

Low-Density Parity-Check Codes from Combinatorial Designs

Sarah Johnson
B.E. (Hons I)

*A thesis submitted in partial fulfilment
of the requirements for the degree of*

Doctor of Philosophy

School of Electrical Engineering
and Computer Science

The University of Newcastle
Callaghan, N.S.W. 2308
Australia

April, 2004



The UNIVERSITY
of NEWCASTLE
AUSTRALIA

I hereby certify that the work embodied in this thesis is the result of original research and has not been submitted for a higher degree to any other University or Institution.

(Sarah Johnson)

to Mum and Dad

ACKNOWLEDGEMENTS

Dr. Steven Weller's continual support, encouragement and unending flow of ideas made this thesis possible. I could not have asked for a better supervisor. Thank you Steve.

My thanks go also to my co-supervisor A/Prof. Brett Ninness who set me on the right path from the very beginning.

My research was funded through an Australian Postgraduate Award and by a CSIRO Telecommunications and Industrial Physics Postgraduate Scholarship. The Center for Integrated Dynamics and Control (CIDAC) provided assistance for conference travel. I am very grateful for this support. Thanks in particular to my CSIRO supervisor Dr. Terry Percival.

I would also like to thank three researchers whose generosity in making their excellent LDPC coding resources widely available made my job a lot easier: Prof. Radford M. Neal and Prof. David MacKay for their on-line repositories of LDPC-related software and Prof. Rüdiger Urbanke for sharing *ldpcOpt* with the rest of us.

I am very grateful to Dr. Pascal Vontobel whose very careful reading of my thesis and numerous suggestions were invaluable.

Thanks to the postgrads; Adrian, Angus, Daniel, Ele, Glenn, Hernan, Josh, Juan, Juan Carlos, Katrina, Leif, Osvaldo, Pratik, Ray, Rob, Simon, Stuart, Terry and Tristan for many helpful discussions and for their friendship, thanks guys I had a lot of fun.

Most of all, thank you Ro, your love and support makes everything seem easy.

CONTENTS

Acknowledgements	vii
Abstract	xv
1 Introduction	1
1.1 Forward error correction	1
1.2 Problem statement and motivation	3
1.3 Thesis overview	4
1.4 Thesis contributions	6
1.4.1 Related publications	7
2 Codes and designs	9
2.1 Low-density parity-check codes	9
2.1.1 Sum-product decoding	11
2.1.2 Design and analysis of LDPC codes	15
2.2 Combinatorial designs	23
2.2.1 Steiner 2-designs	23
2.2.2 Partial geometries	27
2.2.3 Codes from designs	29

3	LDPC codes from Steiner systems	31
3.1	Introduction	31
3.2	LDPC codes from Steiner 2-designs	32
3.2.1	The rate of codes from Steiner 2-designs	35
3.2.2	The girth of codes from Steiner 2-designs	35
3.2.3	The stopping set distribution of codes from Steiner 2-designs	37
3.2.4	The minimum distance of codes from Steiner 2-designs	38
3.3	LDPC codes from Steiner triple systems	39
3.3.1	Bounding the rate of STS LDPC codes	39
3.3.2	Configurations and the minimum distance of STS LDPC codes	41
3.3.3	The stopping set distribution of STS LDPC codes	44
3.3.4	STS LDPC codes from low rank STS designs	46
3.3.5	The decoding performance of STS LDPC codes	47
3.4	Discussion	49
4	LDPC codes from designs with large block size	57
4.1	Introduction	57
4.2	Column weight versus decoding performance	58
4.3	LDPC codes with column weights > 3	61
4.4	LDPC codes from oval designs	65
4.4.1	Simulation results for oval LDPC codes	67
4.5	LDPC codes from unital designs	70

4.5.1	Simulation results for unital LDPC codes	72
4.6	Linearly dependent parity-checks	73
4.7	Discussion	75
5	A generalization to partial geometries	93
5.1	Introduction	93
5.2	Codes from partial geometries	94
5.2.1	Minimum distance	96
5.2.2	Linearly dependent rows in H	100
5.3	Proper partial geometries	101
5.3.1	Simulation results for proper partial geometry LDPC codes	102
5.4	Transversal designs	105
5.4.1	Simulation results for TD LDPC codes	110
5.5	Discussion	111
6	Code design using resolvability	121
6.1	Introduction	121
6.2	LDPC codes from Kirkman triple systems	124
6.2.1	The minimum distance of KTS LDPC codes	125
6.2.2	The decoding performance of KTS LDPC codes	126
6.2.3	The implementation complexity of KTS LDPC codes	127
6.3	Codes from cyclically resolvable cyclic designs	128
6.3.1	Encoding LDPC codes from cyclically resolvable cyclic designs	130

6.4	Resolvability and other designs	135
6.4.1	LDPC codes from Kirkman quadruple systems	135
6.4.2	LDPC codes from resolvable oval designs	136
6.4.3	LDPC codes from resolvable transversal designs	138
6.5	Discussion	140
7	Cyclic and quasi-cyclic LDPC codes	151
7.1	Introduction	151
7.2	Cyclic LDPC codes	152
7.2.1	Defining the parity-check polynomial	153
7.2.2	The impact of cyclic parity-checks on decoding performance	158
7.3	Quasi-cyclic LDPC codes with H a row of circulants	160
7.3.1	Quasi-cyclic codes from difference families	162
7.3.2	The girth of quasi-cyclic LDPC codes from difference sets	163
7.3.3	Regular quasi-cyclic LDPC codes	166
7.3.4	Irregular quasi-cyclic LDPC codes	168
7.4	Discussion	170
8	Concluding remarks	183
8.1	Future directions	185
A	Construction of designs	187
A.1	Steiner 2-designs	187

A.1.1 Projective geometries	187
A.1.2 Ovals	188
A.1.3 Resolvable ovals	191
A.1.4 Unitals	192
A.2 Proper partial geometries	193
A.3 Kirkman triple systems from mixed difference systems	195
Glossary	199
References	202

ABSTRACT

Recent breakthroughs in forward error correction, in the form of low-density parity-check (LDPC) and turbo codes, have seen the promise of near Shannon limit information transmission realised. The deterministic construction of good codes, for LDPC codes in particular, is currently the subject of intense interest in the research and development community. This thesis adds to this field, developing methods and supporting theory for the design of algebraic LDPC codes using incidence structures from combinatorial designs. The benefits of this approach are guaranteed code properties, parameter flexibility, ease of implementation and reduced encoding complexity.

In the first part of this thesis Steiner 2-designs and partial geometries are used to provide systematic constructions of regular LDPC codes with Tanner graphs free of 4-cycles and with deterministic code properties. Drawing extensively from existing incidence structures we present new families of algebraic LDPC codes with benefits over traditional randomly constructed regular codes, particularly where small-to-medium length codes are concerned. We focus in particular on exploring the effect of different design properties on the code parameters by deriving the rate, minimum distance, girth, and stopping set distribution of LDPC codes from designs.

In the second part of this thesis resolvable designs are employed to create regular LDPC codes free of 4-cycles with the parameter flexibility and decoding performance of randomly constructed regular LDPC codes. Employing resolvability greatly extends the class of LDPC codes that can be algebraically constructed. These codes can offer significant implementation advantages due to the deterministic structure of their resolution classes and in some cases their simple encoding.

A shortcoming of LDPC codes is their potentially high encoding complexity, which is generally quadratic in the code length. Finding computationally efficient encoders is therefore critical for LDPC codes to be considered as serious contenders for future generations of forward error correction devices. As a result the purpose of the last part of this thesis is to design codes which have the graph-based properties necessary to perform well with sum-product decoding but which are also cyclic or quasi-cyclic and so can be encoded simply. The main focus is regular codes but we also extend our construction of quasi-cyclic codes to produce irregular LDPC codes with simple linear-time encoding circuits.

INTRODUCTION

Error control codes underpin the success of digital communications and storage systems. Future applications will demand increasingly effective error correction, yet at the same time traditional forward error correction systems fall well short of fundamental capacity limits established some fifty years ago. Recent years have witnessed a recognition of the importance of iterative decoding algorithms operating on codes defined on graphs for practical error correction close to the fundamental limits. Iteratively decoded block codes, called low-density parity-check (LDPC) codes, were first presented by Gallager [42] and recently shown to provide error correction within a fraction of a decibel of the Shannon limit. These codes are typically constructed pseudo-randomly; however, some work has shown that good LDPC codes can also be constructed algebraically. In this thesis we design new algebraic LDPC codes using incidence structures from combinatorial designs with the aim of producing codes which perform as well as randomly constructed regular LDPC codes but with deterministic constructions and guaranteed properties.

1.1 Forward error correction

The fundamental performance limits of forward error correction were set down by Shannon in his 1948 paper, *A mathematical theory of communication* [93]. Shannon proved that, by employing forward error correction, arbitrarily reliable communication is possible through channels which corrupt the data sent over them only if information is transmitted at a rate less than the *capacity* of the channel. Furthermore, if the data is transmitted at a rate less than the channel capacity, Shannon's theorem also establishes that an error correction code and decoder must exist for which the number of errors that can not be corrected goes to zero.

Unfortunately Shannon's proof is non-constructive. That is, it provides no explicit guidance as to how such a code can be produced and decoding algorithms which can be applied in practice are not guaranteed. The challenges to communicating reliably are therefore twofold: the first is to design suitable codes, and the second is to devise

methods for decoding the channel output to recover the codeword sent, and to do so without excessive decoder complexity.

From the very earliest error-correcting codes of Golay and Hamming, through to the early 1990s, the approach to this problem has been overwhelmingly algebraic in nature. Codewords are formed from strings of symbols chosen from a finite field and decoding with the lowest probability of error requires that the codeword which is closest to the corrupted word received from the channel is chosen, a process called *maximum likelihood* decoding.

The downside of this strategy is that for all but the smallest codes the complexity of explicitly comparing the received word to every codeword in the code becomes infeasible. The classical strategy for error correction coding is then to design codes with the best possible distance distribution and enough structure to enable approximate maximum likelihood decoding with feasible complexity.

The introduction of *turbo codes* by Berrou, Glavieux and Thitimajshima in 1993 heralded a fundamental departure from algebraic approaches to code design [7, 6]. Through the ingenious use of parallel concatenation of simple constituent codes and a pseudo-random block interleaver, Berrou et al. devised codes which approached the capacity limits promised by Shannon, yet which were iteratively decodable with manageable complexity.

The decade since the introduction of turbo codes has seen a radical re-evaluation of what constitutes “good” codes, how such codes might be designed, and how to decode them effectively. A central aspect of this re-evaluation has been the recognition of the importance of iterative decoding algorithms operating on codes defined on graphs [1]. It was therefore more than a little surprising for researchers in the mid-1990s [73, 71] to discover that iterative decoding of simple constituent codes had been first presented some 40 years earlier by Elias [37] for product codes and subsequently developed by Gallager [42, 43] for parity-check codes, and generalized by Tanner [100] who represented the codes and their decoding on graphs. The codes we consider in this thesis were first presented by Gallager in his 1963 PhD thesis [43], and were termed “low-density parity-check” codes to reflect the sparsity of the parity-check matrices from which they are defined.

For high-performance applications, LDPC codes are seen as competitors to turbo codes. LDPC codes are capable of outperforming turbo codes for block lengths greater than around 10^5 , and the error floors of LDPC codes at bit error rates below about 10^{-5} are typically much less pronounced than those of turbo codes. In general turbo codes require many fewer iterations (although they are of higher complexity) than LDPC codes to converge. However, the actual number of iterations carried out in a turbo decoding will depend on the stopping criterion employed as, unlike for LDPC codes, convergence to a

valid codeword is not readily determined during turbo decoding. Moreover, the inherent parallelism of the sum-product decoding algorithm is more readily exploited with LDPC codes than their turbo counterparts, where block interleavers pose formidable challenges to achieving high throughput [9].

The best known error-correction performance on the additive white Gaussian noise (AWGN) channel has been achieved with an LDPC code, albeit one with an impractically long block length and high implementation complexity [25]. The absence of an explicit interleaver, such as those required by turbo codes, leads to highly parallel (and therefore low latency) decoder implementations in application-specific integrated circuits [9]. LDPC codes are also capable of exceptional performance on channels where data is not just corrupted but may be lost entirely, so-called *erasure* channels. This opens the way to new application domains such as reliable Internet multicasting where whole packets of lost data are reconstructed without the network overhead of retransmission [18]. The challenge is to devise new LDPC codes with sufficient flexibility to cope with the myriad of applications opening up in future generation wireless communications, data storage and the Internet.

1.2 Problem statement and motivation

Traditionally, LDPC codes have been defined pseudo-randomly. This has proven very effective at producing good decoding performances, particularly for very long codes. However, a random construction method also means that code properties are not guaranteed for any individual code, and are thus not easy to control or even determine. As well, implementation complexity is increased by the need to store, and encode, codes described by random parity-check matrices.

In particular, their potentially high *encoding* complexity, which is in general quadratic in the code length, is a serious shortcoming of LDPC codes and compares poorly with the linear time encoding of turbo codes. Finding computationally efficient encoders is therefore important for LDPC codes to be considered as serious contenders for replacing turbo codes in future generations of forward error correction devices. Several approaches have been suggested, including the manipulation of the parity-check matrix to establish that while the complexity is, strictly speaking, quadratic, the actual number of encoding operations grows essentially linearly with code length [89].

A very different approach to the encoding complexity problem is to employ LDPC codes with algebraic structure to guarantee linear-time encoding. In [103, 105, 102] Tanner considered binary quasi-cyclic, or more generally group invariant, LDPC codes, which can be analyzed using his generalized transform methods [101]. An existing family of cyclic codes, the finite geometry codes, were shown to be very good LDPC codes in the

work of Lucas et al. and Kou et al. [70, 60] and a small finite geometry code was earlier implemented with sum-product decoding in [56]. The work of Lucas et al. and Kou et al. demonstrates that as well as providing implementation advantages the algebraic finite geometry codes can significantly outperform equivalent length and rate randomly constructed codes when both are decoded with the sum-product algorithm [70, 60]. The excellent performance of the finite geometry codes was the motivation for this work, by showing that codes which perform excellently with sum-product decoding can be constructed without the need for random interleavers or random parity-check matrices.

With this background in mind the problem considered in this thesis is the design of algebraic error correction codes for use with sum-product decoding or, more succinctly, the design of LDPC codes. The central idea of this thesis is to apply combinatorial theory to the design of new algebraic LDPC codes. The potential benefits of considering codes from combinatorial designs are guaranteed code properties, parameter flexibility, and ease of implementation including reduced storage requirements and reduced encoding complexity. The addition of algebraic structure required to meet these goals must be balanced against its effect on the decoding performance of the code, and we consider what structure is beneficial in an LDPC code as well as how to achieve it.

1.3 Thesis overview

We begin in the following chapter by presenting the background material to be used throughout this thesis. A brief introduction to low-density parity-check codes and combinatorial designs is presented, including an overview of specific ways in which designs have historically been related to error correction codes. The remainder of this thesis can then be divided into three main parts. In the first part, Chapters 3–5, the incidence structures of combinatorial designs, with known algebraic constructions, are used to derive algebraic LDPC codes. In the second part, presented in Chapter 6, a compromise between completely deterministic codes and parameter flexibility is achieved by constructing LDPC codes using design resolvability. In the final part of this thesis we focus on linear-time encodable LDPC codes by using difference structures to derive new constructions for cyclic and quasi-cyclic LDPC codes.

More specifically the contents of each chapter are as follows.

In Chapter 3 combinatorial designs are employed to create new error correction codes for use with sum-product decoding. The problem of constructing LDPC codes with Tanner graphs free of 4-cycles has a solution in a well studied problem in combinatorics, that of constructing Steiner 2-designs. In particular, Steiner triple systems are used to provide systematic constructions of high rate $(3,r)$ -regular LDPC codes with Tanner graphs free of 4-cycles and with deterministic code properties. We focus also on exploring

the effect of different Steiner 2-design parameters on the properties of the LDPC codes constructed using them.

In Chapter 4 the design of LDPC codes using Steiner 2-designs with large block length is considered. The relationship between the column weight of the parity-check matrix and the decoding performance of the codes is considered and the role of rank deficient parity-check matrices examined. Two new classes of LDPC codes are presented, both with Tanner graphs free of 4-cycles, good minimum distances and a large portion of linearly dependent rows in their parity-check matrix.

Using Steiner 2-designs a large class of LDPC codes are produced but only for a limited range of (mostly high) rates. By generalizing to partial geometry designs in Chapter 5, a wider flexibility in choice of code parameters is achieved. This represents a generalization to a larger class of LDPC codes which include both the codes from Steiner 2-designs presented in Chapters 3 and 4 and the codes from generalized quadrangles presented by Vontobel and Tanner [118]. Furthermore the graph based properties of partial geometries enable good minimum distance, girth and rate properties to be derived.

In Chapter 6 resolvable designs are employed to create LDPC codes with the parameter flexibility and decoding performance of the randomly constructed codes. By employing resolvability we can greatly extend the class of low-density parity-check codes that can be algebraically constructed. The resulting codes are $(3, \rho)$ -regular or $(4, \rho)$ -regular with Tanner graphs free of 4-cycles, for any value of ρ and for a flexible choice of code lengths. Further, cyclically resolvable cyclic designs are used to construct LDPC codes with simple shift-register encoding circuits.

In Chapter 7 the focus is on designing codes with the structure to enable very low complexity encoding. Codes which have the graph-based properties necessary to perform well with the sum-product decoding algorithm but which are also cyclic or quasi-cyclic are designed. We make extensive use of combinatorial structures such as difference sets and difference families to design the new codes. The translates of these sets provide us with the incidence structures necessary for systematic constructions of regular LDPC codes with Tanner graphs free of 4-cycles and with deterministic code properties. Although the main focus is regular codes we also extend our construction of quasi-cyclic codes to produce irregular LDPC codes with simple shift-register encoding circuits.

In each of the Chapters 3–7 the advantages of the LDPC codes presented are demonstrated by deriving the parameters of the codes, by simulating their decoding performance, and by considering their implementation, particularly in respect to code description, storage and encoding. For ease of reference the simulation plots have been placed at the end of each chapter.

A discussion and future directions is presented in Chapter 8. Finally, Appendix A presents constructions for the combinatorial designs we use in this thesis, followed by a glossary and bibliography.

1.4 Thesis contributions

This section outlines the original contributions of this thesis. Some parts of this thesis have been published in the open literature and this section also serves to identify them, with the associated citations listed in the following Section 1.4.1.

Chapter 2 is of a background nature and no new results are presented, parts of this chapter appeared in the book chapter [B1].

Chapter 3 considers the design of LDPC codes using structures from combinatorics. The original contribution of this chapter is an analysis of the role of Steiner 2-designs in constructing LDPC codes and the recognition of the link between design structures such as configurations and LDPC code properties such as stopping set distribution and minimum distance. An expression for the exact number of 6-cycles in the Tanner graph of LDPC codes from Steiner 2-designs is also presented. Parts of the material in this chapter appeared in a less general form in [J2].

Chapter 4 follows on from Chapter 3 to consider LDPC codes with larger block size. The original contribution of this chapter is the presentation of two new families of LDPC codes, namely oval codes and unital codes. Chapter 4 represents the combination of several papers. The material on oval codes was presented in the conference paper [C3] and subsequently the journal paper [J3]. The material on unital codes is presented in [C8].

The original contribution presented in Chapter 5 is the generalization to codes from the field of partial geometries which includes as subclasses the codes from Steiner 2-designs and codes from generalized quadrangles. An expression for the exact number of minimum size cycles in the Tanner graphs of LDPC codes from partial geometries is presented, a lower bound on minimum distance is presented and both lower and upper bounds on code rate are derived. New LDPC codes from partial geometries, codes from transversal designs and codes from proper partial geometries, are presented. Chapter 5 is primarily based on the journal paper [J4] which itself is based on the conference paper [C5].

In Chapter 6 the original contribution is the innovation of employing the resolvability of combinatorial designs to construct regular LDPC codes free of 4-cycles. New LDPC codes are presented which compare well with the flexibility of parameter choice and the decoding performance of the traditional randomly constructed codes. Linear time encod-

ing circuits are also presented for a subset of these codes. Chapter 6 is based on the journal paper [J2] which in turn is based on conference papers [C1] and [C2].

In Chapter 7 the original contribution is the design of new cyclic and quasi-cyclic regular LDPC codes, and the presentation of a simple construction of irregular quasi-cyclic LDPC codes. The material on cyclic codes is presented in [C7] while the material on quasi-cyclic LDPC codes is mainly based on [C4]. The extension to irregular quasi-cyclic LDPC codes was published in [J1].

1.4.1 Related publications

Book chapters

- [B1] S. J. Johnson and S. R. Weller. Low-density parity-check codes: Design and decoding, in *The Wiley Encyclopedia of Telecommunications*, J. Proakis (Ed.), John Wiley & Sons, 2003.

Journal papers

- [J1] S. J. Johnson and S. R. Weller. A family of irregular LDPC codes with low encoding complexity. *IEEE Communications Letters*. vol. 7(2), pp. 79–81, February 2003.
- [J2] S. J. Johnson and S. R. Weller. Resolvable 2-designs for low-density parity-check codes. *IEEE Transactions on Communications*. Vol 51 (9), pp 1413 - 1419, September 2003.
- [J3] S. R. Weller and S. J. Johnson, Iterative decoding of codes from oval designs. *European Transactions on Telecommunications*. Vol 14 (5), pp 399–409, October 2003.
- [J4] S. J. Johnson and S. R. Weller, Codes for iterative decoding from partial geometries. *IEEE Transactions on Communications* To appear, February 2004.

Conference papers

- [C1] S. J. Johnson and S. R. Weller. Regular low-density parity-check codes from combinatorial designs. In *Proc. IEEE Information Theory Workshop*, pp. 90-92, Cairns, Australia, September 2001.
- [C2] S. J. Johnson and S. R. Weller. Construction of low-density parity-check codes from Kirkman triple systems. In *Proc. IEEE Globecom Conf.*, San Antonio, TX, November 2001. vol. 2 pp. 970-974.

- [C3] S. R. Weller and S. J. Johnson. Iterative decoding of codes from oval designs. In *Defense Applications of Signal Processing (DASP)*, 2001.
- [C4] S. J. Johnson and S. R. Weller. Quasi-cyclic LDPC codes from difference families. In *Proc. 3rd Australian Communications Theory Workshop (AusCTW'2002)*, pp. 15-17, Canberra, Australia, February 2002.
- [C5] S. J. Johnson and S. R. Weller. Codes for iterative decoding from partial geometries. In *Proc. IEEE International Symposium on Information Theory (ISIT)*, p. 310, Lausanne, Switzerland, June 30 - July 5, 2002.
- [C6] R. A. Brown, S. J. Johnson and S. R. Weller. Performance of space-time block codes with finite geometry LDPC outer codes. In *Proc. 4th Australian Communications Theory Workshop (AusCTW)*, pp. 93–98, Melbourne, Australia, February 2003.
- [C7] S. J. Johnson and S. R. Weller. Can cyclic codes be useful low-density parity-check codes? In *Proc. 4th Australian Communications Theory Workshop (AusCTW'2003)*, pp. 81–86, Melbourne, Australia, February 2003.
- [C8] S. J. Johnson and S. R. Weller. High-rate LDPC codes from unital designs. *Proc. IEEE Globecom Conf.* San Fransisco, 1 – 5 December 2003.

CODES AND DESIGNS

This chapter presents the background material to be used throughout this thesis. A brief introduction to the theory of error correction codes is presented before low-density parity-check codes are introduced. Next the combinatorial designs employed in this thesis are described before we overview the ways in which combinatorial designs have historically been related to error correction codes.

