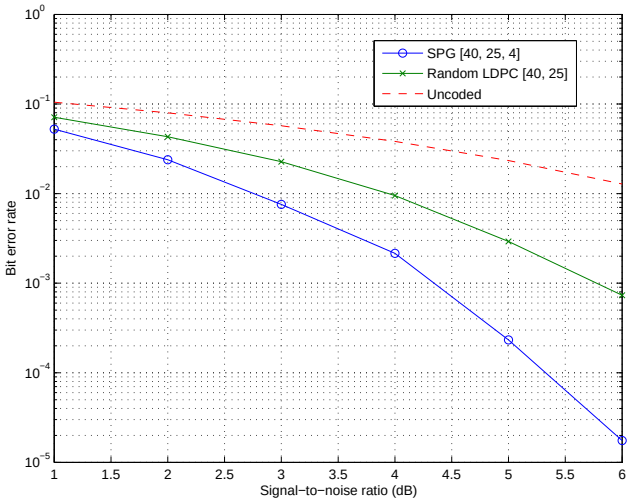

Figure 5.11: The decoding performance of the SPG code $\mathcal{C}_{Q_2^-}$ on an AWGN channel using iterative probabilistic decoding with a maximum of 10 iterations. The random code has column weight 3.

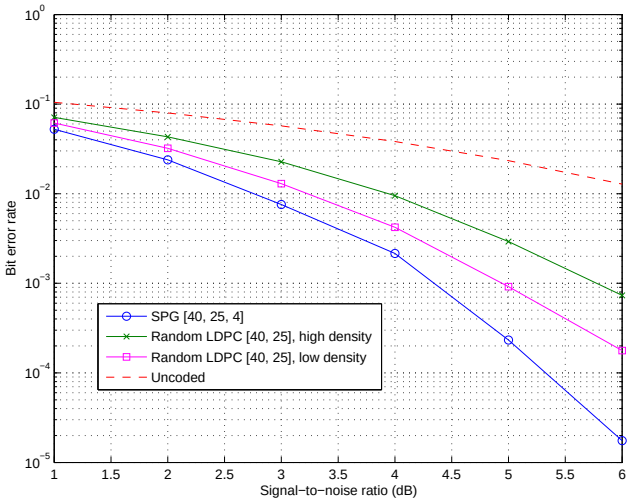

Figure 5.12: The decoding performance of the SPG code $\mathcal{C}_{Q_2^-}$ on an AWGN channel using iterative probabilistic decoding with a maximum of 10 iterations.

factors:

- The Tanner graph of $\mathcal{C}_{Q_4^-}$ has girth 8 and is therefore free of 6-cycles.
- The parity-check matrix of $\mathcal{C}_{Q_4^-}$ contains linearly dependent rows.
- The minimum distance of $\mathcal{C}_{Q_4^-}$ is 8, which by Lemma 3.1 is the largest possible value for a code derived from the linear representation of a set of points in PG$(3, 4)$.

These factors outweigh the slightly lower density of the randomly-generated matrix ($3/227 \approx 0.0132$ as opposed to $4/256 = 0.015625$ for the SPG code).

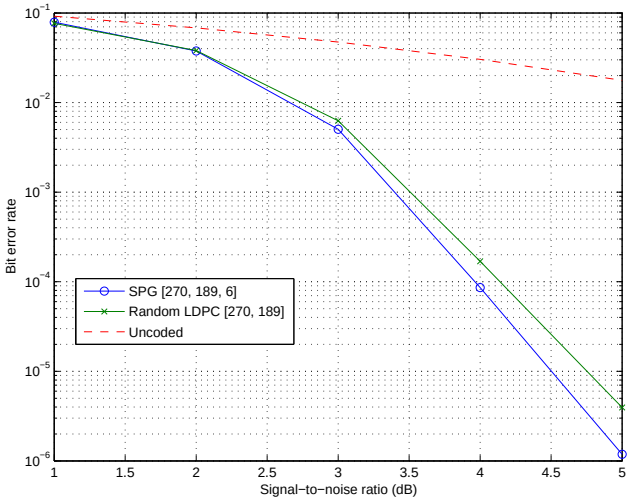

Figure 5.13: The decoding performance of the SPG code $\mathcal{C}_{Q_3^-}$ on an AWGN channel using iterative probabilistic decoding with a maximum of 10 iterations.

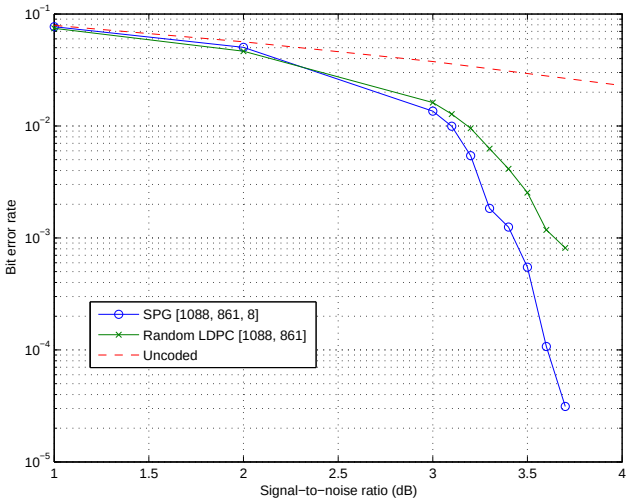

Figure 5.14: The decoding performance of the SPG code $\mathcal{C}_{Q_4^-}$ on an AWGN channel using log-likelihood iterative probabilistic decoding with a maximum of 10 iterations.

5.6 The code $\mathcal{C}_{TO_8}$

The first value of q for which a Tits ovoid exists is $q = 8$. The code $\mathcal{C}_{TO_8}$ is an (8, 65)-regular [33280, 29722, 16] code of rate $29722/33280 \approx 0.893$. Its parity-check matrix is a 4096×33280 matrix with 2-rank $33280 - 29722 = 3558$.

The next shortest code in the family $\mathcal{C}_{TO_q}$ is $\mathcal{C}_{TO_{32}}$, which has length 33587200 and was therefore too long to test using the available software.

The performance of $\mathcal{C}_{TO_8}$ was compared with a randomly constructed LDPC code of the same rate and length. Its parity-check matrix was a 3558×33280 matrix with column weight 3 and row weights that ranged between 14 and 46. A Magma computation showed that the matrix had full 2-rank.

Because of the length of these codes the log-likelihood version of the decoding algorithm was used to perform the simulations. The results are shown in Figure 5.15.

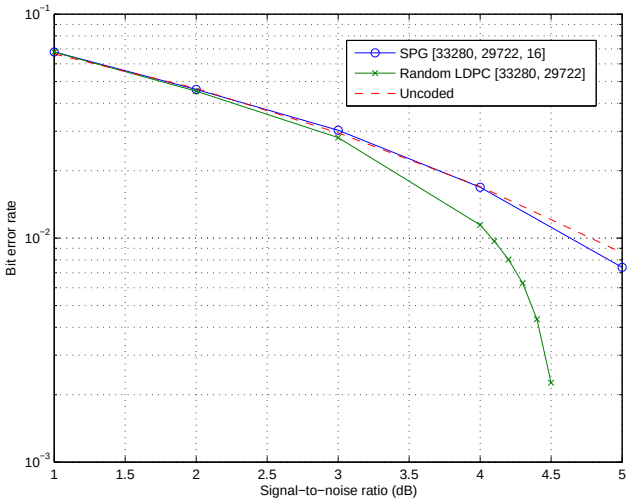

Figure 5.15: The decoding performance of the SPG code $\mathcal{C}_{TO_8}$ on an AWGN channel using log-likelihood iterative probabilistic decoding with a maximum of 10 iterations.

In this simulation $\mathcal{C}_{TO_8}$ performed poorly, hardly improving on the bit error rate of the uncoded signal. By contrast the randomly constructed code

performed well. The probable explanation for this is the greater density of the parity-check matrix of $\mathcal{C}_{TO_8}$ compared to the random code: the parity-check matrix of $\mathcal{C}_{TO_8}$ has density $8/4096 = 1/512$ whereas the random matrix has density $3/3558 = 1/1186$.

5.7 The code $\mathcal{C}_{C_{11}}$

The code $\mathcal{C}_{C_{11}}$ is a (3, 11)-regular [891, 648, 6] code of rate $648/891 \approx 0.727$. Its parity-check matrix is a 243×891 matrix of full 2-rank.

The performance of $\mathcal{C}_{C_{11}}$ was compared with a randomly constructed LDPC code of the same rate and length. Its parity-check matrix was a 243×891 matrix with column weight 3 and row weights that ranged between 4 and 22. A Magma computation showed that the matrix had full 2-rank.

Because of the relatively long length of these codes the log-likelihood version of the decoding algorithm was used to perform the simulations.

The results are shown in Figure 5.16. The performance of the SPG code and the random code were similar. This was perhaps to be expected as the parity-check matrices both had density $3/243 = 1/81$ and they both had column weight equal to 3.

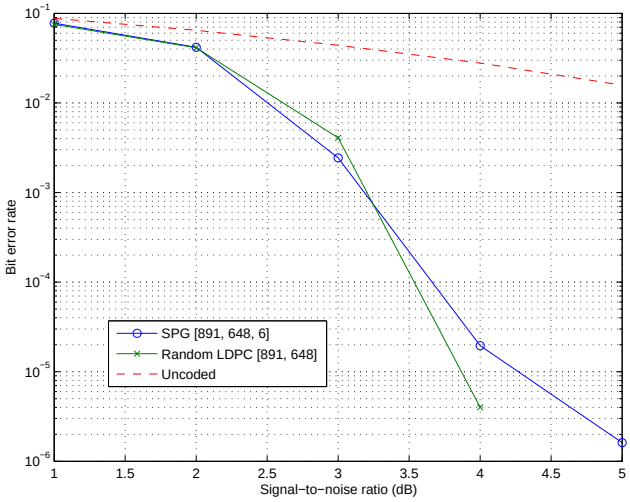

Figure 5.16: The decoding performance of the SPG code $\mathcal{C}_{C_{11}}$ on an AWGN channel using log-likelihood iterative probabilistic decoding with a maximum of 10 iterations.

5.8 The code $\mathcal{C}_{C_{56}}$

The code $\mathcal{C}_{C_{56}}$ is a (3, 56)-regular [13608, 12879, $4 \leq d_{min} \leq 6$] code of rate $12879/13608 \approx 0.946$. Its parity-check matrix is a 729×13608 matrix of full 2-rank.

The performance of $\mathcal{C}_{C_{56}}$ was compared with a randomly constructed LDPC code of the same rate and length. Its parity-check matrix was a 729×13608 matrix with column weight 3 and row weights that ranged between 37 and 85. A Magma computation showed that the matrix had full 2-rank.

Because of the relatively long length of these codes the log-likelihood version of the decoding algorithm was used to perform the simulations. The results are shown in Figure 5.17.

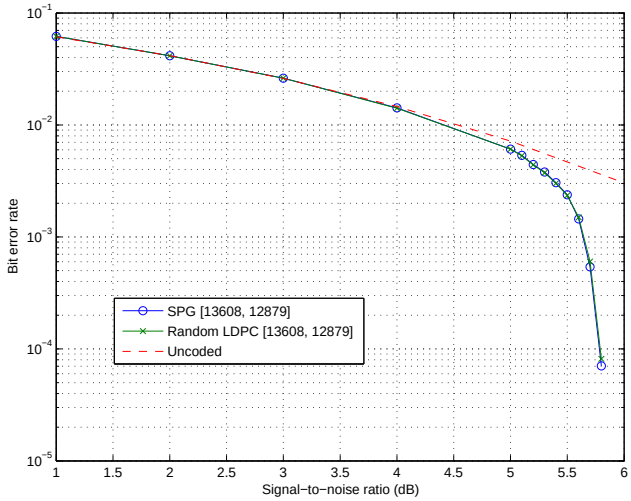

Figure 5.17: The decoding performance of the SPG code $\mathcal{C}_{C_{56}}$ on an AWGN channel using log-likelihood iterative probabilistic decoding with a maximum of 10 iterations. The performance of $\mathcal{C}_{C_{56}}$ and the random code were extremely similar over this range of signal-to-noise ratios.

The performance of $\mathcal{C}_{C_{56}}$ and the random code were extremely similar. The density of the parity-check matrices for $\mathcal{C}_{C_{56}}$ and the randomly constructed code are both 1/243, both matrices have column weight equal to 3, and both matrices have full 2-rank. These factors would seem to explain the

similar performance of the two codes.

5.9 The code $\mathcal{C}_{C_{78}}$

The code $\mathcal{C}_{C_{78}}$ is a (4, 78)-regular code of length $n = 79872$. Unfortunately it was not possible to simulate the performance of this code because of its length.

5.10 The codes $\mathcal{C}_{U_{2,q}2}$

The codes $\mathcal{C}_{U_{2,4}}$ and $\mathcal{C}_{U_{2,9}}$ were tested.

5.10.1 $\mathcal{C}_{U_{2,4}}$

The code $\mathcal{C}_{U_{2,4}}$ is a (4,9)-regular [144, 96, 6] code of rate 2/3. Its parity-check matrix is a 64×144 matrix of 2-rank $144 - 96 = 48$.

The performance of $\mathcal{C}_{U_{2,4}}$ was compared with a randomly constructed LDPC code of the same rate and length. Its parity-check matrix was a 48×144 matrix of column weight 3 and row weights that ranged between 4 and 23. A Magma computation showed that the matrix had full 2-rank.

The results are shown in Figure 5.18.

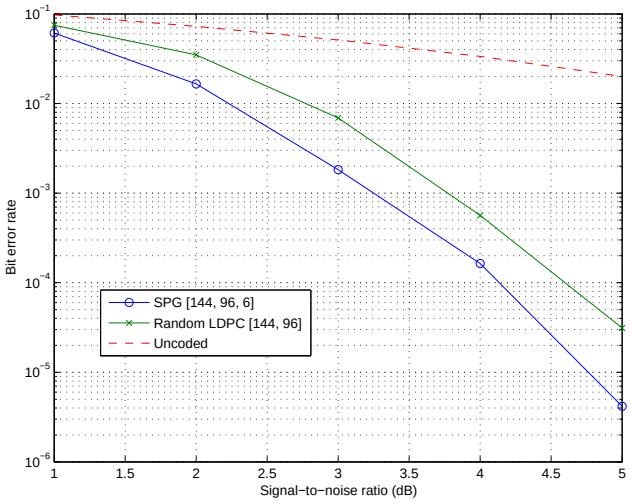

Figure 5.18: The decoding performance of the SPG code $\mathcal{C}_{U_{2,4}}$ on an AWGN channel using iterative probabilistic decoding with a maximum of 10 iterations.

The code $\mathcal{C}_{U_{2,4}}$ performed better than the randomly generated code. In this case the parity-check matrices for both the SPG code and the randomly generated code have density 1/16. However the parity-check matrix of the random code contained some 4-cycles. In addition the parity-check matrix of the SPG code contains linearly dependent rows.

5.10.2 $\mathcal{C}_{U_{2,9}}$

The code $\mathcal{C}_{U_{2,9}}$ is a (9,28)-regular [2268, 1539] code of rate $1539/2268 \approx 0.679$. Its parity-check matrix is a 729×2268 matrix of full 2-rank. By Result 4.11 we have $d_{min} = 12, 14, 16$ or 18.

The performance of $\mathcal{C}_{U_{2,9}}$ was compared with a randomly constructed LDPC code of the same rate and length. Its parity-check matrix was a 729×2268 matrix with 2267 columns of weight 3 and one column of weight 4. The row weights ranged between 2 and 20. A Magma calculation showed that the matrix had full 2-rank.

Because of the relatively long length of these codes the log-likelihood version of the decoding algorithm was used to perform the simulations. The results are shown in Figure 5.19.

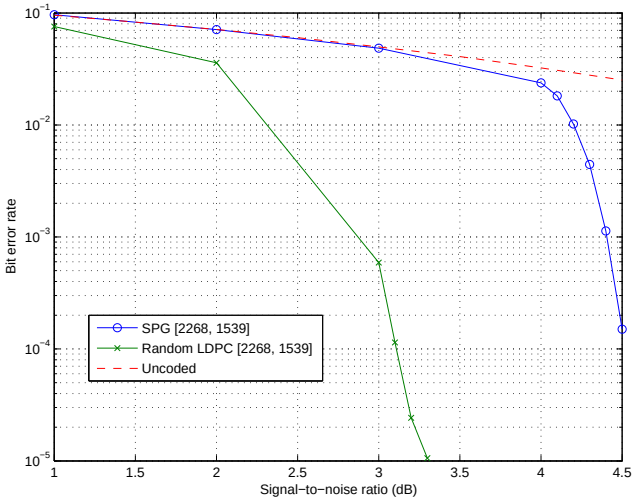

Figure 5.19: The decoding performance of the SPG code $\mathcal{C}_{U_{2,9}}$ on an AWGN channel using log-likelihood iterative probabilistic decoding with a maximum of 10 iterations.

In this case the randomly generated code performed better than the SPG code. We note that the density of the random matrix is approximately 1/243, compared to 1/81 for the SPG code. The greater column weight and density of the SPG code gives rise to a higher number of short cycles in the Tanner graph and degrades the performance of the code.

5.11 The codes $\mathcal{C}_{B_{2,q^2}}$

The codes $\mathcal{C}_{B_{2,4}}$ and $\mathcal{C}_{B_{2,9}}$ were tested.

5.11.1 $\mathcal{C}_{B_{2,4}}$

The code $\mathcal{C}_{B_{2,4}}$ is a (4, 7)-regular [112, 67, 6] code of rate 0.598. Its parity-check matrix is a 64×112 matrix of 2-rank $112 - 67 = 45$.

The performance of $\mathcal{C}_{B_{2,4}}$ was compared with a randomly constructed LDPC code of the same rate and length. Its parity-check matrix was a 45×112 matrix of column weight 3 and row weights that ranged between 2 and 14. A Magma computation showed that the matrix had full 2-rank.

The results are shown in Figure 5.20, which shows that $\mathcal{C}_{B_{2,4}}$ performed better than the random code. This can be attributed to two factors. Firstly the parity-check matrix of the SPG code has density 1/16 compared to a density of 1/15 for the random code. Secondly the parity-check matrix for the SPG code contains linearly dependent rows.

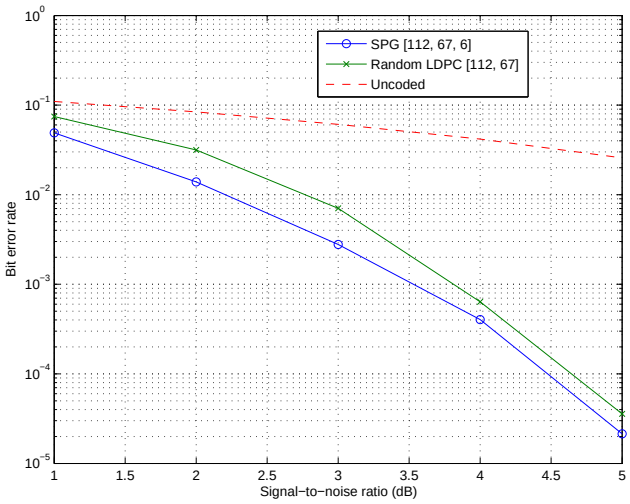

Figure 5.20: The decoding performance of the SPG code $\mathcal{C}_{B_{2,4}}$ on an AWGN channel using iterative probabilistic decoding with a maximum of 10 iterations.

5.11.2 $\mathcal{C}_{B_{2,9}}$

The code $\mathcal{C}_{B_{2,9}}$ is a (9, 13)-regular [1053, 324] code of rate 0.308. Its parity-check matrix is a 729×1053 matrix of full 2-rank. By result 4.15 we have $d_{min} = 12, 14$ or 16.

The performance of $\mathcal{C}_{B_{2,9}}$ was compared with a randomly constructed LDPC code of the same rate and length. Its parity-check matrix was a 729×1053 matrix with 985 columns of weight 3, 65 columns of weight 4 and 3 columns of weight 5. The row weights ranged between 2 and 12. A Magma calculation showed that the matrix had full 2-rank.

Because of the relatively long length of these codes the log-likelihood version of the decoding algorithm was used to perform the simulations. The results are shown in Figure 5.21.

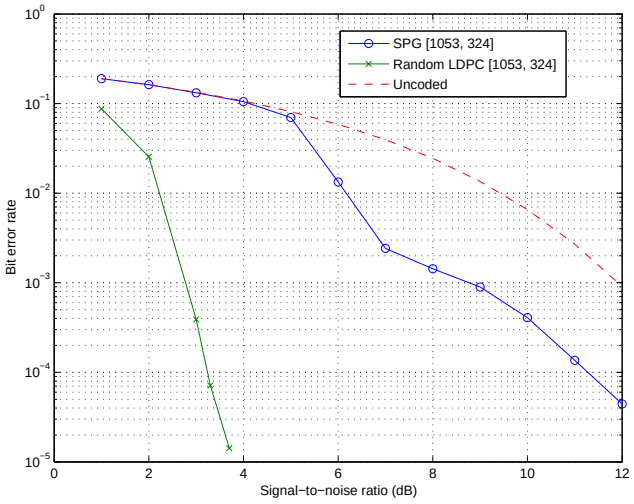

Figure 5.21: The decoding performance of the SPG code $\mathcal{C}_{B_{2,9}}$ on an AWGN channel using log-likelihood iterative probabilistic decoding with a maximum of 10 iterations.