2.1 Low-density parity-check codes

The essential idea of forward error correction is to deliberately introduce redundancy into a digital message so that the message can be correctly inferred at the receiver, even when some of the symbols are corrupted during transmission or storage. More specifically, a q -ary $[n, k, d]$ block error correction code with rate $R = k/n$, maps a message of k symbols into a codeword of $n > k$ symbols where each symbol is one of q possible elements. The *decoder* receives a length n vector, which is not necessarily a codeword, and uses the structure of the code to determine which message was sent. The gains to data reliability afforded by employing error correction can be used to reduce the required transmission power or bandwidth, or increase data storage efficiency.

The *Hamming distance* between two codewords is the number of symbols in which they differ. The *minimum distance* of the code, d , is the smallest Hamming distance between any pair of codewords in the code and is one measure of the error correction capability of the code. In general, for a code with minimum distance d , t bit errors can always be corrected by choosing the closest codeword, in Hamming distance, to the received vector whenever

$$t \leq \lfloor (d - 1)/2 \rfloor, \quad (2.1)$$

where $\lfloor x \rfloor$ is the largest integer that is at most x .

To illustrate, a simple linear binary code, with elements from the binary Galois field, $\text{GF}(2)$, is defined to have the following structure:

$$c = c_1 \ c_2 \ c_3 \ c_4 \ c_5 \ c_6,$$

where each symbol c_i is either 0 or 1, and the codeword, c , is constrained by three parity-check equations:

$$\begin{aligned} c_1 \oplus c_2 \oplus c_4 &= 0 \\ c_2 \oplus c_3 \oplus c_5 &= 0 \\ c_1 \oplus c_2 \oplus c_3 \oplus c_6 &= 0 \end{aligned}$$

The notation \oplus represents modulo-2 addition, which is equal to 1 if the ordinary sum is odd and 0 if the ordinary sum is even. The parity-check equations can be re-written in matrix form:

$$\underbrace{\begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}}_H \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad (2.2)$$

where the *parity-check matrix*, H , represents the parity-check equations which define the code. Thus a vector $r = [r_1 \ r_2 \ r_3 \ r_4 \ r_5 \ r_6]$ is a codeword if and only if it satisfies the constraint

$$Hr^T = 0. \quad (2.3)$$

To generate the codeword for a given message, the code constraints can be rewritten in the form

$$\begin{aligned} c_4 &= c_1 \oplus c_2 \\ c_5 &= c_2 \oplus c_3 \\ c_6 &= c_1 \oplus c_2 \oplus c_3 \end{aligned} \quad (2.4)$$

where bits c_1, c_2 , and c_3 contain the 3-bit message, and the parity-check bits c_4, c_5 and c_6 are a function of this message. Thus, for example, the message 110 produces parity-check bits $c_4 = 1 \oplus 1 = 0$, $c_5 = 1 \oplus 0 = 1$ and $c_6 = 1 \oplus 1 \oplus 0 = 0$, and hence the codeword 110010. As the code is linear, matrix notation can again be used,

$$\begin{bmatrix} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \end{bmatrix} = \begin{bmatrix} c_1 & c_2 & c_3 \end{bmatrix} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}}_G, \quad (2.5)$$

where the *generator matrix* of the code, G , represents a basis for the one-to-one mapping of messages onto codewords.

An error correction code can be described by more than one parity-check matrix or generator matrix. A matrix H is a valid parity-check matrix for a code provided that (2.3) holds for all codewords in the code. Likewise two matrices generate the same code if they map every message to the same codeword. Two parity-check matrices for the same

code need not even have the same number of rows; however the rank over $\text{GF}(q)$ of both must be the same, since the number of message symbols, k , in a q -ary code is

$$k = n - \text{rank}_q(H), \quad (2.6)$$

where $\text{rank}_q(H)$ is the number of rows in H which are linearly dependent over $\text{GF}(q)$.

A parity-check matrix is *regular* if each code symbol is contained in a fixed number, w_c , of parity checks and each parity-check equation contains a fixed number, w_r , of codeword symbols. If a code is described by a regular parity-check matrix it is called a (w_c, w_r) -*regular* code otherwise it is an *irregular* code.

A regular parity-check matrix for the binary code of (2.2) with $w_c = 2$, $w_r = 3$ and $\text{rank}_2(H) = 3$ is:

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}. \quad (2.7)$$

An LDPC code is simply a block code with a parity-check matrix which is sparse, that is, the majority of entries must be zero. What sets LDPC codes apart from traditional codes is the way in which they are decoded which in turn has implications for which sparse parity-check matrices make good LDPC codes. In the following sections the sum-product decoding algorithm is introduced and the properties of a good LDPC code are discussed.

2.1.1 Sum-product decoding

Decoding with the sum-product algorithm make essential use of *graphs* to represent codes, passing messages along the edges of the graph. In his work, Gallager used a graphical representation of the bit and parity-check sets of regular LDPC codes, to describe the application of iterative decoding. However, the systematic study of codes on graphs due to Tanner who, in 1981 [100], extended the single parity-check constraints of Gallager's LDPC codes to arbitrary linear code constraints and formalized the use of bipartite graphs for describing families of codes [100]. The use of codes defined on graphs increased dramatically with the independent rediscovery of LDPC codes by researchers in the mid-1990s [73, 71], and graph-based representations of codes are now an integral feature in the development of both the theoretical understanding and practical implementation of iterative decoders.

A *Tanner graph* is a graphical representation of a code representing the interaction between codeword symbols and code constraints. In the case of binary codes constrained by parity-check equations the Tanner graph is bipartite consisting of n *bit vertices* (or

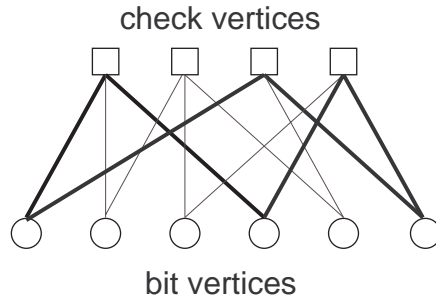


Figure 2.1: The Tanner graph representation of the parity-check matrix in (2.7). A 6-cycle is shown in bold.

bit nodes), and m *parity-check vertices* (or *check nodes*) where there is a parity-check vertex for every parity-check equation in H and a bit vertex for every codeword bit. Each parity-check vertex is connected by an edge to each of the bit vertices which correspond to the code bits included in that parity-check equation. A code parity-check matrix is thus uniquely described by a Tanner graph, although translating from a graph back to a matrix requires an ordering of nodes corresponding to an ordering of rows and columns in H .

A *cycle* in a graph is a sequence of connected vertices which start and end at the same vertex in the graph and contain no other vertices more than once. The length of the cycle is the number edges it contains and the *girth* of a graph is the length of its smallest cycle. The girth of an LDPC code, C , is defined by considering the Tanner graph corresponding to the parity-check matrix of C . As the Tanner graph is bipartite, the length of a cycle must be even and at least 4. The Tanner graph of the parity-check matrix (2.7) is shown in Fig. 2.1 with a 6-cycle highlighted.

The aim of sum-product decoding is to compute the *a posteriori probability* (APP) for each codeword bit, $P_i = P\{c_i = 1 | \mathcal{N}, y_i\}$, which is the probability that the i th codeword bit is a 1 conditional on the received signal y_i and on the event \mathcal{N} that all parity-check constraints are satisfied. The *intrinsic* or *a priori probability*, P_i^{int} , is the original bit probability independent of knowledge of the code constraints, and the *extrinsic* probability P_i^{ext} represents what has been learnt from the parity checks. The extrinsic bit information obtained from a parity-check constraint is calculated independently of the a priori value for that bit at the start of the iteration. However, the extrinsic information provided in subsequent iterations remains independent of the original a priori bit probability only until that information is returned via a cycle. If the Tanner graph of the code is cycle free the probabilities remain independent and the exact APP value is calculated for each bit.

For a binary valued random variable, if p is the probability of a 1, then $1 - p$ is the probability of a 0 which is represented as a log-likelihood ratio (LLR) by

$$\text{LLR}(p) = \log_e \left(\frac{1-p}{p} \right). \quad (2.8)$$

The sign of $\text{LLR}(p)$ is the hard decision and the magnitude $|\text{LLR}(p)|$ is the reliability of this decision.

The extrinsic probability that bit i is a 1 from the j th parity-check equation is the probability that an odd number of the other codeword bits in check j are 1 [43]:

$$P_{i,j}^{\text{ext}} = \frac{1}{2} + \frac{1}{2} \prod_{i' \in B_j, i' \neq i} (1 - 2P_{i'}^{\text{int}}). \quad (2.9)$$

The set B_j is the set of column locations of the bits in the j th parity-check equation of the code. Similarly, A_i is the set of row locations of the parity-check equations which check on the i th bit of the code. Using the result that

$$\tanh \left(\frac{1}{2} \log_e \left(\frac{1-p}{p} \right) \right) = 1 - 2p,$$

equation (2.9) can be put into log-likelihood notation to give [43]:

$$\text{LLR}(P_{i,j}^{\text{ext}}) = \log_e \left(\frac{1 + \prod_{i' \in B_j, i' \neq i} \tanh(\text{LLR}(P_{i'}^{\text{int}})/2)}{1 - \prod_{i' \in B_j, i' \neq i} \tanh(\text{LLR}(P_{i'}^{\text{int}})/2)} \right).$$

The estimated APP of the i th bit at each iteration is then simply the sum of the intrinsic and extrinsic LLRs for that bit.

More formally, the sum-product algorithm is as follows:

Step 1 (in the first iteration) *Initialization*: The initial message, $L_{i,j}$, sent from bit node i to the check node j is R_i , the LLR of the (soft) received signal y_i given knowledge of the channel properties. For an AWGN channel with signal-to-noise ratio E_b/N_0 this is:

$$L_{i,j} = R_i = 4y_i \frac{E_b}{N_0}. \quad (2.10)$$

Step 2 *Check-to-bit*: The extrinsic message from check node j to bit node i is the LLR of the probability that parity-check j is satisfied if bit i is assumed to be a 1:

$$E_{i,j} = \log_e \left(\frac{1 + \prod_{i' \in B_j, i' \neq i} \tanh(L_{i',j}/2)}{1 - \prod_{i' \in B_j, i' \neq i} \tanh(L_{i',j}/2)} \right). \quad (2.11)$$

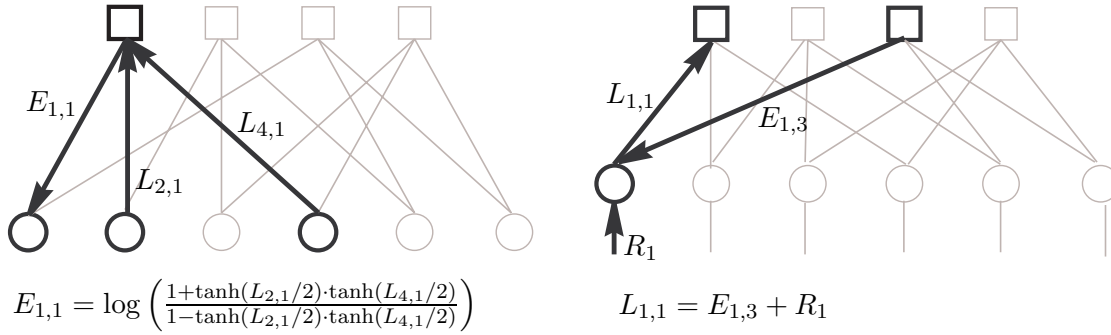


Figure 2.2: An example of the messages for sum-product decoding. Calculation of the extrinsic message sent to bit 1 depends on messages from bits 2 and 4 but not from bit 1. Similarly, the message sent to check 1 is independent of the message just received from it.

Step 3 Codeword test: The estimated APP of each bit is the sum of the intrinsic and extrinsic LLRs:

$$L_i = \sum_{j \in A_i} E_{i,j} + R_i. \quad (2.12)$$

For each bit a hard decision is made:

$$z_i = \begin{cases} 1, & L_i \leq 0 \\ 0, & L_i > 0. \end{cases}$$

Then, if $z = [z_1, \dots, z_n]$ is a valid codeword ($H z^T = 0$), or if the maximum number of allowed iterations have been completed, the algorithm terminates.

Step 1 (in all iterations after the first) *Bit-to-check:* For every iteration after the first, the message $L_{i,j}$ sent by each bit node to the check nodes to which it is connected is similar to (2.12), except that, to preserve independence, bit i sends to check node j a LLR calculated without using the information from check node j :

$$L_{i,j} = \sum_{j' \in A_i, j' \neq j} E_{i,j'} + R_i. \quad (2.13)$$

The application of equations (2.11) and (2.13) to the code in (2.7) is demonstrated in Fig. 2.2.

To illustrate the operation of the sum-product decoder, we take the code of (2.7) and assume that the codeword $c = 001011$ is sent. Suppose that the channel is AWGN with $E_b/N_0 = 1.25$ and the received signal is $y = -0.1, 0.5, -0.8, 1.0, -0.7, 0.5$. There are now two bits in error if the hard decision of the signal, $r = 101010$, is considered: bits 1 and 6. Fig. 2.3 illustrates the operation of the sum-product decoding algorithm, as

described in equations (2.10)–(2.13), to decode this received signal which terminates in three iterations. The existence of an exact termination rule for the sum-product algorithm has two important benefits; the first is that a failure to converge is always detected, and the second is that unnecessary additional iterations once a solution has been found are avoided.

There are a rich variety of extensions to the sum-product algorithm. The *min-sum* algorithm, for example, simplifies the calculation of (2.11) by recognizing that the term corresponding to the smallest $L_{i',j}$ dominates the product term and so the product can be approximated by a minimum; the resulting algorithm thus requires calculation of only minimums and additions. An alternative approach, designed to bridge the gap between the error performance of sum-product decoding and that of maximum-likelihood (ML) decoding, finishes each iteration of sum-product decoding with ordered statistic decoding, with the algorithm terminating when a specified number of iterations have returned the same codeword [41]. However, in this thesis the focus is on code design, and so in the following we employ only the sum-product decoding algorithm as presented above.

2.1.2 Design and analysis of LDPC codes

Traditionally, LDPC codes have been designed by first choosing the required block length and node degree distributions, then pseudo-randomly constructing a parity-check matrix, or Tanner graph with these properties. A generator matrix for the code can then be found using Gaussian elimination [71]. Gallager, for example, considered the ensemble of all (w_r, w_c) -regular matrices with rows divided into w_c submatrices, the first containing w_r copies of the identity matrix and with subsequent submatrices being random column permutations of the first [43].

Analysis is difficult due to the random nature of the construction and so code properties are generally discussed in terms of the ensemble of all possible matrices with a given set of row and column weights. Using ensembles of matrices defined in this way Gallager found the maximum crossover probability of the binary symmetric channel (BSC) for which LDPC codes can be used to transmit information reliably using a simple iterative hard decision decoding algorithm.

Luby et al. extended the class of LDPC ensembles to those with irregular node degrees and showed that irregular codes are capable of outperforming regular codes [68, 69]. In extending Gallager’s analysis to irregular ensembles, Luby et al. introduced tools based on linear programming to design irregular code ensembles for which the maximum allowed crossover probability of the binary symmetric channel is optimized. Resulting from this work are the “tornado codes,” a family of codes that approach the capacity of the erasure channel and can be encoded and decoded in linear time [18, 68].

$$\begin{array}{l}
\text{Iteration 1} \\
R = [-0.5000 \quad 2.5000 \quad -4.0000 \quad 5.0000 \quad -3.5000 \quad 2.5000] \\
\quad [1 \ 0 \ 1 \ 0 \ 1 \ 0] \text{ as a hard decision} \\
E = \begin{bmatrix} 2.4217 & -0.4930 & \cdot & -0.4217 & \cdot & \cdot \\ \cdot & 3.0265 & -2.1892 & \cdot & -2.3001 & \cdot \\ -2.1892 & \cdot & \cdot & \cdot & -0.4217 & 0.4696 \\ \cdot & \cdot & 2.4217 & -2.3001 & \cdot & -3.6869 \end{bmatrix} \\
L = [-0.2676 \quad 5.0334 \quad -3.7676 \quad 2.2783 \quad -6.2217 \quad -0.7173] \\
z = [1 \ 0 \ 1 \ 0 \ 1 \ 1] \\
Hz^T = [1 \ 0 \ 1 \ 0]^T \Rightarrow \text{Continue} \\
L = \begin{bmatrix} -2.6892 & 5.5265 & \cdot & 2.6999 & \cdot & \cdot \\ \cdot & 2.0070 & -1.5783 & \cdot & -3.9217 & \cdot \\ 1.9217 & \cdot & \cdot & \cdot & -5.8001 & -1.1869 \\ \cdot & \cdot & -6.1892 & 4.5783 & \cdot & 2.9696 \end{bmatrix} \\
\text{Iteration 2} \\
E = \begin{bmatrix} 2.6426 & -2.0060 & \cdot & -2.6326 & \cdot & \cdot \\ \cdot & 1.4907 & -1.8721 & \cdot & -1.1041 & \cdot \\ 1.1779 & \cdot & \cdot & \cdot & -0.8388 & -1.9016 \\ \cdot & \cdot & 2.7877 & -2.9305 & \cdot & -4.3963 \end{bmatrix} \\
L = [3.3206 \quad 1.9848 \quad -3.0845 \quad -0.5630 \quad -5.4429 \quad -3.7979] \\
z = [0 \ 0 \ 1 \ 1 \ 1 \ 1] \\
Hz^T = [1 \ 0 \ 0 \ 1]^T \Rightarrow \text{Continue} \\
L = \begin{bmatrix} 0.6779 & 3.9907 & \cdot & 2.0695 & \cdot & \cdot \\ \cdot & 0.4940 & -1.2123 & \cdot & -4.3388 & \cdot \\ 2.1426 & \cdot & \cdot & \cdot & -4.6041 & -1.8963 \\ \cdot & \cdot & -5.8721 & 2.3674 & \cdot & 0.5984 \end{bmatrix} \\
\text{Iteration 3} \\
E = \begin{bmatrix} 1.9352 & 0.5180 & \cdot & 0.6515 & \cdot & \cdot \\ \cdot & 1.1733 & -0.4808 & \cdot & -0.2637 & \cdot \\ 1.8332 & \cdot & \cdot & \cdot & -1.3362 & -2.0620 \\ \cdot & \cdot & 0.4912 & -0.5948 & \cdot & -2.3381 \end{bmatrix} \\
L = [3.2684 \quad 4.1912 \quad -3.9896 \quad 5.0567 \quad -5.0999 \quad -1.9001] \\
z = [0 \ 0 \ 1 \ 0 \ 1 \ 1] \\
Hz^T = [0 \ 0 \ 0 \ 0]^T \Rightarrow \text{Terminate}
\end{array}$$

Figure 2.3: Operation of sum-product decoding with the code from (2.7) when the codeword [001011] is sent through an AWGN channel with $E_b/N_0 = 1.25$ and the vector [-0.1 0.5 -0.8 1.0 -0.7 0.5] is received. The sum-product decoder converges to the correct codeword after three iterations.

Richardson and Urbanke extended the work of Luby et al. to any binary-input memoryless channel and to soft decision decoding [88, 85]. They determined the capacity of iterative decoders applied to LDPC code ensembles by a method called density evolution. For sum-product decoding density evolution makes it possible to determine the corresponding capacity to any degree of accuracy and hence determine the ensemble with node degree distribution which gives the best capacity. Once a code ensemble has been chosen a code from that ensemble is realized pseudo-randomly. By carefully choosing a code from an optimized ensemble Chung et al. have demonstrated the best performance to date of LDPC codes in terms of approaching the Shannon limit [25].

The relationship between LDPC codes and their decoding is closely associated with the graph-based representations of these codes. The most obvious example of this is the link between the existence of cycles in the Tanner graph of the code to both the analysis and performance of sum-product decoding of the code [100, 38]. By proving the convergence of the sum-product algorithm for codes whose graphs are free of cycles, Tanner was the first to formally recognize the importance of cycle-free graphs in the context of iterative decoding [100].

The effect of cycles on the practical performance of LDPC codes was demonstrated by simulation experiments when LDPC codes were rediscovered by MacKay and Neal [73] in the mid-1990s, and the beneficial effects of using graphs free of short cycles were shown [71]. Given the detrimental effects of cycles on the convergence of iterative decoders, it is natural to seek strong codes whose Tanner graphs are free of cycles. An important negative result in this direction was established by Etzion et al. [38], who showed that for linear codes of rate $k/n \geq 0.5$ which can be represented by a Tanner graph without cycles, the minimum distance is at most 2. With a small number of exceptions [122], LDPC codes are constructed in the most part with the object of obtaining Tanner graphs free of 4-cycles.

As the existence of cycles in a graph makes analysis of the decoding algorithm difficult, most analysis consider the asymptotic performance of iterative decoding on graphs with asymptotically unbounded girth. There has been very little analysis regarding the convergence of iterative decoding methods on graphs with cycles. Gallager suggested that the dependencies introduced by cycles have a relatively minor effect and tend to cancel each other out somewhat. This “seems to work” philosophy has underlined the performance of sum-product decoding on graphs with cycles for much of the (short) history of the topic. It is only recently that exact analysis on the expected performance of codes with cycles has emerged.

This technique, called finite length analysis [34, 87, 86], has suggested a new measure of the performance of an LDPC code, namely its *stopping set* distribution. A stopping set is a set of bit nodes with the property that every check node connected to a bit node

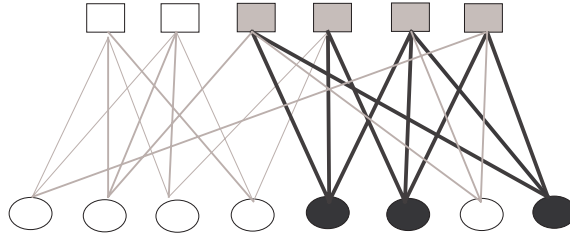


Figure 2.4: The Tanner graph of a length 8 code with 6 rows in the parity-check matrix with column weight 3 and row weight 4. A stopping set of size three is shown in bold.

in the stopping set is connected to at least two such nodes. Fig. 2.4 shows the Tanner graph of a length 8 code with 6 parity-check equations and filled bit nodes representing a stopping set of size 3.

Finite length analysis gives the exact average bit and block error probabilities for any regular ensemble of LDPC codes over the binary erasure channel (BEC) when decoded iteratively. The strategy of sum-product decoding on the binary erasure channel is a simple one. If only one of the bits in a given parity-check equation is erased that bit can be determined by choosing the value which satisfies the parity-check. However, if more than one bit in the parity-check equation is erased, no correction can be made. Thus, on the binary erasure channel, an erased codeword bit can be corrected if and only if it is the sole erased bit in any parity-check equation. If the set of code bits which are erased contains a stopping set, none of the bits in the stopping set can be corrected. Conversely, if there is no stopping set in the set of erased bits they can eventually all be corrected.

More precisely, the probability of bit erasure for a given code C of length n on a binary erasure channel with erasure probability ϵ is [34]:

$$P(C, \epsilon) = \sum_{v=0}^n \binom{n}{v} \epsilon^v (1 - \epsilon)^{n-v} \left(\frac{N(v)}{T(v)} \right).$$

$T(v)$ is the total of number of ways a bit set of size v could be constructed over all possible codes in the ensemble and $N(v)$ is the number of those ways which result in the v points being a maximal stopping set. Thus $N(v)/T(v)$ can be considered the probability that a given set of v points is a stopping set.

Thus the expected performance of an ensemble of LDPC codes on a binary erasure channel can be enumerated exactly using the average stopping set distribution of all codes in the ensemble. The girth and minimum distance of a code play a role in its performance on a binary erasure channel only in so much as they influence the stopping set distribution of the code.

The lack of any obvious algebraic structure in pseudo-randomly constructed LDPC codes makes the calculation of minimum distance infeasible for long codes, and most analyses focus on the average distance function for an ensemble of LDPC codes. However, performance results suggest that minimum distance properties are simply not as important for LDPC codes as for traditional codes. Indeed, it has recently been established, for codes with full rank parity-check matrices, that to achieve capacity on the binary erasure channel when using irregular LDPC codes, the codes can not have large minimum distances [35].