The random code performed better than the SPG code. Again this is due to the fact that the parity-check matrix of the random code has a much lower density (≈ 0.0042) than the parity-check matrix of the SPG code (≈ 0.0123).

Chapter 6

Other Semipartial Geometries

In this section we discuss some examples of semipartial geometries that do not arise from SPG-reguli, and their associated codes. The examples are taken from [3, Chapter 10].

6.1 The codes $\mathcal{C}_{U_{2,3}(m)}$

Let $U_m = \{1, 2, 3, \ldots, m\}$, $m \geq 4$. Let $\mathcal{P}$ be the set of pairs of U_m, let $\mathcal{B}$ be the set of unordered triples of U_m, and let I be the inclusion relation. Then $U_{2,3}(m) = (\mathcal{P}, \mathcal{B}, \mathrm{I})$ is an $\mathrm{spg}(2, m-3, 2, 4)$. See [3].

It is usual to denote these semipartial geometries by $\mathcal{C}_{U_{2,3}(n)}$. However we shall use n to denote the length of the resulting codes, and we therefore denote the codes by $\mathcal{C}_{U_{2,3}(m)}$ to avoid confusion.

We note that if $m > 4$ $U_{2,3}(m)$ is a proper semipartial geometry since $\mu \neq (t+1)\alpha$ for all $m > 4$.

6.1.1 The matrix H

For all $m \geq 4$ we obtain an SPG code $\mathcal{C}_{U_{2,3}(m)}$ for which $v = \binom{m}{2}$, $b = n = \binom{m}{3}$, $\gamma = 3$ and $\rho = m - 2$. We have $H_d = \frac{6}{m(m-1)}$.

6.1.2 Girth

Since $\alpha = 2$ the Tanner graph contains 6-cycles. From Lemma 1.1 we have

$$N_6 = \frac{bt(\alpha-1)}{3}\binom{s+1}{2} = (m^4 - 6m^3 + 11m^2 - 6m)/6.$$

6.1.3 Minimum distance and codeword weights

We have the following Lemmas.

Lemma 6.1. *For all $m \geq 4$ the minimum distance of the code $\mathcal{C}_{U_{2,3}(m)}$ is 4 and $\mathcal{C}_{U_{2,3}(m)}$ contains $\binom{m}{4}$ minimum weight codewords.*

Proof. A codeword in $\mathcal{C}_{U_{2,3}(m)}$ corresponds to a subset $L \subset \mathcal{L}$ such that every set in $\mathcal{P}$ is a subset of an even number of the sets in L. Consider the set $L = \{\{1,2,3\},\{1,2,4\},\{1,3,4\},\{2,3,4\}\} \subset \mathcal{L}$. Every set in $\mathcal{P}$ is a subset of 0 or 2 sets in L. Hence $\mathcal{C}_{U_{2,3}(m)}$ contains a codeword of length 4 and $d_{min} \geq 4$. It is clear that there can be no smaller subset of $\mathcal{L}$ with this property. Hence $d_{min} = 4$.

We can choose L in $\binom{m}{4}$ different ways, each corresponding to a different minimum weight codeword. □

Lemma 6.2. *All codewords in $\mathcal{C}_{U_{2,3}(m)}$ have even weight.*

Proof. Let $L \subset \mathcal{L}$ correspond to a codeword and let $t = |L|$. Each set in L contains 3 pairs, which means that counting repetitions there are $3t$ pairs contained in the sets of L. Since L corresponds to a codeword any pair must occur an even number of times, which means that the $3t$ pairs can be split into sets of even size. Hence $2 \mid 3t$ and t is even. □

The following lemmas give us information about the weight distribution of the codes $\mathcal{C}_{U_{2,3}(m)}$.

Lemma 6.3. *Let $\mathcal{C} = \mathcal{C}_{U_{2,3}(m)}$, m even. If $\mathcal{C}$ contains x codewords of weight t, $0 \leq t \leq n$ then it contains x codewords of weight $n - t$.*

Proof. $\mathcal{L}$ contains $\binom{m}{3}$ sets and each of these sets contains 3 pairs, which means that counting repetitions there are $3 \times \binom{m}{3}$ pairs contained in the sets of $\mathcal{L}$. Therefore each set in $\mathcal{P}$ is contained in $\frac{3\times\binom{m}{3}}{\binom{m}{2}} = m - 2$ sets of $\mathcal{L}$. Now consider a subset $L \subset \mathcal{L}$ corresponding to a codeword of weight $t = |L|$. Each set in $\mathcal{P}$ is contained in an even number of the sets in L. Since each set in $\mathcal{P}$ is contained in $m - 2$ sets of $\mathcal{L}$, and $m - 2$ is even, it follows that each set of $\mathcal{P}$ is contained in an even number of the sets in $\mathcal{L}\backslash L$. Hence $\mathcal{L}\backslash L$ corresponds to a codeword of weight $|\mathcal{L}| - |L| = n - t$. □

Corollary 6.1. *For m even $\mathcal{C}_{U_{2,3}(m)}$ contains the all 1s codeword.*

Lemma 6.4. *Let $\mathcal{C} = \mathcal{C}_{U_{2,3}(m)}$, m odd. Then the complement of any codeword of $\mathcal{C}$ is not a codeword.*

Proof. As for Lemma 6.4. In this case $m-2$ is odd and so each set of $\mathcal{P}$ is contained in an odd number of the sets in $\mathcal{L}\backslash L$. Hence $\mathcal{L}\backslash L$ cannot correspond to a codeword. □

6.1.4 2-rank of H

Let H_m denote the parity-check matrix of the code $\mathcal{C}_{U_{2,3}(m)}$, $m \geq 4$.

We have the following Theorem.

Theorem 6.1. *For $m \geq 4$ the 2-rank of H_m is $\binom{m-1}{2}$.*

Proof. For $m = 4$ we have

$$H_4 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}.$$

By inspection we see that $\text{rank}_2(H_4) = 3 = \binom{3}{2}$.

For $m \geq 5$ we note that H_m has the following structure:

$$H_m = \begin{bmatrix} H_{m-1} & I_{\binom{m-1}{2}} \\ O & A \end{bmatrix}.$$

Here $I_{\binom{m-1}{2}}$ is the $\binom{m-1}{2} \times \binom{m-1}{2}$ identity matrix, O is an $(m-1) \times \binom{m-1}{3}$ all-zero matrix, and A is a $(2, m-2)$-regular $(m-1) \times \binom{m-1}{2}$ matrix. The rows of H_m are arranged with the last $m-1$ rows corresponding to pairs that contain m. Similarly the last $\binom{m-1}{2}$ columns correspond to the triples containing m.

By inspection the first $\binom{m-1}{2}$ rows of H_m are linearly independent, so $\text{rank}_2(H_m) \geq \binom{m-1}{2}$. Since any of the last $m-1$ rows of H_m is a linear combination of $m-2$ of the first $\binom{m-1}{2}$ rows of H_m it follows that $\text{rank}_2(H_m) = \binom{m-1}{2}$.

The matrix H_5 is shown in Table 6.1 as an illustration.

□

Table 6.1: H_5

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

6.1.5 Code dimension

The dimension of $\mathcal{C}_{U_{2,3}(m)}$ follows from the last section.

Corollary 6.2. *The dimension of* $\mathcal{C}_{U_{2,3}(m)}$ *is* $\binom{m-1}{3}$.

Proof. The dimension is given by

$$k = n - \text{rank}_2(H_m) = \binom{m}{3} - \binom{m-1}{2} = \binom{m-1}{3}.$$

□

We note that this means that the dimension of $\mathcal{C}_{U_{2,3}(m)}$ is the same as the length of $\mathcal{C}_{U_{2,3}(m-1)}$, $m \geq 5$.

Table 6.2 shows the dimension for small values of m.

6.2 The code $\mathcal{C}_{\overline{M(7)}}$

Three sporadic SPG codes can be obtained from the *Moore graphs* $M(2), M(3)$ and $M(7)$. Moore graphs are graphs with valency $k > 1$, girth 5 and with the minimum number of vertices, which is $k^2 + 1$. It is known that necessarily $k \in \{2, 3, 7, 57\}$; however a Moore graph with $k = 57$ is not known to exist.

$M(2)$ is a trivial cycle graph of length 5. $M(3)$ is known as the Petersen graph. $M(7)$ is known as the Hoffman-Singleton graph.

For each of the Moore graphs Γ there is an associated semipartial geometry, denoted by $\overline{M(k)}$. $\mathcal{P}$ is the set of vertices Γ and $\mathcal{B}$ is the set $\{\Gamma(x) : x \in \mathcal{P}\}$, where $\Gamma(x)$ is the set of vertices adjacent to x. I is the natural incidence relation. $\overline{M(k)} = (\mathcal{P}, \mathcal{B}, \mathrm{I})$ is an $\mathrm{spg}(k-1, k-1, k-1, (k-1)^2))$. See [3].

For $k = 2, 3, 7$ we obtain an SPG code $\mathcal{C}_{\overline{M(k)}}$ for which $v = b = n = k^2+1$, $\gamma = \rho = k$. We have $H_d = \frac{k}{k^2+1}$.

Table 6.2: Dimension of $\mathcal{C}_{U_{2,3}(m)}$

m	length n	dimension k
5	10	4
6	20	10
7	35	20
8	56	35
9	84	56
10	120	84
11	165	120
12	220	165
13	286	220
14	364	286
15	455	364
16	560	455
17	680	560
18	816	680
19	969	816
20	1140	969

The length of the SPG-code $\mathcal{C}_{\overline{M(2)}}$ is 5 and the length of $\mathcal{C}_{\overline{M(3)}}$ is 10. These codes are too short to be of interest and we therefore restrict our attention to the code $\mathcal{C}_{\overline{M(7)}}$.

6.2.1 The matrix H

The code $\mathcal{C}_{\overline{M(7)}}$ has length $n = 50$ and $H_d = 7/50$.

A parity-check matrix H was constructed. This was achieved by labeling vertices from 0 to 49 and listing the corresponding blocks $\Gamma(0)$ to $\Gamma(49)$ (the blocks were taken from [20]), and then using Magma to construct the incidence matrix. The points and blocks are shown in Table 6.3.

6.2.2 Girth

Since $\alpha = 6$ the Tanner graph of $\mathcal{C}_{\overline{M(7)}}$ contains 6-cycles. Using Lemma 1.1 we find that the Tanner graph contains 10500 6-cycles.

6.2.3 Minimum distance and codeword weights

Using Magma, the minimum distance of $\mathcal{C}_{\overline{M(7)}}$ was found to be 8. The weight distribution of the code is shown in Table 6.4.

6.2.4 2-rank of H and code dimension

Using Magma the 2-rank of H was found to be 22. The code dimension is $50 - 22 = 28$.

6.3 The codes $\mathcal{C}_{\mathrm{LP}(n,q)}$

Define $\mathcal{P}$ as the set of lines of $\mathrm{PG}(n,q)$ ($n \geq 4$), $\mathcal{B}$ as the set of planes of $\mathrm{PG}(n,q)$, and I as the inclusion relation. Then $\mathrm{LP}(n,q) = (\mathcal{P}, \mathcal{B}, \mathrm{I})$ is an $\mathrm{spg}(q(q+1), \frac{q^{n-1}-1}{q-1} - 1, q+1, (q+1)^2)$. See [3].

Table 6.3: Points and lines of $\overline{M(7)}$

x	$\Gamma(x)$		x	$\Gamma(x)$
0	{1,11, 14, 26, 38, 46, 49}		25	{2,7,16,24,26,33,37}
1	{0,2, 6 ,19, 22 ,40,43}		26	{0,4,21,25,27,30,35}
2	{1,3,9,13,25,31,47}		27	{9,15,19,26,28,32,45}
3	{2,4,11,18,34,42,45}		28	{13,24,27,29,34,40,49}
4	{3,5,8,23,26,40,48}		29	{8,11,22,28,30,37,47}
5	{4,6,13,17,32,37,46}		30	{6,18,26,29,31,41,44}
6	{1,5,7,10,15,30,34}		31	{2,20,23,30,32,36,49}
7	{6,8,12,25,39,45,49}		32	{5,11,27,31,33,39,43}
8	{4,7,9,14,20,29,43}		33	{14,22,25,32,34,41,48}
9	{2,8,10,17,27,38,41}		34	{3,6,20,28,33,35,38}
10	{6,9,11,21,24,36,48}		35	{12,17,26,34,36,43,47}
11	{0,3,10,12,16,29,32}		36	{10,14,31,35,37,40,45}
12	{7,11,13,19,23,35,41}		37	{5,19,25,29,36,38,42}
13	{2,5,12,14,21,28,44}		38	{0,9,23,34,37,39,44}
14	{0,8,13,15,18,33,36}		39	{7,18,21,32,38,40,47}
15	{6,14,16,23,27,42,47}		40	{1,4,16,28,36,39,41}
16	{11,15,17,20,25,40,44}		41	{9,12,30,33,40,42,46}
17	{5,9,16,18,22,35,49}		42	{3,15,21,37,41,43,49}
18	{3,14,17,19,24,30,39}		43	{1,8,24,32,35,42,44}
19	{1,12,18,20,27,37,48}		44	{13,16,30,38,43,45,48}
20	{8,16,19,21,31,34,46}		45	{3,7,22,27,36,44,46}
21	{10,13,20,22,26,39,42}		46	{0,5,20,24,41,45,47}
22	{1,17,21,23,29,33,45}		47	{2,15,29,35,39,46,48}
23	{4,12,15,22,24,31,38}		48	{4,10,19,33,44,47,49}
24	{10,18,23,25,28,43,46}		49	{0,7,17,28,31,42,48}

6.3.1 The matrix H

We have $v = \frac{(q^{n+1}-1)(q^n-1)}{(q^2-1)(q-1)}$, $b = n = \frac{(q^{n+1}-1)(q^n-1)(q^{n-1}-1)}{(q^3-1)(q^2-1)(q-1)}$, $\gamma = \frac{q^{n-1}-1}{q-1}$ and $\rho = q^2 + q + 1$.

We note that for $n = 4$ H is a square matrix (that is $v = b$). For $n \geq 5$ we have $b > v$.

The matrix density is

$$H_d = \frac{(q^3-1)^2(q^2-1)}{(q^{n+1}-1)(q^n-1)(q^{n-1}-1)}.$$

Table 6.4: Weight distribution of $\mathcal{C}_{\overline{M(7)}}$

Weight	Number of codewords
0	1
8	750
10	7770
12	52675
14	449550
16	2311575
18	8624700
20	22661835
22	42102850
24	57811425
26	58254700
28	42267025
30	22408050
32	8633100
34	2316300
36	470625
38	61950
40	525
42	50

6.3.2 Girth

Since $\alpha \geq 3$ the Tanner graph contains 6-cycles. From Lemma 1.1 we have

$$N_6 = \frac{bt(\alpha-1)}{3}\binom{s+1}{2} = \frac{q^3(q^{n+1}-1)(q^n-1)(q^{n-1}-1)(q^{n-2}-1)}{6(q-1)^4}.$$

6.3.3 Minimum distance

Lemma 1.3 gives $d_{min} \geq \frac{q^{n-1}-1}{q-1}+1$. This is a better bound than that provided by Lemma 1.4.

Magma was only able to calculate the minimum distance of $\mathcal{C}_{\mathrm{LP}(4,2)}$ and $\mathcal{C}_{\mathrm{LP}(4,3)}$. The results are displayed in Table 6.5. The lower bound of Lemma 1.3 is sharp for $\mathcal{C}_{\mathrm{LP}(4,2)}$ but not for $\mathcal{C}_{\mathrm{LP}(4,3)}$.

Table 6.5: Minimum distance of $\mathcal{C}_{\mathrm{LP}(n,q)}$

n	q	d_{min}
4	2	8
4	3	40

6.3.4 2-rank of H and dimension of $\mathcal{C}_{\mathrm{LP}(n,q)}$

Lemma 1.7 does not apply for any values of n and q.

The code dimension and 2-rank of H were calculated for small values of n and q using Magma. The results are displayed in Table 6.6.

Table 6.6: $\mathcal{C}_{\mathrm{LP}(n,q)}$: code dimension and 2-rank of H

n	q	dimension k	2-rank of H
4	2	79	76
4	3	120	1090
5	2	974	421

6.4 The codes $\mathcal{C}_{\overline{W(2n+1,q)}}$

Let σ be a symplectic polarity of PG$(2n+1, q), n \geq 1$. Let $\mathcal{P}$ be the point set of PG$(2n+1, q)$, $\mathcal{B}$ be the set of lines that are not totally isotropic with respect to σ, and I the incidence relation of PG$(2n+1, q)$. Then $\overline{W(2n+1, q)} = (\mathcal{P}, \mathcal{B}, \mathrm{I})$ is an spg$(q, q^{2n}-1, q, q^{2n}(q-1))$. See [3].

6.4.1 The matrix H

We have $v = (q^{2n+2}-1)/(q-1)$, $b = n = q^{2n}(q^{2n+2}-1)/(q^2-1)$, $\gamma = q+1$ and $\rho = q^{2n}$.

The matrix density is

$$H_d = (q^2-1)/(q^{2n+2}-1).$$

6.4.2 Girth

Since $\alpha \geq 2$ the Tanner graph contains 6-cycles. From Lemma 1.1 we have

$$N_6 = \frac{bt(\alpha-1)}{3}\binom{s+1}{2} = \frac{q^{2n+1}(q^{2n}-1)(q^{2n+2}-1)}{6}.$$

6.4.3 Minimum distance

Lemma 1.3 gives $d_{min} \geq q+2$. This is a better bound than that provided by Lemma 1.4.

Magma was only able to calculate the minimum distance of $\mathcal{C}_{\overline{W(2n+1,q)}}$ for small values of n and q. The results are displayed in Table 6.7.

Table 6.7: Minimum distance of $\mathcal{C}_{\overline{W(2n+1,q)}}$

n	q	d_{min}
1	2	4
1	3	10
1	4	6
2	2	4

We observe that for even values of q the lower bound of $q+2$ provided by Lemma 1.3 is sharp in all cases.

6.4.4 2-rank of H and dimension of $\mathcal{C}_{\overline{W(2n+1,q)}}$

Lemma 1.7 does not apply since

$$s + t\alpha + t + 1 - \mu = 2q^{2n} \equiv 0 \bmod 2.$$

The code dimension and 2-rank of H were calculated for small values of n and q using Magma. The results are displayed in Table 6.8.

Table 6.8: $\mathcal{C}_{\overline{W(2n+1,q)}}$: code dimension and 2-rank of H

n	q	dimension k	2-rank of H
1	2	10	10
1	3	51	39
1	4	212	60
1	5	495	155
1	7	2051	399
1	8	3760	400
1	9	6642	5823
2	2	336	280
2	3	7371	7008

Based on the results in Table 6.8 we make the following conjecture for odd values of q.

Conjecture 6.1. *For q odd*

- $\text{rank}_2(H) = v - 1 = \frac{q^{2n+2}-1}{q-1} - 1$;
- *The dimension of $\mathcal{C}_{\overline{W(2n+1,q)}}$ is $b - v + 1 = \frac{(q^{2n+2}-1)(q^{2n}-q-1)}{q^2-1} + 1$.*

We note that for odd q all the rows in H sum to zero since $\gamma = q + 1$ is even, which shows that $\text{rank}_2(H) < v$, that is H does not have full 2-rank. It follows that Conjecture 6.1 is true if and only if it is not possible to find a non-empty proper subset $\overline{R}$ of the set of rows R of a parity-check matrix of $\mathcal{C}_{\overline{W(2n+1,q)}}$, q odd, such that the the rows in $\overline{R}$ sum to zero. We note that if such a set $\overline{R}$ were to exist then:

- $\overline{R}$ must contain an even number of rows (by a counting argument, since the row weight is q^{2n} which is odd).

- The vectors in $R \backslash \overline{R}$ would also sum to zero. This means that there would exist a set $\overline{R}$ with $|\overline{R}| \leq v/2$.
- $\overline{R}$ would have to contain at least $q^{2n} + 1$ rows, since the row weight is q^{2n}.

6.5 The codes $\mathcal{C}_{NQ^+(2n-1,2)}$

Let $Q^+(2n-1,2)$ be a non-singular hyperbolic quadric in PG$(2n-1,2)$, $n \geq 3$. Let $\mathcal{P}$ be the set of points of PG$(2n-1,2)$ that do not lie on $Q^+(2n-1,2)$, let $\mathcal{L}$ be the set of lines of PG$(2n-1,2)$ that do not intersect $Q^+(2n-1,2)$, and let I be the incidence relation of PG$(2n-1,2)$. Then $NQ^+(2n-1,2) = (\mathcal{P}, \mathcal{B}, \mathrm{I})$ is an spg$(2, 2^{2n-3}-2^{n-2}-1, 2, 2^{2n-3}-2^{n-1})$. See [3].

6.5.1 The matrix H

We have $v = 2^{2n} - 1 - (2^{n-1}+1)(2^n-1) = 2^{n-1}(2^n-1)$. Therefore $b = v(t+1)/(s+1) = 2^{n-1}(2^n-1)(2^{2n-3}-2^{n-2})/3 = 2^{2n-3}(2^n-1)(2^{n-1}-1)/3$. Also $\gamma = 3$ and $\rho = 2^{2n-3}-2^{n-2}$. The matrix density is $H_d = 3/(2^{n-1}(2^n-1))$.

6.5.2 Girth

Since $\alpha = 2$ the Tanner graph contains 6-cycles. From Lemma 1.1 we have

$$N_6 = \frac{bt(\alpha-1)}{3}\binom{s+1}{2} = bt = 2^{2n-3}(2^n-1)(2^{n-1}-1)(2^{2n-3}-2^{n-2}-1)/3.$$

6.5.3 Minimum distance

We have the following Theorem.

Theorem 6.2. *The minimum distance of $\mathcal{C}_{NQ^+(2n-1,2)}$ is 4, for all $n \geq 3$.*

Proof. Suppose that the hyperbolic quadric is given by the canonical form (see [13, Theorem 5.16])

$$Q^+(2n-1,2) = X_0X_1 + X_2X_3 + \ldots + X_{2n-2}X_{2n-1}.$$

Then $P = (1,0,0,\ldots,0)$ lies on the quadric. Consider the plane Π defined by P and the points $Q = (0,0,1,1,0,\ldots,0)$ and $R = (0,0,0,1,1,1,0,\ldots,0)$. (For example in PG$(5,2)$ $P = (1,0,0,0,0,0)$, $Q = (0,0,1,1,0,0)$ and $R = (0,0,0,1,1,1)$. In PG$(7,2)$ $P = (1,0,0,0,0,0,0,0)$, $Q = (0,0,1,1,0,0,0,0)$ and $R = (0,0,0,1,1,1,0,0)$, and so on.) Then $\Pi \cap Q^+(2n-1,2) = \{P\}$. Let $L \subset \mathcal{L}$ be the set of the four lines of Π that do not contain P. Then every point of $\mathcal{P}$ lies on either 0 or 2 lines of L. Hence $\mathcal{C}_{NQ^+(2n-1,2)}$ contains a codeword of weight $|L| = 4$.

From Lemma 1.3 we have $d_{min} \geq 4$, from which it follows that $d_{min} = 4$. □

6.5.4 2-rank of H and dimension of $\mathcal{C}_{NQ^+(2n-1,2)}$

Lemma 1.7 does not apply since $s + t\alpha + t + 1 - \mu \equiv 0 \bmod 2$.

The code dimension and 2-rank of H could only be calculated for $n = 3$ and $n = 4$ using Magma. The results are displayed in Table 6.9.

Table 6.9: $\mathcal{C}_{NQ^+(2n-1,2)}$: code dimension and 2-rank of H

n	dimension k	2-rank of H
3	35	21
4	1008	112

6.6 The codes $\mathcal{C}_{NQ^-(2n-1,2)}$

Let $Q^-(2n-1,2)$ be a non-singular elliptic quadric in PG$(2n-1,2)$, $n \geq 2$. Let $\mathcal{P}$ be the set of points of PG$(2n-1,2)$ that do not lie on $Q^-(2n-1,2)$, let $\mathcal{L}$ be the set of lines of PG$(2n-1,2)$ that do not intersect $Q^-(2n-1,2)$, and let I be the incidence relation of PG$(2n-1,2)$. Then $NQ^-(2n-1,2) = (\mathcal{P}, \mathcal{B}, \text{I})$ is an spg$(2, 2^{2n-3}+2^{n-2}-1, 2, 2^{2n-3}+2^{n-1})$. See [3].

6.6.1 The matrix H

We have $v = 2^{2n} - 1 - (2^n+1)(2^{n-1}-1) = 2^{n-1}(2^n+1)$. Therefore $b = v(t+1)/(s+1) = 2^{n-1}(2^n+1)(2^{2n-3}+2^{n-2})/3 = 2^{2n-3}(2^n+1)(2^{n-1}+1)/3$. Also $\gamma = 3$ and $\rho = 2^{2n-3}+2^{n-2}$. The matrix density is $H_d = 3/(2^{n-1}(2^n+1))$.

6.6.2 Girth

Since $\alpha = 2$ the Tanner graph contains 6-cycles. From Lemma 1.1 we have

$$N_6 = \frac{bt(\alpha-1)}{3}\binom{s+1}{2} = bt = 2^{2n-3}(2^n+1)(2^{n-1}+1)(2^{2n-3}+2^{n-2}-1)/3.$$

6.6.3 Minimum distance

We have the following Theorem.

Theorem 6.3. *The minimum distance of $\mathcal{C}_{NQ^-(2n-1,2)}$ is 4, for all $n \geq 2$.*

Proof. Suppose that the elliptic quadric is given by the canonical form (see [13, Theorem 5.16])

$$Q^-(2n-1,2) = X_0^2 + X_0X_1 + X_1^2 + X_2X_3 + ... + X_{2n-2}X_{2n-1}.$$

Then the point $P = (0,0,1,0,\ldots,0)$ lies on the quadric. Consider the plane Π defined by P and the two points $Q = (1,0,0,\ldots,0)$ and $R = (0,1,0,\ldots,0)$. (For example in PG$(3,2)$ $P = (0,0,1,0)$, $Q = (1,0,0,0)$ and $R = (0,1,0,0)$.) Then $\Pi \cap Q^-(2n-1,2) = \{P\}$. Let $L \subset \mathcal{L}$ be the set of the four lines of Π that do not contain P. Then every point of $\mathcal{P}$ lies on either 0 or 2 lines of L. Hence $\mathcal{C}_{NQ^-(2n-1,2)}$ contains a codeword of weight $|L| = 4$.

From Lemma 1.3 we have $d_{min} \geq 4$, from which it follows that $d_{min} = 4$. □

6.6.4 2-rank of H and dimension of $\mathcal{C}_{NQ^-(2n-1,2)}$

Lemma 1.7 does not apply since $s + t\alpha + t + 1 - \mu \equiv 0 \bmod 2$.

The code dimension and 2-rank of H could only be calculated for $n = 2$, $n = 3$ and $n = 4$ using Magma. The results are displayed in Table 6.10.

Table 6.10: $\mathcal{C}_{NQ^-(2n-1,2)}$: code dimension and 2-rank of H

n	dimension k	2-rank of H
2	4	6
3	90	30
4	1504	128

6.7 The codes $\mathcal{C}_{H_q^{(n+1)*}}$

Let $\mathcal{P}$ be the set of lines of a projective space $\Sigma = \mathrm{PG}(n+1, q)$ skew to a fixed projective space $H = \mathrm{PG}(n-1, q)$. Let $\mathcal{B}$ be the set of planes of Σ that intersect H in exactly one point. Then $H_q^{(n+1)*} = (\mathcal{P}, \mathcal{B}, \mathrm{I})$ is an $\mathrm{spg}(q^2-1, (q^n-1)/(q-1)-1, q, q(q+1))$. See [3].

6.7.1 The matrix H

We have $v = q^{2n}$ and $b = q^{2(n-1)}(q^n-1)/(q-1)$. Also $\gamma = q^2$ and $\rho = (q^n-1)/(q-1)$. The matrix density is $H_d = q^{2(1-n)}$.

From these parameters it follows that for any q we have $b \geq v \Leftrightarrow n \geq 3$.

6.7.2 Girth

Since $\alpha = q \geq 2$ the Tanner graph contains 6-cycles. From Lemma 1.1 we have

$$N_6 = \frac{bt(\alpha-1)}{3}\binom{s+1}{2} = q^{2n}(q^n-1)(q^n-q)(q+1)/6.$$

6.7.3 Minimum distance

Lemma 1.3 gives $d_{min} \geq q^2+1$.

Lemma 1.4 only provides a meaningful (non-negative) bound for $n = 2$. However $v > b$ for all q when $n = 2$.

Magma was only able to calculate the minimum distance of $\mathcal{C}_{H_q^{(n+1)*}}$ for small values of n and q. The results are displayed in Table 6.11.

Table 6.11: Minimum distance of $\mathcal{C}_{H_q^{(n+1)*}}$

n	q	d_{min}
2	2	6
3	2	6
2	3	18
3	3	18
2	4	20
2	5	50

6.7.4 2-rank of H and dimension of $\mathcal{C}_{H_q^{(n+1)*}}$

The code dimension and 2-rank of H could only be calculated for small values of n and q using Magma. The results are displayed in Table 6.12.

Table 6.12: $\mathcal{C}_{H_q^{(n+1)*}}$: code dimension and 2-rank of H

n	q	dimension k	2-rank of H
2	2	4	8
3	2	67	45
4	2	746	214
5	2	7001	935
2	3	3	33
3	3	324	729
2	4	24	56
3	4	3451	1925
2	5	5	145

Chapter 7

Code simulation results 2

This chapter provides some simulation results that test the performance of some of the SPG codes presented in the previous chapter. As in Chapter 5 the codes were compared with a randomly generated code and an uncoded signal transmitted on an AWGN channel.

7.1 The codes $\mathcal{C}_{U_{2,3}(m)}$

The results of Section 6.1.3 show that the structure of the codes for m odd and m even are rather different. Therefore four codes were tested in total, two with m odd and two with m even: $\mathcal{C}_{U_{2,3}(7)}$, $\mathcal{C}_{U_{2,3}(8)}$, $\mathcal{C}_{U_{2,3}(17)}$ and $\mathcal{C}_{U_{2,3}(18)}$.

7.1.1 $\mathcal{C}_{U_{2,3}(7)}$

The code $\mathcal{C}_{U_{2,3}(7)}$ is a (3, 5)-regular [35, 20, 4] code of rate $4/7 \approx 0.571$. Its parity-check matrix is a 21×35 matrix with 2-rank $35 - 20 = 15$.

The performance of $\mathcal{C}_{U_{2,3}(7)}$ was compared with a randomly constructed LDPC code that had a 15×35 parity-check matrix with column weight 3 and row weights that ranged between 1 and 22. The 2-rank of this matrix was calculated using Magma and was found to be 15, the same as for $\mathcal{C}_{U_{2,3}(7)}$. The rate and length of the randomly constructed code were therefore the same as for $\mathcal{C}_{U_{2,3}(7)}$. The matrix contained some 4-cycles.

The results are shown in Figure 7.1. The code $\mathcal{C}_{U_{2,3}(7)}$ performed well in comparison with the randomly generated code. This can be attributed to the following reasons.

- The Tanner graph of the randomly generated code contains some 4-cycles, whereas the Tanner graph of $\mathcal{C}_{U_{2,3}(7)}$ has girth 6.

- The parity-check matrix of $\mathcal{C}_{U_{2,3}(7)}$ is less dense (1/7 compared to 1/5 for the random code).

- The parity-check matrix of $\mathcal{C}_{U_{2,3}(7)}$ contains linearly dependent rows.

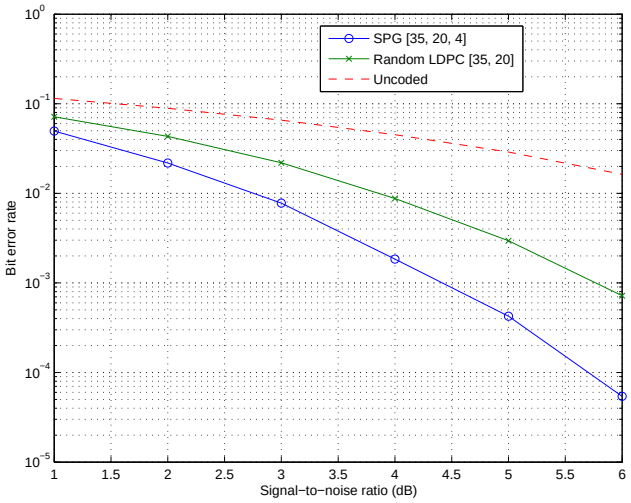

Figure 7.1: The decoding performance of the SPG code $\mathcal{C}_{U_{2,3}(7)}$ on an AWGN channel using iterative probabilistic decoding with a maximum of 10 iterations.

7.1.2 $\mathcal{C}_{U_{2,3}(8)}$

The code $\mathcal{C}_{U_{2,3}(8)}$ is a (3, 6)-regular [56, 35, 4] code of rate $35/56 = 0.625$. Its parity-check matrix is a 28×56 matrix with 2-rank $56 - 35 = 21$.

The performance of $\mathcal{C}_{U_{2,3}(8)}$ was compared with a randomly constructed LDPC code that had a 21×56 parity-check matrix with column weight 3 and row weights that ranged between 2 and 28. The 2-rank of this matrix was calculated using Magma and was found to be 21, the same as for $\mathcal{C}_{U_{2,3}(8)}$. The rate and length of the randomly constructed code were therefore the same as for $\mathcal{C}_{U_{2,3}(8)}$. The matrix contained some 4-cycles.

The results are shown in Figure 7.2. The code $\mathcal{C}_{U_{2,3}(8)}$ performed well in comparison with the randomly generated code. This can be attributed to the following reasons.

- The Tanner graph of the randomly generated code contains some 4-cycles, whereas the Tanner graph of $\mathcal{C}_{U_{2,3}(8)}$ has girth 6.
- The parity-check matrix of $\mathcal{C}_{U_{2,3}(8)}$ is less dense (3/28 compared to 3/21 for the random code).
- The parity-check matrix of $\mathcal{C}_{U_{2,3}(8)}$ contains linearly dependent rows.

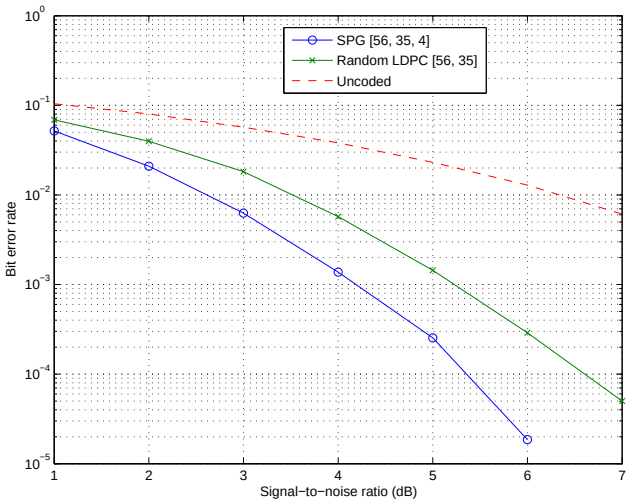

Figure 7.2: The decoding performance of the SPG code $\mathcal{C}_{U_{2,3}(8)}$ on an AWGN channel using iterative probabilistic decoding with a maximum of 10 iterations.

7.1.3 $\mathcal{C}_{U_{2,3}(17)}$

The code $\mathcal{C}_{U_{2,3}(17)}$ is a (3, 15)-regular [680, 560, 4] code of rate $560/680 = 14/17 \approx 0.824$. Its parity-check matrix is a 136×680 matrix with 2-rank $680 - 560 = 120$.

The performance of $\mathcal{C}_{U_{2,3}(17)}$ was compared with a randomly constructed LDPC code that had a 120×680 parity-check matrix with column weight

3 and row weights that ranged between 8 and 28. The matrix was found to have full 2-rank using Magma, showing that the rate and length of the randomly constructed code were the same as for $\mathcal{C}_{U_{2,3}(17)}$. The Tanner graph contained no 4-cycles.

Because of the relatively long length of these codes the log-likelihood version of the decoding algorithm was used to perform the simulations.

The results are shown in Figure 7.3. The code $\mathcal{C}_{U_{2,3}(17)}$ performed slightly better than the randomly generated code for low signal-to-noise ratios. However above about 4dB the performance of $\mathcal{C}_{U_{2,3}(17)}$ degrades as the code exhibits an error floor.

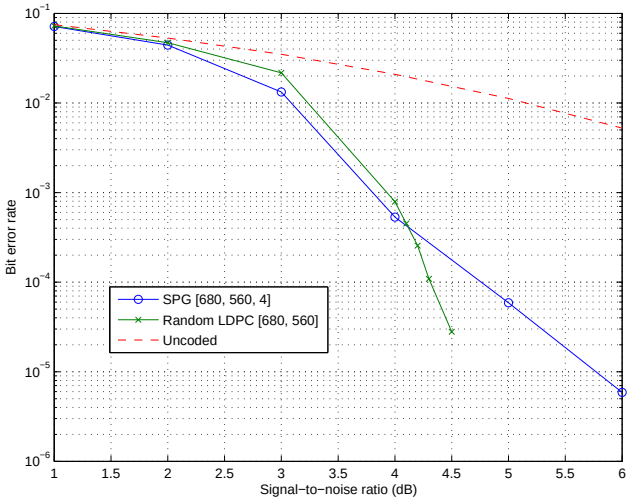

Figure 7.3: The decoding performance of the SPG code $\mathcal{C}_{U_{2,3}(17)}$ on an AWGN channel using log-likelihood iterative probabilistic decoding with a maximum of 10 iterations.

7.1.4 $\mathcal{C}_{U_{2,3}(18)}$

The code $\mathcal{C}_{U_{2,3}(18)}$ is a (3, 16)-regular [816, 680, 4] code of rate $680/816 = 5/6 \approx 0.833$. Its parity-check matrix is a 153×816 matrix with 2-rank $816 - 680 = 136$.

The performance of $\mathcal{C}_{U_{2,3}(18)}$ was compared with a randomly constructed LDPC code that had a 136×816 parity-check matrix with column weight

3 and row weights that ranged between 8 and 28. The matrix was found to have full 2-rank using Magma, showing that the rate and length of the randomly constructed code were the same as for $\mathcal{C}_{U_{2,3}(18)}$. The Tanner graph contained no 4-cycles.

Because of the relatively long length of these codes the log-likelihood version of the decoding algorithm was used to perform the simulations.

The results are shown in Figure 7.4. The code $\mathcal{C}_{U_{2,3}(18)}$ performs less well than the randomly generated code.

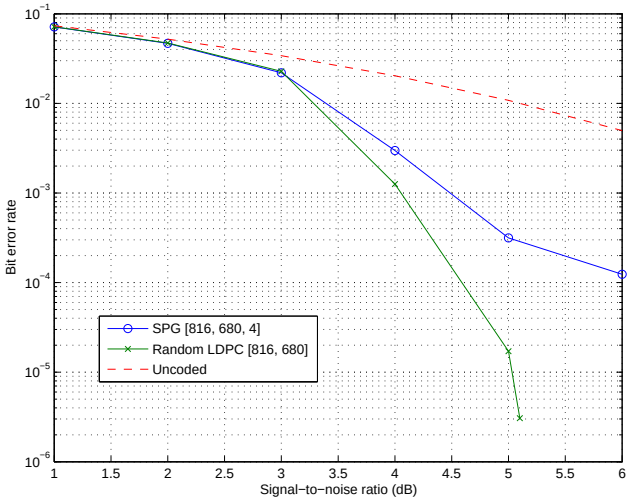

Figure 7.4: The decoding performance of the SPG code $\mathcal{C}_{U_{2,3}(18)}$ on an AWGN channel using log-likelihood iterative probabilistic decoding with a maximum of 10 iterations.

7.1.5 Summary

The codes $\mathcal{C}_{U_{2,3}(m)}$ perform very well for low values of m and therefore seem to provide a good source of short LDPC codes. The codes perform similarly for both even and odd values of m, despite the different structures of the codes.

7.2 The code $\mathcal{C}_{\overline{M(7)}}$

The code $\mathcal{C}_{\overline{M(7)}}$ is a (7,7)-regular [50, 28,8] code of rate 0.56. Its parity-check matrix is a 50×50 matrix with 2-rank $50 - 28 = 22$.

The performance of $\mathcal{C}_{\overline{M(7)}}$ was compared with a randomly constructed LDPC code that had a 22×50 parity-check matrix with column weight 3 and row weights that ranged between 2 and 19. The 2-rank of this matrix was calculated using Magma and was found to be 22, the same as for $\mathcal{C}_{\overline{M(7)}}$. The rate and length of the randomly constructed code were therefore the same as for $\mathcal{C}_{\overline{M(7)}}$. The matrix contained some 4-cycles.

The results are shown in Figure 7.5. The code $\mathcal{C}_{\overline{M(7)}}$ performed well in comparison with the randomly generated code. This can be attributed to the following reasons.

- The Tanner graph of the randomly generated code contains some 4-cycles, whereas the Tanner graph of $\mathcal{C}_{\overline{M(7)}}$ has girth 6.
- The parity-check matrix of $\mathcal{C}_{\overline{M(7)}}$ contains a large number of linearly dependent rows.
- The code $\mathcal{C}_{\overline{M(7)}}$ has a large minimum distance.

The matrix densities were similar (7/50 for $\mathcal{C}_{\overline{M(7)}}$ and 3/22 for the randomly generated code).

7.3 The codes $\mathcal{C}_{\mathrm{LP}(n,q)}$

The codes $\mathcal{C}_{\mathrm{LP}(4,2)}$ and $\mathcal{C}_{\mathrm{LP}(4,3)}$ were tested.

7.3.1 $\mathcal{C}_{\mathrm{LP}(4,2)}$

The code $\mathcal{C}_{\mathrm{LP}(4,2)}$ is a (7,7)-regular [155, 79, 8] code of rate $79/155 \approx 0.510$. Its parity-check matrix is a square 155×155 matrix with 2-rank $155 - 79 = 76$.