The nature of the random LDPC codes means that no bound on minimum distance can be given for any particular randomly constructed code, however Gallager's expression for the average minimum distance of an ensemble of codes [43] gives a good idea of what minimum distances are achievable. For large n most codes in the ensemble will have a minimum distance close to $\delta_{\gamma r}n$ where $\delta_{\gamma r} \neq 0$ is the value of λ which, for a given row weight r and column weight γ , gives [43]

$$B_{\gamma r}(\lambda) = (\gamma - 1)H(\lambda) - \frac{\gamma}{r}(\mu(s) + (r - 1)\ln 2) + \gamma s\lambda = 0,$$

where

$$\lambda = \frac{\mu'(s)}{r} = l/n.$$

Gallager demonstrated that for large enough n a reasonable proportion of all codes with a given set of parameters can achieve a minimum distance $d = \delta_{\gamma r}n$, and so we assume that a code with this minimum distance can be constructed without too many attempts. The smaller the code length, the larger the variation in minimum distance of individual codes from the ensemble, however for small n a large number of attempts to construct a "good" code is more reasonable and so we assume that it is possible, without too much difficulty, to construct a random parity-check matrix with d at least $\delta_{\gamma r}n$.

Given this reasoning, Fig. 2.5 shows the minimum distance values which can be achieved for random, column weight 3, LDPC codes at various rates and for lengths up to 5000. Column weight 3 codes are shown as these are the randomly constructed regular LDPC codes generally considered best for use with sum-product decoding [71].

Overall, in designing new LDPC codes the following code properties are generally considered:

- **Girth:** Cycles in the Tanner graph affect decoding convergence, and the smaller the code girth, the larger the effect on decoding performance [100, 77, 71]. However the avoidance of all cycles is also not desirable since a linear code of rate $k/n \geq 0.5$ without any cycles has a minimum distance of at most 2 [38]. The best approach is then to avoid small cycles in the Tanner graphs of LDPC codes.

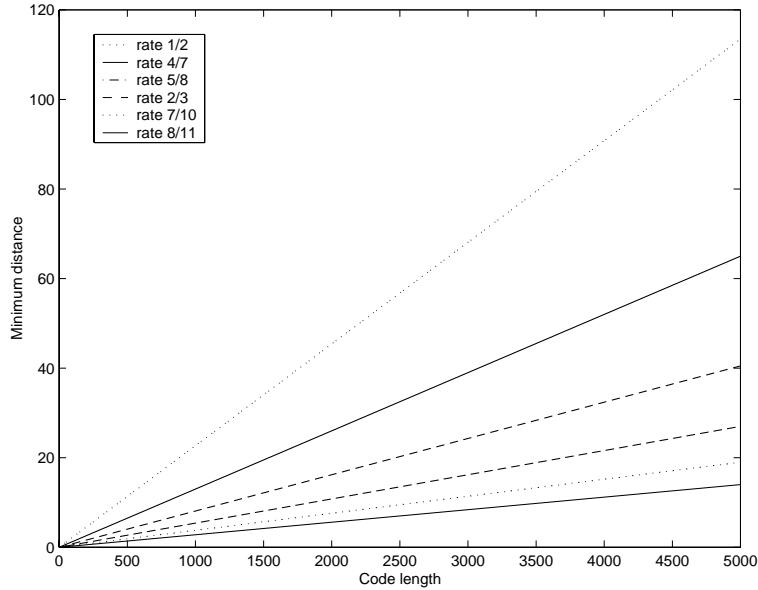


Figure 2.5: Minimum distance estimates for ensembles of LDPC codes with column weight 3

- **Expansion:** Sipser and Spielman [96] showed that the expansion of the graph is a significant factor in the application of iterative decoding. Using only a simple hard decision decoding algorithm they proved that a fixed fraction of errors in an LDPC code can be corrected in linear time provided that the Tanner graph of the code is a good enough expander. That is, any subset S of bit vertices of size m or less is connected to at least $\epsilon|S|$ constraint vertices, for some defined m and ϵ [96].
- **Stopping set distribution:** As described above, the stopping set distribution of an LDPC code determines its performance on the binary erasure channel, however the extent to which it affects the performance of the codes on other channels is not yet determined. Nevertheless, the minimum stopping set size, S_{\min} , is an estimate of a code's performance with sum-product decoding in a similar manner as minimum distance is an estimate of a codes performance with maximum likelihood decoding [34, 87, 86].
- **Minimum distance:** The code minimum distance plays less of a role in the performance of sum-product decoding than maximum likelihood decoding and is therefore less important for LDPC codes. However, a poor minimum distance will affect the performance of LDPC codes at very high signal-to-noise ratios, causing an error floor.
- **Density:** A sparse parity-check matrix is essential for the low complexity operation of a sum-product decoder. However, there is a tradeoff between the density of H and the code minimum distance and stopping set size.

While the first five properties presented concern the decoding performance of the LDPC codes, the following two consider their implementation. Since very long codes are decoded effectively with sum-product decoding, the extent to which they can be effectively encoded and stored will be important factors in how widely LDPC codes are employed.

- **Encoding:** Perhaps the biggest deterrent to the implementation of LDPC codes is the lack of an efficient encoding algorithm for them. While applying Gaussian elimination to transform the parity-check matrix into upper triangular form and using back substitution is effective, the new matrix is no longer sparse and the computational complexity of encoding is $O(n^2)$. The requirement then is to produce codes with the structure to enable linear encoding complexity, either by enabling a sparse generator matrix to be constructed using only row and column swaps as in [89] or by employing the shift invariance of cyclic codes or, more generally, constructing group invariant codes as in [101].
- **Implementation:** Where randomly constructed LDPC codes are employed the entire matrix must be stored, with repercussions for the system implementation requirements. There are two main ways that the implementation complexity can be improved. If hard-wiring of the Tanner graph is employed, as in [9], regularity of the LDPC codes will translate directly into regularity in the VLSI layout, improving parallelization. Alternatively, where on-line construction of the codes is employed, deterministic codes that can be completely described by only a small number of parameters will prove very beneficial.

Since their introduction by Gallager, LDPC codes have for the most part been constructed pseudo-randomly with modifications made to Gallager's original construction to improve code properties such as girth or expansion [52, 71, 73, 96, 97, 78]. This is done in one of two ways: either by pseudo-randomly assigning entries in a sparse binary parity-check matrix or by randomly assigning edges in a bipartite graph, the code Tanner graph. In this thesis the traditional pseudo-random LDPC codes constructed by MacKay and Neal [71, 73, 81] will be used as a point of comparison to the new algebraic LDPC codes we construct. MacKay and Neal's construction of a length n , rate $\approx R$ LDPC code, with column weight γ and as close to regular as possible is as follows:

Step 1 an all zero $n(1 - R) \times n$ matrix is constructed

Step 2 γ ones are placed in each column of H with an attempt made to keep the number of ones in each row approximately the same

Step 3 extra ones are randomly added so that the weight of each row is greater than one

Step 4 to remove 4-cycles, ones are moved randomly within columns involved in the cycle.

While random constructions of LDPC codes have produced very good codes, particularly for quite long lengths, it is sometimes difficult to construct 4-cycle free codes randomly, particularly for high rate codes. As well, there is a trade off between removing code cycles and obtaining code regularity, as the process of removing 4-cycles can cause the row weights in particular to be less regular. Finally, randomly constructed codes also have implementation disadvantages stemming from the difficulty associated with storing and encoding codes described by random parity-check matrices.

Algebraic constructions of LDPC codes have received much less attention but have produced some very good LDPC codes with simple encoding and decoding. Tanner founded the topic of algebraic methods for constructing graphs suitable for sum-product decoding in [100], and more recently examined bipartite graphs defining binary quasi-cyclic codes [106, 103, 105] which can be analyzed using his generalized transform methods [101]. Other authors have looked at existing algebraic structures to produce LDPC codes free of 4-cycles, such as the array codes considered in [39, 40] and the finite geometry codes shown to make excellent LDPC codes by Lucas et al. and Kou et al. in [70, 60]. The length 73 finite geometry code has also been implemented on an integrated circuit using iterative decoding by Karplus and Krit [56].

Graph-based constructions for codes with good girth have been presented by Margulis [74], and extended by Rosenthal and Vontobel [90] and Lafferty and Rockmore [62]. Vontobel and Tanner have also used finite generalized polygons to construct LDPC codes which give the maximum possible girth for a graph with given diameter [118]. Other constructions for LDPC codes have been presented which have a mixture of algebraic and randomly constructed portions [10].

An important outcome of the work with algebraic codes was the demonstration that highly redundant parity-check matrices can lead to very good iterative decoding performances without the need for very long block lengths [70, 118]. While the probability of a random graph having a highly redundant parity-check matrix is vanishingly small, the field of combinatorial designs offers a rich source of algebraic constructions for matrices which are both sparse and redundant.

The premise of this thesis is that combinatorial constructions of LDPC codes can provide significant improvements to randomly constructed LDPC codes, particularly when small to medium code lengths are concerned. A background to these combinatorial designs is presented in the remainder of this chapter.

2.2 Combinatorial designs

A combinatorial design is an assignment of a set of objects into subsets subject to some defined condition on the size, structure or incidence of the subsets. A simple combinatorial problem for example would be to arrange a set of seven academics into seven committees with three academics in each committee, every academic on the same number of committees, and each pair of academics serving together in exactly one committee.

Formally, an *incidence structure* $(\mathcal{P}, \mathcal{B}, \mathcal{I})$ consists of a finite non-empty set \mathcal{P} of points (academics) and a finite non-empty set \mathcal{B} of subsets of those points called blocks (committees), together with an incidence relation $\mathcal{I} \subseteq \mathcal{P} \times \mathcal{B}$. A point P and block B are *incident*, denoted $P \in B$, if and only if $(P, B) \in \mathcal{I}$. A *design* \mathcal{D} is an incidence structure with a constant number of points per block and no repeated blocks. A design is *regular*, if the number of points in each block, and the number of blocks which contain each point, designated γ and r respectively, are the same for every point and block in the design. In the field of combinatorial designs the block size is usually denoted by the symbol k , however we use γ in this thesis to avoid confusion with the use of k for the number of message symbols in the code, and with Gallager's use of k to denote the row weight of his LDPC codes.

Every design can be represented by a $v \times b$ binary matrix N , $v = |\mathcal{P}|$, $b = |\mathcal{B}|$ called an *incidence matrix*, where each column in N represents a block B_j of the design and each row a point P_i with the (i, j) th entry of N a 1 if point i is contained in block j otherwise it is 0. Thus

$$N_{i,j} = \begin{cases} 1 & \text{if } P_i \in B_j, \\ 0 & \text{otherwise.} \end{cases} \quad (2.14)$$

The *incidence graph* of \mathcal{D} has vertex set $\mathcal{P} \cup \mathcal{B}$, with two vertices x and y connected if and only if $x \in \mathcal{P}$, $y \in \mathcal{B}$ and $x \in y$, or $x \in \mathcal{B}$, $y \in \mathcal{P}$ and $y \in x$, and is thus a bipartite graph.

In this section two classes of designs, Steiner 2-designs and partial geometries, will be considered further before the links between designs and codes are discussed.

2.2.1 Steiner 2-designs

For a t -*design* every set of t points are incident in a constant number, λ , of blocks together and thus for 2-designs, every *pair* of points occur in λ blocks together. 2-designs are also called balanced incomplete block designs (BIBDs), and are denoted as $2-(v, b, r, \gamma, \lambda)$ or $2-(v, \gamma, \lambda)$ designs. The designs of interest in this thesis are the 2-designs with λ equal to 1, called *Steiner 2-designs*, in which every pair of points occur together in exactly one block and so each pair of blocks intersect in at most one position.

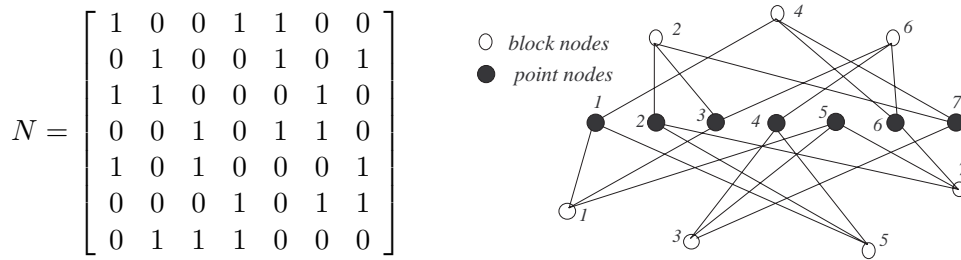


Figure 2.6: An incidence matrix and incidence graph of the $2-(7, 3, 1)$ design in Example 2.2.1. The graph nodes are numbered to indicate the ordering of point nodes into rows and block nodes into columns of N .

The parameters of a $2-(v, b, r, \gamma, \lambda)$ design are constrained by

$$vr = b\gamma, \quad (2.15)$$

as vr and $b\gamma$ are each the number of non-zero entries in the incidence matrix. Further, any given point P is in a pair with $\gamma - 1$ points in each of the r blocks containing it and so there are $r(\gamma - 1)$ pairs involving P . However, P must be paired with each of the $v - 1$ other points exactly λ times and so

$$\lambda(v - 1) = r(\gamma - 1). \quad (2.16)$$

Thus the choice of any three of the parameters completely specifies the design with the remaining two parameters determined by (2.16) and (2.15).

Example 2.2.1 *As a solution to our problem of organizing academics, the set of points (academics)*

$$\mathcal{P} = \{1, 2, 3, 4, 5, 6, 7\}$$

can be formed into a design with blocks (committees),

$$\mathcal{B} = \{[1, 3, 5], [2, 3, 7], [4, 5, 7], [1, 6, 7], [1, 2, 4], [3, 4, 6], [2, 5, 6]\}. \quad (2.17)$$

This design can easily be seen to satisfy the regularity constraint with $\gamma = 3$ points in every block (3 academics in every committee) and each point in exactly $r = 3$ blocks (each academic on exactly three committees). Each pair of points (academics) occurs in one block (committee) together thus the blocks in \mathcal{B} form a t -design or, more precisely, a 2-design with parameters $2-(7, 3, 3, 1)$. An incidence matrix and incidence graph for this design are shown in Fig 2.6 using an arbitrary numbering of blocks 1–7 respectively from left to right.

Three types of designs in particular will be applied to LDPC code construction in the following chapters:

- **Resolvable designs:** A design is resolvable if the blocks of the design can be arranged into r groups, called resolution classes, such that the $\frac{v}{r}$ blocks of each resolution class are disjoint, and each class contains every point precisely once. Not all designs are resolvable, those that can be are restricted to designs with parameters $v \equiv 0 \pmod{r}$. Design resolvability plays a key role in the LDPC codes constructed in Chapter 6.
- **Configurations:** A (v, b, r) -configuration is a set of b lines on v points such that each line contains r points and each point is in r lines [26, page 253]. A configuration differs from a design in that each pair of points are connected in *at most* one line together. Alternatively, an *n-line configuration* can be defined as any subset of n blocks of a design [51]. In this case the requirement that each point is contained in a fixed number of lines (blocks) will not necessarily hold. The number and type of these n -line configurations in STS designs will be useful in this work to analyze the distance and stopping set distributions of LDPC codes from the designs.
- **Difference families:** Designs from difference families will prove very useful in Chapter 7 for the design of cyclic and quasi-cyclic LDPC codes.

Definition 2.2.1 [3] *The t γ -element subsets of an Abelian group \mathcal{G} , D_1, \dots, D_t with $D_i = \{d_{i,1}, d_{i,2}, \dots, d_{i,\gamma}\}$ form a (v, γ, λ) difference family if the differences $d_{i,x} - d_{i,y}$, ($i = 1, \dots, t$; $x, y = 1, \dots, \gamma, x \neq y$) give each non-zero element of \mathcal{G} exactly λ times.*

The set $D+g = \{d_1+g, d_2+g, \dots, d_k+g\}$, $g \in \mathcal{G}$, is a translate of D and the translates $D, D+g_1, \dots, D+g_{v-1}$ of the sets of a difference family make up the blocks of a design. Difference families defined on any group isomorphic to $Z_v = \{0, 1, 2, \dots, v-1\}$ produce cyclic designs. As in this case a translate is a cyclic shift of D . Where the difference family consists of just one set, called a *difference set*, the design incidence matrix consists of a single circulant matrix.

An extension to the method of difference families, called *mixed difference systems*, allows several copies of each element of an Abelian group to be used. For \mathcal{G} an Abelian group of order v let $\mathcal{H} = \mathcal{G} \times Z_t$. Then \mathcal{H} consists of tv elements, t copies of each element of \mathcal{G} . An element $(a, i) \in \mathcal{H}$ represents the i th copy of the element a in \mathcal{G} .

Definition 2.2.2 [3] For $\mathcal{H} = \mathcal{G} \times Z_t$, the k -subsets $D_1, \dots, D_s \in \mathcal{H}$ form a mixed difference system if there exists an integer λ such that for every $i \in \{1, 2, \dots, t\}$, every element $g \in \mathcal{G}$ occurs λ times as the difference $(x, i) - (y, i)$ where $g = x - y$ and $(x, i), (y, i) \in D_1, \dots, D_s$.

As for difference families the translates of the set D_l are the sets $D_l + g := \{(x + g, i) : (x, i) \in D_l\}$ for all $g \in \mathcal{G}$. The translates of the sets of a mixed difference system form the blocks of a 2 - $(tv, sv, sk/t, k, \lambda)$ design with point set the elements of \mathcal{H} [3, Theorem 2.4.1]. The resolvable Steiner 2 - $(v, 3, 1)$ -designs we employ in Chapter 6 are constructed using these mixed difference families.

An area closely related to designs is that of finite geometries (see e.g. [4]). The finite projective geometry of a vector space V of dimension $m + 1$, $\text{PG}(V)$, has as elements the subspaces of V . The points of $\text{PG}(V)$ are the 1-dimensional subspaces of V , the lines are 2-dimensional subspaces of V , the planes are 3-dimensional subspaces of V and so on to hyperplanes the m -dimensional subspaces of V . The incidence between elements of $\text{PG}(V)$ corresponds to containment between subspaces of V . Thus a point P is incident with a line L in $\text{PG}(V)$ if the 1-dimensional subspace corresponding to P is contained in the 2-dimensional subspace corresponding to L . For V a vector space of dimension $m + 1$ over the field $F = \text{GF}(q)$, the projective geometry is often written $\text{PG}(m, q)$. A Euclidean geometry $\text{EG}(V)$ has as elements the cosets $x + U$ of the subspaces U of V where x is any vector in V and incidence is again given by containment.

The projective and Euclidean geometries are an important source of designs. These designs can be formed by taking as points of the design the points of the geometries and as blocks the lines, planes or hyperplanes of the geometry with the incidence of the geometry carried into the design. Of interest in this thesis are the designs consisting of the points and lines of $\text{PG}(2, q)$ which are *finite projective planes* of order q . $\text{PG}(2, q)$ is a set of $q^2 + q + 1$ lines and $q^2 + q + 1$ points such that every line passes through exactly $q + 1$ points and every point is incident on exactly $q + 1$ lines. Further, any pair of points in the plane must be incident together in exactly one line. With these properties, the points and lines of a projective plane are the points and blocks of a 2 - $(q^2 + q + 1, q + 1, 1)$ design with the incidence of the design given by the incidence of the plane. Fig. 2.7 shows the typical representation of the finite projective plane of order 3. Many of the designs considered in Chapter 4 are derived using a finite projective plane as the starting point. The designs of points and lines of $\text{PG}(m, 2)$ will also be important in this thesis. These designs are the classical Steiner triple systems or 2 - $(v, 3, 1)$ designs.

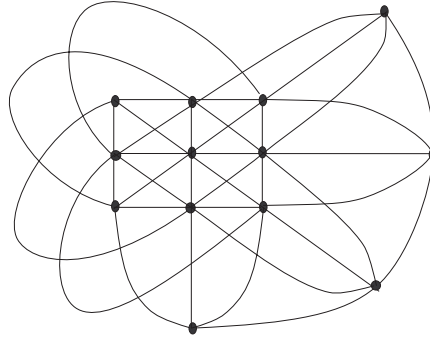


Figure 2.7: The finite projective plane of order 3 consists of 13 points on 13 lines

2.2.2 Partial geometries

Partial geometries, first presented in [12], are a set of points, and subsets of those points, called blocks or lines, completely specified by three parameters, s , t , and α . That is, a partial geometry, denoted $\text{pg}(s, t, \alpha)$, satisfies the following properties [19, p. 33]:

- P1.** Each point P is incident with $t + 1$ blocks and each block B is incident with $s + 1$ points.
- P2.** Any two blocks have at most one point in common.
- P3.** For any non-incident point-block pair (P, B) the number of blocks incident with P and intersecting B equals some constant α .

Partial geometries differ most noticeably from finite geometries in that a pair of points in the geometry need not have a line defined through them, a property contrary to the traditional definition of a geometry, and the reason for their name. However, the subset of the partial geometries with $\alpha = s + 1$ are exactly Steiner 2-designs since if a point P is not incident in a block B , every block incident with P must intersect B and thus every pair of points must appear in a block together.

Our use of partial geometries, in Chapter 5, makes essential use of the graphs defined by geometries. Any design can be described by a *point graph* \mathcal{G} which has vertex set $\mathcal{V} = \mathcal{P}$ and $v = |\mathcal{V}|$ vertices. An edge connects two vertices if the corresponding points are incident with the same block $B \in \mathcal{B}$, and it is said that the vertices (and corresponding points) are *connected*. The *adjacency matrix* of \mathcal{G} is then a $v \times v$ matrix A , indexed by the vertices of \mathcal{G} , and defined by

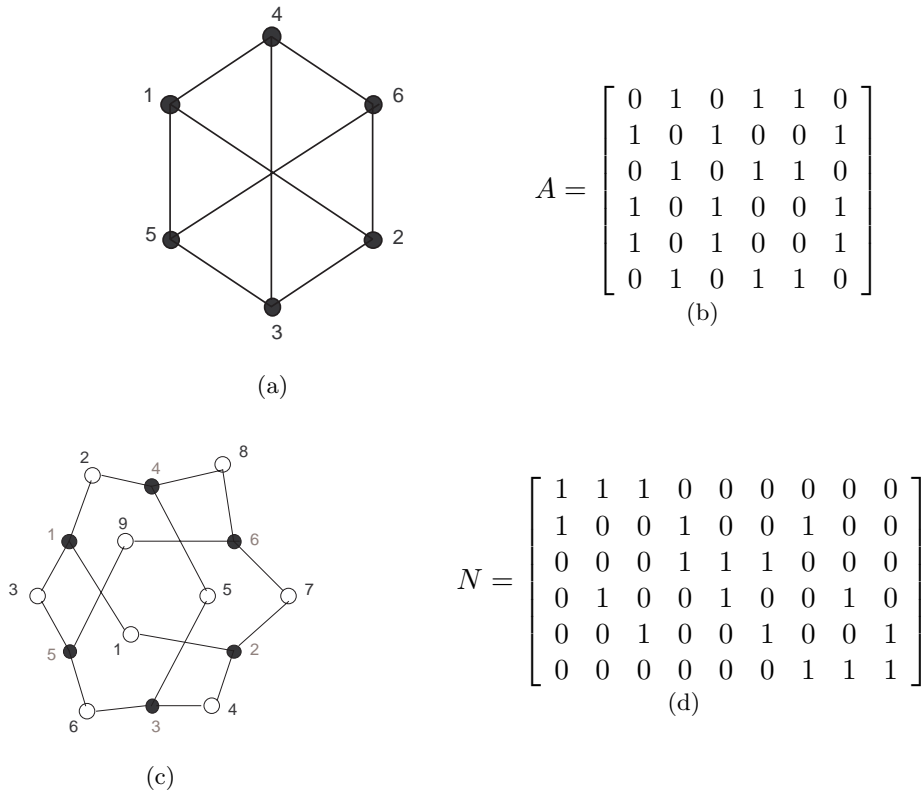


Figure 2.8: The partial geometry $pg(1,2,1)$. Part (a) shows the incidence of points and lines of the partial geometry and can also be thought of as the vertices and edges of the graph which is the point graph of the geometry. Taking an arbitrary numbering of points as shown, the adjacency matrix is given in part (b). In parts (c) and (d) an incidence graph and incidence matrix are given for the arbitrary ordering of the lines of the geometry into blocks of the design as shown.

$$A_{i,j} = \begin{cases} 1 & \text{if } P_i, P_j \in B \text{ for some } B \in \mathcal{B}, i \neq j \\ 0 & \text{otherwise.} \end{cases} \tag{2.18}$$

Fig 2.8 shows the incidence matrix, incidence graph, adjacency graph and adjacency matrix of a $pg(1,2,1)$ design.