The performance of $\mathcal{C}_{\mathrm{LP}(4,2)}$ was compared with a randomly constructed LDPC code that had a 76×155 parity-check matrix. The matrix had 153 columns of weight 3 and 2 columns of weight 4. The row weights ranged between 1 and 12. The matrix was free of 4-cycles.

The 2-rank of this matrix was calculated using Magma and was found to be 76, the same as for $\mathcal{C}_{\mathrm{LP}(4,2)}$. The rate and length of the randomly constructed code were therefore the same as for $\mathcal{C}_{\mathrm{LP}(4,2)}$.

The results are shown in Figure 7.6. The code $\mathcal{C}_{\mathrm{LP}(4,2)}$ performed well in comparison with the randomly generated code. The primary reason for this is that the parity-check matrix of $\mathcal{C}_{\mathrm{LP}(4,2)}$ contains a large number of linearly dependent rows. This outweighed the fact that the parity-check matrix of the randomly generated code was less dense than the parity-check matrix of $\mathcal{C}_{\mathrm{LP}(4,2)}$ (approximately 0.040 for the randomly generated code, compared to 0.045 for $\mathcal{C}_{\mathrm{LP}(4,2)}$).

7.3.2 $\mathcal{C}_{\mathrm{LP}(4,3)}$

The code $\mathcal{C}_{\mathrm{LP}(4,3)}$ is a (13,13)-regular [1210, 120, 40] code of rate $120/1210 \approx 0.099$. Its parity-check matrix is a square 1210×1210 matrix with 2-rank $1210 - 120 = 1090$.

The performance of $\mathcal{C}_{\mathrm{LP}(4,3)}$ was compared with a randomly constructed LDPC code that had a 1090×1210 parity-check matrix. The matrix had 1012 columns of weight 3, 185 columns of weight 4 and 13 columns of weight 5. The row weights ranged between 2 and 11. The matrix was free of 4-cycles.

The 2-rank of this matrix was calculated using Magma and was found to be 1090, the same as for $\mathcal{C}_{\mathrm{LP}(4,3)}$. The rate and length of the randomly constructed code were therefore the same as for $\mathcal{C}_{\mathrm{LP}(4,3)}$.

Because of the relatively long length of these codes the log-likelihood version of the decoding algorithm was used to perform the simulations.

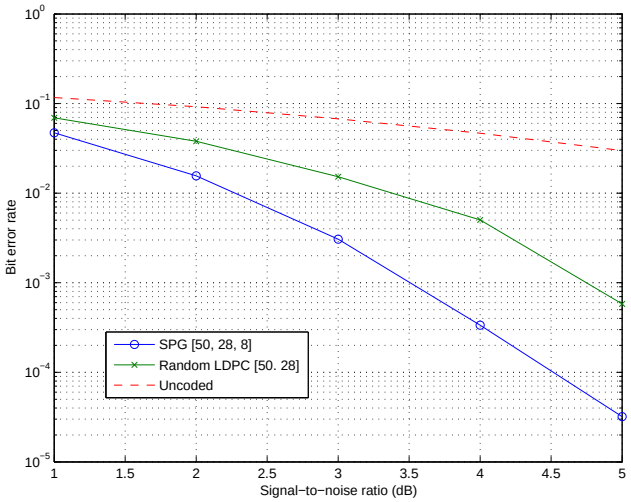

Figure 7.5: The decoding performance of the SPG code $\mathcal{C}_{\overline{M(7)}}$ on an AWGN channel using iterative probabilistic decoding with a maximum of 10 iterations.

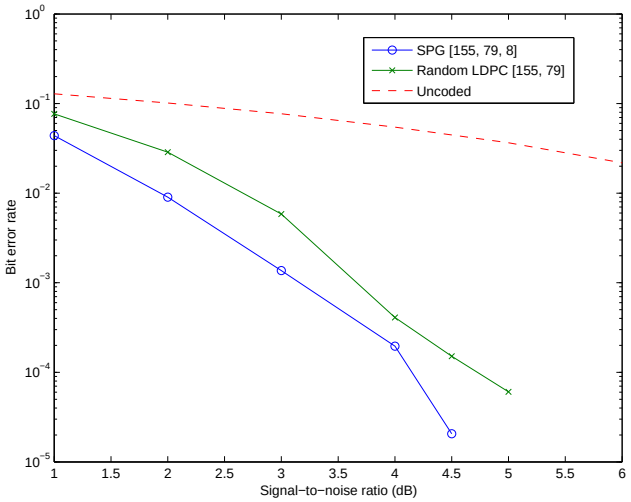

Figure 7.6: The decoding performance of the SPG code $\mathcal{C}_{\mathrm{LP}(4,2)}$ on an AWGN channel using iterative probabilistic decoding with a maximum of 10 iterations.

The results are shown in Figure 7.7. The code $\mathcal{C}_{\mathrm{LP}(4,3)}$ performed very poorly, producing a bit error rate that was almost the same as the uncoded signal. The reason for this appears to be that the density of the parity-check matrix for $\mathcal{C}_{\mathrm{LP}(4,3)}$ is much greater than for the randomly constructed matrix. The density of the matrix for $\mathcal{C}_{\mathrm{LP}(4,3)}$ is $13/1210 \approx 0.011$, whereas the density of the randomly constructed matrix is $(1012 \times 3 + 185 \times 4 + 13 \times 5)/(1090 \times 1210) \approx 0.0029$. The SPG matrix therefore contains nearly four times as many 1s as the random matrix, which accounts for the significantly better performance of the randomly constructed code. Also most of the columns in the randomly constructed matrix have weight 3, compared to a column weight of 13 for the $\mathcal{C}_{\mathrm{LP}(5,2)}$ matrix: as discussed in [15], LDPC codes that have parity-check matrices with column weights equal to 3 perform well under iterative decoding.

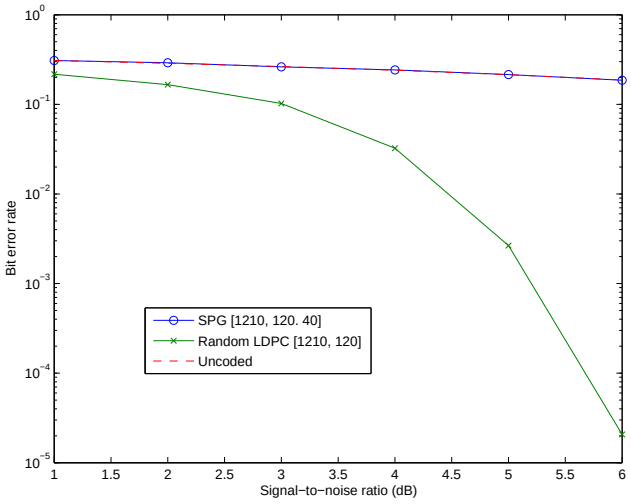

Figure 7.7: The decoding performance of the SPG code $\mathcal{C}_{\mathrm{LP}(4,3)}$ on an AWGN channel using log-likelihood iterative probabilistic decoding with a maximum of 10 iterations.

7.3.3 $\mathcal{C}_{\mathrm{LP}(5,2)}$

The code $\mathcal{C}_{\mathrm{LP}(5,2)}$ is a (7,15)-regular [1395, 974] code of rate $974/1395 \approx 0.698$. Its parity-check matrix is a 651×1395 matrix with 2-rank $1395 - 974 = 421$.

The performance of $\mathcal{C}_{\mathrm{LP}(5,2)}$ was compared with a randomly constructed LDPC code that had a 421×1395 parity-check matrix. The matrix had column weight equal to 3. The row weights ranged between 4 and 19. The matrix was free of 4-cycles.

The 2-rank of this matrix was calculated using Magma and was found to be 421, the same as for $\mathcal{C}_{\mathrm{LP}(5,2)}$. The rate and length of the randomly constructed code were therefore the same as for $\mathcal{C}_{\mathrm{LP}(5,2)}$.

Because of the relatively long length of these codes the log-likelihood version of the decoding algorithm was used to perform the simulations.

The results are shown in Figure 7.8. The code $\mathcal{C}_{\mathrm{LP}(5,2)}$ performed very poorly, producing a bit error rate that was almost the same as the uncoded signal. The reason for this appears to be that the density of the parity-check matrix for $\mathcal{C}_{\mathrm{LP}(5,2)}$ is greater than the density of the randomly constructed matrix. The density of the matrix for $\mathcal{C}_{\mathrm{LP}(5,2)}$ is $7/651 \approx 0.0107$, whereas the density of the randomly constructed matrix is $3/421 \approx 0.00713$. Also the column weight of the randomly constructed matrix is 3, which is much better for iterative decoding than the column weight of 7 of the $\mathcal{C}_{\mathrm{LP}(5,2)}$ matrix (see [15]).

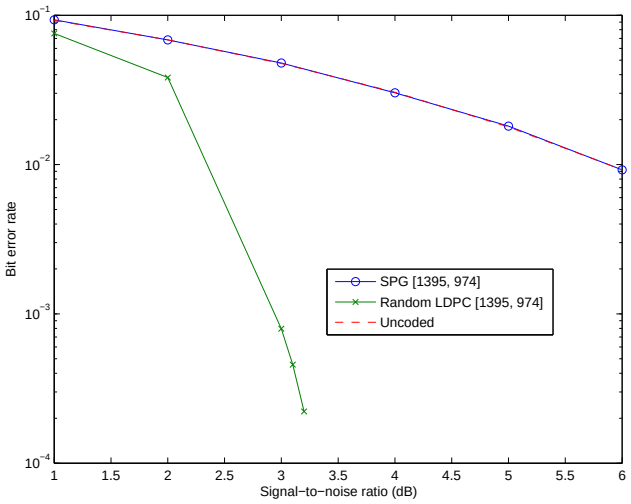

Figure 7.8: The decoding performance of the SPG code $\mathcal{C}_{\mathrm{LP}(5,2)}$ on an AWGN channel using log-likelihood iterative probabilistic decoding with a maximum of 10 iterations.

7.4 The codes $\mathcal{C}_{\overline{W(2n+1,q)}}$

The codes $\mathcal{C}_{\overline{W(3,3)}}$, $\mathcal{C}_{\overline{W(3,4)}}$, $\mathcal{C}_{\overline{W(3,5)}}$ and $\mathcal{C}_{\overline{W(5,2)}}$ were tested.

7.4.1 $\mathcal{C}_{\overline{W(3,2)}}$

The code $\mathcal{C}_{\overline{W(3,2)}}$ is a [20, 10, 4] code of rate 0.5. Its parity-check matrix is a 15×20 (3,4)-regular matrix with 2-rank $20 - 10 = 10$.

The software that generated the random LDPC codes did not produce a code with rate equal to 0.5. Therefore the performance of $\mathcal{C}_{\overline{W(3,2)}}$ was compared with two randomly constructed codes, one of rate 0.55 and the other of rate 0.45. The first code had a 10×20 parity-check matrix with 2-rank equal to 9. The matrix had column weight equal to 3 and row weights that ranged between 1 and 18. The matrix contained some 4-cycles. The second code had an 11×20 parity-check matrix with full 2-rank. This matrix also had column weight equal to 3 and row weights that ranged between 1 and 18, and contained some 4-cycles.

The results are shown in Figure 7.9. The code $\mathcal{C}_{\overline{W(3,2)}}$ performed well in comparison to the randomly generated codes. This can probably be attributed to the fact that the parity-check matrix had a lower density than the randomly constructed matrices. Furthermore the matrix contained a number of linearly dependent rows.

7.4.2 $\mathcal{C}_{\overline{W(3,3)}}$

The code $\mathcal{C}_{\overline{W(3,3)}}$ is a (4,9)-regular [90, 51, 10] code of rate $51/90 = 17/30 \approx 0.567$. Its parity-check matrix is a 40×90 matrix with 2-rank $90 - 51 = 39$.

The performance of $\mathcal{C}_{\overline{W(3,3)}}$ was compared with a randomly constructed LDPC code that had a 39×90 parity-check matrix. The matrix had column weight equal to 3 and row weights that ranged between 2 and 14. The matrix contained some 4-cycles. The matrix was found to have full 2-rank using Magma. The rate and length of the randomly constructed code were therefore the same as for $\mathcal{C}_{\overline{W(3,3)}}$.

The results are shown in Figure 7.10. The code $\mathcal{C}_{\overline{W(3,3)}}$ performed less well than the randomly generated code. This can be attributed to the following factors.

- The density of the matrix for $\mathcal{C}_{\overline{W(3,3)}}$ is $4/40 = 0.1$ compared to $3/39 \approx 0.077$ for the randomly constructed code.

- The randomly constructed matrix had column weight equal to 3.

These factors outweigh the fact that the the randomly constructed matrix contained some 4-cycles, and also the fact that the matrix of the SPG code contained an extra linearly dependent row.

7.4.3 $\mathcal{C}_{\overline{W(3,4)}}$

The code $\mathcal{C}_{\overline{W(3,4)}}$ is a (5, 16)-regular [272, 212, 6] code of rate $212/272 = 53/68 \approx 0.779$. Its parity-check matrix is an 85×272 matrix with 2-rank $272 - 212 = 60$.

The performance of $\mathcal{C}_{\overline{W(3,4)}}$ was compared with a randomly constructed LDPC code that had a 60×212 parity-check matrix. The matrix had column weight equal to 3 and row weights that ranged between 7 and 32. The matrix contained some 4-cycles. The matrix was found to have full 2-rank using Magma. The rate and length of the randomly constructed code were therefore the same as for $\mathcal{C}_{\overline{W(3,4)}}$.

The results are shown in Figure 7.11. The code $\mathcal{C}_{\overline{W(3,4)}}$ performed well compared to the randomly generated code at signal-to-noise ratios greater than 2dB. The explanation for this appears to be that the parity-check matrix of $\mathcal{C}_{\overline{W(3,4)}}$ contains a number of linearly dependent rows and also that the randomly constructed matrix contains some 4-cycles. In this case these factors appear to outweigh the column weight of 3 of the randomly constructed matrix and the fact that the random matrix is less dense than the SPG matrix (the densities are 0.05 and 0.059 respectively).

Comparing the performance of $\mathcal{C}_{\overline{W(3,4)}}$ with $\mathcal{C}_{\overline{W(3,3)}}$ we see that there is a delicate balance in the way that the various properties of the matrices affect performance. It seems that the significant factor in the better performance of $\mathcal{C}_{\overline{W(3,4)}}$ compared to $\mathcal{C}_{\overline{W(3,3)}}$ is that the parity-check matrix of $\mathcal{C}_{\overline{W(3,4)}}$ contains a higher proportion of linearly dependent rows.

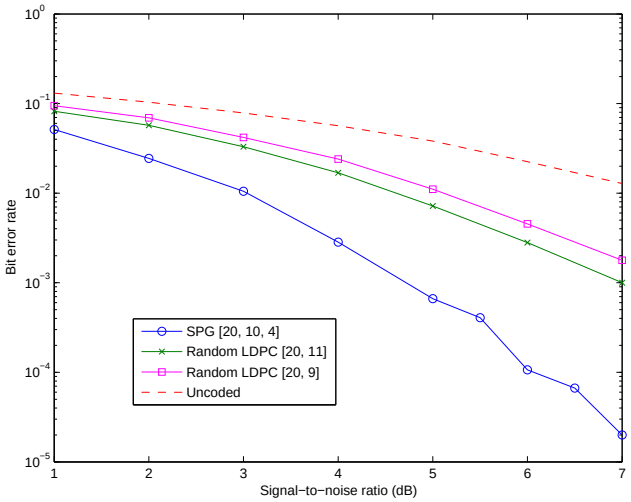

Figure 7.9: The decoding performance of the SPG code $\mathcal{C}_{\overline{W(3,2)}}$ on an AWGN channel using iterative probabilistic decoding with a maximum of 10 iterations.

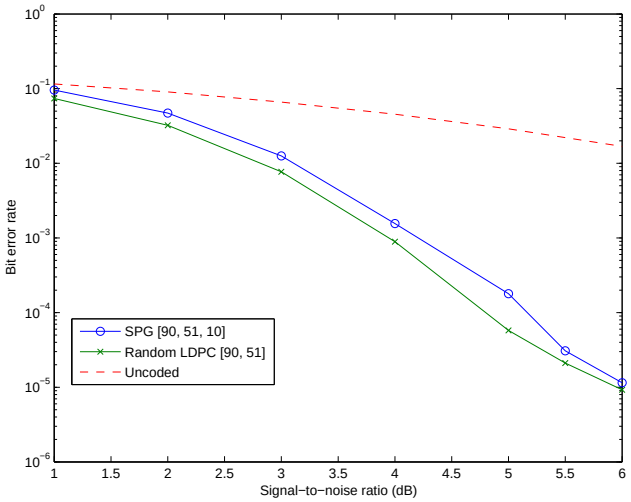

Figure 7.10: The decoding performance of the SPG code $\mathcal{C}_{\overline{W(3,3)}}$ on an AWGN channel using iterative probabilistic decoding with a maximum of 10 iterations.

7.4.4 $\mathcal{C}_{\overline{W(3,5)}}$

The code $\mathcal{C}_{\overline{W(3,5)}}$ is a (6,25)-regular [650, 495] code of rate $495/650 = 99/130 \approx 0.762$. Its parity-check matrix is a 156×650 matrix with 2-rank $650 - 495 = 155$.

The performance of $\mathcal{C}_{\overline{W(3,5)}}$ was compared with a randomly constructed LDPC code that had a 155×650 parity-check matrix. The matrix had column weight equal to 3 and row weights that ranged between 6 and 22. The matrix was free of 4-cycles. The matrix was found to have full 2-rank using Magma. The rate and length of the randomly constructed code were therefore the same as for $\mathcal{C}_{\overline{W(3,5)}}$.

Because of the relatively long length of these codes the log-likelihood version of the decoding algorithm was used to perform the simulations.

The results are shown in Figure 7.12. The randomly constructed code outperformed $\mathcal{C}_{\overline{W(3,5)}}$ for all values of the signal-to-noise ratio. The main reason for this is that the density of the randomly constructed matrix is nearly half that of the SPG matrix (the densities are 0.0194 and 0.385 respectively).

7.4.5 $\mathcal{C}_{\overline{W(5,2)}}$

The code $\mathcal{C}_{\overline{W(5,2)}}$ is a (3, 16)-regular [336, 280, 4] code of rate $280/336 = 5/6 \approx 0.833$. Its parity-check matrix is a 63×336 matrix with 2-rank $336 - 280 = 56$.

The performance of $\mathcal{C}_{\overline{W(5,2)}}$ was compared with a randomly constructed LDPC code that had a 56×336 parity-check matrix. The matrix had column weight equal to 3 and row weights that ranged between 6 and 68. The matrix contained some 4-cycles. The matrix was found to have full 2-rank using Magma. The rate and length of the randomly constructed code were therefore the same as for $\mathcal{C}_{\overline{W(5,2)}}$.

The results are shown in Figure 7.13. The code $\mathcal{C}_{\overline{W(5,2)}}$ performed well compared to the randomly generated code at signal-to-noise ratios greater than 2dB. The explanation for this appears to be that the parity-check matrix of $\mathcal{C}_{\overline{W(5,2)}}$ contains a large number of linearly dependent rows and also that the randomly constructed matrix contains some 4-cycles. Also the density of the parity-check matrix for $\mathcal{C}_{\overline{W(5,2)}}$ is $3/63 = 1/21 \approx 0.048$ whereas the randomly constructed matrix has density $3/56 \approx 0.054$.

Interestingly the randomly constructed code performed slightly better than $\mathcal{C}_{\overline{W(5,2)}}$ at low signal-to-noise ratios. The reason for this is unclear.

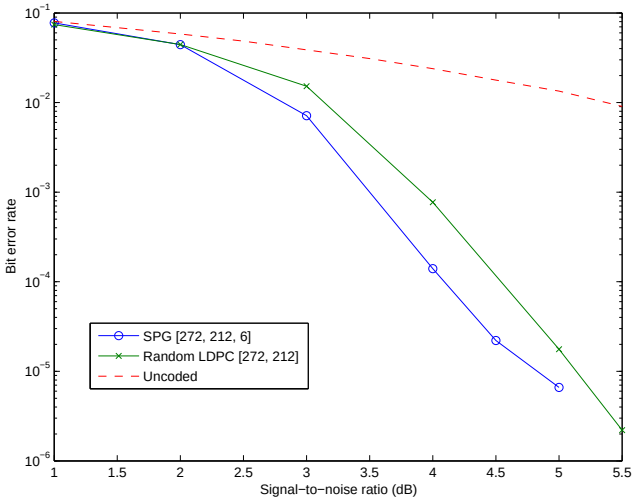

Figure 7.11: The decoding performance of the SPG code $\mathcal{C}_{\overline{W(3,4)}}$ on an AWGN channel using iterative probabilistic decoding with a maximum of 10 iterations.