A graph \mathcal{G} is said to be *regular* if each vertex is connected to exactly n_1 other vertices. If further, any two connected vertices of \mathcal{G} are both connected together to exactly p_1 other vertices, and any two unconnected vertices are both connected to exactly p_2 vertices together, the graph is *strongly regular* [114].

The point graph of a partial geometry $\text{pg}(s, t, \alpha)$ is strongly regular with parameters [26]:

$$\begin{aligned} n_1 &= s(t+1), & p_1 &= s-1+t(\alpha-1), & p_2 &= \alpha(t+1), \\ |\mathcal{P}| &= \frac{(s+1)(st+\alpha)}{\alpha} & \text{and} & & |\mathcal{B}| &= \frac{(t+1)(st+\alpha)}{\alpha}. \end{aligned} \quad (2.19)$$

To see this, note that a point, P_i , of the partial geometry is incident in $t+1$ blocks, and in each of these blocks connected to s other points. There can be no overlap of these $s(t+1)$ points, since the point P_i is in every block and property **P2** of partial geometries must hold. Thus we have that each vertex of the graph is connected to exactly $n_1 = (t+1)s$ other vertices. A pair of connected points (P_i, P_j) are each connected to the $s-1$ other points on the same block in which they are connected, and, by property **P3** of partial geometries, in each of the t other blocks incident on point P_i there are $\alpha-1$ points connected to the point P_j and so P_i and P_j are connected together to exactly $p_1 = s-1+t(\alpha-1)$ other vertices. Finally, consider an unconnected pair of points (P_i, P_j) . The point P_i is incident with $t+1$ blocks each of which is not incident with the point P_j . Again by property **P3** each of these blocks is incident with α points which are also connected to P_j and so two unconnected points are both connected together to exactly $p_2 = (t+1)\alpha$ other vertices.

For a given set of parameters that define a design there may be more than one way to arrange the point set into blocks which satisfy the requirements of the design. Two designs are defined as being *isomorphic* if there is a bijection which maps points to points such that every block in the original design is a block in the new design. Much of the work in the field of combinatorial designs, including that relating to codes, is aimed at proving the existence or non existence of designs with a specified parameter set and in counting the number of non-isomorphic designs if they exist.

2.2.3 Codes from designs

There are two main focuses of previous work linking codes and designs. Firstly, the properties of the codes defined by designs are employed to characterise the designs. Secondly, designs are used to give a combinatorial description for existing algebraic codes such as the Reed Muller codes (see e.g. [4, 110]). The monograph by Assmus and Key [4] gives an excellent treatment of this topic.

From a coding theory perspective, the blocks of a design can be associated with the codewords of a code. The codes defined in this way using the designs from projective and Euclidean geometries, are called *geometric codes*, and are particularly important in coding theory. The minimum weight codewords of the geometric codes are exactly the incidence vectors of the blocks of a projective or Euclidean geometry design [32, 33]. The designs of the geometries $\text{PG}(m, 2)$ have as blocks the minimum weight codewords

of Reed-Muller and punctured Reed Muller codes. While the generalized Reed-Muller codes have as minimum weight codewords the blocks of the geometries $\text{PG}(m, q)$ [4]. The geometric point of view can achieve simplifications for the analysis of the Reed-Muller and generalized Reed-Muller codes (see [4, Chapter 5] for a detailed description).

From a combinatorial perspective, a design is generally associated with a code of length v defined as the column space of the design incidence matrix N , called its *block code*. The block code of a design can be thought of as the code with generator matrix given by N^T . Most of the general results about designs in codes including the celebrated Assmus–Mattson theorem [5] consider these block codes (see e.g. [4]). However, the codes spanned by the row space of N , called *point codes*, have also been used to provide useful information about the design structure (see e.g. [109] and [110]).

Less well known are the dual codes of the block and point codes. The dual of the block codes have as their dual space the column space of N^T . Thus the transpose of the incidence matrix of the design is the parity-check matrix of the code. The dual of the point code, using the incidence matrix of a design as the parity-check matrix of the code, are more rarely considered. However these are the codes which have the properties required to perform well with sum-product decoding and so the codes from designs considered in this thesis are defined in this way.

Designs have also played a role in defining new codes such as in the case of the difference set cyclic codes first presented by Rudolph and Weldon [91, 119] (see e.g. [65] for a detailed description). In this case the codes were defined using the transpose of the incidence matrix of the projective geometry design $\text{PG}(2, q)$ as the code parity-check matrix. The properties of these projective geometry designs are well suited to the majority logic decoding algorithm. More recently these codes have had an important impact on the field of iterative decoding when it was shown by Lucas et al. and Kou et al. that the properties that make them majority logic decodable also make them excellent LDPC codes [70, 60]. The success of the finite geometry codes as LDPC codes are the motivation for our work in designing error correction codes using combinatorial designs with the aim of devising powerful new algebraic LDPC codes.

LDPC CODES FROM STEINER SYSTEMS

In this chapter combinatorial designs are employed to create new error correction codes for use with sum-product decoding. Steiner 2-designs are used to provide systematic constructions for regular LDPC codes with Tanner graphs free of 4-cycles and with deterministic code properties. The success of employing Steiner 2-designs to create $(3,r)$ -regular LDPC codes is demonstrated by deriving the code properties such as minimum distance, girth, and stopping set distribution, and by simulating their decoding performance. We focus also on exploring the effect of different design properties on the LDPC codes derived from them. In particular, we see that design configurations play an important role in determining the distance and stopping set distributions of the LDPC codes from designs.

3.1 Introduction

The idea of designing LDPC codes using combinatorial designs derives from the observation that the incidence matrix of a combinatorial design is reminiscent of the parity-check matrix of a regular LDPC code. Both are sparse binary matrices with constant row and column weights and, more significantly, the desire for LDPC codes with Tanner graphs free of 4-cycles is analogous to a well studied problem in combinatorics, that of constructing Steiner 2-designs.¹

¹While we made this observation independently, several other researchers made the same connection at a similar time. This connection was perhaps first made by MacKay in his 2000 paper using the incidence matrix of Steiner triple system (STS) designs for high-rate LDPC codes [72]. That same year Mittelholzer, in an IBM research report [80], demonstrated the link between projective geometry LDPC codes [70] and balanced incomplete block designs, and generated new LDPC codes from further BIBDs, extending the work of [72]. The use of Steiner triple systems as LDPC codes was also the central theme in a presentation by Vasic in September of 2001 at the Cairns Information Theory Workshop, and later published in [115]. In this work the cyclic nature of certain STS designs was suggested as a means to produce quasi-cyclic

A Steiner 2-design is a balanced incomplete block design with the property that every pair of points in the design is incident in exactly one block together and so the incidence graph of a Steiner 2-design has girth greater than 4 (see section 2.2.1). When we use the incidence matrix of a Steiner 2-design as the parity-check matrix of an LDPC code, the design incidence graph is the Tanner graph of the code. Thus existing algebraic constructions for Steiner 2-designs yield algebraic constructions for LDPC codes which are both regular and have Tanner graphs free of 4-cycles. These LDPC codes are in combinatorial design terminology the dual of the point codes.

The properties of combinatorial designs will also enable us to derive expressions for the code properties as a function of the design parameters and an important part of this work will be to determine which designs have the best structure for LDPC codes. This requires first determining the design properties that translate into the required code properties before secondly finding specific designs with these properties.

The focus of this chapter then is to define the properties of codes derived from combinatorial designs, and subsequently evaluate their decoding performance. Firstly, Section 3.2 presents expressions for the properties of LDPC codes from Steiner 2-designs as a function of the design parameters. Following this, Steiner triple systems in particular are considered in Section 3.3 and we see that they provide a construction for excellent $(3,r)$ -regular LDPC codes. Lastly, we conclude with a discussion in Section 3.4.

3.2 LDPC codes from Steiner 2-designs

As an example of the role of a Steiner 2-design in defining an LDPC code, Fig. 3.1 shows an LDPC code created from the $2-(v, 3, 1)$ design shown in Fig. 2.6, taking the incidence matrix of the design as the parity-check matrix of the code. At first glance the code presented may seem trivial, due to the equal number of rows and columns in H , however, the rank over $\text{GF}(2)$ of H is 4 and thus the code has a dimension of 3. A valid generator matrix for this code, constructed using Gaussian elimination, is given by:

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

The incidence matrix, N , of any t -design, $t-(v, b, r, \gamma, \lambda)$ has v rows, length b and

LDPC codes. At the same workshop, in the paper [53], we exploited a different family of STS designs, called Kirkman triple systems which have the property of resolvability, allowing construction of LDPC codes with a wider range of lengths and rates but which are still regular. Finally, the concluding remarks of the paper by Kou et al. [60], published in November of 2001, make reference to the possible use of BIBDs to construct LDPC codes, and there has been much subsequent work in this area.

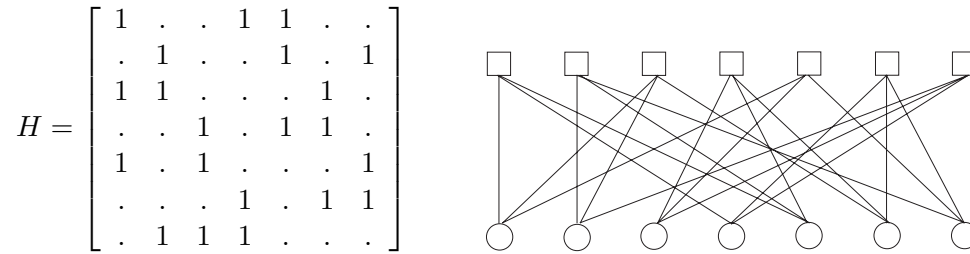


Figure 3.1: The parity-check matrix and Tanner graph of a $[7,3,4]$ code from the $2-(7,3,1)$ design in Example 2.2.1. Zero entries in the parity-check matrix are represented by dots.

constant column and row weights, γ and r respectively. Further, the incidence matrix will be sparse if t -designs with γ small relative to v are chosen. A 4-cycle is a pair of code bits which occur together in the same two checks. So in order to avoid 4-cycles in the LDPC codes from designs, we require that no pair of points in the design occur together in more than one block. The incidence between points and blocks in a t -design are controlled by the choices of t and λ ; every set of t points occurs together in λ blocks. So to avoid a pair of points together in two blocks requires $t = 2$ and $\lambda = 1$, so that two points occur in exactly one block together. The designs with $t = 2$ are 2-designs and 2-designs with $\lambda = 1$ are Steiner 2-designs. Thus the LDPC codes defined using the incidence matrix of a Steiner 2-design as the code parity-check matrix are regular with Tanner graphs free of 4-cycles.

Note that it is possible to define a 4-cycle free regular LDPC code with parity-check matrix given by either the incidence matrix of a Steiner 2-design or its transpose. However, Steiner 2-designs generally have more blocks than points and so N and not N^T is used as the parity-check matrix of the code to ensure a non-zero number of code message bits. Using $H = N$, a Steiner 2-design with parameters $(v, b, r, \gamma, 1)$ will produce a code with a (γ, r) -regular parity-check matrix with v parity-check equations and length $n = b$.

We have narrowed our consideration of t -designs to only those designs with $t = 2$ and $\lambda = 1$. In the following sections the impact on the LDPC code properties of each choice of the remaining t -design parameters; v , b , r and γ , are considered. First though we consider the possible design parameters.

If a $2-(v, \gamma, 1)$ design exists then the requirements (2.15) and (2.16) must be satisfied and thus the parameters of a Steiner 2-designs are constrained by:

$$(v - 1) \equiv 0 \pmod{(\gamma - 1)} \quad (3.1)$$

and

$$v(v - 1) \equiv 0 \pmod{(\gamma(\gamma - 1))}. \quad (3.2)$$

Different families of Steiner 2-designs are classified by the relationship between the parameters v and γ . (The parameters r and b are dependent on the choices of v and γ by (2.16) and (2.15)). The families of Steiner 2-designs considered in this thesis are:

- oval designs, which are specified by $v = (\gamma - 1)(\gamma - 2)/2$ for all integers γ ,
- unital designs, which are specified by $v = \gamma^3 - 3\gamma^2 + 3\gamma$ for all integers γ ,
- Steiner triple systems (STS), which have constant $\gamma = 3$,
- Steiner 2-designs with fixed $\gamma \geq 4$,
- projective planes, or projective geometry (PG) designs, which are specified by $v = (\gamma - 1)^2 + \gamma$ for all $\gamma - 1$ a prime power, and finally
- affine planes, or affine (Euclidean) geometry designs, which are specified by $v = (\gamma - 1)^2$ for all $\gamma - 1$ a prime power.

The projective planes are $2-(q^2+q+1, q+1, 1)$ designs defined by the points and lines of projective geometries, and the affine (Euclidean) planes are $2-(q^2, q^2+q, q+1, q, 1)$ designs derived from Euclidean geometries. The projective planes have been widely considered as error correction codes, firstly as one step majority logic decodable codes [91, 119], and more recently as LDPC codes [70, 59, 60]. Further, the finite Euclidean geometry codes presented in [60, 67] are closely related to affine planes. We will not present more on Euclidean geometry or projective geometry LDPC codes in this thesis, other than as a point of comparison, as they are well represented in the LDPC coding theory literature [70, 59, 60, 66, 67, 61, 122].

There is often more than one arrangement of v points into size γ blocks that gives a Steiner 2-design. For example, there are exactly 2 non-isomorphic STS designs on 13 points, 80 non-isomorphic STS designs on 15 points and 11,084,874,829 non-isomorphic STS designs on 19 points [57]. Non-isomorphic Steiner 2-designs with the same parameters can be differentiated by, among other things, the rank of their incidence matrices, the number of configurations in the design and the properties of their automorphism group. Each of which will be important in determining the performance of the codes derived from the different Steiner 2-designs.

Where the particular $2-(v, \gamma, 1)$ design is not important we will denote by $\mathcal{C}(v, \gamma)$ an LDPC code defined as the dual of the point code of a Steiner 2-design with parameters v and γ .

3.2.1 The rate of codes from Steiner 2-designs

The incidence matrices of Steiner 2-designs have the maximum number of columns possible in a binary matrix with v rows, column weight γ and column intersection at most 1. This implies that the LDPC codes from Steiner 2-designs are the highest rate possible for 4-cycle free codes with a column weight γ and v parity-check equations. The one caveat is that rank deficient parity-check matrices will increase the code rate and indeed many of the codes from Steiner 2-designs do have higher rates than $\frac{b-v}{b}$.

As there may be linearly dependent rows in N , the dimension of the code is given by $k = n - \text{rank}_2(N)$. The rank of an incidence matrix is trivially at most v and so the dimension of the codes, $\mathcal{C}(v, \gamma)$, is at least $b - v$. Thus using (2.15) and (2.16) the code rate, R , is lower bounded by:

$$R \geq \frac{b-v}{b} \geq 1 - \frac{v\gamma(\gamma-1)}{v(v-1)} \geq 1 - \frac{\gamma(\gamma-1)}{v-1}. \quad (3.3)$$

LDPC codes from Steiner 2-designs with smaller column weight will be higher rate and similarly the rate will increase with code length for a given column weight.

A lower bound on the 2-rank of the incidence matrix of a Steiner 2- $(v, b, r, \gamma, 1)$ design is [16, Theorem 4.2]

$$\text{Rank}_2(N) \geq (\gamma-1)\sqrt{(r-1)r/\gamma},$$

and so an upper bound on the rate of the LDPC codes from Steiner 2-designs is

$$R \leq 1 - \frac{\gamma(\gamma-1)\sqrt{(v-1)(v-\gamma)/\gamma}}{v(v-1)}.$$

3.2.2 The girth of codes from Steiner 2-designs

The Tanner graph of the codes, $\mathcal{C}(v, \gamma)$ is the incidence graph of the Steiner 2- $(v, \gamma, 1)$ designs which defines them. The girth of the incidence graph of a design is not a focus of combinatorics, however the design properties allow us to determine some of the graph properties.

As no two points in a Steiner 2-design can be incident in two blocks together, 4-cycles are automatically avoided in the Tanner graph of all LDPC codes obtained in this way. However the requirement that every pair of points occur in exactly one block together guarantees the existence of 6-cycles in the Tanner graph of the codes. Consider any choice of three points P_1, P_2, P_3 not all in the same block in a Steiner 2-design. Every pair of points must occur in a block together so there is a block containing P_1 and P_2 and similarly a block containing P_2 and P_3 and a block containing P_1 and P_3 . These three blocks form a 6-cycle in the incidence graph of the design, as shown in Fig 3.2.

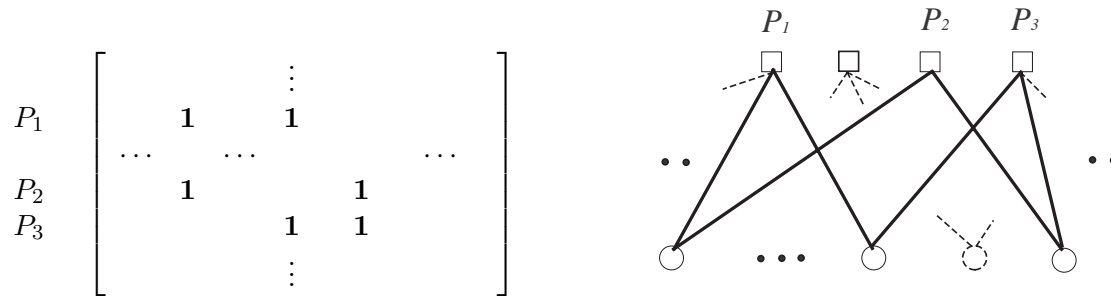


Figure 3.2: A 6-cycle through points (checks) P_1, P_2 and P_3 .

To extend this argument to counting the exact number of 6-cycles requires only to consider how many times such a combination occurs in a given design.

Lemma 3.2.1 *The exact number of 6-cycles, N_6 , in the Tanner graph of a code with parity-check matrix the incidence matrix of a $2-(v, b, r, \gamma, 1)$ design is:*

$$N_6 = \frac{b}{3} \binom{\gamma}{2} (r-1)(\gamma-1).$$

Proof. Every pair of points is incident together in exactly one block of the design. Take one pair (P_1, P_2) of points in a block B . The point P_1 is incident in $r-1$ blocks other than the block B , none of which contain the point P_2 . Similarly, the point P_2 is incident in $r-1$ blocks other than B none of which contain the point P_1 . Since every pair of points must be incident in a block together, all the $(r-1)(\gamma-1)$ points other than P_1 incident in the first set of blocks must be incident in the second set of blocks so as to be incident with the point P_2 . Likewise all the $(r-1)(\gamma-1)$ points other than P_2 incident in the second set of blocks must be incident in the first set of blocks so as to be incident with the point P_1 . Thus for each one of these $(r-1)(\gamma-1)$ points, P_i , there is a 6-cycle from P_1 to P_2 to P_i which includes block B and the block in each set containing P_i . For each of the points P_i the pair of blocks that complete the 6-cycle will be unique, otherwise the two points with a common pair of blocks will form a 4-cycle. In a block of the design there are $\binom{\gamma}{2}$ different pairs of points. There are b blocks in total, and given a single 6-cycle includes 3 blocks, the result follows. \square

Lemma 3.2.1 can be re-written by substituting for b and r using (2.15) and (2.16) to give the number of 6-cycles in a Steiner 2-design in terms of only the parameters v and γ :

$$N_6(\mathcal{C}(v, \gamma)) = \binom{\gamma}{2} \frac{v(v-1)(v-\gamma)}{3\gamma(\gamma-1)}. \quad (3.4)$$

This result is true of all the Steiner 2-designs regardless of the structure of the design.

As would be expected, the larger the column weight, and the more rows, in the design the more 6-cycles.

3.2.3 The stopping set distribution of codes from Steiner 2-designs

A stopping set is a set of codeword bits with the property that every parity-check equation connected to a codeword bit in the stopping set is connected to at least two codeword bits in the set. Recently, it was shown in [83] that for codes which are free of 4-cycles, the size of the smallest stopping set is at least one greater than the minimum column weight of H and thus the codes presented in this chapter have minimum stopping set size

$$S_{\min} \geq \gamma + 1. \quad (3.5)$$

More generally, a stopping set of size S will require a set of S columns in H such that every row which intersects a column in the set intersects at least two columns in the set.

Recall from Section 2.2.1 that a subset of the columns of the incidence matrix of a t -design is called a configuration, and so the type and distribution of configurations in the Steiner 2-designs will determine the stopping sets distribution in the LDPC codes. Some configurations, called *constant* configurations, occur the same number of times in every possible design, other configurations vary in number in across different, non-isomorphic designs. In the terminology of configurations a block is often called a line and so a configuration of l blocks is called an l -line configuration. A (k, l) -configuration in a Steiner 2-design is simply a set of l blocks whose union contains precisely k points. The degree of a point is the number of configuration blocks which contain it.

Lemma 3.2.2 *An LDPC code $\mathcal{C}(v, \gamma)$ with minimum stopping set size S_{\min} requires a 2 - $(v, \gamma, 1)$ design without any l -line configurations, $2 \leq l < S_{\min}$, which have all point degrees at least 2.*

Proof. The proof is simply that these configurations are exactly the stopping sets in the LDPC codes. The lines of the configuration are a set of codeword bits and the points of the configuration the set of code parity-check equations through those bits. If all points in the configuration have degree at least 2, the resulting checks in the code are connected to at least two of the codeword bits in the set and it is thus a stopping set. \square

3.2.4 The minimum distance of codes from Steiner 2-designs

Massey's lower bound on minimum distance can be applied to LDPC codes from Steiner 2-designs. Define a set of parity-check sums as *orthogonal on a codeword bit e* if e is involved in every parity-check equation in the set and no other codeword bit is checked by more than one of the parity-check equations in the set.

Lemma 3.2.3 [75] *If in a linear code there are at least $d - 1$ check sums orthogonal on each digit, then the code has minimum distance at least d .*

By definition no two points in a Steiner 2-design share more than one block in common and so no two parity-check equations in an LDPC code, $\mathcal{C}(v, \gamma)$, will share more than one codeword bit in common. Thus the set of γ parity-check sums on each codeword bit in the code must be orthogonal and Massey's bound gives the minimum distance of the LDPC codes from Steiner 2-designs as:

$$d \geq \gamma + 1. \tag{3.6}$$

To increase this bound on the minimum distance we observe that the distribution of configurations in a Steiner 2-design exactly determine the LDPC code distance distribution.

Lemma 3.2.4 *An LDPC code $\mathcal{C}(v, \gamma)$ with minimum distance d requires a 2 -($v, \gamma, 1$) design without any l -line configurations, $2 \leq l < d$, which have all point degrees even.*

Proof. As for Lemma 3.2.2 the proof is simply that a weight d codeword will occur if there are d columns of H which sum together to give zero modulo 2 and so an d -line configuration will result in codeword of weight d if each point in the configuration has even degree. \square

We will use this strategy in later sections to bound the minimum distance of certain families of LDPC codes from designs.