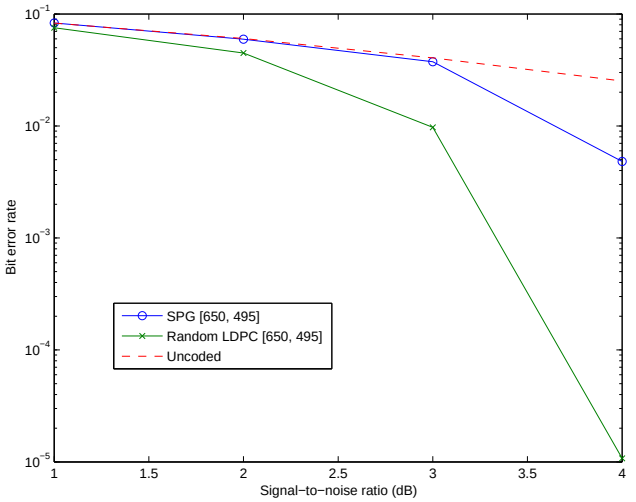

Figure 7.12: The decoding performance of the SPG code $\mathcal{C}_{\overline{W(3,5)}}$ on an AWGN channel using log-likelihood iterative probabilistic decoding with a maximum of 10 iterations.

7.5 The codes $\mathcal{C}_{NQ^+(2n-1,2)}$

The codes $\mathcal{C}_{NQ^+(5,2)}$ and $\mathcal{C}_{NQ^+(7,2)}$ were tested.

7.5.1 $\mathcal{C}_{NQ^+(5,2)}$

The code $\mathcal{C}_{NQ^+(5,2)}$ is a [56, 35, 4] code of rate 0.625. Its parity-check matrix is a 28×56 (3,6)-regular matrix with 2-rank $56 - 35 = 21$.

The performance of $\mathcal{C}_{NQ^+(5,2)}$ was compared with a randomly constructed LDPC code that had a 21×56 parity-check matrix. The matrix had column weight equal to 3 and row weights that ranged between 2 and 28. The matrix contained some 4-cycles. The matrix was found to have full 2-rank using Magma. The rate and length of the randomly constructed code were therefore the same as for $\mathcal{C}_{NQ^+(5,2)}$.

The results are shown in Figure 7.14. The code $\mathcal{C}_{NQ^+(5,2)}$ performed well in comparison to the randomly generated code. This can be attributed to the following factors.

- The density of the matrix for $\mathcal{C}_{NQ^+(5,2)}$ is $3/28 \approx 0.107$ compared to $3/21 \approx 0.143$ for the randomly constructed code.
- The matrix for $\mathcal{C}_{NQ^+(5,2)}$ contained a number of linearly dependent rows.
- The randomly constructed matrix contained some 4-cycles.

7.5.2 $\mathcal{C}_{NQ^+(7,2)}$

The code $\mathcal{C}_{NQ^+(7,2)}$ is a [1120,1008, 4] code of rate 0.9. Its parity-check matrix is a 120×1120 (3,28)-regular matrix with 2-rank $1120 - 1008 = 112$.

The performance of $\mathcal{C}_{NQ^+(7,2)}$ was compared with a randomly constructed LDPC code that had a 112×1120 parity-check matrix. The matrix had column weight equal to 3 and row weights that ranged between 17 and 102. The matrix contained some 4-cycles. The matrix was found to have full 2-rank using Magma. The rate and length of the randomly constructed code were therefore the same as for $\mathcal{C}_{NQ^+(7,2)}$.

Because of the relatively long length of these codes the log-likelihood version of the decoding algorithm was used to perform the simulations.

The results are shown in Figure 7.15. The code $\mathcal{C}_{NQ^+(7,2)}$ performed poorly in comparison to the randomly generated code, hardly improving on the bit

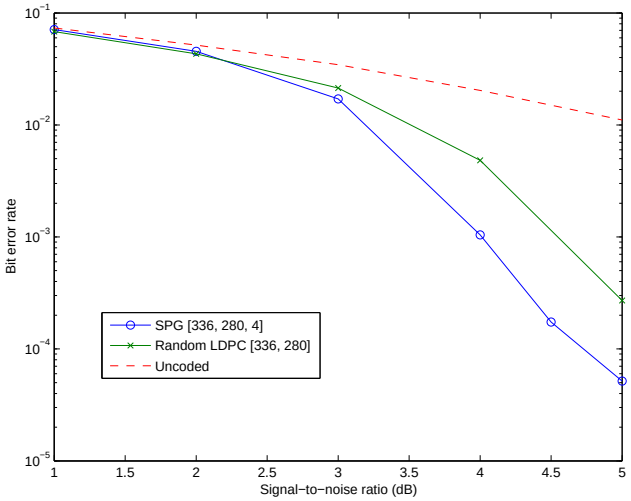

Figure 7.13: The decoding performance of the SPG code $\mathcal{C}_{\overline{W(5,2)}}$ on an AWGN channel using iterative probabilistic decoding with a maximum of 10 iterations.

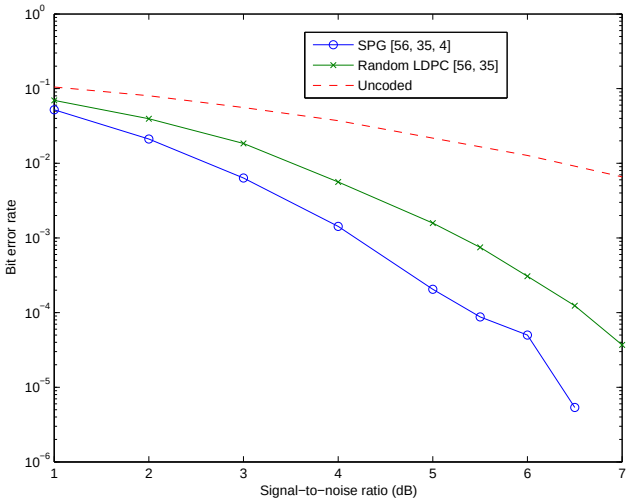

Figure 7.14: The decoding performance of the SPG code $\mathcal{C}_{NQ^+(5,2)}$ on an AWGN channel using iterative probabilistic decoding with a maximum of 10 iterations.

error rate of the uncoded signal. It is unclear to me why this should have been the case, particularly since the SPG matrix was less dense than the randomly constructed matrix and the randomly constructed matrix contained 4-cycles.

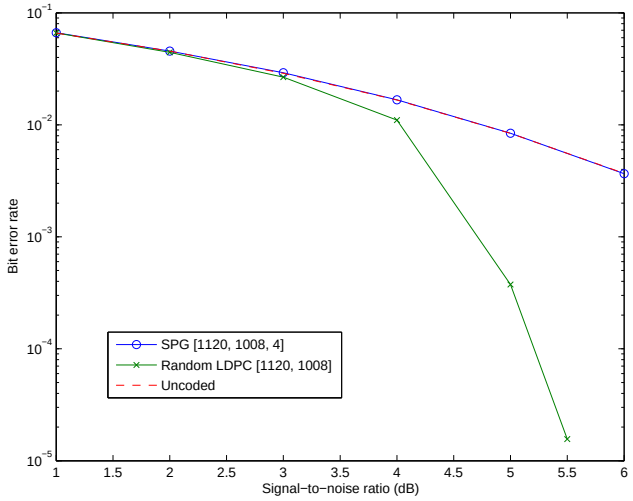

Figure 7.15: The decoding performance of the SPG code $\mathcal{C}_{NQ^+(7,2)}$ on an AWGN channel using log-likelihood iterative probabilistic decoding with a maximum of 10 iterations.

7.6 The codes $\mathcal{C}_{NQ^-(2n-1,2)}$

The codes $\mathcal{C}_{NQ^-(5,2)}$ and $\mathcal{C}_{NQ^-(7,2)}$ were tested.

7.6.1 $\mathcal{C}_{NQ^-(5,2)}$

The code $\mathcal{C}_{NQ^-(5,2)}$ is a [120, 90, 4] code of rate 0.75. Its parity-check matrix is a 36×120 (3,10)-regular matrix with 2-rank $120 - 90 = 30$.

The performance of $\mathcal{C}_{NQ^-(5,2)}$ was compared with a randomly constructed LDPC code that had a 30×120 parity-check matrix. The matrix had column weight equal to 3 and row weights that ranged between 3 and 50. The matrix contained some 4-cycles. The matrix was found to have full 2-rank using Magma. The rate and length of the randomly constructed code were therefore the same as for $\mathcal{C}_{NQ^-(5,2)}$.

The results are shown in Figure 7.16. The code $\mathcal{C}_{NQ^-(5,2)}$ performed well in comparison to the randomly generated code. This can be attributed to the following factors.

- The density of the matrix for $\mathcal{C}_{NQ^-(5,2)}$ is $3/36 \approx 0.0833$ compared to $3/30 = 0.1$ for the randomly constructed code.
- The matrix for $\mathcal{C}_{NQ^-(5,2)}$ contained a number of linearly dependent rows.
- The randomly constructed matrix contained some 4-cycles.

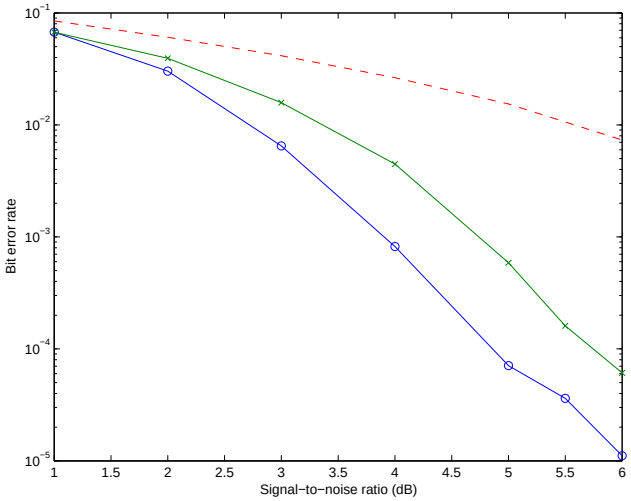

Figure 7.16: The decoding performance of the SPG code $\mathcal{C}_{NQ^-(5,2)}$ on an AWGN channel using iterative probabilistic decoding with a maximum of 10 iterations.

7.6.2 $\mathcal{C}_{NQ^-(7,2)}$

The code $\mathcal{C}_{NQ^-(7,2)}$ is a [1632,1504, 4] code of rate 0.921 (approximately). Its parity-check matrix is a 136×1632 (3,36)-regular matrix with 2-rank $1632 - 1504 = 128$.

The performance of $\mathcal{C}_{NQ^-(7,2)}$ was compared with a randomly constructed LDPC code that had a 128×1632 parity-check matrix. The matrix had column weight equal to 3 and row weights that ranged between 16 and 150.

The matrix contained some 4-cycles. The matrix was found to have full 2-rank using Magma. The rate and length of the randomly constructed code were therefore the same as for $\mathcal{C}_{NQ^-(7,2)}$.

Because of the relatively long length of these codes the log-likelihood version of the decoding algorithm was used to perform the simulations.

The results are shown in Figure 7.17. The code $\mathcal{C}_{NQ^-(7,2)}$ performed poorly in comparison to the randomly generated code, hardly improving on the bit error rate of the uncoded signal. As with $\mathcal{C}_{NQ^+(7,2)}$ the SPG matrix was less dense than the randomly constructed matrix and the randomly constructed matrix contained 4-cycles, and I am therefore unable to explain the poor performance of the SPG code compared to the randomly generated code.

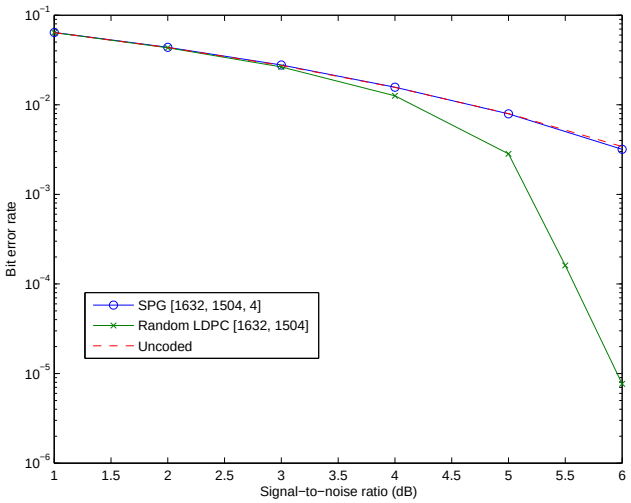

Figure 7.17: The decoding performance of the SPG code $\mathcal{C}_{NQ^-(7,2)}$ on an AWGN channel using log-likelihood iterative probabilistic decoding with a maximum of 10 iterations.

7.7 The codes $\mathcal{C}_{H_q^{(n+1)*}}$

The codes $\mathcal{C}_{H_2^{(4)*}}$, $\mathcal{C}_{H_2^{(5)*}}$ and $\mathcal{C}_{H_3^{(4)*}}$ were tested.

7.7.1 $\mathcal{C}_{H_2^{(4)*}}$

The code $\mathcal{C}_{H_2^{(4)*}}$ is a [112, 67, 6] code of rate 0.598 (approximately). Its parity-check matrix is a 64×112 (4,7)-regular matrix with 2-rank $112 - 67 = 45$.

The performance of $\mathcal{C}_{H_2^{(4)*}}$ was compared with a randomly constructed LDPC code that had a 45×112 parity-check matrix. The matrix had column weight equal to 3 and row weights that ranged between 2 and 14. The matrix contained some 4-cycles. The matrix was found to have full 2-rank using Magma. The rate and length of the randomly constructed code were therefore the same as for $\mathcal{C}_{H_2^{(4)*}}$.

The results are shown in Figure 7.18. The code $\mathcal{C}_{H_2^{(4)*}}$ performed well in comparison to the randomly generated code. This can be attributed to the following factors.

- The density of the matrix for $\mathcal{C}_{NQ^-(5,2)}$ is $4/64 = 1/16$ compared to $3/45 = 1/15$ for the randomly constructed code.
- The matrix for $\mathcal{C}_{H_2^{(4)*}}$ contained a number of linearly dependent rows.
- The randomly constructed matrix contained some 4-cycles.

7.7.2 $\mathcal{C}_{H_2^{(5)*}}$

The code $\mathcal{C}_{H_2^{(5)*}}$ is a [960,746] code of rate 0.777 (approximately). Its parity-check matrix is a 256×960 (4,7)-regular matrix with 2-rank $960 - 746 = 214$.

The performance of $\mathcal{C}_{H_2^{(5)*}}$ was compared with a randomly constructed LDPC code that had a 214×960 parity-check matrix. The matrix had column weight equal to 3 and row weights that ranged between 4 and 22. The matrix contained some 4-cycles. The matrix was found to have full 2-rank using Magma. The rate and length of the randomly constructed code were therefore the same as for $\mathcal{C}_{H_2^{(5)*}}$.

The results are shown in Figure 7.19. The code $\mathcal{C}_{H_2^{(5)*}}$ performed poorly in comparison to the randomly generated code, hardly improving on the bit error rate of the uncoded signal. I am unable to explain the poor performance of the SPG code compared to the randomly generated code.

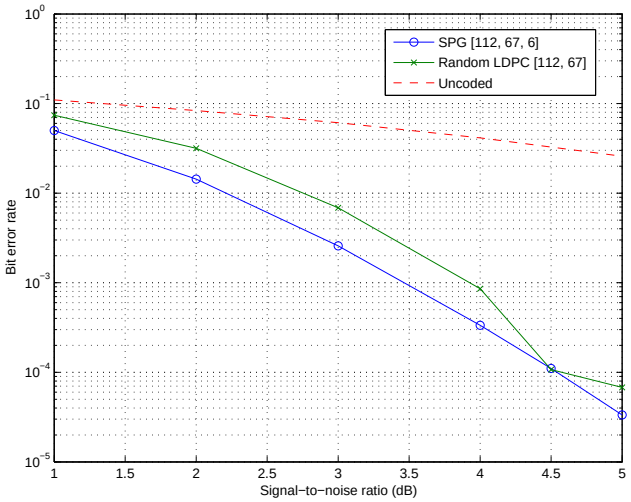

Figure 7.18: The decoding performance of the SPG code $\mathcal{C}_{H_2^{(4)*}}$ on an AWGN channel using iterative probabilistic decoding with a maximum of 10 iterations.

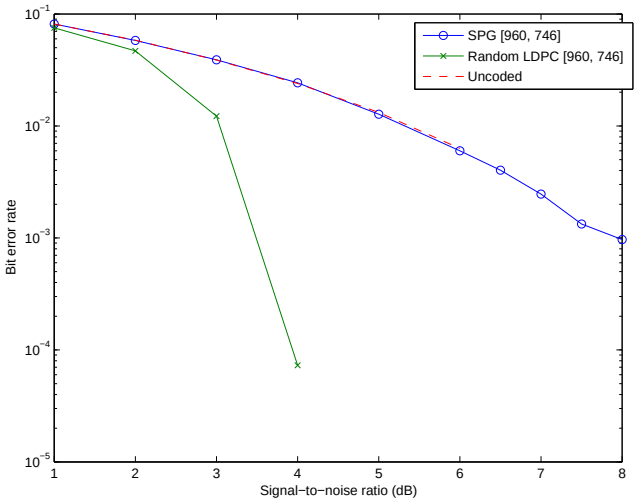

Figure 7.19: The decoding performance of the SPG code $\mathcal{C}_{H_2^{(5)*}}$ on an AWGN channel using iterative probabilistic decoding with a maximum of 10 iterations.

7.7.3 $\mathcal{C}_{H_3^{(4)*}}$

The code $\mathcal{C}_{H_3^{(4)*}}$ is a [1053,324] code of rate 0.307 (approximately). Its parity-check matrix is a 729×1053 (9,13)-regular matrix with 2-rank $1053 - 324 = 729$.

The performance of $\mathcal{C}_{H_3^{(4)*}}$ was compared with a randomly constructed LDPC code that had a 729×1053 parity-check matrix. The matrix had column weights equal to 3 and 4 row weights that ranged between 2 and 11. The matrix contained some 4-cycles. The matrix was found to have full 2-rank using Magma. The rate and length of the randomly constructed code were therefore the same as for $\mathcal{C}_{H_3^{(4)*}}$.

The results are shown in Figure 7.20. The code $\mathcal{C}_{H_3^{(4)*}}$ performed poorly in comparison to the randomly generated code, hardly improving on the bit error rate of the uncoded signal.

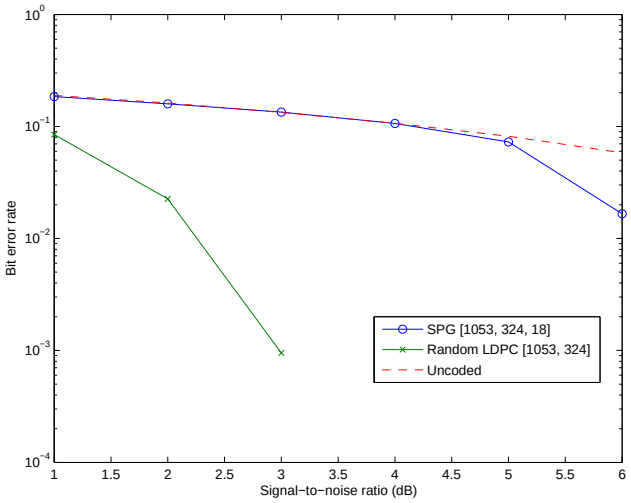

Figure 7.20: The decoding performance of the SPG code $\mathcal{C}_{H_3^{(4)*}}$ on an AWGN channel using log-likelihood iterative probabilistic decoding with a maximum of 10 iterations.

Chapter 8

Appendix: Magma Code

The following programs print parity-check matrices in alist format of the codes that are discussed in the thesis. Some of the programs also calculate properties of the codes such as the minimum distance.

8.1 $\mathcal{C}_{K_{q+1,q}}$

```
/*
Program to find minimum distance and dimension of code
obtained from linear representation of PG(2,q)
*/

print "Input q";
read q1;
q := StringToInteger(q1);
k := GF(q);
VS4 := VectorSpace(k,4);

/*
PG2: set of points of PG(2,q)
P: PG(3,q)\PG(2,q) as an enumerated sequence
*/
PG2 := {Normalize(v) : v in VS4 | v ne 0 and v[1] eq 0};
P := Setseq({v: v in VS4 | v[1] eq 1});
```

```
/*
line: returns points on a line, given v,w on the line
*/
line := function(v,w);
        l := {};
        w1:=Normalize(w);
        Include(~l,w1);
        for a in k do
                j := v + a*w;
                s := VS4!j;
                t := Normalize(s);
                Include(~l, t);
        end for;
return l;
end function;

/*
Lines: set of all lines not in PG(2,q)
PG2Lines: Lines that contain a point of PG(2,q)
B: PG2Lines expressed in terms of P
*/
Lines := {line(v,w) : v,w in VS4 | Normalize(v) ne Normalize(w)
                                                and w[1] ne 0 and v ne 0};
PG2Lines := {l: l in Lines | #(l meet PG2) ne 0};
B := {{i:i in [1..#P] | P[i] in l} : l in PG2Lines};

S := IncidenceStructure<#P|B>;
C := Dual(LinearCode(ChangeRing(IncidenceMatrix(S),GF(2))));
M:=ChangeRing(IncidenceMatrix(S),GF(2));

print "q: " cat IntegerToString(q);
print "length: " cat IntegerToString(#B);
print "dimension: " cat IntegerToString(Dimension(C));
print "minimum distance: " cat IntegerToString(MinimumDistance(C));

/*
Print the matrix to file in alist format
```

```
See MacKay http://www.inference.phy.cam.ac.uk/mackay/codes/alist.html
*/

col_weight := q;
row_weight := q^2+q+1;

f := Open("alist", "w");

Puts(f, Sprint(#B) cat " " cat Sprint(#P));
Puts(f, Sprint(col_weight) cat " " cat Sprint(row_weight));

s := Sprint(row_weight) cat " ";
Put(f, s^#P);
Put(f, "\n");

s := Sprint(col_weight) cat " ";
Put(f, s^#B);
Put(f, "\n");

for i in [1..#P] do
        temp_vector := M[i];
        for j in [1..#B] do
                if temp_vector[j] eq 1 then
                        Put(f,Sprint(j) cat " ");
                else
                end if;
        end for;
        Put(f, "\n");
end for;

for i in [1..#B] do
        temp_vector := Transpose(M)[i];
        for j in [1..#P] do
if temp_vector[j] eq 1 then
Put(f,Sprint(j) cat " ");
else
end if;
```

```
end for;
Put(f, "\n");
end for;
```

8.2 $\mathcal{C}_{K_{q,q}}$

```
/*
Program to find minimum distance and dimension of code
obtained from linear representation of AG(2,q)
*/

print "Input q";
read q1;
q := StringToInteger(q1);
k := GF(q);
VS4 := VectorSpace(k,4);

/*
AG2: set of points of AG(2,q)
P: PG(3,q)\PG(2,q) as an enumerated sequence
*/
AG2 := {v : v in VS4 | v[1] eq 0 and v[2] eq 1};
P := Setseq({v: v in VS4 | v[1] eq 1});

/*
line: returns points on a line, given v,w on the line
*/
line := function(v,w);
        l := {};
        w1:=Normalize(w);
        Include(~l,w1);
        for a in k do
                j := v + a*w;
                s := VS4!j;
                t := Normalize(s);
                Include(~l, t);
        end for;
return l;
end function;

/*
```

```
Lines: set of all lines not in PG(2,q)
AG2Lines: Lines that contain a point of AG(2,q)
B: AG2Lines expressed in terms of P
*/
Lines := {line(v,w) : v,w in VS4 | Normalize(v) ne Normalize(w)
                                                  and w[1] ne 0 and v ne 0};
AG2Lines := {l: l in Lines | #(l meet AG2) ne 0};
B := {{i:i in [1..#P] | P[i] in l} : l in AG2Lines};

S := IncidenceStructure<#P|B>;
C := Dual(LinearCode(ChangeRing(IncidenceMatrix(S),GF(2))));
M:=ChangeRing(IncidenceMatrix(S),GF(2));

print "q: " cat IntegerToString(q);
print "length: " cat IntegerToString(#B);
print "dimension: " cat IntegerToString(Dimension(C));
print "minimum distance: " cat IntegerToString(MinimumDistance(C));

/*
Print the matrix to file in alist format
See MacKay http://www.inference.phy.cam.ac.uk/mackay/codes/alist.html
*/

col_weight := q;
row_weight := q^2;

f := Open("alist", "w");

Puts(f, Sprint(#B) cat " " cat Sprint(#P));
Puts(f, Sprint(col_weight) cat " " cat Sprint(row_weight));

s := Sprint(row_weight) cat " ";
Put(f, s^#P);
Put(f, "\n");

s := Sprint(col_weight) cat " ";
Put(f, s^#B);
```

```
Put(f, "\n");

for i in [1..#P] do
        temp_vector := M[i];
        for j in [1..#B] do
                if temp_vector[j] eq 1 then
                        Put(f,Sprint(j) cat " ");
                else
                end if;
        end for;
        Put(f, "\n");
end for;

for i in [1..#B] do
        temp_vector := Transpose(M)[i];
        for j in [1..#P] do
if temp_vector[j] eq 1 then
Put(f,Sprint(j) cat " ");
else
end if;
end for;
Put(f, "\n");
end for;
```

8.3 $\mathcal{C}_{K_{2^m,2^h}}, 1 < m < h$

```
/*
Program to find minimum distance and dimension of code
obtained from linear representation of 'Denniston type'
maximal arc
*/


print "Input even q";
read q1;
q := StringToInteger(q1);
k := GF(q);
a := NormalElement(k);

print "Input h, 2<2^h<q";
read h1;
h := StringToInteger(h1);
b := {};

for i in [1..(h-1)] do
j:=2^i;
Include(~b, a^j);
end for;

j:={};
Include(~j,k!0);
Include(~j,k!1);
for i in b do
for t in j do
Include(~j, t + i);
end for;
end for;

VS4 := VectorSpace(k,4);

/*
```

```
Given even q, calculate r such that rx^2 + xy + y^2 is irreducible.
*/

if not IsSquare(q) then
        r:=1;
else
        i:=1;
        g:=PrimitiveElement(k);
        while Trace(g^i) ne 1 do
        i:=i+1;
        end while;
        r:=g^i;
end if;

/*
maxarc: set of points of 'Denniston type' maximal arc
P: PG(3,q)\PG(2,q) as an enumerated sequence
*/

maxarc := {v : v in VS4 | v[1] eq 0 and v[2] eq 1 and (r*v[3]^2 + v[3]*v[4] +v[4

P := Setseq({v: v in VS4 | v[1] eq 1});

/*
line: returns points on a line, given v,w on the line
*/

line := function(v,w);
        l := {};
        w1:=Normalize(w);
        Include(~l,w1);
        for a in k do
                j := v + a*w;
                s := VS4!j;
                t := Normalize(s);
                Include(~l, t);
        end for;
```

```
return l;
end function;

/*
Lines: set of all lines not in PG(2,q)
maxarclines: Lines that contain a point of the maximal arc
B: maxarclines expressed in terms of P
*/

Lines := {line(v,w) : v,w in VS4 | Normalize(v) ne Normalize(w)
                                                  and w[1] ne 0 and v ne 0};
maxarclines := {l: l in Lines | #(l meet maxarc) ne 0};
B := {{i:i in [1..#P] | P[i] in l} : l in maxarclines};

S := IncidenceStructure<#P|B>;
C := Dual(LinearCode(ChangeRing(IncidenceMatrix(S),GF(2))));
M:=ChangeRing(IncidenceMatrix(S),GF(2));


print "q: " cat IntegerToString(q);
print "length: " cat IntegerToString(#B);
print "dimension: " cat IntegerToString(Dimension(C));
print "minimum distance: " cat IntegerToString(MinimumDistance(C));

/*
Print the matrix to file in alist format
See MacKay http://www.inference.phy.cam.ac.uk/mackay/codes/alist.html
*/

col_weight := q;
row_weight := #B/q^2;

f := Open("alist", "w");

Puts(f, Sprint(#B) cat " " cat Sprint(#P));
Puts(f, Sprint(col_weight) cat " " cat Sprint(row_weight));
```

```
s := Sprint(row_weight) cat " ";
Put(f, s^#P);
Put(f, "\n");

s := Sprint(col_weight) cat " ";
Put(f, s^#B);
Put(f, "\n");

for i in [1..#P] do
        temp_vector := M[i];
        for j in [1..#B] do
                if temp_vector[j] eq 1 then
                        Put(f,Sprint(j) cat " ");
                else
                end if;
        end for;
        Put(f, "\n");
end for;

for i in [1..#B] do
        temp_vector := Transpose(M)[i];
        for j in [1..#P] do
if temp_vector[j] eq 1 then
Put(f,Sprint(j) cat " ");
else
end if;
end for;
Put(f, "\n");
end for;
```

8.4 $\mathcal{C}_{K_{2,2^h}}$

```
/*
Program to find minimum distance and dimension of code
obtained from linear representation of hyperoval
*/

print "Input q";
read q1;
q := StringToInteger(q1);
k := GF(q);

VS4 := VectorSpace(k,4);

/*
hyp: set of points of hyperoval
P: PG(3,q)\PG(2,q) as an enumerated sequence
*/

hyp := {v : v in VS4 | v[1] eq 0 and v[2] eq 1 and v[4] eq v[3]^2};
Include(~hyp, VS4![0,0,0,1]);
Include(~hyp, VS4![0,0,1,0]);
P := Setseq({v: v in VS4 | v[1] eq 1});

/*
line: returns points on a line, given v,w on the line
*/
line := function(v,w);
        l := {};
        w1:=Normalize(w);
        Include(~l,w1);
        for a in k do
                j := v + a*w;
                s := VS4!j;
                t := Normalize(s);
                Include(~l, t);
        end for;
```

```
return l;
end function;

/*
Lines: set of all lines not in PG(2,q)
hyplines: Lines that contain a point of the hyperoval
B: hyplines expressed in terms of P
*/
Lines := {line(v,w) : v,w in VS4 | Normalize(v) ne Normalize(w)
                                                  and w[1] ne 0 and v ne 0};
hyplines := {l: l in Lines | #(l meet hyp) ne 0};
B := {{i:i in [1..#P] | P[i] in l} : l in hyplines};

S := IncidenceStructure<#P|B>;
C := Dual(LinearCode(ChangeRing(IncidenceMatrix(S),GF(2))));
M:=ChangeRing(IncidenceMatrix(S),GF(2));

print "q: " cat IntegerToString(q);
print "length: " cat IntegerToString(#B);
print "dimension: " cat IntegerToString(Dimension(C));
print "minimum distance: " cat IntegerToString(MinimumDistance(C));

/*
Print the matrix to file in alist format
See MacKay http://www.inference.phy.cam.ac.uk/mackay/codes/alist.html
*/

col_weight := q;
row_weight := q+2;

f := Open("alist", "w");

Puts(f, Sprint(#B) cat " " cat Sprint(#P));
Puts(f, Sprint(col_weight) cat " " cat Sprint(row_weight));

s := Sprint(row_weight) cat " ";
Put(f, s^#P);
```

```
Put(f, "\n");

s := Sprint(col_weight) cat " ";
Put(f, s^#B);
Put(f, "\n");

for i in [1..#P] do
        temp_vector := M[i];
        for j in [1..#B] do
                if temp_vector[j] eq 1 then
                        Put(f,Sprint(j) cat " ");
                else
                end if;
        end for;
        Put(f, "\n");
end for;

for i in [1..#B] do
        temp_vector := Transpose(M)[i];
        for j in [1..#P] do
if temp_vector[j] eq 1 then
Put(f,Sprint(j) cat " ");
else
end if;
end for;
Put(f, "\n");
end for;
```

8.5 $\mathcal{C}_{Q_q^-}$

```
/*
Program to find dimension of code obtained from
linear representation of elliptic quadric. (We know
that the minimum distance is 2q, so no need to find
this.)
*/

print "Input q";
read q1;
q:=StringToInteger(q1);
k := GF(q);

/*
Given q, calculate d such that dx^2 + xy + y^2 is irreducible.
For even q, trace d equals 1. For odd q, 1-4d is a non-square.
*/

if IsEven(q) and not IsSquare(q) then
        d:=1;
elif IsEven(q) and IsSquare(q) then
        i:=1;
        g:=PrimitiveElement(k);
        while Trace(g^i) ne 1 do
        i:=i+1;
        end while;
        d:=g^i;
else
        g:=PrimitiveElement(k);
        i:=q-1;
        while IsSquare(1-4*g^i) do
        i:=i-1;
        end while;
        d:=g^i;
end if;
```

```
VS5 := VectorSpace(k, 5);

/*
Q: set of points of elliptic quadric in PG(3,q)
P: PG(4,q)\PG(3,q) as an enumerated sequence
*/

Q := {Normalize(v) : v in VS5 | v ne 0 and v[1] eq 0 and
d*v[2]^2 + v[2]*v[3] + v[3]^2 + v[4]*v[5] eq 0};
P := Setseq({v: v in VS5 | v[1] eq 1});

/*
line: returns points on a line, given v,w on the line
*/
line := function(v,w);
l := {};
w1:=Normalize(w);
Include(~l,w1);
for a in k do
j := v + a*w;
s := VS5!j;
t := Normalize(s);
Include(~l, t);
end for;
return l;
end function;

/*
Lines: set of all lines not in PG(3,q)
QLines: Lines that contain a point of the elliptic quadric
B: QLines expressed in terms of P
*/
Lines := {line(v,w) : v,w in VS5 | Normalize(v) ne Normalize(w)
and w[1] ne 0 and v ne 0};
QLines := {l: l in Lines | #(l meet Q) ne 0};
B := {{i:i in [1..#P] | P[i] in l} : l in QLines};
```

```
S := IncidenceStructure<#P|B>;
C:=Dual(LinearCode(ChangeRing(IncidenceMatrix(S),GF(2))));
M:=ChangeRing(IncidenceMatrix(S),GF(2));

print "q: " cat IntegerToString(q);
print "length: " cat IntegerToString(#B);
print "dimension: " cat IntegerToString(Dimension(C));

/*
Print the matrix to file in alist format
See MacKay http://www.inference.phy.cam.ac.uk/mackay/codes/alist.html
*/

col_weight := q;
row_weight := q^2+1;

f := Open("alist", "w");

Puts(f, Sprint(#B) cat " " cat Sprint(#P));
Puts(f, Sprint(col_weight) cat " " cat Sprint(row_weight));

s := Sprint(row_weight) cat " ";
Put(f, s^#P);
Put(f, "\n");

s := Sprint(col_weight) cat " ";
Put(f, s^#B);
Put(f, "\n");

for i in [1..#P] do
        temp_vector := M[i];
        for j in [1..#B] do
                if temp_vector[j] eq 1 then
                        Put(f,Sprint(j) cat " ");
                else
                end if;
```

```
        end for;
        Put(f, "\n");
end for;

for i in [1..#B] do
        temp_vector := Transpose(M)[i];
        for j in [1..#P] do
if temp_vector[j] eq 1 then
Put(f,Sprint(j) cat " ");
else
end if;
end for;
Put(f, "\n");
end for;
```

8.6 $\mathcal{C}_{TO_q}$

```
/*
Program to find dimension of code obtained from
linear representation of Tits ovoid. (We know
that the minimum distance is 2q, so no need to find
this.)
*/

print "q of form 2^(2e+1), e>0. Input e:";
read e1;
e:=StringToInteger(e1);
print "input q:";
read q1;
q:=StringToInteger(q1);
k := GF(q);

VS5 := VectorSpace(k, 5);

/*
TO: set of points of Tits ovoid in PG(3,q)
P: PG(4,q)\PG(3,q) as an enumerated sequence
*/

TO := {Normalize(v) : v in VS5 | v ne 0 and v[1] eq 0 and v[2] eq 1 and
v[3] eq (v[5]*v[4] + v[5]^(e+3) + v[4] ^(e+1))};
P := Setseq({v: v in VS5 | v[1] eq 1});
Include(~TO, VS5![0,0,1,0,0]);

/*
line: returns points on a line, given v,w on the line
*/
line := function(v,w);
l := {};
w1:=Normalize(w);
Include(~l,w1);
for a in k do
```

```
j := v + a*w;
s := VS5!j;
t := Normalize(s);
Include(~l, t);
end for;
return l;
end function;

/*
Lines: set of all lines not in PG(3,q)
TOLines: Lines that contain a point of the elliptic quadric
B: TOLines expressed in terms of P
*/
Lines := {line(v,w) : v,w in VS5 | Normalize(v) ne Normalize(w)
and w[1] ne 0 and v ne 0};
TOLines := {l: l in Lines | #(l meet TO) ne 0};
B := {{i:i in [1..#P] | P[i] in l} : l in TOLines};

S := IncidenceStructure<#P|B>;
C:=Dual(LinearCode(ChangeRing(IncidenceMatrix(S),GF(2))));
M:=ChangeRing(IncidenceMatrix(S),GF(2));

print "q: " cat IntegerToString(q);
print "length: " cat IntegerToString(#B);
print "dimension: " cat IntegerToString(Dimension(C));

/*
Print the matrix to file in alist format
See MacKay http://www.inference.phy.cam.ac.uk/mackay/codes/alist.html
*/

col_weight := q;
row_weight := q^2+1;

f := Open("alist", "w");

Puts(f, Sprint(#B) cat " " cat Sprint(#P));
```

```
Puts(f, Sprint(col_weight) cat " " cat Sprint(row_weight));

s := Sprint(row_weight) cat " ";
Put(f, s^#P);
Put(f, "\n");

s := Sprint(col_weight) cat " ";
Put(f, s^#B);
Put(f, "\n");

for i in [1..#P] do
        temp_vector := M[i];
        for j in [1..#B] do
                if temp_vector[j] eq 1 then
                        Put(f,Sprint(j) cat " ");
                else
                end if;
        end for;
        Put(f, "\n");
end for;

for i in [1..#B] do
        temp_vector := Transpose(M)[i];
        for j in [1..#P] do
                if temp_vector[j] eq 1 then
                        Put(f,Sprint(j) cat " ");
                else
                end if;
        end for;
        Put(f, "\n");
end for;
```

8.7 $\mathcal{C}_{C_{11}}$

```
/*
Program to find minimum distance and dimension of code
obtained from linear representation of 11-cap in PG(4,3)
*/

k:= GF(3);
VS6 := VectorSpace(k,6);

/*
c11: set of points of the 11-cap
Quadrilateral is {(0,0,0,1,0,0), (0,0,0,0,1,0),
(0,0,0,0,0,1), (0,0,0,1,1,1)}, with diagonal points
(0,0,0,0,1,1), (0,0,0,1,1,0),(0,0,0,1,0,1).
Line not in plain: {(0,1,0,0,0,0), (0,0,1,0,0,0),
(0,1,1,0,0,0),(0,1,2,0,0,0)}.

P: PG(5,3)\PG(4,3) as an enumerated sequence
*/

c11 := {VS6![0,0,0,0,1,1],
VS6![0,0,0,1,1,0],
VS6![0,0,0,1,0,1],
VS6![0,1,0,1,0,0],
VS6![0,1,0,2,0,0],
VS6![0,0,1,0,1,0],
VS6![0,0,1,0,2,0],
VS6![0,1,1,0,0,1],
VS6![0,1,1,0,0,2],
VS6![0,1,2,1,1,1],
VS6![0,1,2,2,2,2]};

P := Setseq({v: v in VS6 | v[1] eq 1});

/*
line: returns points on a line, given v,w on the line
```

```
*/

line := function(v,w);
        l := {};
        w1:=Normalize(w);
        Include(~l,w1);
        for a in k do
                j := v + a*w;
                s := VS6!j;
                t := Normalize(s);
                Include(~l, t);
        end for;
return l;
end function;

/*
L: set of all lines not in PG(4,q)
c11lines: element of L that contain a point of the 11-cap
B: c11lines  expressed in terms of P
*/

L := {line(v,w) : v,w in VS6 | Normalize(v) ne Normalize(w)
                                                and w[1] ne 0 and v ne 0};
c11lines := {l: l in L | #(l meet c11) ne 0};
B := {{i:i in [1..#P] | P[i] in l} : l in c11lines};

S := IncidenceStructure<#P|B>;
C := Dual(LinearCode(ChangeRing(IncidenceMatrix(S),GF(2))));
M:=ChangeRing(IncidenceMatrix(S),GF(2));

print "length: " cat IntegerToString(#B);
print "dimension: " cat IntegerToString(Dimension(C));
print "minimum distance: " cat IntegerToString(MinimumDistance(C));

/*
Print the matrix to file in alist format
See MacKay http://www.inference.phy.cam.ac.uk/mackay/codes/alist.html
```

```
*/

col_weight := 3;
row_weight := 11;

f := Open("alist", "w");

Puts(f, Sprint(#B) cat " " cat Sprint(#P));
Puts(f, Sprint(col_weight) cat " " cat Sprint(row_weight));

s := Sprint(row_weight) cat " ";
Put(f, s^#P);
Put(f, "\n");

s := Sprint(col_weight) cat " ";
Put(f, s^#B);
Put(f, "\n");

for i in [1..#P] do
        temp_vector := M[i];
        for j in [1..#B] do
                if temp_vector[j] eq 1 then
                        Put(f,Sprint(j) cat " ");
                else
                end if;
        end for;
        Put(f, "\n");
end for;

for i in [1..#B] do
        temp_vector := Transpose(M)[i];
        for j in [1..#P] do
                if temp_vector[j] eq 1 then
                        Put(f,Sprint(j) cat " ");
                else
                end if;
        end for;
```

```
        Put(f, "\n");
end for;
```