The nature of the random LDPC codes means that no bound on minimum distance can be given for a particular randomly constructed code, however Gallager's expression for the average minimum distance of an ensemble of codes [43] gives a good idea of what minimum distances are achievable and so serves as a reasonable comparison point (see Fig. 2.5). For example, a length 1000, rate-1/2 code requires a Steiner 2-design with $\gamma = 23$ to guarantee a better minimum distance than that possible using a $(3, r)$ -regular random construction, whereas for a rate 7/10 code of the same length a better minimum distance can be achieved by a code from a Steiner 2-design with only $\gamma = 3$. In general,

the higher the code rate the smaller the minimum distances that are possible for randomly constructed codes.

3.3 LDPC codes from Steiner triple systems

The focus of most randomly constructed regular LDPC codes is centered on codes with column weights of 3 and so Steiner 2-designs with $\gamma = 3$, *Steiner triple systems* (STS) or $2-(v, b, r, 3, 1)$ designs, seem a natural source for algebraic LDPC codes. The origin of Steiner triple systems is a prize problem in the Lady's and Gentleman's Diary in 1844 which asked for a system of γ -subsets of a v -set such that no t -subset occurs more than once in any of the γ -subsets. The case $t = 2$, $\gamma = 3$, of course being a $2-(v, 3, 1)$ design. The Rev. T. Kirkman showed in 1847 that a $2-(v, 3, 1)$ design could exist only when $v \equiv 1, 3 \pmod{6}$ and constructed these designs in all possible cases. Kirkman's work was generally overlooked and six years later Steiner, after whom the designs are named, posed the existence of STS designs but did not give a proof [3].

The requirement that $v \equiv 1, 3 \pmod{6}$ for Steiner triple systems follows easily from (3.1) and (3.2). From (2.15) there are

$$b = \frac{\lambda v(v-1)}{\gamma(\gamma-1)} = \frac{v(v-1)}{6} \quad (3.7)$$

blocks in the design and from (2.16) the row weight of the design is

$$r = \frac{3b}{v} = \frac{3v(v-1)}{6v} = \frac{v-1}{2}. \quad (3.8)$$

Steiner triple systems have previously been considered as LDPC codes by MacKay in [72] where a small upper bound on minimum distance of 10 was proved. As well, LDPC codes from STS designs which are cyclic were later presented by Vasic in [115] and the LDPC code from the length 1044 cyclic Steiner triple system was presented by Ammar et al. in [2] where simulation results show a decoding performance within 1.6 dB of the Shannon limit at a bit error rate (BER) of 10^{-4} .

Here we show that the upper bound on minimum distance presented by MacKay is not a significant hindrance to code performance and we look more closely at the structure of Steiner triple systems to determine which of the many different types of STS designs will make the best LDPC codes. An LDPC code with parity-check matrix the incidence matrix of an STS design we call an STS LDPC code.

3.3.1 Bounding the rate of STS LDPC codes

The incidence matrices of Steiner triple systems have the maximum number of columns possible in a binary matrix with v rows, column weight 3 and column intersection at most

1. Thus the maximum length, N_m , of an LDPC code with column weight 3 and m parity-check equations is called the Steiner bound (see e.g. [52])

$$N_m = m(m-1)/6.$$

This implies that the LDPC codes from Steiner triple systems are the highest rate possible for 4-cycle free codes with a column weight γ and v parity-check equations [72]. Thus the rate (assuming a full rank parity-check matrix),

$$R = \frac{b-v}{b} = \frac{v(v-1)/6 - v}{v(v-1)/6} = \frac{v-7}{v-1}$$

is considered the upper bound for LDPC codes with length $v(v-1)/6$, column weight 3, and no 4-cycles (see e.g. [52, Fig. 1]).

However, this is not necessarily the case as the rank of the STS LDPC parity-check matrix plays an important role in increasing the code rate. As we will see in Section 3.3.4, LDPC codes from carefully chosen Steiner triple systems have higher rates than $\frac{b-v}{b}$. Moreover it is not impossible that there exists a binary parity-check matrix with column weight 3 and no 4-cycles, which is not the incidence matrix of a Steiner triple system (and perhaps has more rows), but which has a lower rank than the equivalent length Steiner triple system incidence matrix and so has a higher rate.

In general, full rank incidence matrices are guaranteed for all choices of $v \equiv 1$ or $9 \pmod{12}$ while for $v \equiv 3$ or $7 \pmod{12}$ the 2-rank of N depends on the structure of the STS design (and hence its construction) and is bounded by [4]

$$\text{rank}_2(N) \geq v - \log_2(v+1). \quad (3.9)$$

Thus, the rate of STS LDPC codes can be given exactly

$$R = \frac{v-7}{v-1} \quad (3.10)$$

for STS LDPC codes with $v \equiv 1$ or $9 \pmod{12}$ parity-check equations and bounded by

$$\frac{v-7}{v-1} \leq R \leq \frac{v(v-7) + \log_2(v+1)}{v(v-1)} \quad (3.11)$$

for STS LDPC codes with $v \equiv 3$ or $7 \pmod{12}$ parity-check equations.

Fig. 3.3 shows the lengths and rates of the available codes from STS designs for all lengths up to 2000. For simplicity the points plotted correspond to full rank designs, the designs not considered will add codes at slightly higher rates. We can see that as longer STS designs are considered the STS LDPC codes quickly become high rate.

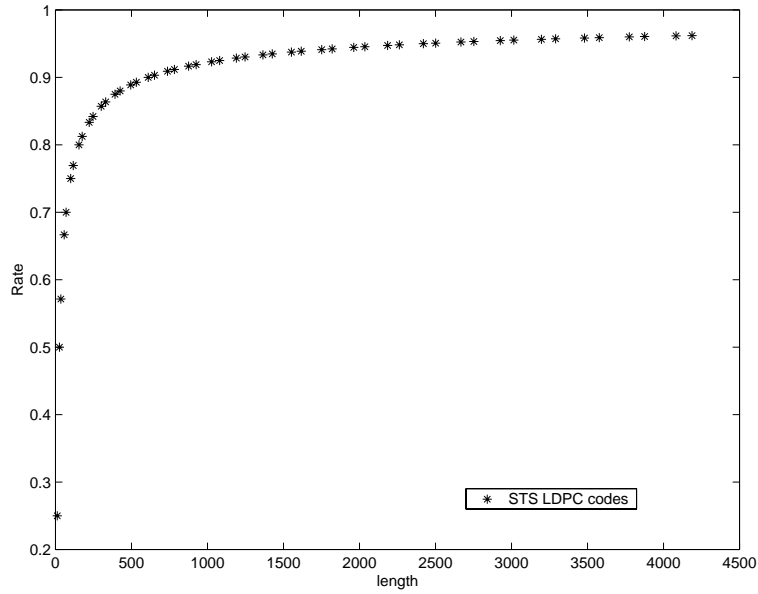


Figure 3.3: Rates and lengths of LDPC codes from Steiner triple systems

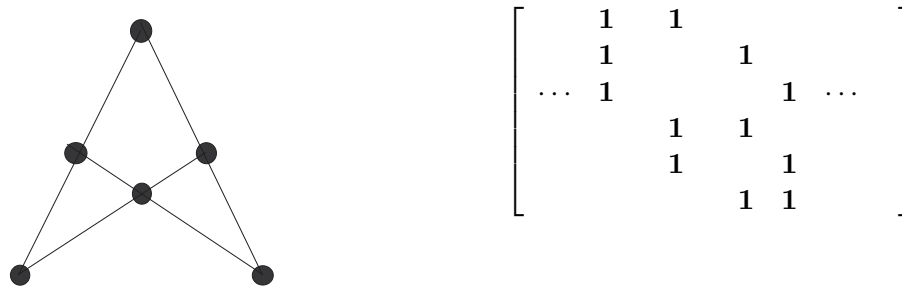


Figure 3.4: A Pasch configuration. The typical point-line representation and the resulting non zero entries in the incidence matrix. (Zero entries omitted for clarity)

3.3.2 Configurations and the minimum distance of STS LDPC codes

The column weight of the parity-check matrix of codes from STS designs is 3 and so a lower bound on minimum distance is 4, from (3.6). This bound can be increased to 6 by considering configurations in Steiner triple systems. In the following it will be shown that there is only one collection of blocks in a Steiner triple system which cause a weight 4 codeword; this configuration of blocks is called a *Pasch* configuration. Fig. 3.4 shows a point-line representation of a Pasch configuration and the portion of the incidence matrix corresponding to the Pasch configuration.

Lemma 3.3.1 *STS LDPC codes with no weight 4 codewords can be constructed from STS*

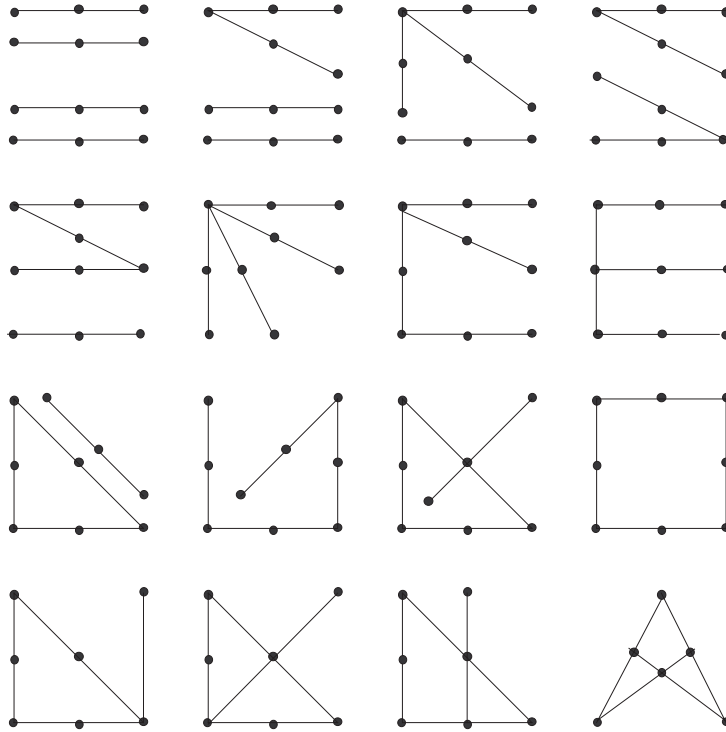


Figure 3.5: All the 4-line configurations possible in an STS design. The configurations all involve four blocks however the number of points in the configuration depends on how many points are common to more than one block. The configurations are ordered by the number of points they contain, ending with the Pasch configuration with the minimum possible six points, all of them included in more than one configuration block.

designs without Pasch configurations. These codes will have a minimum distance of at least 6.

Proof. To show that a Pasch configuration results in a weight four codeword is trivial, simply sum the columns of Pasch configuration in the incidence (parity-check) matrix in Fig 3.4. The positions of the four columns in the Pasch configuration are the codeword bit positions of the non-zero bits in the resulting weight four codeword. To show that by removing Pasch configurations the code minimum distance is at least 6 requires that no other configuration will give a weight-4 codeword and that a weight-5 codeword can not exist. Fig. 3.5 shows all the possible 4-line configurations in an STS design (see e.g. [50]); it is easy to see that only the Pasch configuration includes each point in an even number of lines. Lastly, five columns of an STS incidence matrix will contain 15 non-zero entries, and there is no way to divide 15 entries amongst a set of points so that each contains an even number. \square

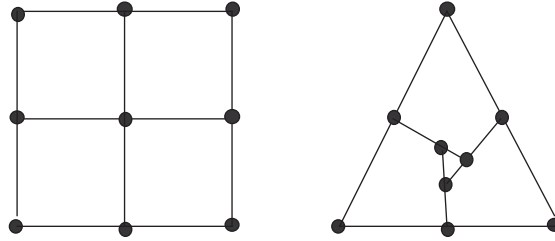


Figure 3.6: The two 6-line configurations in an STS design that will lead to weight 6 codewords in an STS LDPC code.

The Steiner triple systems free of Pasch configurations are termed *anti-Pasch* STS designs. These designs will generate STS LDPC codes with minimum distance at least 6. The minority of STS designs are anti-Pasch, for example, of the 11,084,874,829 non-isomorphic Steiner triple systems on 19 points exactly 2,591 are anti-Pasch [57]. Fortunately, it has recently been proven that anti-Pasch STS designs do exist for all v for which STS exist [46]. We propose that by using these STS designs the resulting STS LDPC codes will produce better decoding performances due to the absence of weight 4 codewords.

An examination of the five possible 6-line configurations having all points in at least two lines (see e.g. [51]) shows that only two of them involve each point in an even number of lines. Thus there are only two possible configurations of blocks in an STS design which will result in a weight 6 codeword in the STS LDPC code. Fig 3.6 shows these two configurations. Unfortunately there is as yet no known construction for STS designs which lack these configurations.

At the opposite end of the spectrum are the STS LDPC codes with the maximum number of weight four codewords, or equivalently the STS designs with the maximum number of Pasch configurations, which is given by [98]:

$$\frac{v(v-1)}{12} \left(\frac{v-1}{2} - 1 \right).$$

MacKay in [72] upper bounded the minimum distance of STS LDPC codes to 10 by counting the number of a particular arrangements of the blocks in an STS design which he called (5,1)-near codewords. The near codewords have the property that of the points contained in the five blocks all but one has degree 2. The two particular configurations which correspond to MacKay's (5,1)-near codewords are shown in Fig. 3.7, one of which is called the *mitre* configuration and the other we will call the *near-codeword* configuration. The point with odd degree is labelled point 1. If two of these configurations are aligned so that both share point 1 in common a 10-line configuration that leads to a weight 10

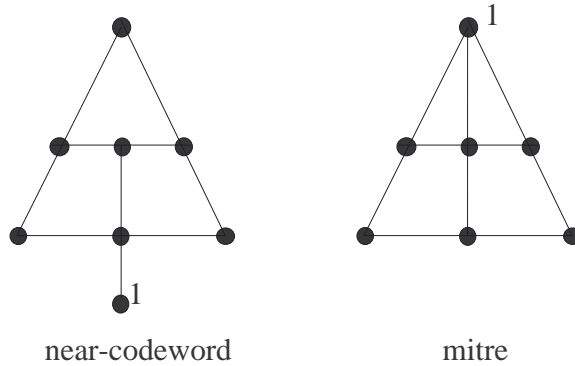


Figure 3.7: The two 5-line configurations in an STS design which are (5,1)-near codewords.

codeword is formed. To prove that codewords of Hamming weight at least 10 must occur in an STS LDPC code MacKay counted the number of these (5,1)-near codewords. He showed that there are more mitre and near-codeword configurations in an STS design than there are points and so there must be at least one point in the design where the odd degree points of two of the configurations are aligned, forming a weight 10 codeword in the STS LDPC code. We will give a slightly different proof of the number of near codewords in an STS design in the following section once basis configurations have been defined.

STS LDPC codes thus have minimum distances between 6 and 10, however, given the extremely high rates of the STS LDPC codes this minimum distance bound compares very favorably to the minimum distances achievable by random LDPC codes. In fact we will see in the following section that STS LDPC codes significantly outperform randomly constructed LDPC codes for all the lengths for which we have constructed LDPC codes, which is up to code lengths of 10,292.

3.3.3 The stopping set distribution of STS LDPC codes

Configurations play a further role in STS LDPC codes due to their relationship to stopping sets. An S -stopping set free STS code requires an STS design which contains no configurations of S or fewer blocks that have all point degrees of at least 2 (Lemma 3.2.2). Of the possible 4-line configurations in an STS design, Fig 3.5, the Pasch configuration is the only one with all point degrees of at least two. The 4-stopping set free designs are then simply the anti-Pasch STS designs considered earlier. Of the 5-line configurations only one, the mitre configuration, is a stopping set. Thus 5-stopping set free designs require that the design is both anti-Pasch and anti-mitre. Such an STS design is called 5-sparse in design terminology. This terminology stems from the work of Erdős who conjectured

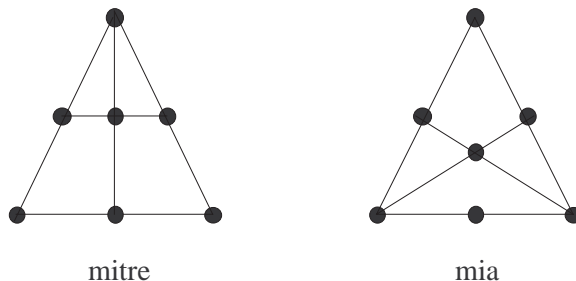


Figure 3.8: The two five line Erdős configurations, mitre and mia. The mitre configuration is the only possible configuration of points (checks) and lines (bits) that results in a stopping set of size 5 in an STS LDPC code.

in 1976 that for every r there is an integer $v_0(r)$ so that for every $v > v_0(r)$ there is an $\text{STS}(v)$ with the property that for $2 \leq j \leq r$ no $j + 2$ points carry j triples. Such an $\text{STS}(v)$ is r -sparse, and a configuration of j triples on just $j + 2$ points is an Erdős configuration.

An r -sparse design will not necessarily provide an LDPC code free of stopping sets on r bits. For example the 6-line configurations in Fig 3.6 will both lead to stopping sets while neither is an Erdős configuration. Similarly not all Erdős configurations lead to stopping sets, the 5-line mia configuration for example (Fig 3.8) will not result in a 5-bit stopping set in the STS LDPC code. However in the case of 5-sparse STS designs the two requirements do overlap. This is because the mia configuration can be obtained from the Pasch configuration by adding a line and so designs free of Pasch configurations are also free of mia configurations.

The Netto triple systems of orders congruent to 19 (mod 24) are anti-Pasch [28] and all Netto triple systems are anti-mitre [27] thus there is an infinite class of 5-sparse triple systems in the Netto triple systems with $v \equiv 19 \pmod{24}$. As well many 5-sparse STS designs with small v have been found and are presented in [27]. We will use some of the known 5-sparse STS designs to simulate the performance of STS LDPC codes without small stopping sets in the following section. There are five 6-line configurations that need to be removed from a 6-stopping set free design and unfortunately there is again at present no such STS design known.

Interestingly, the configurations that are important for stopping sets are also important for counting the frequencies of all the configurations in a design. Any constant n -configuration together with all m -line configurations, $m \leq n$, having all vertices of degree at least 2, form a generating set for the n -line configurations [51]. For the STS LDPC codes this means that in theory the process of finite length analysis can be applied directly to an individual STS LDPC code once the generating set is known. Unfortu-

nately, however, determining the generating set in practice is not easy, and has only been accomplished for small n (≤ 6).

The generating set for all 5-line configurations in STS designs is given in [29]. The number of each 5-line configuration is given as a function of the number of Pasch and mitre configurations in the design, n_p and n_m respectively. For example, the number of 5-line near-codeword configurations in an STS design on v points is:

$$n_M = \frac{v(v-1)(v-3)}{4} - 6n_p - 3n_m.$$

To lower bound the minimum distance of STS LDPC codes using this result, note that if the STS design has Pasch configurations the minimum distance is 6 so the interesting designs are those which are anti-Pasch, in which case the number of configurations which give (5,1)-near codewords is

$$n_M + 3n_m = \frac{v(v-1)(v-3)}{4}.$$

Thus for all v for which anti-Pasch STS designs exists there are more (5,1)-near codewords in the design than there are points (v) and so a subset of blocks must exist which produces a weight 10 codeword in the LDPC code from the STS design.

3.3.4 STS LDPC codes from low rank STS designs

It is conjectured that highly redundant parity-check matrices can lead to very good iterative decoding performances without the need for long codes [70, 118]. This encourages the consideration of STS designs with minimum rank incidence matrices.

From (3.9) the maximum number of linearly dependent rows in the incidence matrix of an STS design is $\log_2(v+1)$. The STS designs with this rank are derived from the points and lines of $\text{PG}(m, 2)$ and called *classical* STS designs [4]. The classical STS designs are the designs of points and lines of $\text{PG}(m, 2)$ and so exist for all $v \equiv 2^m - 1$, m an integer.

The LDPC codes from classical STS designs are possibly the highest rate LDPC codes with column weight 3 and no 4-cycles. Unfortunately however the classical STS designs also have a very high number of Pasch configurations and thus a large number of minimum weight (four) codewords are present in the STS LDPC codes from the classical STS designs. For example, an examination of the 80 Steiner triple systems on 15 points reveals that the classical STS designs are the designs with the largest number of Pasch configurations (105) of all the STS designs and this is true of all classical STS designs [98].

3.3.5 The decoding performance of STS LDPC codes

A few STS designs have been selected to simulate the performance of STS LDPC codes when decoded using the sum-product decoding algorithm on an additive white Gaussian noise (AWGN) channel, Figs. 3.9–3.15. In each simulation a maximum number of iterations has been set and the standard stopping criterion for LDPC codes, $zH' = 0$, is applied to terminate the decoding early if the hard decision of the bit probabilities, z , is a valid codeword.

The STS LDPC codes have been compared to regular randomly generated codes of the same rate and length constructed using the method of [71, 73] (see Section 2.1) using source code from [81]. For these high rate codes it is not possible to remove all 4-cycles from the LDPC codes using the random construction and in fact the process of moving ones within columns serves to produce poorer performing LDPC codes at these very high rates. Thus in this section the randomly constructed codes we simulate have regular parity-check matrices without repeated blocks and all contain 4-cycles. The decoding complexity of the STS and random LDPC codes is similar as the density of the parity-check matrix is the same for both and so the number of operations per iteration for each code will be equal.

The STS codes outperform the equivalent length and rate randomly constructed codes for all the code lengths we have simulated. There are two explanations for this, the first is that at these rates it is practically impossible to construct 4-cycle free LDPC codes randomly. Secondly, randomly constructed LDPC codes at these rates have poor expected minimum distance values, certainly much less than the upper bound on STS LDPC codes at these lengths. For the smallest STS code the union bound is also shown, in Fig. 3.9, demonstrating that the algebraic structure of the STS design does give a “better” code than the random code, in terms of maximum likelihood decoding performance, but at the same time this structure also gives a better sum-product decoding performance.

An examination of the expected minimum distance for ensembles of $(3,r)$ -regular random LDPC codes with the same length and rate as the STS LDPC codes demonstrates this. Table 3.1 gives Gallager’s expected minimum distance (see Section 2.1.2) for LDPC ensembles with parameters corresponding to the STS LDPC codes simulated in this thesis. We see that for the high rate LDPC codes only very small minimum distances (certainly less than 6) are expected. Thus for LDPC codes with the rates and lengths of the STS LDPC codes the linear increase of minimum distance with code length expected of randomly constructed LDPC codes is outweighed by the corresponding increase in code rate with length.

As can be seen in Figs. 3.16 and 3.17 the relative performance of the STS LDPC codes on the AWGN channel is reflected in their performance on the BEC channel. The

(γ, r)	Length (n)	Rate	$\delta_{\gamma,r}$	Expected minimum distance ($\delta_{\gamma,r}n$)
(3,6)	26	0.5000	0.0227	0.5902
(3,7)	35	0.5714	0.0130	0.455
(3,9)	57	0.6667	0.0054	0.3078
(3,10)	70	0.7000	0.0038	0.2660
(3,15)	155	0.8000	9.9718×10^{-4}	0.1546
(3,19)	247	0.8421	4.6807×10^{-4}	0.1156
(3,31)	651	0.8571	1.0085×10^{-4}	0.0657
(3,40)	1080	0.8800	4.5898×10^{-5}	0.0496
(3,87)	5075	0.8800	4.2705×10^{-6}	0.0217
(3,124)	10292	0.8750	$1.4617e \times 10^{-6}$	0.0150

Table 3.1: The expected minimum distance of random LDPC ensembles with the same rate, length, column weight and row weight as the STS LDPC codes simulated in this section.

STS LDPC codes show a large improvement in decoding performance over the randomly constructed codes and the 5-sparse STS LDPC codes perform slightly better than the codes from STS designs which contain mitre configurations.

Fig. 3.16 shows the decoding performance of the length 57 STS and random LDPC codes simulated on a BEC channel. Also shown is the average performance of the relevant code ensembles calculated using finite length analysis (see Section 2.1.2). Using finite length analysis allows us to compare the performance of individual codes to the expected average performance of all LDPC codes with the same parameters. The ensemble of all possible $m \times n$ parity-check matrices with column weight γ and row weight r is denoted by (m, n, γ, r) . An ensemble with parameter $S_{\min} \neq 1$ is the subset of the original ensemble which contains only those matrices with no stopping sets of size smaller than S_{\min} .

Both the randomly constructed and STS codes in Fig. 3.16 come from the $(19,57,3,9)$ ensemble. Also shown is the expected performance of the subset of this ensemble with minimum stopping set size 5 and minimum stopping set size 6. As would be expected the anti-Pasch STS code performs close the average of the ensemble with minimum stopping set size 5 and the 5-sparse STS code performs close to the average of the ensemble with minimum stopping set size 6. The randomly constructed code is not free of 4-cycles, but has no repeated blocks and so is free only of stopping sets of size 1 and 2, and its

performance is not as good as that of the STS codes.

Similarly, Fig. 3.17 shows the decoding performance of the length 247 STS and random LDPC codes simulated on a BEC channel along with the expected average performance of the $(39,247,3,19)$ code ensemble. Again, due to their better minimum stopping set size, the STS LDPC codes out-perform the randomly constructed LDPC codes. The results indicate that the excellent performance of the STS LDPC codes in general is due to the fact that the STS designs allow us to obtain 4-cycle free LDPC codes. Indeed, the closeness in the performance between the different STS codes and their respective ensemble averages suggests that, at least at these lengths, the algebraic structure of the codes is not a hindrance to code performance.

3.4 Discussion

In this chapter we have examined the use of Steiner 2-designs to construct algebraic LDPC codes. Firstly, the structure of Steiner 2-designs was used to derive lower bounds for minimum distance, minimum stopping set size and code rate and an expression for the exact number of minimum weight cycles in LDPC codes from designs.

We then looked in particular at Steiner triple systems to construct $(3,r)$ -regular LDPC codes. These STS LDPC codes give excellent decoding performances, significantly better than those achievable for randomly constructed codes of the same lengths and rates. The STS LDPC codes demonstrate that algebraic LDPC codes constructed using combinatorial designs can significantly outperform random LDPC codes for a wide range of code lengths. This is despite the results of [72] showing that the minimum distance of STS LDPC codes is at most 10. At the high rates of the STS LDPC codes a minimum distance of 10 is also improbable for randomly constructed codes. In fact it is difficult to randomly construct LDPC codes with minimum distances as good as the lower bound for the STS LDPC codes.

Further, there is additional performance benefit to be gained by considering the structure of the many non-isomorphic Steiner 2-designs available for a given choice of design parameters. In particular, the recognition of the link between design configurations and stopping sets in LDPC codes enabled us to design STS LDPC codes with improved minimum distance and minimum stopping set size. By choosing STS LDPC codes from anti-Pasch and 5-sparse STS designs, improved decoding performances are achieved.

Lastly, we considered designs with lower rank incidence matrices to obtain better LDPC codes. Unfortunately, however the STS designs with minimum rank incidence matrices also contain the maximum number of Pasch configurations and hence the maximum number of low weight (four) codewords in the resulting codes. Thus, for STS designs, the

performance benefits of linearly dependent rows in H are quickly eroded as the signal-to-noise ratio of the channel decreases. However, in the following chapter, we consider Steiner 2-designs with larger block size that produce incidence matrices with many more linearly dependent rows and no low weight codewords.

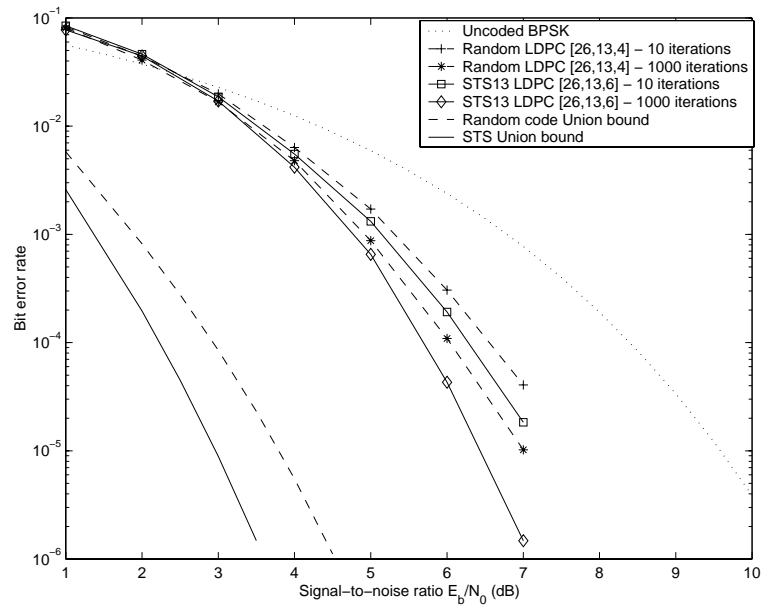


Figure 3.9: The decoding performance of the length-26 anti-Pasch STS LDPC code and of a random length-26 LDPC code on an AWGN channel using sum-product decoding are compared to the union bound for both codes.

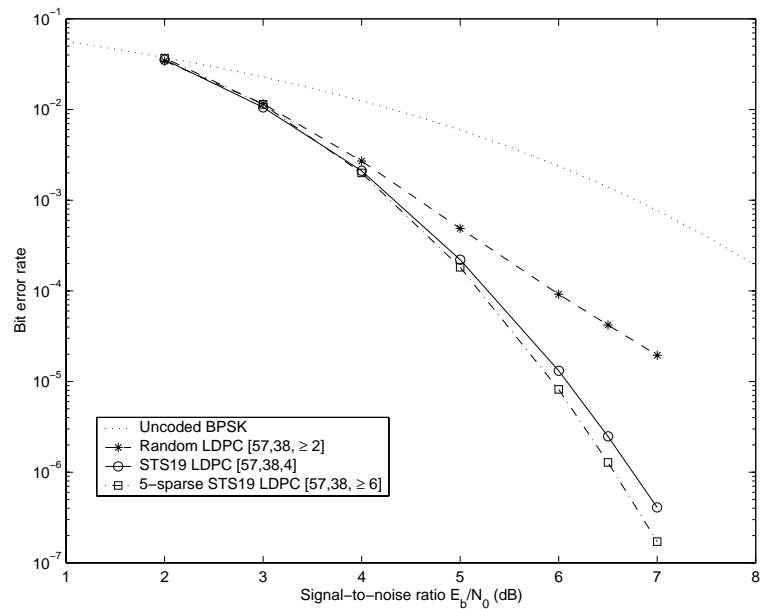


Figure 3.10: The decoding performance of length-57 STS LDPC codes on an AWGN channel using sum-product decoding with a maximum of 10 iterations.

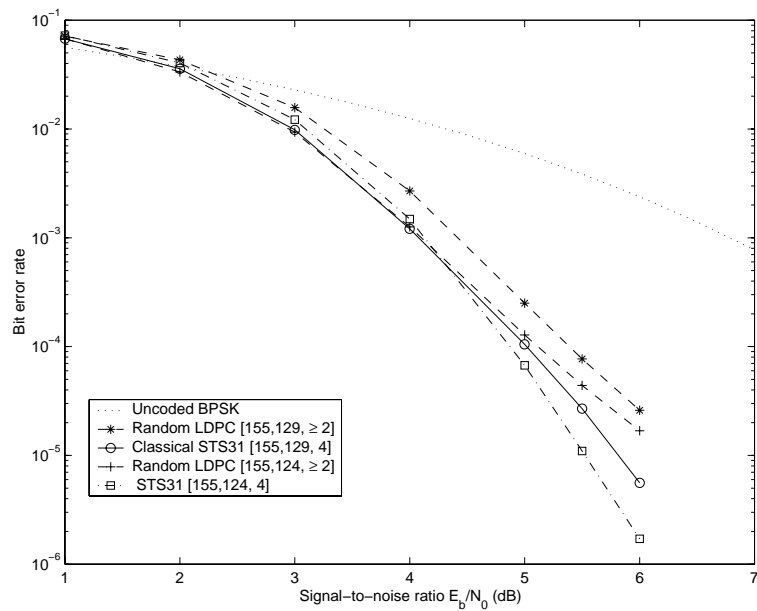


Figure 3.11: The decoding performance of length-155 STS LDPC codes on an AWGN channel using sum-product decoding with a maximum of 10 iterations.

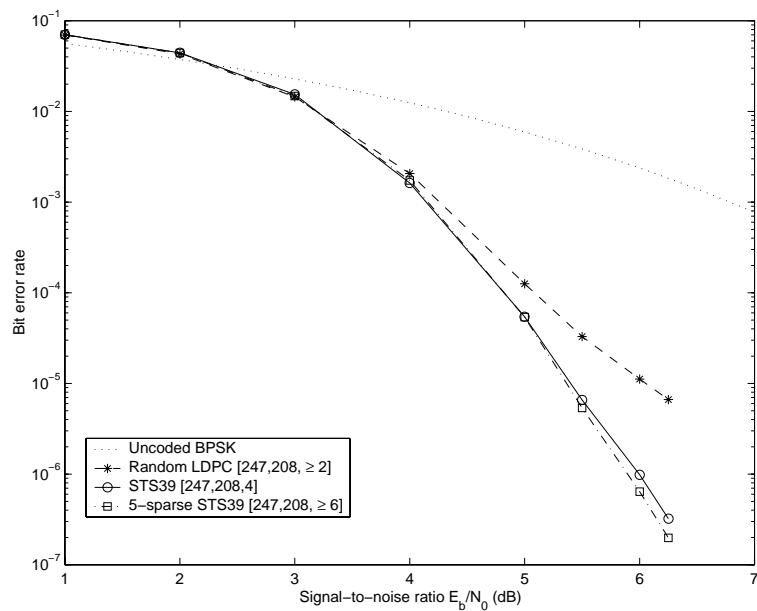


Figure 3.12: The decoding performance of length-247 STS LDPC codes on an AWGN channel using sum-product decoding with a maximum of 10 iterations.

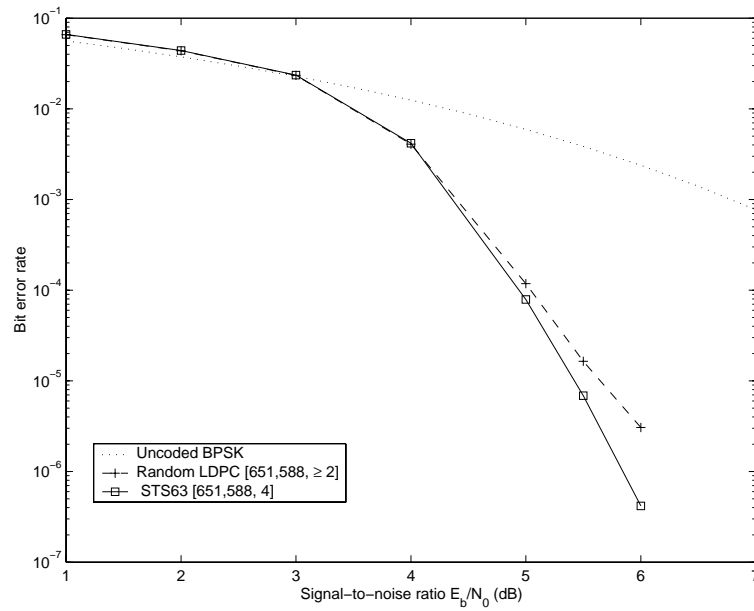


Figure 3.13: The decoding performance of length-651 STS LDPC codes on an AWGN channel using sum-product decoding with a maximum of 10 iterations.

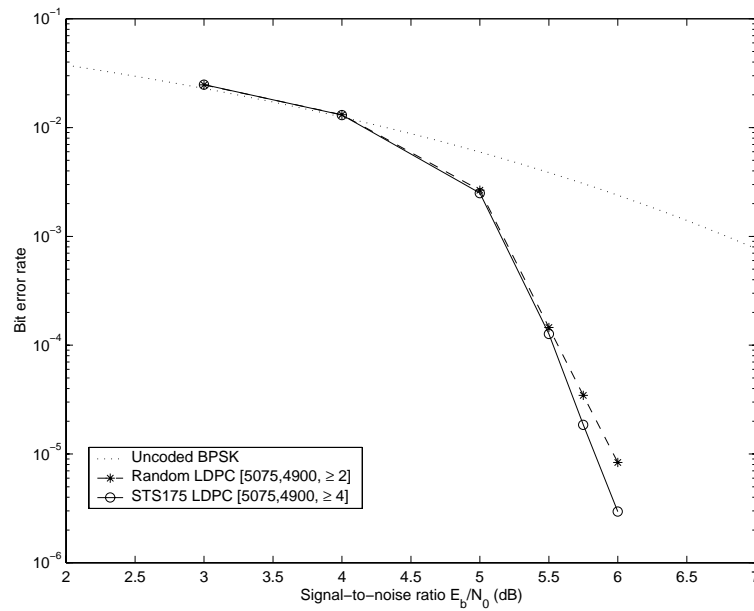


Figure 3.14: The decoding performance of length-5075 STS LDPC codes on an AWGN channel using sum-product decoding with a maximum of 10 iterations.

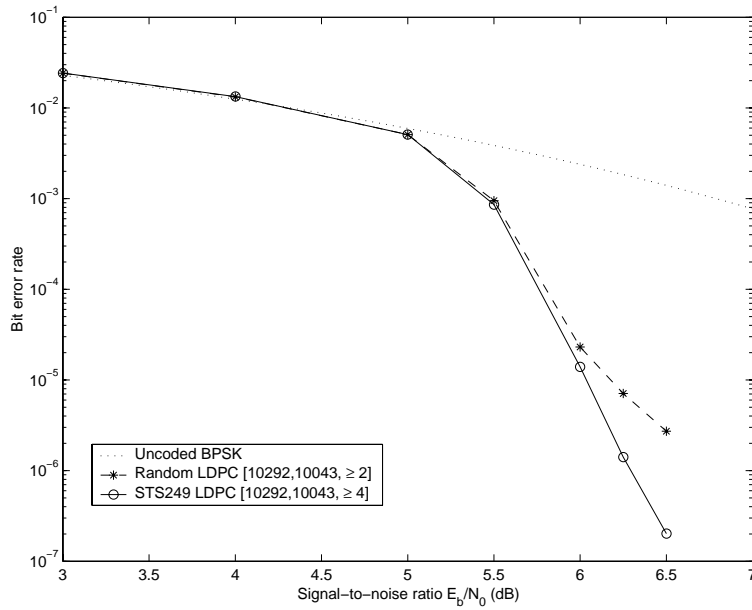


Figure 3.15: The decoding performance of length-10292 STS LDPC codes on an AWGN channel using sum-product decoding with a maximum of 10 iterations.

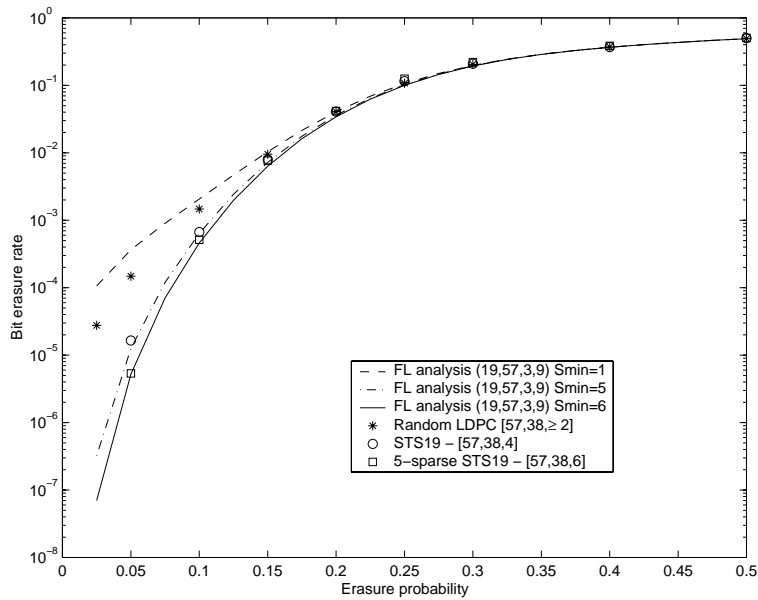


Figure 3.16: The erasure correction performance of $(3,9)$ -regular, length-57 LDPC codes on a binary erasure channel. Continuous curves show the average erasure correction performance of an ensemble while individual symbols give the simulated erasure correction performance of individual codes.

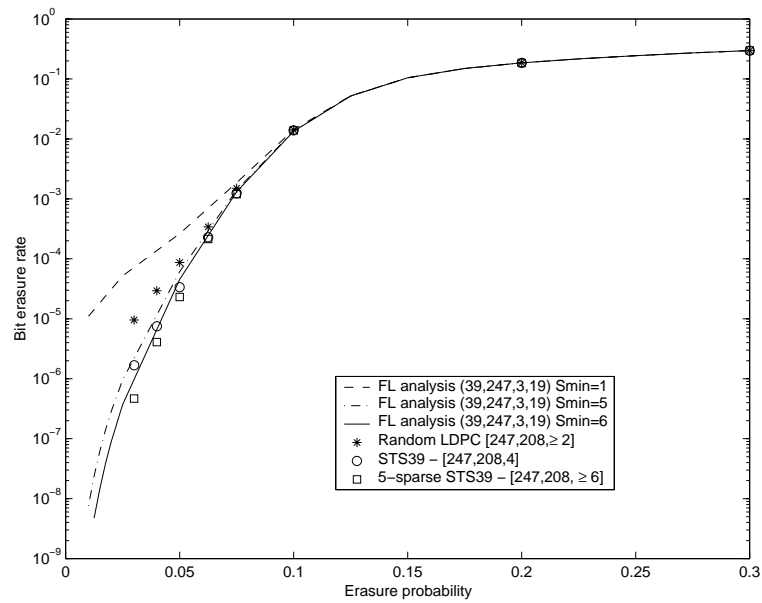


Figure 3.17: The erasure correction performance of (3,19)-regular length-247 LDPC codes on a binary erasure channel. Continuous curves show the average erasure correction performance of an ensemble while individual symbols give the simulated erasure correction performance of individual codes.

LDPC CODES FROM DESIGNS WITH LARGE BLOCK SIZE

In this chapter we extend upon the results of the previous chapter to explore the use of Steiner 2-designs with column weight greater than 3 to provide systematic constructions of regular LDPC codes. Increasing column weight improves code properties such as minimum distance, and stopping set distribution, but also increases the density of the code parity-check matrix. This yields excellent decoding performances in high signal-to-noise ratio channels but poorer decoding performances at low signal-to-noise ratios. Motivated by the success of the LDPC codes from finite geometries we focus on LDPC codes whose parity-check matrices include a large portion of linearly dependent rows. Two new families of LDPC codes with rank deficient parity-check matrices are presented, both with Tanner graphs free of 4-cycles and with deterministic code properties.

4.1 Introduction

The motivation for considering designs with larger block size to construct LDPC codes is that codes with larger minimum distances can be achieved. Codes from Steiner 2-designs with block size γ , that is $2-(v, \gamma, 1)$ designs, will have a minimum distance $d \geq \gamma + 1$ as well as a minimum stopping set size $S_{\min} \geq \gamma + 1$.

For the most part randomly constructed regular LDPC codes are considered best with column weight 3. MacKay found that decoding results became steadily worse as column weight was increased above 3 [71]. However, the Euclidean and projective geometry LDPC codes have large column weights yet produce excellent LDPC codes [60].

Most results suggest that the benefits of larger column weight depend on both the channel and the code rate. For example, in [36] it was shown that at very high rates array codes with column weight 4 performed well with sum-product decoding, and in [8] it was shown that at low signal-to-noise ratios on a partial response channel LDPC codes with

column weight 2 outperformed the LDPC codes with column weight 3.

In the following section we examine the role of column weight in determine code performance before looking at some specific classes of combinatorial designs in Sections 4.3–4.5. We find that some very good (γ, r) -regular LDPC codes can be produced for values of $\gamma > 3$ which outperform the existing $(3, r)$ -regular LDPC codes.

4.2 Column weight versus decoding performance

In this section we seek to determine the role of column weight in determining the decoding performance of regular LDPC codes to demonstrate the usefulness, or otherwise, of Steiner 2-designs with larger γ for designing LDPC codes.

An examination of Gallager’s expected minimum distance for LDPC ensembles (see Section 2.1.2) shows that the expected minimum distance of LDPC code ensembles increases with the column weight of H . Thus for maximum-likelihood decoding LDPC codes with larger column weight will give better decoding performances. For sum-product decoding however, larger column weights can be detrimental as the code density increases with γ .

To consider the effect of LDPC code column weight on sum-product decoding performance we use density evolution [88, 85] to determine the performance threshold of regular LDPC ensembles with varying column weight. Density evolution determines, in the case of very long codes, the column weights which give the best decoding thresholds (see Section 2.1.2). The threshold represents the value of the channel noise standard deviation below which it is possible to decode with zero error using the sum-product algorithm given enough iterations and a long enough code. There are however limitations to this analysis for our purpose. In particular, only infinite length codes are considered, while we are interested in the performance of short-to-medium length codes, and further the analysis assumes codes which are free of cycles and have full rank parity-check matrices; neither of which is necessarily true of algebraic LDPC codes. These considerations aside we present the trends observed using density evolution for an understanding of the performance of LDPC codes in the limit of infinite code length.

For a range of rates the threshold performance of regular LDPC codes on an AWGN channel, calculated using the *ldpcOpt* program at [112], is shown in Fig. 4.1. For most code rates a column weight of 3 gives the lowest threshold value for regular LDPC codes with the exception of rates ≥ 0.95 for which the threshold for column weight 4 regular codes is slightly higher than the threshold for column weight 3 regular codes.

On a binary erasure channel the results are similar. For a range of rates the threshold

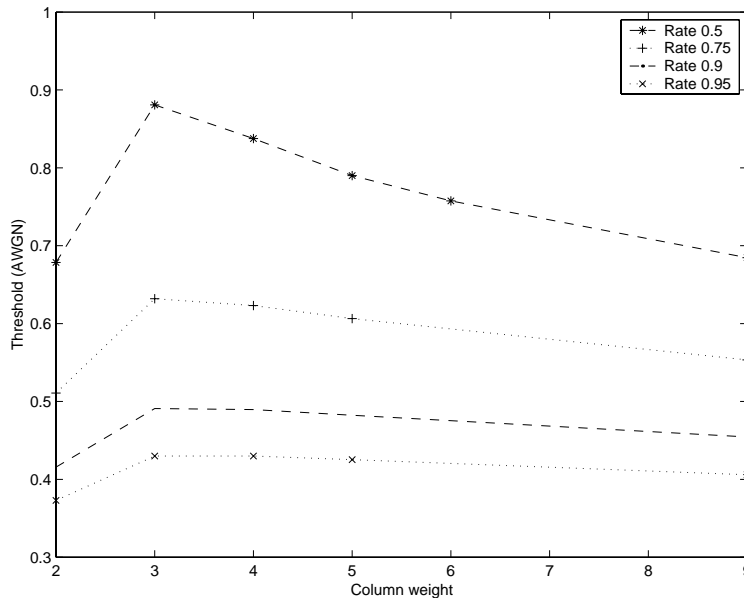


Figure 4.1: The threshold of regular LDPC codes on an AWGN channel calculated using density evolution at [112].

performance of regular LDPC codes on a binary erasure channel, calculated using the *ldp-cOpt* program at [112], is shown in Fig. 4.2. The threshold represents the channel erasure probability below which it is possible to decode with zero error using the sum-product algorithm given enough iterations and a long enough code. Again the best thresholds achieved by regular LDPC codes are for codes with column weight 3.