8.8 $\mathcal{C}_{C_{56}}$

```
/*
Program to find minimum distance and dimension of code
obtained from linear representation of 56-cap in PG(5,3)
*/

k:= GF(3);
VS7 := VectorSpace(k,7);

/*
c56: points of the 56-cap. (See Hill's
article, On the largest size of cap in S_{5,3}).
P: PG(6,3)\PG(5,3) as an enumerated sequence
*/

c56 := {
VS7![0,1,2,0,0,0,0],
VS7![0,0,1,2,0,0,0],
VS7![0,0,0,1,2,0,0],
VS7![0,0,0,0,1,2,0],
VS7![0,0,0,0,0,1,2],
VS7![0,1,1,1,1,1,2],
VS7![0,1,2,2,2,2,2],
VS7![0,1,0,2,0,2,0],
VS7![0,0,1,0,2,0,2],
VS7![0,1,1,2,1,0,1],
VS7![0,1,0,0,2,0,1],
VS7![0,1,0,1,1,2,1],
VS7![0,1,0,1,0,0,2],
VS7![0,1,2,1,2,1,1],
VS7![0,1,1,2,0,0,0],
VS7![0,0,1,1,2,0,0],
VS7![0,0,0,1,1,2,0],
VS7![0,0,0,0,1,1,2],
VS7![0,1,1,1,1,2,2],
VS7![0,1,2,2,2,2,0],
```

```
VS7![0,0,1,2,2,2,2],
VS7![0,1,1,0,2,0,1],
VS7![0,1,0,0,1,2,1],
VS7![0,1,0,1,1,0,2],
VS7![0,1,2,1,2,2,1],
VS7![0,1,0,2,0,2,2],
VS7![0,1,2,1,0,1,0],
VS7![0,0,1,2,1,0,1],
VS7![0,1,1,0,2,0,2],
VS7![0,1,2,2,1,0,1],
VS7![0,1,0,2,2,0,1],
VS7![0,1,0,1,2,2,1],
VS7![0,1,0,1,0,2,2],
VS7![0,1,2,1,2,1,0],
VS7![0,0,1,2,1,2,1],
VS7![0,1,1,1,2,0,0],
VS7![0,0,1,1,1,2,0],
VS7![0,0,0,1,1,1,2],
VS7![0,1,1,1,2,2,2],
VS7![0,1,2,2,2,0,0],
VS7![0,0,1,2,2,2,0],
VS7![0,0,0,1,2,2,2],
VS7![0,1,1,2,2,0,0],
VS7![0,0,1,1,2,2,0],
VS7![0,0,0,1,1,2,2],
VS7![0,1,1,1,2,2,0],
VS7![0,0,1,1,1,2,2],
VS7![0,1,1,2,2,2,0],
VS7![0,0,1,1,2,2,2],
VS7![0,1,2,2,0,2,1],
VS7![0,1,0,2,2,1,2],
VS7![0,1,2,1,0,0,2],
VS7![0,1,2,0,2,1,1],
VS7![0,1,0,2,1,2,0],
VS7![0,0,1,0,2,1,2],
VS7![0,1,1,2,1,0,2]
};
```

```
P := Setseq({v: v in VS7 | v[1] eq 1});

/*
line: returns points on a line, given v,w on the line
*/

line := function(v,w);
        l := {};
        w1:=Normalize(w);
        Include(~l,w1);
        for a in k do
                j := v + a*w;
                s := VS7!j;
                t := Normalize(s);
                Include(~l, t);
        end for;
return l;
end function;

/*
L: set of all lines not in PG(5,3)
c56lines: elements of L that contain a point of the 56-cap
B: c56lines  expressed in terms of P
*/

L := {line(v,w) : v,w in VS7 | Normalize(v) ne Normalize(w)
                                                and w[1] ne 0 and v ne 0};
c56lines := {l: l in L | #(l meet c56) ne 0};
B := {{i:i in [1..#P] | P[i] in l} : l in c56lines};

S := IncidenceStructure<#P|B>;
C := Dual(LinearCode(ChangeRing(IncidenceMatrix(S),GF(2))));
M:=ChangeRing(IncidenceMatrix(S),GF(2));

print "length: " cat IntegerToString(#B);
print "dimension: " cat IntegerToString(Dimension(C));
```

```
print "minimum distance: " cat IntegerToString(MinimumDistance(C));

/*
Print the matrix to file in alist format
See MacKay http://www.inference.phy.cam.ac.uk/mackay/codes/alist.html
*/

col_weight := 3;
row_weight := 56;

f := Open("alist", "w");

Puts(f, Sprint(#B) cat " " cat Sprint(#P));
Puts(f, Sprint(col_weight) cat " " cat Sprint(row_weight));

s := Sprint(row_weight) cat " ";
Put(f, s^#P);
Put(f, "\n");

s := Sprint(col_weight) cat " ";
Put(f, s^#B);
Put(f, "\n");

for i in [1..#P] do
        temp_vector := M[i];
        for j in [1..#B] do
                if temp_vector[j] eq 1 then
                        Put(f,Sprint(j) cat " ");
                else
                end if;
        end for;
        Put(f, "\n");
end for;

for i in [1..#B] do
        temp_vector := Transpose(M)[i];
        for j in [1..#P] do
```

```
                if temp_vector[j] eq 1 then
                        Put(f,Sprint(j) cat " ");
                else
                end if;
        end for;
        Put(f, "\n");
end for;
```

8.9 $\mathcal{C}_{C_{78}}$

```
/*
Program to find minimum distance and dimension of code
obtained from linear representation of 78-cap in PG(5,4)
*/

k:= GF(4);
a:=PrimitiveElement(k);
b:=a^2;

VS7 := VectorSpace(k,7);

/*
c78: points of the 78-cap.
P: PG(6,4)\PG(5,4) as an enumerated sequence
*/

c78 := {
VS7![0,1,0,0,0,0,0],
VS7![0,1,b,b,a,1,0],
VS7![0,0,0,1,b,1,0],
VS7![0,1,a,1,0,b,1],
VS7![0,1,1,a,0,a,a],
VS7![0,1,0,a,1,a,b],
VS7![0,1,a,1,a,1,0],
VS7![0,1,0,b,1,b,0],
VS7![0,0,0,1,b,a,a],
VS7![0,0,0,0,1,a,a],
VS7![0,0,1,0,0,1,1],
VS7![0,1,1,a,1,a,0],
VS7![0,0,0,1,1,0,0],
VS7![0,1,1,b,a,0,b],
VS7![0,1,0,b,1,0,0],
VS7![0,1,1,b,1,a,b],
VS7![0,1,b,1,b,b,a],
VS7![0,1,b,a,0,1,b],
```

```
VS7![0,1,1,b,0,1,0],
VS7![0,1,1,a,a,a,0],
VS7![0,1,0,1,a,a,1],
VS7![0,1,1,a,0,0,b],
VS7![0,1,b,b,0,1,1],
VS7![0,1,0,0,1,b,0],
VS7![0,1,a,0,1,0,b],
VS7![0,1,a,0,0,a,b],
VS7![0,1,a,b,a,1,a],
VS7![0,1,b,a,a,1,1],
VS7![0,1,1,0,b,1,a],
VS7![0,1,0,b,1,0,1],
VS7![0,0,1,1,0,0,1],
VS7![0,1,b,0,1,b,0],
VS7![0,1,1,a,b,1,0],
VS7![0,1,a,0,b,0,0],
VS7![0,1,b,0,a,a,b],
VS7![0,1,1,a,a,0,a],
VS7![0,1,a,0,b,0,b],
VS7![0,1,1,1,0,a,0],
VS7![0,0,1,b,1,a,a],
VS7![0,1,b,a,0,1,1],
VS7![0,1,b,1,a,0,b],
VS7![0,1,a,0,1,0,a],
VS7![0,1,b,1,b,a,0],
VS7![0,1,0,b,1,a,b],
VS7![0,1,a,a,0,1,b],
VS7![0,1,b,a,1,a,a],
VS7![0,1,1,a,1,0,b],
VS7![0,0,1,0,0,1,0],
VS7![0,1,a,a,1,a,1],
VS7![0,1,b,1,a,b,1],
VS7![0,1,a,b,b,0,a],
VS7![0,1,b,0,a,1,0],
VS7![0,1,a,0,1,a,a],
VS7![0,1,a,a,b,1,1],
VS7![0,1,1,0,b,a,b],
```

```
VS7![0,1,b,a,0,b,1],
VS7![0,1,0,b,0,1,1],
VS7![0,1,1,b,a,1,1],
VS7![0,1,a,0,a,a,0],
VS7![0,0,1,a,b,a,b],
VS7![0,0,1,b,0,1,1],
VS7![0,1,b,1,0,0,1],
VS7![0,1,a,1,0,0,1],
VS7![0,1,1,b,a,1,a],
VS7![0,1,1,b,b,a,b],
VS7![0,1,1,0,1,0,0],
VS7![0,0,1,a,b,b,0],
VS7![0,1,1,b,a,b,1],
VS7![0,1,b,0,0,0,b],
VS7![0,1,0,0,a,b,1],
VS7![0,1,b,0,0,1,b],
VS7![0,0,1,0,1,0,b],
VS7![0,1,b,b,0,b,b],
VS7![0,1,a,a,a,a,1],
VS7![0,1,a,a,1,1,1],
VS7![0,0,1,b,b,0,a],
VS7![0,0,1,a,b,a,1],
VS7![0,1,a,b,1,0,0]
};

P := Setseq({v: v in VS7 | v[1] eq 1});

/*
line: returns points on a line, given v,w on the line
*/

line := function(v,w);
        l := {};
        w1:=Normalize(w);
        Include(~l,w1);
        for a in k do
                j := v + a*w;
```

```
                s := VS7!j;
                t := Normalize(s);
                Include(~l, t);
        end for;
return l;
end function;

/*
L: set of all lines not in PG(5,4)
c78lines: elements of L that contain a point of the 78-cap
B: c78lines  expressed in terms of P
*/

L := {line(v,w) : v,w in VS7 | Normalize(v) ne Normalize(w)
                                                and w[1] ne 0 and v ne 0};
c78lines := {l: l in L | #(l meet c78) ne 0};
B := {{i:i in [1..#P] | P[i] in l} : l in c78lines};

S := IncidenceStructure<#P|B>;
C := Dual(LinearCode(ChangeRing(IncidenceMatrix(S),GF(2))));
M:=ChangeRing(IncidenceMatrix(S),GF(2));

print "length: " cat IntegerToString(#B);
print "dimension: " cat IntegerToString(Dimension(C));
print "minimum distance: " cat IntegerToString(MinimumDistance(C));

/*
Print the matrix to file in alist format
See MacKay http://www.inference.phy.cam.ac.uk/mackay/codes/alist.html
*/

col_weight := 4;
row_weight := 78;

f := Open("alist", "w");

Puts(f, Sprint(#B) cat " " cat Sprint(#P));
```

```
Puts(f, Sprint(col_weight) cat " " cat Sprint(row_weight));

s := Sprint(row_weight) cat " ";
Put(f, s^#P);
Put(f, "\n");

s := Sprint(col_weight) cat " ";
Put(f, s^#B);
Put(f, "\n");

for i in [1..#P] do
        temp_vector := M[i];
        for j in [1..#B] do
                if temp_vector[j] eq 1 then
                        Put(f,Sprint(j) cat " ");
                else
                end if;
        end for;
        Put(f, "\n");
end for;

for i in [1..#B] do
        temp_vector := Transpose(M)[i];
        for j in [1..#P] do
                if temp_vector[j] eq 1 then
                        Put(f,Sprint(j) cat " ");
                else
                end if;
        end for;
        Put(f, "\n");
end for;
```

8.10 $\mathcal{C}_{U_{2,q^2}}$

```
/*
Program to find minimum distance and dimension of code
obtained from linear representation of Hermitian curve
in PG(2,q^2)
*/

print "Input characteristic p";
read p1;
p:=StringToInteger(p1);
print "Input h (even)";
read h1;
print "Input h/2";
read h2;
h:=StringToInteger(h1);
h3:= StringToInteger(h2);
k := GF(p^h);
r:=p^h3;

VS4 := VectorSpace(k, 4);

/*
H: set of points of Hermitian curve in PG(2,q^2)
P: PG(3,q^2)\PG(2,q^2) as an enumerated sequence
*/

H := {Normalize(v) : v in VS4 | v ne 0 and v[1] eq 0 and
(v[2]^(r+1) + v[3]^(r+1) + v[4]^(r+1)) eq 0};
P := Setseq({v: v in VS4 | v[1] eq 1});

/*
line: returns points on a line, given v,w on the line
*/
line := function(v,w);
l := {};
w1:=Normalize(w);
```

```
Include(~l,w1);
for a in k do
j := v + a*w;
s := VS4!j;
t := Normalize(s);
Include(~l, t);
end for;
return l;
end function;

/*
Lines: set of all lines not in PG(3,q^2)
HLines: Lines that contain a point of the Hermitian curve
B: HLines expressed in terms of P
*/
Lines := {line(v,w) : v,w in VS4 | Normalize(v) ne Normalize(w)
and w[1] ne 0 and v ne 0};
HLines := {l: l in Lines | #(l meet H) ne 0};
B := {{i:i in [1..#P] | P[i] in l} : l in HLines};

S := IncidenceStructure<#P|B>;
C:=Dual(LinearCode(ChangeRing(IncidenceMatrix(S),GF(2))));
M:=ChangeRing(IncidenceMatrix(S),GF(2));

print "q^2: " cat IntegerToString(p^h);
print "length: " cat IntegerToString(#B);
print "dimension: " cat IntegerToString(Dimension(C));
print "minimum distance: " cat IntegerToString(MinimumDistance(C));

/*
Print the matrix to file in alist format
See MacKay http://www.inference.phy.cam.ac.uk/mackay/codes/alist.html
*/

col_weight := r^2;
row_weight := r^3+1;
```

```
f := Open("hermalist", "w");

Puts(f, Sprint(#B) cat " " cat Sprint(#P));
Puts(f, Sprint(col_weight) cat " " cat Sprint(row_weight));

s := Sprint(row_weight) cat " ";
Put(f, s^#P);
Put(f, "\n");

s := Sprint(col_weight) cat " ";
Put(f, s^#B);
Put(f, "\n");

for i in [1..#P] do
        temp_vector := M[i];
        for j in [1..#B] do
                if temp_vector[j] eq 1 then
                        Put(f,Sprint(j) cat " ");
                else
                end if;
        end for;
        Put(f, "\n");
end for;

for i in [1..#B] do
        temp_vector := Transpose(M)[i];
        for j in [1..#P] do
                if temp_vector[j] eq 1 then
                        Put(f,Sprint(j) cat " ");
                else
                end if;
        end for;
        Put(f, "\n");
end for;
```

8.11 $\mathcal{C}_{B_{2,q^2}}$

```
/*
Program to find minimum distance and dimension of code
obtained from linear representation of Baer subplane
of PG(2,q^2)
*/

print "Input characteristic p";
read p1;
p:=StringToInteger(p1);
print "Input h (even)";
read h1;
print "Input h/2";
read h2;
h:=StringToInteger(h1);
h3:= StringToInteger(h2);
b := GF(p^h3);
k := ext< b|2>;

VS4k := VectorSpace(k, 4);
VS4b := VectorSpace(b, 4);

/*
Baer: set of points of Baer subplane  in PG(2,q^2)
P: PG(3,q^2)\PG(2,q^2) as an enumerated sequence
*/

Baer := {Normalize(v) : v in VS4b | v ne 0 and v[1] eq 0};
P := Setseq({v: v in VS4k | v[1] eq 1});

/*
line: returns points on a line, given v,w on the line
*/
line := function(v,w);
l := {};
w1:=Normalize(w);
```

```
Include(~l,w1);
for a in k do
j := v + a*w;
s := VS4k!j;
t := Normalize(s);
Include(~l, t);
end for;
return l;
end function;

/*
Lines: set of all lines not in PG(2,q^2)
BaerLines: Lines that contain a point of the Baer subplane
B: BaerLines expressed in terms of P
*/
Lines := {line(v,w) : v,w in VS4k | Normalize(v) ne Normalize(w)
and w[1] ne 0 and v ne 0};
BaerLines := {l: l in Lines | #(l meet Baer) ne 0};
B := {{i:i in [1..#P] | P[i] in l} : l in BaerLines};

S := IncidenceStructure<#P|B>;
C:=Dual(LinearCode(ChangeRing(IncidenceMatrix(S),GF(2))));
M:=ChangeRing(IncidenceMatrix(S),GF(2));

print "q^2: " cat IntegerToString(p^h);
print "length: " cat IntegerToString(#B);
print "dimension: " cat IntegerToString(Dimension(C));
print "minimum distance: " cat IntegerToString(MinimumDistance(C));

/*
Print the matrix to file in alist format
See MacKay http://www.inference.phy.cam.ac.uk/mackay/codes/alist.html
*/

col_weight := p^h;
row_weight := p^h + p^h3 +1;
```

```
f := Open("baeralist", "w");

Puts(f, Sprint(#B) cat " " cat Sprint(#P));
Puts(f, Sprint(col_weight) cat " " cat Sprint(row_weight));

s := Sprint(row_weight) cat " ";
Put(f, s^#P);
Put(f, "\n");

s := Sprint(col_weight) cat " ";
Put(f, s^#B);
Put(f, "\n");

for i in [1..#P] do
        temp_vector := M[i];
        for j in [1..#B] do
                if temp_vector[j] eq 1 then
                        Put(f,Sprint(j) cat " ");
                else
                end if;
        end for;
        Put(f, "\n");
end for;

for i in [1..#B] do
        temp_vector := Transpose(M)[i];
        for j in [1..#P] do
                if temp_vector[j] eq 1 then
                        Put(f,Sprint(j) cat " ");
                else
                end if;
        end for;
        Put(f, "\n");
end for;
```

8.12 $\mathcal{C}_{U_{2,3}(m)}$

```
/*
Print the matrix to file in alist format for the code U_{2,3}(n)
See MacKay http://www.inference.phy.cam.ac.uk/mackay/codes/alist.html
(Note that n in this program is the same as m in the thesis.)
*/

print "Input n";
read n1;
n := StringToInteger(n1);

S := {1..n};
sub2S := Subsets(S,2);
sub3S := Subsets(S,3);

P:=Setseq({v: v in sub2S});

B := {{i:i in [1..#P] | P[i] subset b} : b in sub3S};

S := IncidenceStructure<#P|B>;
C:=Dual(LinearCode(ChangeRing(IncidenceMatrix(S),GF(2))));
M:=ChangeRing(IncidenceMatrix(S),GF(2));

col_weight := 3;
row_weight := n-2;

f := Open("alist", "w");

Puts(f, Sprint(#B) cat " " cat Sprint(#P));
Puts(f, Sprint(col_weight) cat " " cat Sprint(row_weight));

s := Sprint(row_weight) cat " ";
Put(f, s^#P);
Put(f, "\n");

s := Sprint(col_weight) cat " ";
```

```
Put(f, s^#B);
Put(f, "\n");

for i in [1..#P] do
        temp_vector := M[i];
        for j in [1..#B] do
                if temp_vector[j] eq 1 then
                        Put(f,Sprint(j) cat " ");
                else
                end if;
        end for;
        Put(f, "\n");
end for;

for i in [1..#B] do
        temp_vector := Transpose(M)[i];
        for j in [1..#P] do
                if temp_vector[j] eq 1 then
                        Put(f,Sprint(j) cat " ");
                else
                end if;
        end for;
        Put(f, "\n");
end for;
```

8.13 $\mathcal{C}_{\overline{M(7)}}$

```
/*
Program to find dimension, minimum distance and weight
distribution of code obtained from the Moore graph M(7)
*/

P:=SetToIndexedSet(P1);
B := {
{1,11,14,26,38,46,49},{2,7,16,24,26,33,37},
{0,2,6,19,22,40,43},{0,4,21,25,27,30,35},
{1,3,9,13,25,31,47},{9,15,19,26,28,32,45},
{2,4,11,18,34,42,45},{13,24,27,29,34,40,49},
{3,5,8,23,26,40,48},{8,11,22,28,30,37,47},
{4,6,13,17,32,37,46},{6,18,26,29,31,41,44},
{1,5,7,10,15,30,34},{2,20,23,30,32,36,49},
{6,8,12,25,39,45,49},{5,11,27,31,33,39,43},
{4,7,9,14,20,29,43},{14,22,25,32,34,41,48},
{2,8,10,17,27,38,41},{3,6,20,28,33,35,38},
{6,9,11,21,24,36,48},{12,17,26,34,36,43,47},
{0,3,10,12,16,29,32},{10,14,31,35,37,40,45},
{7,11,13,19,23,35,41},{5,19,25,29,36,38,42},
{2,5,12,14,21,28,44},{0,9,23,34,37,39,44},
{0,8,13,15,18,33,36},{7,18,21,32,38,40,47},
{6,14,16,23,27,42,47},{1,4,16,28,36,39,41},
{11,15,17,20,25,40,44},{9,12,30,33,40,42,46},
{5,9,16,18,22,35,49},{3,15,21,37,41,43,49},
{3,14,17,19,24,30,39},{1,8,24,32,35,42,44},
{1,12,18,20,27,37,48},{13,16,30,38,43,45,48},
{8,16,19,21,31,34,46},{3,7,22,27,36,44,46},
{10,13,20,22,26,39,42},{0,5,20,24,41,45,47},
{1,17,21,23,29,33,45},{2,15,29,35,39,46,48},
{4,12,15,22,24,31,38},{4,10,19,33,44,47,49},
{10,18,23,25,28,43,46},{0,7,17,28,31,42,48}
};
```

```
S := IncidenceStructure<P|B>;
C:=Dual(LinearCode(ChangeRing(IncidenceMatrix(S),GF(2))));
M:=ChangeRing(IncidenceMatrix(S),GF(2));

print "weight distribution:";
print WeightDistribution(C);
print "dimension: " cat IntegerToString(Dimension(C));
print "minimum distance: " cat IntegerToString(MinimumDistance(C));

/*
Print the matrix to file in alist format
See MacKay http://www.inference.phy.cam.ac.uk/mackay/codes/alist.html
*/

col_weight := 7;
row_weight := 7;

f := Open("alist", "w");

Puts(f, Sprint(#B) cat " " cat Sprint(#P));
Puts(f, Sprint(col_weight) cat " " cat Sprint(row_weight));

s := Sprint(row_weight) cat " ";
Put(f, s^#P);
Put(f, "\n");

s := Sprint(col_weight) cat " ";
Put(f, s^#B);
Put(f, "\n");

for i in [1..#P] do
        temp_vector := M[i];
        for j in [1..#B] do
                if temp_vector[j] eq 1 then
                        Put(f,Sprint(j) cat " ");
                else
                end if;
```

```
        end for;
        Put(f, "\n");
end for;

for i in [1..#B] do
        temp_vector := Transpose(M)[i];
        for j in [1..#P] do
                if temp_vector[j] eq 1 then
                        Put(f,Sprint(j) cat " ");
                else
                end if;
        end for;
        Put(f, "\n");
end for;

//end for;
```