Thus analysis in the limit of code length indicates that for regular codes on a BEC or AWGN channel a column weight of 3 will give the best decoding performance in most situations with the exception being very high rate codes where columns of weight 4 can be beneficial.

In considering finite length codes however, it is often codes with large column weight which perform best in low noise channels, particularly where the absence of 4-cycles can guarantee these codes a much larger minimum distance than for the same length codes with column weight 3 (see e.g. [60]). While there is as yet no general analysis of the performance of finite length LDPC code ensembles on an AWGN channel, for the binary erasure channel at least, an indication of the effects of increasing column weight on finite length codes containing cycles can be determined by using finite length analysis [34, 87, 86] (see Section 2.1.2). Finite length analysis provides the expected performance on the binary erasure channel of the ensemble of all LDPC codes with a given length, rate, and column weight. One of the major benefits of this analysis is that the effects of changing a code parameter averaged over all codes in an ensemble is now possible to

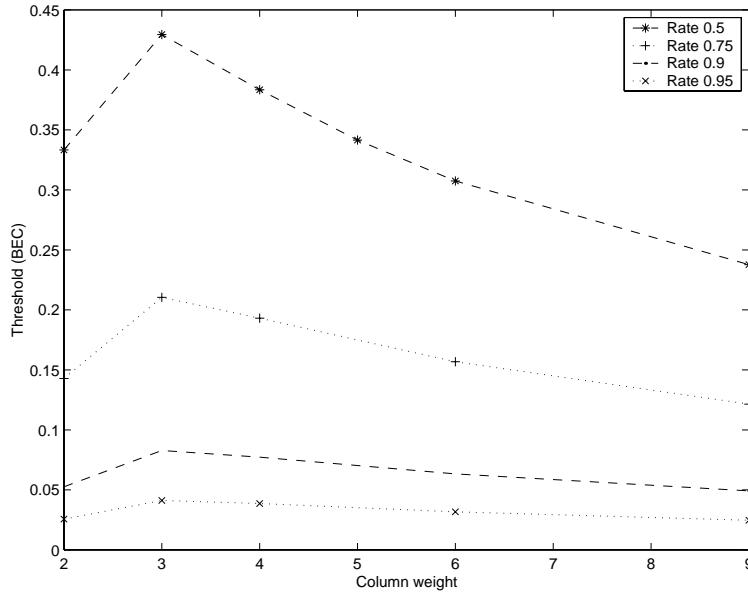


Figure 4.2: The threshold of regular LDPC codes on a binary erasure channel calculated using density evolution at [112].

observe for ensembles with finite length.

Fig 4.3 shows the expected performance of ensembles of length 100 rate- $\frac{1}{2}$ regular LDPC codes on a binary erasure channel calculated using finite length analysis. In this case the ensembles of column weight 3 codes do not always perform better than the ensembles of codes with larger column weights. In fact at smaller erasure probabilities the longer column weight codes may well give the best performance. This is even more apparent when the effect of removing small cycles is considered, as shown in Fig 4.4. The difference between the trends observed using density evolution and using finite length analysis is most likely due to the large effect of cycles, through stopping sets, on the performance of sum-product decoding on the binary erasure channel.

The negative effect of larger column weights in high noise channels can be explained by observing that if any two code bits in a given parity-check are in error (or erased) it is more difficult (impossible) to determine and correct the erroneous bit using that parity-check equation. In codes with large column weight each bit is connected to more parity checks, increasing the likelihood that two bits in any given parity checks are in error. This is less of a problem in high signal-to-noise ratio channels where there are only very few code bits affected by the noise. Indeed in these channels a large column weight can improve decoding convergence due to the larger number of parity-check equations on each code bit.

Overall, analysis of LDPC codes on both the AWGN and binary erasure channel

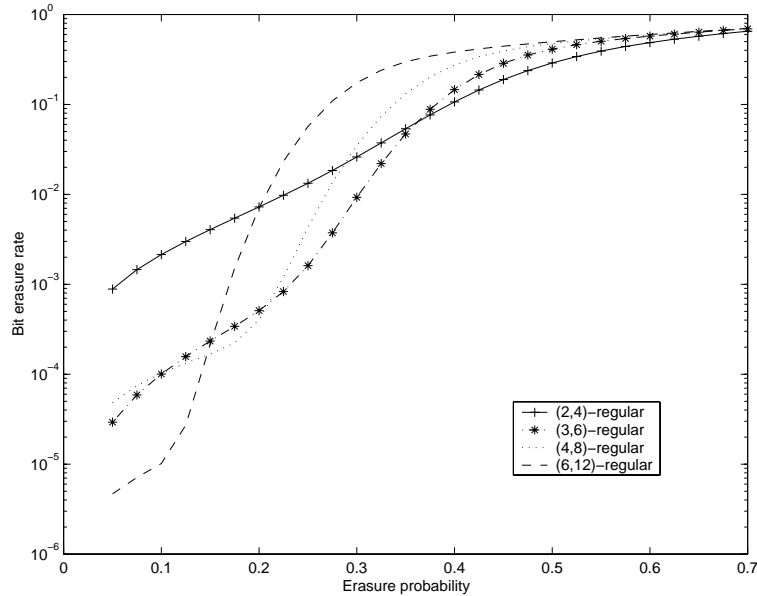


Figure 4.3: The expected performance of ensembles of length 100, rate half, regular LDPC codes on a BEC calculated using finite length analysis.

indicate that there may be benefit in considering high rate LDPC codes with column weight greater than 3. For the Steiner 2-designs in particular, the absence of 4-cycles guarantees a better minimum distance and minimum stopping set size for codes with larger column weight and we consider codes from Steiner 2- $(v, \gamma, 1)$ designs with $\gamma > 3$ in the remainder of this chapter.

4.3 LDPC codes with column weights > 3

We begin with Steiner 2-designs with structure similar to the STS designs but with column weights greater than 3. The encouraging performance of column weight 4 array codes [36] means that 2- $(v, 4, 1)$ designs in particular may provide us with good LDPC codes.

The necessary conditions for the existence of 2- $(v, 4, 1)$ designs are that (3.1)

$$(v - 1) \equiv 0 \pmod{3}$$

and that (3.2)

$$v(v - 1) \equiv 0 \pmod{12},$$

which are satisfied for any $v \equiv 1$ or $4 \pmod{12}$. This condition was shown to be sufficient by Hanani [48] in 1960 and so a 2- $(v, 4, 1)$ design exists for all $v \equiv 1$ or $4 \pmod{12}$.

The terminology Steiner quadruple systems (SQS) is already in use in combinatorics

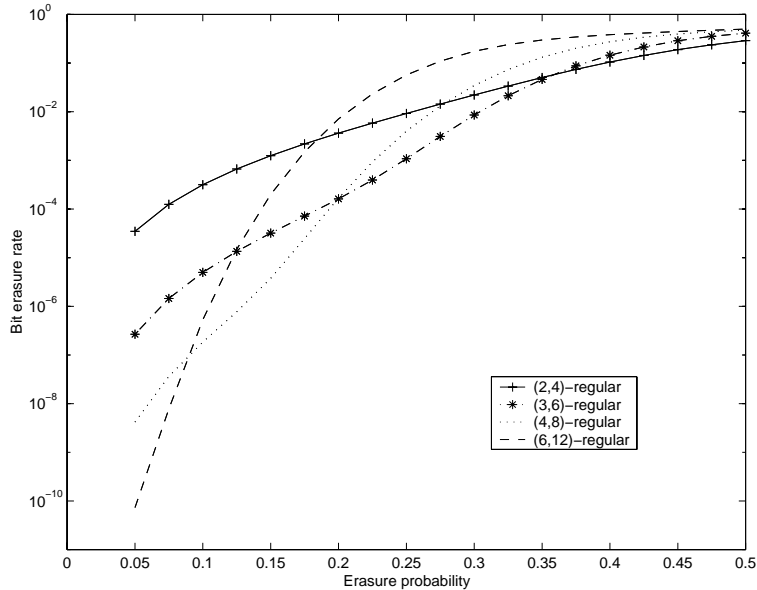


Figure 4.4: The expected performance of ensembles of length 100, rate half, regular LDPC codes with minimum stopping set size $\gamma + 1$ on a BEC calculated using finite length analysis.

to apply to designs derived from Steiner triple systems and is not applied to $2-(v, 4, 1)$ designs. We will thus use the notation SFS in the following to refer to Steiner 2-designs with block size four. From (2.15) there are

$$b = \frac{\lambda v(v-1)}{\gamma(\gamma-1)} = \frac{v(v-1)}{12} \quad (4.1)$$

blocks in an SFS design and from (2.16) each point of an SFS design is in

$$r = \frac{4b}{v} = \frac{4v(v-1)}{12v} = \frac{v-1}{3}, \quad (4.2)$$

blocks.

Like the STS designs, SFS designs achieve the upper bound on the number of columns in a binary matrix with column weight 4 and column intersection at most 1 and so most likely give the highest rate 4-cycle free LDPC codes with column weight 4. The SFS codes will have a minimum distance of at least 5 (Lemma 3.2.3) and minimum stopping set size of at least 5 also. Using the results of Section 3.2 the parameters of the LDPC codes constructed from SFS designs are:

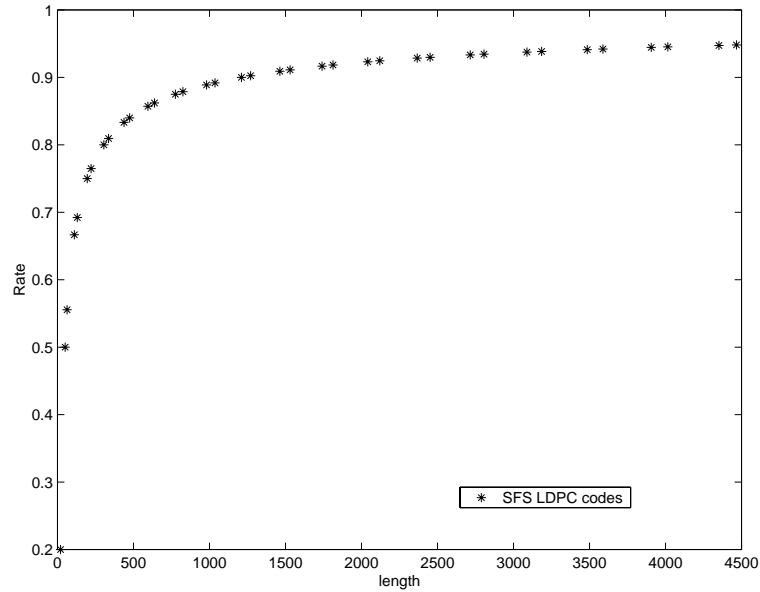


Figure 4.5: Rates and lengths of LDPC codes from Steiner 2-designs with $\gamma = 4$

Number of parity-check equations:	v
Length:	$v(v-1)/12$
Rate:	$R \approx (v-13)/(v-1)$
Minimum distance:	$d \geq 5$
Row weight of the parity-check matrix:	$(v-1)/3$
Column weight of the parity-check matrix:	4
Density of the parity-check matrix:	$4/v$

Fig. 4.5 shows the lengths and rates of the codes from SFS designs for all lengths up to 2000 assuming a full rank incidence matrix.

The performance of SFS LDPC codes on the AWGN channel, when decoded using the sum-product decoding algorithm [71], have been compared to that of randomly generated codes of the same rate and length with results shown in Figs. 4.9–4.11. In each simulation a maximum number of iterations has been set and the standard stopping criterion for LDPC codes, $zH' = 0$, is applied to terminate the decoding early if the hard decision of the bit probabilities, z , is a valid codeword.

For the random LDPC codes we have used the construction method from [71, 73] using source code from [81]. At the signal-to-noise ratios we consider the regular randomly constructed codes perform best with column weight 3 and so we have constructed random LDPC codes with this column weight as well. Further, in an attempt to get the best random LDPC codes we have generated random LDPC codes with as few 4-cycles as possible when the removal of 4-cycles produces a better performing LDPC code. However,

there is a trade off between removing code cycles and obtaining code regularity as the process of removing 4-cycles causes the row weight to be more variable in the random codes.

The SFS LDPC codes significantly outperform randomly constructed codes at the lengths simulated and so are promising LDPC codes for high-rate applications. However there is a small decrease in decoding performance at small signal-to-noise ratios demonstrated by the SFS codes compared to the random LDPC codes.

Ammar et al. in [2] presented simulation results for two LDPC codes from $2-(v,5,1)$ designs for lengths 3498 and 6700 and showed that these codes perform within 1.09 dB and 0.73 dB of the Shannon limit at a BER of 10^{-4} respectively. However the performance of the equivalent size random codes is not given in [2] and so it is not clear whether these codes outperform the random codes at these lengths.¹

We see though in simulation results for $2-(v,5,1)$ designs that the trend of improved performance in low noise channels but a decrease in performance in high noise channels, becomes more evident as the column weight is increased. Fig 4.12 for example shows the performance on a binary erasure channel of a length 82 rate- $\frac{1}{2}$ LDPC code from a $2-(41,5,1)$ design. The Steiner LDPC code is compared to randomly constructed LDPC codes with column weights 3 and 5. The random code with column weight 3 is free of 4-cycles and the random code with column weight 5 has as many 4-cycles removed as possible. The Steiner LDPC code performs as expected and outperforms the column weight 3 LDPC code at small erasure probabilities, with a performance degradation in channels with high erasure probabilities.

Already at column weights of 5, it is difficult to simulate these codes at small enough error probabilities to see an improvement in error correction performance for the longer code. Thus, for practical purposes, the observation that randomly constructed codes will perform best with column weight 3 seems reasonable. However, for implementations of LDPC codes in low noise channels, codes with larger column weights should be considered, particularly if high rate codes are employed.

Thus, whether or not LDPC codes from Steiner 2-designs with larger column weight will prove beneficial will depend on the channels under consideration. The higher the signal-to-noise ratio the more likely a larger column weight code will give the best performance. If LDPC codes are employed on binary erasure channels with low erasure probabilities, the Steiner 2-designs offer promise over randomly constructed codes due to both their regularity and guaranteed girth. The larger the column weight required, the more difficult it will be to randomly construct the LDPC codes without 4-cycles, which in

¹It has been pointed out by an examiner of this thesis that a significantly extended version of [2] has been accepted for publication in *IEEE Transactions on Information Theory*.

contrast is guaranteed by the codes from Steiner 2-designs regardless of column weight.

The codes from Euclidean and projective geometries, which have very high column weights, outperformed the randomly constructed codes at much lower error probabilities than we see with the $2-(v, 4, 1)$ and $2-(v, 5, 1)$ designs. The excellent performance of these codes on AWGN channels can not be due solely to their good minimum distances, but rather to the many linearly dependent rows in their parity-check matrices. With this in mind we consider in the following further Steiner 2-designs with both large column weight and many linearly dependent rows in N , of which there are two main families, the oval and unital designs.

4.4 LDPC codes from oval designs

In the search for algebraic LDPC codes two main observations have become apparent. Firstly, large column weight will provide a good minimum distance but will have an adverse effect on the decoding performance of the sum-product algorithm at low signal-to-noise ratios. Secondly, if there are a significant proportion of linearly dependent rows in the parity-check matrix of an LDPC code this may improve its performance to the extent that the benefit of the larger minimum distance can also be realized. With these observations in mind a promising source of LDPC codes presents itself in the form of oval designs, which are Steiner 2-designs with large column weight and a significant proportion of linearly dependent rows.

Oval designs are constructed from finite projective planes which were introduced in Section 2.2.1. An example, $\text{PG}(2, 4)$, is constructed in Appendix A.1.1. Briefly, a projective plane of order q , $\text{PG}(2, q)$, is a set of $q^2 + q + 1$ lines and $q^2 + q + 1$ points such that every line passes through exactly $q + 1$ points and every point is incident on exactly $q + 1$ lines with any pair of points in the plane incident together in exactly one line. An *oval* in a projective plane is a set of $q + 2$ points that meet each line of the plane in 0 or 2 points. The *oval design* is the incidence structure having for points the lines of the plane exterior to \mathcal{O} , and for blocks the points of the plane not on the oval \mathcal{O} , called the retained points. Incidence is given by the incidence of the projective plane; that is, in an oval design, a point is considered to belong to a block if the corresponding point and line in $\text{PG}(2, q)$ are incident. Thus oval designs are Steiner $2-(q(q - 1)/2, q/2, 1)$ designs with $q + 1$ blocks through each point and $q^2 - 1$ blocks (see e.g. [4, Chapter 8]). The family of oval designs constructed in this way have the parameters:

$$v = q(q - 1)/2, \quad b = q^2 - 1, \quad r = q + 1, \quad \gamma = q/2, \quad \lambda = 1.$$

As an example, the oval on 6 points from the projective plane of order 4 is constructed in Appendix A.1.2. It produces a $2-(6, 2, 1)$ oval design with incidence matrix shown in Fig. 4.6.

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

Figure 4.6: The incidence matrix of the 2-(6, 2, 1) oval design from the oval on the plane $\text{PG}(2, 2^2)$.

The well known extended Hamming codes can also be described from ovals in projective planes although they are defined in a different manner to oval designs. The points of the projective plane that form the oval become the columns of the parity-check matrix of the Hamming code. By definition no three points on an oval are collinear, and so no three columns of the extended Hamming code are linearly dependent, and thus a minimum distance of 4 is assured. For example the oval defined on $\text{PG}(2, 2^2)$ gives the extended Hamming [6,3,4] code:

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & \alpha & \beta & 0 & 1 \\ 0 & 1 & \beta & \alpha & 1 & 0 \end{bmatrix}.$$

The LDPC codes from oval designs are constructed differently. The incidence matrix of the design is taken as the parity-check matrix of the code. For an oval on a projective plane of order q , the code will be binary, have length $q^2 - 1$, with $q(q - 1)/2$ parity checks and is $(q/2, q + 1)$ -regular. Since there are $q + 1$ ones per row of H and $q^2 - 1$ rows, H has a density of

$$\frac{q + 1}{q^2 - 1} = \frac{1}{q - 1}. \quad (4.3)$$

Unlike for the STS and SFS LDPC codes the minimum distance bound from Massey for the oval codes (Lemma 3.2.3),

$$d \geq q/2 + 1,$$

increases with the length of the code as

$$d \geq \frac{1}{2}(\sqrt{n + 1} + 1).$$

However, this also means that the density of the parity-check matrix (r/n) decreases only with the square root of the code length as opposed to decreasing proportionally to the code length, which is the case if the column and row weights are fixed and the length of the code is increased.

Aside from increasing minimum distances, an important property of oval designs is the large number of linearly dependent rows in their incidence matrix. The 2-rank of the

m	(v, b, r, γ)	$\text{rank}_2(H)$	$[n, k, d]$
2	(6, 15, 5, 2)	5	[15, 10, 3]
3	(28, 63, 9, 4)	19	[63, 44, 5]
4	(120, 255, 17, 8)	65	[255, 190, 9–10]
5	(496, 1023, 33, 16)	211	[1023, 812, 17–36]
6	(2016, 4095, 65, 32)	665	[4095, 3430, 33–78]

Table 4.1: Parameters of the first five codes from oval designs

incidence matrix of a regular oval design is [20]

$$\text{rank}_2(N) = 3^m - 2^m, \quad \text{where } q = 2^m \quad (4.4)$$

which yields the number of linearly dependent rows in H :

$$2^{2m-1} - 2^{m-1} - 3^m + 2^m, \quad (4.5)$$

and hence the number of code message bits

$$k = (2^m)^2 + 2^m - 3^m - 1. \quad (4.6)$$

Using the results of Section 3.2 and the above, the LDPC codes derived from oval designs, oval LDPC codes, have the following parameters:

Number of parity checks:	$2^{m-1}(2^m - 1)$
Length:	$2^{2m} - 1$
Rate:	$(2^{2m} - 1 + 2^m - 3^m)/(2^{2m} - 1)$
Minimum distance:	$d \geq 2^{m-1} + 1$
Row weight of the parity-check matrix:	$2^m + 1$
Column weight of the parity-check matrix:	2^{m-1}
Density:	$1/(2^m - 1)$

Table 4.1 shows the parameters of the oval designs and the corresponding oval LDPC codes for $m = 2, \dots, 6$. For the two shortest codes, the minimum distance has been obtained by exhaustive computation. For the longer codes, corresponding to $m \geq 4$ in Table 4.1, the range of minimum distances has been obtained using Magma [14] for an upper bound, and Massey's lower bound applied to oval LDPC codes.

4.4.1 Simulation results for oval LDPC codes

The performance of oval LDPC codes on the AWGN channel, when decoded using the sum-product decoding algorithm [71, 73], have been compared to that of randomly

generated codes of the same rate and length and existing algebraic LDPC codes from Steiner triple systems and Euclidean geometry (EG) codes [59, 60]. The results are shown in Figs. 4.13–4.19. In each simulation a maximum number of iterations has been set and the standard stopping criterion for LDPC codes, $zH' = 0$, is applied to terminate the decoding early if the hard decision of the bit probabilities, z , is a valid codeword.

For the random LDPC codes we have used the construction method from [71, 73] using source code from [81]. At the signal-to-noise ratios we consider the regular randomly constructed LDPC codes perform best with column weight 3 and so we have constructed random LDPC codes with this column weight rather than constructing random codes with the column weight of the oval codes. Further, in an attempt to get the best random LDPC codes we have generated random LDPC codes with as few 4-cycles as possible in the cases where the removal of 4-cycles produces a better performing LDPC code. However, there is a tradeoff between removing code cycles and obtaining code regularity as the process of removing 4-cycles causes the row weight to be more variable.

The more parity checks per bit there are in H , the more calculations that are required to perform an iteration of the sum-product decoding algorithm. The effect this has on decoding complexity at various signal-to-noise ratios is also shown in this section. The average number of multiplication floating point operations (flops) required to decode a codeword can be calculated by estimating the number of multiplication flops for one iteration of the sum-product algorithm as $6n\gamma$ [71] and counting the number of iterations required to decode each codeword. As the decoding complexity is used solely to give a comparison between LDPC codes with different column weights, we do not employ the more precise estimates of decoding complexity, such as in [41], which include operations other than multiplication.

Fig. 4.13 shows the performance of the oval-[63, 44, 5] code compared to that of a randomly generated length 63 rate-0.7 LDPC code on an AWGN channel. The oval code is (4, 9)-regular and the randomly generated LDPC code has column weights of 3 and row weights of 9 and 10. The same rate STS LDPC code from the STS(21) is also shown. The oval code out-performs the STS code, even though it is slightly shorter, which is attributed to its many extra parity-check equations caused by the low rank of the oval incidence matrix. Fig. 4.14 shows the decoding complexity of the oval and STS codes. Despite its larger column weight the oval LDPC code has a lower decoding complexity at some signal-to-noise ratios due to its faster convergence.

Fig. 4.15 shows the performance of the oval [255, 190, ≥ 9] code compared to that of a randomly generated LDPC code in the AWGN channel. The oval code is an (8, 17)-regular code, and the randomly generated LDPC code has column weight 3 and row weights between 7 and 18. The variable row weights in the randomly constructed LDPC codes is the by-product of removing all 4-cycles from the code. The oval code has a

higher column weight than the random code and so has a higher decoding complexity per iteration. However, in some cases the faster convergence of the oval code gives the two codes a similar decoding complexity overall, Fig. 4.16. Also shown is the same length, but lower rate, (16,16)-regular Euclidean geometry code compared to a randomly constructed code of the same rate and length. The oval code provides the same gain in performance as the EG code but at a higher rate. It also does this with a much lower computational complexity gain over the random LDPC codes.