8.14 $\mathcal{C}_{\mathrm{LP}(n,q)}$

```
/*
Program to find dimension and minimum distance of code
obtained from LP(n,q)
*/

print "Input q";
read q1;
q := StringToInteger(q1);
k := GF(q);
print "Input n>=5, (n-1) is dimension of the projective space";
read n1;
n := StringToInteger(n1);
V:=VectorSpace(k,n);
Vnorm := {v:v in V | v eq Normalize(v) and v ne 0};

/*
line: returns points on a line, given independent v,w on the line
*/
line := function(v,w);
        l := {};
        w1:=Normalize(w);
        Include(~l,w1);
        for a in k do
                j := v + a*w;
                s := V!j;
                t := Normalize(s);
                Include(~l, t);
        end for;
return l;
end function;

/*
plane: returns points on a plane, given independent v,w,x on the plane
*/
plane := function(v,w,x);
```

```
p := {};
l := line(v,w);
for a in l do
j := line(a,x);
for m in j do
n := V!m;
q := Normalize(n);
Include(~p,q);
end for;
end for;
return p;
end function;

/*
P1: set of lines of PG(n,q)
P: P1 as an enumerated sequence
*/
P1 := {line(v,w) : v,w in Vnorm | v ne w};
P := Setseq({p: p in P1});

/*
B1: set of planes of PG(n,q)
B: B1 expressed in terms of P
*/

B1 := {plane(v,w,x) : v,w,x in Vnorm | v ne w and x notin line (v,w)};
B := {{i:i in [1..#P] | P[i] subset b} : b in B1};

S := IncidenceStructure<#P|B>;
C := Dual(LinearCode(ChangeRing(IncidenceMatrix(S),GF(2))));
M:=ChangeRing(IncidenceMatrix(S),GF(2));

print "q: " cat IntegerToString(q);
print "length: " cat IntegerToString(#B);
print "dimension: " cat IntegerToString(Dimension(C));
print "minimum distance: " cat IntegerToString(MinimumDistance(C));
```

```
/*
Print the matrix to file in alist format
See MacKay http://www.inference.phy.cam.ac.uk/mackay/codes/alist.html
*/

col_weight := ((q^(n-2)-1)/(q-1));
row_weight := q^2+q+1;

f := Open("alist", "w");

Puts(f, Sprint(#B) cat " " cat Sprint(#P));
Puts(f, Sprint(col_weight) cat " " cat Sprint(row_weight));

s := Sprint(row_weight) cat " ";
Put(f, s^#P);
Put(f, "\n");

s := Sprint(col_weight) cat " ";
Put(f, s^#B);
Put(f, "\n");

for i in [1..#P] do
        temp_vector := M[i];
        for j in [1..#B] do
                if temp_vector[j] eq 1 then
                        Put(f,Sprint(j) cat " ");
                else
                end if;
        end for;
        Put(f, "\n");
end for;

for i in [1..#B] do
        temp_vector := Transpose(M)[i];
        for j in [1..#P] do
if temp_vector[j] eq 1 then
Put(f,Sprint(j) cat " ");
```

```
else
end if;
end for;
Put(f, "\n");
end for;
```

8.15 $\mathcal{C}_{\overline{W(2n+1,q)}}$

```
/*
Program to find dimension and minimum distance of code
obtained from symplectic polarity
*/

print "Input q";
read q1;
q := StringToInteger(q1);
k := GF(q);
print "Input n>=3, n odd (dimension of the projective space)";
read n1;
n := StringToInteger(n1)+1;
V:=VectorSpace(k,n);
Vnorm := {v:v in V | v eq Normalize(v) and v ne 0};

/*
line: returns points on a line, given independent v,w on the line
*/
line := function(v,w);
        l := {};
        w1:=Normalize(w);
        Include(~l,w1);
        for a in k do
                j := v + a*w;
                s := V!j;
                t := Normalize(s);
                Include(~l, t);
        end for;
return l;
end function;

/*
hyp: returns points on polar prime of a point. Uses canonical form
for symplectic polarity, see JWPH Theorem 5.26
*/
```

```
hyp := function(v);
h := {};
for a in Vnorm do
j:=0;
i:=1;
while i lt n do
j:=j+ v[i]*a[i+1] - v[i+1]*a[i];
i:=i+2;
end while;
if j eq 0 then
 Include(~h,a);
end if;
end for;
return h;
end function;

/*
P1: point set of PG(n,q)
P: P1 as an enumerated sequence
*/
P1:=Vnorm;
P:=Setseq(P1);

/*
Lines: set of lines of PG(n,q)
*/
Lines := {line(v,w) : v,w in Vnorm | v ne w};

/*
B1: set of non totally isotropic lines of PG(n,q)
B: B1 expressed in terms of P
*/

B1 := {};
for l in Lines do
L := Setseq(l);
x := hyp(L[1]) meet hyp(L[2]);
```

```
if l notsubset x then
Include(~B1,l);
end if;
end for;
B := {{i:i in [1..#P] | P[i] in b} : b in B1};

S := IncidenceStructure<#P|B>;
C := Dual(LinearCode(ChangeRing(IncidenceMatrix(S),GF(2))));
M:=ChangeRing(IncidenceMatrix(S),GF(2));

print "q: " cat IntegerToString(q);
print "length: " cat IntegerToString(#B);
print "dimension: " cat IntegerToString(Dimension(C));
print "minimum distance: " cat IntegerToString(MinimumDistance(C));

/*
Print the matrix to file in alist format
See MacKay http://www.inference.phy.cam.ac.uk/mackay/codes/alist.html
*/

col_weight := q+1;
row_weight := q^(n-2);

f := Open("alist", "w");

Puts(f, Sprint(#B) cat " " cat Sprint(#P));
Puts(f, Sprint(col_weight) cat " " cat Sprint(row_weight));

s := Sprint(row_weight) cat " ";
Put(f, s^#P);
Put(f, "\n");

s := Sprint(col_weight) cat " ";
Put(f, s^#B);
Put(f, "\n");

for i in [1..#P] do
```

```
        temp_vector := M[i];
        for j in [1..#B] do
                if temp_vector[j] eq 1 then
                        Put(f,Sprint(j) cat " ");
                else
                end if;
        end for;
        Put(f, "\n");
end for;

for i in [1..#B] do
        temp_vector := Transpose(M)[i];
        for j in [1..#P] do
if temp_vector[j] eq 1 then
Put(f,Sprint(j) cat " ");
else
end if;
end for;
Put(f, "\n");
end for;
```

8.16 $\mathcal{C}_{NQ^+(2n-1,2)}$

```
/*
Program to find dimension of code obtained from
NQ+(2n-1,2)
*/

print "Input n, n >= 3";
read n1;
n:=StringToInteger(n1);

k := GF(2);


V := VectorSpace(k, 2*n);
PG := {v : v in V | v ne 0};

/*
offQ: set of points not on hyperbolic quadric in PG(2n-1,2)
P: offQ as an enumerated sequence
*/

offQ := {};
for v in PG do
        a:=k!0;
        for i in [1..n] do
                a:= a + (v[2*i]*v[(2*i)-1]);
        end for;
        if a eq 1 then
                Include(~offQ,Normalize(v));
        end if;
end for;

P := Setseq({v: v in offQ});


/*
```

```
Q: set of points of hyperbolic quadric
*/
Q := PG diff offQ;

/*
line: returns points on a line, given v,w on the line
*/
line := function(v,w);
        l := {};
        w1:=Normalize(w);
        Include(~l,w1);
        for a in k do
                j := v + a*w;
                s := V!j;
                t := Normalize(s);
                Include(~l, t);
        end for;
return l;
end function;

/*
Lines: lines of PG(2n-1,2)
offLines: Lines that do not intersect the quadric
B: offLines expressed in terms of P
*/
Lines := {line(v,w) : v,w in PG | v ne w};
offLines := {l: l in Lines | #(l meet Q) eq 0};
B := {{i:i in [1..#P] | P[i] in l} : l in offLines};

S := IncidenceStructure<#P|B>;
C := Dual(LinearCode(ChangeRing(IncidenceMatrix(S),GF(2))));
M:=ChangeRing(IncidenceMatrix(S),GF(2));

print "length: " cat IntegerToString(#B);
print "dimension: " cat IntegerToString(Dimension(C));
//print "minimum distance: " cat IntegerToString(MinimumDistance(C));
```

```
//Print the matrix to file in alist format - see MacKay

col_weight := 3;
row_weight := 2^(2*n-3) - 2^(n-2);

f := Open("alist", "w");

Puts(f, Sprint(#B) cat " " cat Sprint(#P));
Puts(f, Sprint(col_weight) cat " " cat Sprint(row_weight));

s := Sprint(row_weight) cat " ";
Put(f, s^#P);
Put(f, "\n");

s := Sprint(col_weight) cat " ";
Put(f, s^#B);
Put(f, "\n");

for i in [1..#P] do
        temp_vector := M[i];
        for j in [1..#B] do
                if temp_vector[j] eq 1 then
                        Put(f,Sprint(j) cat " ");
                else
                end if;
        end for;
        Put(f, "\n");
end for;

for i in [1..#B] do
        temp_vector := Transpose(M)[i];
        for j in [1..#P] do
                if temp_vector[j] eq 1 then
                        Put(f,Sprint(j) cat " ");
                else
                end if;
```

```
        end for;
        Put(f, "\n");
end for;
```

8.17 $\mathcal{C}_{NQ^-(2n-1,2)}$

```
/*
Program to find dimension of code obtained from
NQ-(2n-1,2)
*/

print "Input n, n>= 2";
read n1;
n:=StringToInteger(n1);

k := GF(2);


V := VectorSpace(k, 2*n);
PG := {v : v in V | v ne 0};

/*
offQ: set of points not on elliptic quadric in PG(2n-1,2)
Canonical form for elliptic quadric: X_0^2 + X_0X_1 + X_1^2 + X_2X_3...,
which since q=2 is simply X_0 + X_1 + SUM(X_iX_i+1) (see JWPH Th 5.16)
P: offQ as an enumerated sequence
*/

offQ := {};
for v in PG do
        a:=k!0;
        a:= v[1] + v[2];
        for i in [1..n] do
                a:= a + (v[2*i]*v[(2*i)-1]);
        end for;
        if a eq 1 then
                Include(~offQ,Normalize(v));
        end if;
end for;

P := Setseq({v: v in offQ});
```

```
/*
Q: set of points of elliptic quadric
*/
Q := PG diff offQ;

/*
line: returns points on a line, given v,w on the line
*/
line := function(v,w);
        l := {};
        w1:=Normalize(w);
        Include(~l,w1);
        for a in k do
                j := v + a*w;
                s := V!j;
                t := Normalize(s);
                Include(~l, t);
        end for;
return l;
end function;

/*
Lines: lines of PG(2n-1,2)
offLines: Lines that do not intersect the quadric
B: offLines expressed in terms of P
*/
Lines := {line(v,w) : v,w in PG | v ne w};
offLines := {l: l in Lines | #(l meet Q) eq 0};
B := {{i:i in [1..#P] | P[i] in l} : l in offLines};

S := IncidenceStructure<#P|B>;
C := Dual(LinearCode(ChangeRing(IncidenceMatrix(S),GF(2))));
M:=ChangeRing(IncidenceMatrix(S),GF(2));

print "length: " cat IntegerToString(#B);
```

```
print "dimension: " cat IntegerToString(Dimension(C));
print "minimum distance: " cat IntegerToString(MinimumDistance(C));


//Print the matrix to file in alist format - see MacKay

col_weight := 3;
row_weight := 2^(2*n-3) + 2^(n-2);

f := Open("alist", "w");

Puts(f, Sprint(#B) cat " " cat Sprint(#P));
Puts(f, Sprint(col_weight) cat " " cat Sprint(row_weight));

s := Sprint(row_weight) cat " ";
Put(f, s^#P);
Put(f, "\n");

s := Sprint(col_weight) cat " ";
Put(f, s^#B);
Put(f, "\n");

for i in [1..#P] do
        temp_vector := M[i];
        for j in [1..#B] do
                if temp_vector[j] eq 1 then
                        Put(f,Sprint(j) cat " ");
                else
                end if;
        end for;
        Put(f, "\n");
end for;

for i in [1..#B] do
        temp_vector := Transpose(M)[i];
        for j in [1..#P] do
                if temp_vector[j] eq 1 then
```

```
                        Put(f,Sprint(j) cat " ");
                else
                end if;
        end for;
        Put(f, "\n");
end for;
```

8.18 $\mathcal{C}_{H_q^{(n+1)*}}$

```
/*
Program to find dimension and minimum distance of code obtained
from H^(n+1)*_q
*/

print "Input n, n>= 2 (S=PG(n+1,q))";
read n1;
n:=StringToInteger(n1);

print "Input q";
read q1;
q:=StringToInteger(q1);

k := GF(q);

/*
S: PG(n+1,q)
H: PG(n-1,q)
*/
V := VectorSpace(k, n+2);

S := {Normalize(v) : v in V | v ne 0};
H := {v: v in S | v[1] eq 0 and v[2] eq 0};

/*
line: returns points on a line, given v,w on the line
*/
line := function(v,w);
        l := {};
        w1:=Normalize(w);
        Include(~l,w1);
        for a in k do
                j := v + a*w;
                s := V!j;
                t := Normalize(s);
```

```
                Include(~l, t);
        end for;
return l;
end function;

/*
plane: returns points on a plane, given independent v,w,x on the plane
*/
plane := function(v,w,x);
        p := {};
        l := line(v,w);
        for a in l do
                j := line(a,x);
                        for m in j do
                                n := V!m;
                                t := Normalize(n);
                                Include(~p,t);
                        end for;
        end for;
return p;
end function;

/*
P1: set of lines of PG(n+1,q)
P2: lines of PG(n+1,q) skew to H
P: P2 as an enumerated sequence
*/
P1 := {line(v,w) : v,w in S | v ne w};
P2:= {x : x in P1 | #(x meet H) eq 0};
P := Setseq({p: p in P2});

/*
B1: set of planes of PG(n+1,q)
B2: planes of PG(n+1,q) intersecting H in exactly one point
B: B1 expressed in terms of P
*/
```

```
B1 := {plane(v,w,x) : v,w,x in S | v ne w and x notin line (v,w)};
B2 := {x : x in B1 | #(x meet H) eq 1};
B := {{i:i in [1..#P] | P[i] subset b} : b in B2};

S := IncidenceStructure<#P|B>;
C := Dual(LinearCode(ChangeRing(IncidenceMatrix(S),GF(2))));
M:=ChangeRing(IncidenceMatrix(S),GF(2));

print "q: " cat IntegerToString(q);
print "length: " cat IntegerToString(#B);
print "dimension: " cat IntegerToString(Dimension(C));
print "minimum distance: " cat IntegerToString(MinimumDistance(C));

/*
Print the matrix to file in alist format
See MacKay http://www.inference.phy.cam.ac.uk/mackay/codes/alist.html
*/

col_weight := q^2;
row_weight := (q^n-1)/(q-1);

f := Open("alist", "w");

Puts(f, Sprint(#B) cat " " cat Sprint(#P));
Puts(f, Sprint(col_weight) cat " " cat Sprint(row_weight));

s := Sprint(row_weight) cat " ";
Put(f, s^#P);
Put(f, "\n");

s := Sprint(col_weight) cat " ";
Put(f, s^#B);
Put(f, "\n");

for i in [1..#P] do
        temp_vector := M[i];
        for j in [1..#B] do
```

```
                if temp_vector[j] eq 1 then
                        Put(f,Sprint(j) cat " ");
                else
                end if;
        end for;
        Put(f, "\n");
end for;

for i in [1..#B] do
        temp_vector := Transpose(M)[i];
        for j in [1..#P] do
                if temp_vector[j] eq 1 then
                        Put(f,Sprint(j) cat " ");
                else
                end if;
        end for;
        Put(f, "\n");
end for;
```

Bibliography

[1] S. Ball, A. Blokhuis, and F. Mazzocca, Maximal arcs in desarguesian planes of odd order do not exist, *Combinatorica* **17** (1997), 31–47.

[2] A. E. Brouwer and C. A. van Eijl, On the p-rank of strongly regular graphs, *Algebra and Combinatorics* **1** (1992), 72–82.

[3] F. Buekenhout (Ed.), *Handbook of Incidence Geometry*, North Holland, Amsterdam, 1995.

[4] R. Calderbank and W. Kantor, The geometry of two-weight codes, *Bull. London Math. Soc.* **18** (1986), 97–122.

[5] P.J. Cameron, Partial quadrangles, *Quart. J. Math. Oxford Ser. (2)* **26** (1975), 61–73.

[6] I. Debroey and J.A. Thas, On semipartial geometries, *J. Combin. Theory Ser. A* **25** (1978), 195–207.

[7] P. Dembowski, *Finite Geometries*, Springer, New York, 1968.

[8] R.H.F. Denniston, Some maximal arcs in finite projective planes, *J. Combin. Theory* **6** (1969), 317–319.

[9] R. Gallager, Low-density parity-check codes, *IRE Trans. Inform. Theory* **IT-8** (1962), 21–28.

[10] N. Hamilton and T. Penttila, Groups of maximal arcs, *J. Combin. Theory Ser. A* **94** (2001), 63–86.

[11] R. Hill, On the largest size of cap in $S_{5,3}$, *Rend. Accad. Naz. Lincei* **54** (1973), 378–384.

[12] ———, Caps and groups, *Atti dei convegni Lincei, Colloquio Internazionale sulle Teorie Combinatorie, Roma, 1973* **17** (1976), 389–394.

[13] J.W.P. Hirschfeld, *Projective Geometries over Finite Fields*, second ed., Oxford University Press, Oxford, 1998.

[14] J.W.P. Hirschfeld and J. A. Thas, *General Galois Geometries*, Oxford University Press, Oxford, 1991.

[15] S. Johnson and S. Weller, High-rate LDPC codes from unital designs, *Proceedings of the IEEE Globecom Conference, San Francisco, CA* **4** (2003), 2036–2040.

[16] ———, Codes for iterative decoding from partial geometries, *IEEE Trans. Comm.* **52** (2004), 236–243.

[17] J. Kim, K. Mellinger, and L. Storme, Small weight codewords in LDPC codes defined by (dual) classical generalized quadrangles, *Des. Codes Cryptogr.* **42** (2007), 73–92.

[18] Y. Kou, S. Lin, and M. Fossorier, Low-density parity-check codes based on finite geometries: a rediscovery and new results, *IEEE Trans. Inform. Theory* **47** (2001), 2711–2736.

[19] X. Li, C. Zhang, and J. Shen, Regular LDPC codes from semipartial geometries, *Acta Appl. Math.* **102** (2008), 25–35.

[20] R. A. Litherland, Symmetric drawings of the Hoffman-Singleton graph, http://www.math.lsu.edu/∼lither/hoff-sing, 2007.

[21] D.J.C. MacKay, Good error-correcting codes based on very sparse matrices, *IEEE Trans. Inform. Theory* **45** (1999), 399–431.

[22] D.J.C. MacKay and R.M. Neal, Near Shannon limit performance of low density parity check codes, *Electronics Letters* **32** (1996), 1645–1646.

[23] J.L. Massey, *Threshold Decoding*, MIT Press, Cambridge, MA, 1963.

[24] R.H. Morelos-Zaragoza, *The Art of Error Correcting Coding*, Wiley, 2002.

[25] G. Pellegrino, Sulle proprietà della 11-calotta completa di $S_{4,3}$ e su un B.I.B.-disegno ad essa collegato, *Boll. Un. Mat. Ital. (4)* **7** (1973), 463–470.

[26] ———, Su una interpretazione geometrica dei gruppi M_{11} ed M_{12} di Mathieu e su alcuni t-(v, k, λ)-disegni deducibili da una $(12)^4_{5,3}$ calotta completa, *Atti Sem. Mat. Fis. Univ. Modena* **23** (1974), 103–117.

[27] V. Pepe, L. Storme, and G. Van de Voorde, Small weight codewords in the LDPC codes arising from linear representations of geometries, *J. Combin. Des.* **17** (2009), 1–24.

[28] H. Tang, J. Xu, S. Lin, and A. Khaled, Codes on finite geometries, *IEEE Trans. Inform. Theory* **51** (2005), 572–596.

[29] R. Tanner, A recursive approach to low complexity codes, *IEEE Trans. Inform. Theory* **IT-27** (1981), 533–547.

[30] ———, Minimum-distance bounds by graph analysis, *IEEE Trans. Inform. Theory* **IT-47** (2001), 808–821.

[31] J.A. Thas, Semipartial geometries and spreads of classical polar spaces, *J. Combin. Theory Ser. A* **35** (1983), 58–66.