Fig. 4.17 shows the decoding performance over an AWGN channel of the oval [1023, 812, 17] code compared with a randomly generated LDPC code. Also shown is the [1023, 781, 33] Euclidean geometry code [59, 60]. The oval code is (16, 33)-regular, the EG code is (32, 32)-regular, and the randomly generated LDPC codes have a column weight of 3 and row weights between 11 and 19. Again, the number of parity checks in H has a significant effect on decoding complexity as shown in Fig. 4.18. Like the EG code the oval code outperforms the equivalent length and rate randomly constructed code only at the higher signal-to noise ratios. The results for the EG code are the same as those presented in [60]. The effects of the longer column weights at low signal-to-noise ratios can be clearly seen with the random codes outperforming the EG and oval codes in high noise channels.

Finally, Fig. 4.19 shows the BER performance over an AWGN channel of the oval [4095, 3430, ≥ 33] code compared with a randomly generated LDPC code. The oval code is (32, 65)-regular and the randomly generated LDPC code has a column weight of 3 and row weights between 16 and 22. Also shown is the equivalent length EG code compared with a randomly generated code. The EG code is (64,64)-regular while the random code has a column weight of 3 and row weights between 14 and 21. At these column weights, 32 and 64, the density of the parity-check matrix, $\frac{32}{2016}$ and $\frac{64}{4095}$ respectively, are very high and so, at the signal-to-noise ratios considered, the EG and oval codes are outperformed by equivalent length and rate randomly constructed column weight 3 codes.

The parameters of the codes from oval designs seem particularly promising for LDPC codes. They have sparse matrices, excellent minimum distances and many linearly dependent rows in their parity-check matrices. However, at longer lengths the performance of the oval codes, as for the Euclidean geometry codes, grows significantly worse relative to the random codes due to the increasing density of their parity-check matrix. The longer oval and EG codes have significantly better minimum distances than that possible with randomly constructed codes and so their disappointing performance is direct evidence of the limitations of comparing LDPC codes by minimum distance alone. The increase in column weight which provides this minimum distance gain also serves to greatly increase the density of H causing the poorer decoding performances at all but very high signal-to-noise ratios.

However, the oval codes do show promise at short lengths with codes from oval designs significantly outperforming the randomly constructed codes on an AWGN channel with sum-product decoding. The linearly dependent rows in the incidence matrices of the oval designs improves the performance of the shorter oval codes compared to equivalent length and rate codes with the same column weight.

Overall, oval designs provide very promising short LDPC codes which perform well on an AWGN channel at all signal-to-noise ratios while the longer oval codes are only useful in channels with very high signal-to-noise ratios. The oval codes may offer an advantage over the existing Euclidean geometry codes due to their higher rate and their lower column weights.

4.5 LDPC codes from unital designs

The final family of Steiner 2-designs we consider have similar properties to the oval designs but with different parameters and so produce new algebraic LDPC codes for more lengths and rates. Unital designs are similar to oval designs in that they are derived from projective planes, have incidence matrices with linearly dependent rows, and column weights that increase with the matrix length. The parameters of the family of unital designs were presented in [116, Table II] where it was assumed that the code rates would be $(b-v)/b$ if codes were constructed from the incidence matrices of unital designs on $v = m^3 + 1$ points. However, for odd m the corresponding incidence matrices are significantly rank deficient for small m and conjectured to be so for larger m [58]. Consequently, LDPC codes from unital designs with odd m benefit from both higher rates and the decoding advantages attributed to a large number of linearly dependent rows in the parity-check matrix.

The construction of unital designs from projective planes is described in Appendix A.1.4. Briefly, a *unitary polarity* in a projective plane of even order $q = m^2$ is a set of points of the plane with cardinality $m^3 + 1$ and the property that every line of the plane meets the set in 1 or $m + 1$ points. A *unital design* has as points the point set of a unitary polarity and for blocks those lines in the projective plane that meet the point set of the unitary polarity in $m + 1$ points [4]. The points and blocks of the design retain the incidence of the points and lines of the plane. By definition the design consists of $m^3 + 1$ points with each block containing $m + 1$ points. Since the lines of the plane that include more than one point in the unitary polarity are retained, every line through a pair of the points in the polarity is a block of the design. Thus every pair of points in the design must occur together in a block and the unital design is a Steiner 2-design with parameters $2-(m^3 + 1, m + 1, 1)$. The number of bits in each check is (2.16)

$$r = m^2,$$

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

Figure 4.7: The incidence matrix of the 2-(9, 3, 1) unital design from the unitary polarity on the plane $\text{GF}(2^2)$.

and the number of blocks is (2.15):

$$b = m^2(m^3 + 1)/(m + 1).$$

The unitary polarity defined on the projective plane of order 4 is constructed in Appendix A.1.4 and produces a 2-(9, 3, 1) unital design with incidence matrix shown in Fig. 4.7.

The unitals described above are *Hermitian unitals*, which are the set of points and lines of a unitary polarity on a desarguesian plane of order m^2 [4]. These unitals exist for all m a prime power. However other unital 2-($m^3 + 1, m + 1, 1$) designs exist, for m not necessarily a prime power, which are not defined on a desarguesian plane [4]. The codes presented in this thesis are all derived from Hermitian unitals constructed using the method of [58].

The density of the parity-check matrix of a unital code is

$$\frac{m + 1}{m^3 + 1},$$

which decreases as the code length increases at a rate proportional to \sqrt{n} . Meanwhile, the minimum distance bound for the unital codes from Massey

$$d \geq m + 2$$

increases with the length of the code in the order of $\sqrt[4]{n}$.

While an explicit formula for the 2-ranks of the incidence matrices of unital designs is not yet available, it is known that the 2-rank can only be less than m^3 if $2|m + 1$ [4, Theorem 8.3.1]. For the unitals presented in Table 4.2 the 2-ranks of the incidence matrices have been calculated in [58]. The conjecture that the 2-rank of those unitals where $2|m + 1$ is

$$m(m^2 - m + 1)$$

m	$(v, b, \gamma, r, \lambda)$	$\text{rank}_2(H)$	$[n, k, d]$
2	(9, 12, 3, 4, 1)	9	[12, 3, 6]
3	(28, 63, 4, 9, 1)	21	[63, 42, 6]
4	(65, 208, 5, 16, 1)	65	[208, 143, 8–10]
5	(126, 525, 6, 25, 1)	105	[525, 420, 7–9]
7	(344, 2107, 8, 49, 1)	301	[2107, 1806, 9–12]
8	(513, 3648, 9, 64, 1)	513	[3648, 3135, ≥ 10]
9	(730, 5913, 10, 81, 1)	657	[5913, 5256, ≥ 11]

Table 4.2: Parameters of the first seven codes from Hermitian unital designs

is proven for the cases with $m \leq 9$.

In summary, the LDPC codes derived from unital designs, unital LDPC codes, have the following parameters:

Number of parity-check equations:	$m^3 + 1$
Length:	$m^2(m^3 + 1)/(m + 1)$
Rate:	$R \geq \frac{m^2 - m - 1}{m^2}$
Minimum distance:	$d \geq m + 2$
Row weight of the parity-check matrix:	m^2
Column weight of the parity-check matrix:	$m + 1$
Density:	$(m + 1)/(m^3 + 1)$

Table 4.2 shows the parameters of the unital designs obtained for values of $m = 2, \dots, 9$, m a prime power. Also shown are the parameters of the associated linear block codes. The range of possible values for the minimum distance of the unital codes has been shortened via exhaustive search using Magma [14]. The small unital codes have minimum distances better than the lower bound. Using unitals produces more algebraic LDPC codes, which have lower densities and higher rates than the oval, EG and PG codes.

4.5.1 Simulation results for unital LDPC codes

The performance of unital LDPC codes on the AWGN channel, when decoded using the sum-product decoding algorithm [71, 73], is compared to that of randomly generated codes of the same rate and length. The results are shown in Figs. 4.21–4.26. The simulation setup, and random code construction is the same as detailed in Section 4.4.1.

Fig. 4.21 shows the performance of the unital [63, 42, 6] code compared to a randomly generated LDPC code on the AWGN channel. The unital code is (4, 9)-regular code and

the randomly generated LDPC code is (3,9)-regular. Also shown is the performance of the equivalent length but slightly higher rate oval LDPC code which is also (4,9)-regular. Probably due to its better minimum distance, the unital code outperforms the oval code at very high signal-to-noise ratios. Fig. 4.23 shows the performance over an AWGN channel of the unital $[525, 420, \geq 7]$ code compared with a randomly generated LDPC code. The unital code is (6, 25)-regular and the randomly generated LDPC code has column weights of 3 and row weights between 9 and 22. The unital code significantly outperforms the random code at medium to high signal-to-noise ratios with some increase in decoding complexity shown in Fig. 4.24, due to the larger column weight and extra parity-checks in the unital code.

Finally, Fig. 4.25 shows the BER performance over an AWGN channel of the unital $[2107, 1806, \geq 9]$ code compared with a randomly generated LDPC code. The unital code is (8, 49)-regular and the randomly generated LDPC code has column weights of 3 and row weights between 16 and 26. As for the longer oval and EG codes the long unital code outperforms the randomly constructed code only in low noise channels and also has a increased decoding complexity as seen in Fig. 4.26.

As the size of the unital increases so too does the column weight of the unital code. This provides a good minimum distance for the code, and hence excellent decoding performance at high signal-to-noise ratios, but also degrades the decoding performance at low signal-to-noise ratios. Overall, unital codes are promising LDPC codes for the short codes at all signal-to-noise ratios and for the longer codes if used in high signal-to-noise ratio channels.

4.6 Linearly dependent parity-checks

An interesting result following from the research into algebraic LDPC codes has been the observation that the existence of linearly dependent rows in the parity-check matrix of an LDPC code are beneficial for the sum-product decoding process [70, 118, 72]. On the one hand it would seem obvious that extra parity-check equations would provide extra extrinsic information to the code bits which would assist in the decoding process. However, because the extra checks are linearly dependent, this extra extrinsic information is a function of the information already available.

The question therefore is: to what extent are linearly dependent rows in the parity-check matrix of an LDPC code of benefit to the decoding process? To consider this question we will look at the performance on the binary erasure channel of some of the algebraic LDPC codes with linearly dependent parity-check equations. The binary erasure channel is useful because the performance of individual codes can be compared to the ensemble average, calculated using finite length analysis.

Since finite length analysis does not depend on the manner of construction of the codes in a given ensemble, only that they have a given number of parity-check equations, length, column weight, row weight, and minimum stopping set size, we expect that an algebraically constructed code will in general perform as close to its ensemble average as a randomly constructed code. We have observed previously, in Section 3.3.5, that the algebraic structure of an LDPC code, where that code has a full rank parity-check matrix, does indeed not greatly alter its performance from that of the ensemble average. This result is further observed in simulations in this and the following chapters.

However, the results of finite length analysis are an average over all codes with a given size and density H , the vast majority of which will be full rank. Thus we expect that any significant differences in performance between codes with rank deficient parity-check matrices and the ensemble average, calculated using finite length analysis, will be due to the linear dependence of the parity-check equations in that code rather than due to the fact that the given code was algebraically constructed.

The ensemble of all possible $m \times n$ parity-check matrices with column weight γ and row weight r is denoted by (m, n, γ, r) . An ensemble with parameter $S_{\min} \neq 1$ is the subset of the original ensemble which contains only those matrices with no stopping sets of size smaller than S_{\min} . For example, the $[63, 44, 5]$ code from the 2 -(28, 4, 1) oval design belongs to the ensemble of all codes with 28 parity-checks, length equal to 63, column weights 4, row weights 6, and minimum stopping set size $S_{\min} = 5$.

To begin we start with the oval and STS codes of Fig 4.13. In Fig 4.27 the performance of these codes, simulated on the BEC, is shown and we see that their relative performance is about the same as on the AWGN channel. To see what effect the linear dependence of the parity-checks in the oval code has on code performance we compare its performance to the ensemble average for codes with the same length, column weight, row weight and number of parity-checks, Fig. 4.28. Codes from this ensemble will generally be of lower rate than the oval code however the object is not to compare the actual codes, but instead to examine the effects when parity-checks are linearly dependent.

The operation of the sum-product decoding algorithm on the BEC is independent of the code rate and so the difference in performance between the oval code and ensemble average is due to the linearly dependent parity-check rows in H . We see that while a full rank binary 28×63 , (4,9)-regular matrix performs close to the ensemble average the oval code performs significantly worse than the average for this ensemble.

To rule out the randomness of the construction as the source of the differing performance we compare the algebraic $[70, 49, 4]$ code from the 2 -(21, 3, 1) STS design to its ensemble average in Fig. 4.29. The STS LDPC code is a very structured code but has no linearly dependent rows in its parity-check matrix and, as expected, performs close to

the full rank random LDPC code from the ensemble, and close to the ensemble average.

Thus, as expected, linearly dependent parity-check equations do not offer the same value extrinsic information as the same number of linearly independent parity-check equations. However the oval code does still out-perform a full rank code of the same length and rate. Thus it would appear that while not as good as linearly independent parity-check equations extra linearly dependent parity-check equations can offer some performance gain without reducing the code rate.

In Fig. 4.30 similar trends to that of the oval codes can be seen in the performance of the length 63 EG LDPC code compared to the performance of a randomly constructed 63×63 , (8,8)-regular, LDPC code and compared to the ensemble average. The linearly dependent checks are not quite as good as linearly independent checks but there are enough of them that the EG code outperforms the same rate randomly constructed code which has fewer checks, all of which are linearly independent.

In Fig 4.31 we look at the performance of the [255,190,9] oval LDPC code compared to a random code from the same ensemble. Again codes from this ensemble will generally be lower rate than the oval code however the object is not to compare the actual codes, but instead to examine the impact of linearly dependent parity-checks. Similarly, Figs 4.32 and 4.33 show the performance of the length 1023 oval and EG codes. Again, for these two oval codes the large portion of linearly dependent rows in their parity-check matrix results in a decoding performance worse than the average for the ensemble of codes with the same number of parity-checks where most of them are linearly independent.

4.7 Discussion

In this chapter we have presented two new infinite classes of error correction codes derived from Steiner 2-designs: oval codes and unital codes. The significant outcome of this approach is the construction of LDPC codes which are both regular and 4-cycle free, particularly for those code parameters where these properties are difficult to achieve using random constructions. This yields codes which both have deterministic constructions and which significantly outperform the traditional randomly constructed LDPC codes of the same rate and length. We found that some very good (γ, r) -regular LDPC codes can be produced for values of $\gamma > 3$ which outperform the existing $(3, r)$ -regular LDPC codes.

There are two main groups of codes from Steiner 2-designs, those with full (or close to full) rank parity-check matrices, and those with a significant proportion of linearly dependent rows in their parity-check matrices. For the codes from designs with (approximately) full rank incidence matrices their performance with sum-product decoding is similar to the randomly constructed codes, which are also full rank. Many of the algebraic codes

show improvements over random LDPC codes due to the guaranteed removal of 4-cycles in Steiner 2-designs. For small to moderate block lengths these LDPC codes from Steiner 2-designs represent a reasonable alternative to randomly constructed LDPC codes as they offer a deterministic construction, regularity and guaranteed girth.

The density of the LDPC parity-check matrix dominates the performance of the LDPC codes at low signal-to-noise ratios, with other properties such as minimum distance and girth playing a greater role as the signal-to-noise ratio is increased. LDPC codes with larger column weight perform poorly in high noise channels but outperform codes with smaller column weight in low noise channels. For the full rank LDPC codes the advantages of a larger column weight can be difficult to show using simulations due to the extremely low bit error rates at which the larger column weight becomes beneficial.

For the second group of codes from Steiner 2-designs the linearly dependent rows in H can provide a significant benefit in decoding performance. We saw that when considering the decoding performance of an LDPC code on a binary erasure channel, the extra linear dependent parity-checks in the oval and unital codes can be advantageous. In this channel adding extra linearly dependent checks to a given parity-check matrix can not weaken its decoding performance (if a given collection of bits was not in a stopping set before the addition of a new parity-check it won't be afterwards) and it can generally improve it (by removing a stopping set). This does not necessarily hold on other channels where the correlation between parity-checks may even worsen the code's performance.

However, even when considering the decoding performance of an LDPC code on an AWGN channel, the linear dependence of the parity-checks are not necessarily a negative aspect of the oval and unital codes since incorporating linearly dependent rows in H produces a higher rate code, which can outperform the equivalent rate randomly constructed code. Furthermore, the linearly dependent checks allow a greater column weight in H without increasing its density and so very good minimum distance and minimum stopping set sizes are achieved. As seen for the small oval, unital, and EG codes a large portion of linearly dependent rows in H can offer improved AWGN decoding performance over the equivalent length and rate codes with full rank parity-check matrices.

For the codes we consider our observations are complicated by the accompanying increase in column weight with code length for these constructions. Thus, although it is not as pronounced as for full rank codes, the performance gain of the longer unital and oval codes, and indeed the existing EG and PG codes, is in general only realised at high signal-to-noise ratios. This becomes more pronounced as the length, and hence column weight, of the codes is increased. Constructions for long codes with small column weight and many linearly dependent parity-check equations unfortunately eludes us.

Overall, the oval and unital codes are promising LDPC codes for the short codes at all

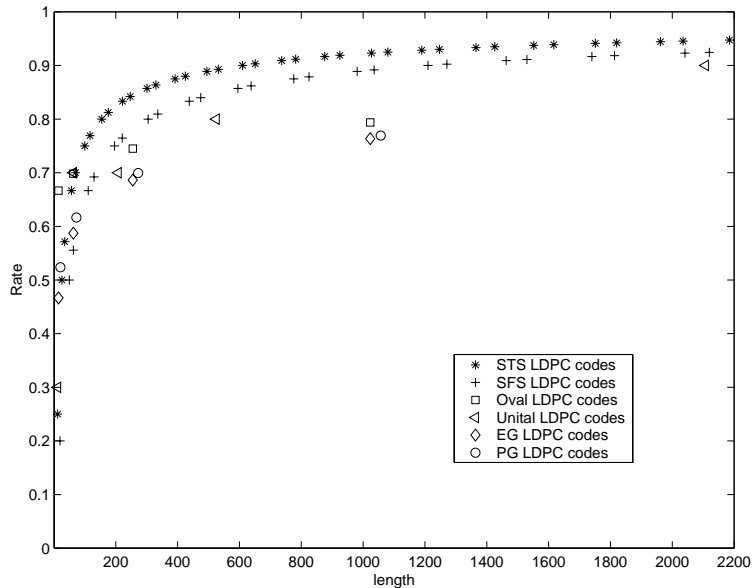


Figure 4.8: Rates and lengths of LDPC codes from Steiner 2-designs

signal-to-noise ratios and for the longer codes if used in high signal-to-noise ratio channels. The longer codes may be particularly beneficial for applications such as deep space or near earth communications. The oval and unital codes can offer a modest advantage over the existing Euclidean and projective geometry codes due to their higher rate, their lower column weights, resulting in lower decoding complexity, and their equally good decoding performances compared to equivalent randomly constructed codes.

All the codes from Steiner 2-designs have Tanner graphs with maximum girth 6, and so they can not show the logarithmic relationship between girth and block length known to be achievable for sufficiently long random codes. Thus, for sufficiently large lengths, randomly constructed codes would be expected to outperform codes from Steiner 2-designs. However, as we saw for the Steiner triple systems, this length may be large enough so that codes from Steiner 2-designs represent the best choice for many practical applications requiring high rate LDPC codes.

The other benefits of the deterministic construction provided by Steiner 2-designs is that the storage requirements necessary to completely describe the LDPC codes are reduced. If storage is a significant issue it is possible to specify only the required value of v and γ , and the entire code can be constructed on-line with some expenditure in terms of computational complexity. Alternatively, where the hard-wiring of the codes Tanner graph is employed [9], the exact regularity of the codes from Steiner 2-designs translates directly into regularity in the layout of an application-specific integrated circuit (ASIC). So in addition to providing guaranteed code properties such as the absence of 4-cycles,

LDPC codes from Steiner 2-designs are significantly easier to implement than randomly constructed codes.

In closing, a final observation we wish to make about codes from designs concerns code parameter availability. Fig. 4.8 shows the parameters of the codes from Steiner 2-designs defined by setting $H = N$. These include the codes presented here as well as the existing codes from finite projective geometries. Also shown are the codes from the finite Euclidean geometries, although not Steiner 2-designs they are closely related to finite affine planes. The codes from Steiner 2-designs are only available for a restricted range of rates and lengths and, particularly for the column weight three designs, these rates are very high even for quite short code lengths. While high rate codes are of importance for applications such as magnetic disk drives, where the non-linearities of the read-write mechanisms deter an increase of bit density to compensate for increased coding redundancy, applications such as wireless communications frequently employ much lower rate error correction codes. The search for a wider range of deterministic and regular LDPC codes is the subject of the following two chapters.

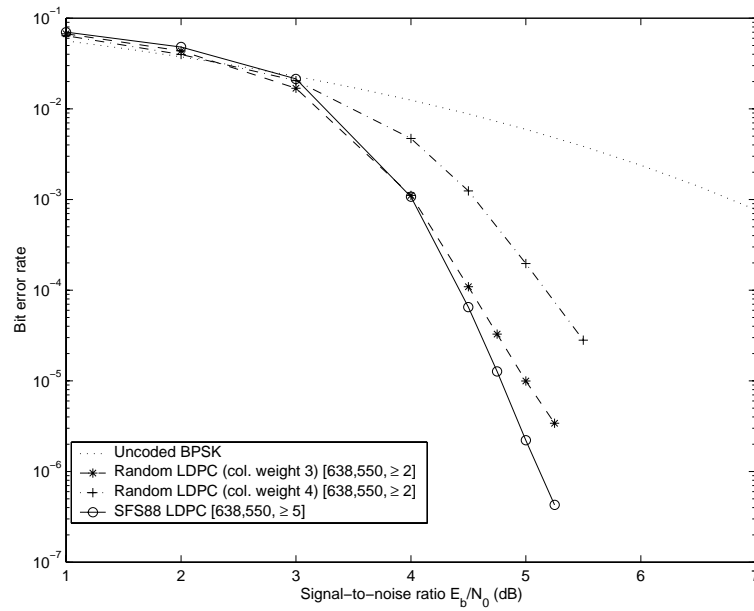


Figure 4.9: The decoding performance of the length-638 SFS LDPC code in an AWGN channel using sum-product decoding with a maximum of 50 iterations.

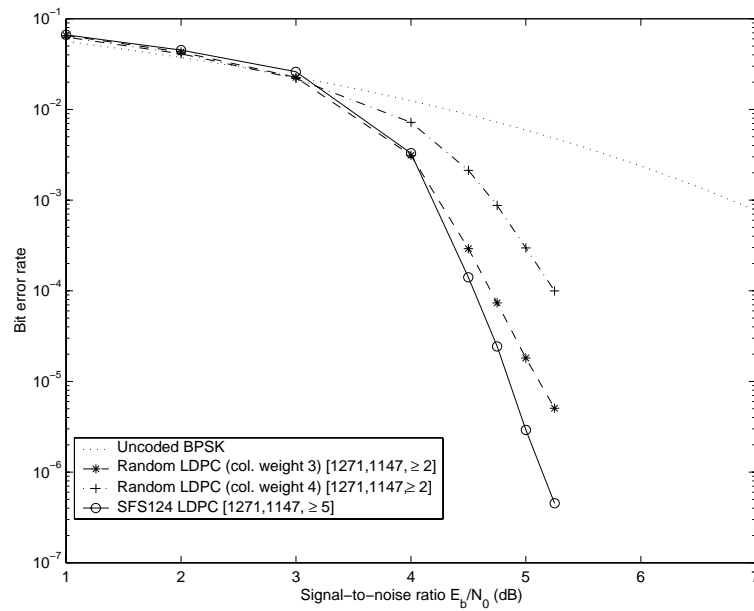


Figure 4.10: The decoding performance of the length-1271 SFS LDPC code in an AWGN channel using sum-product decoding with a maximum of 50 iterations.

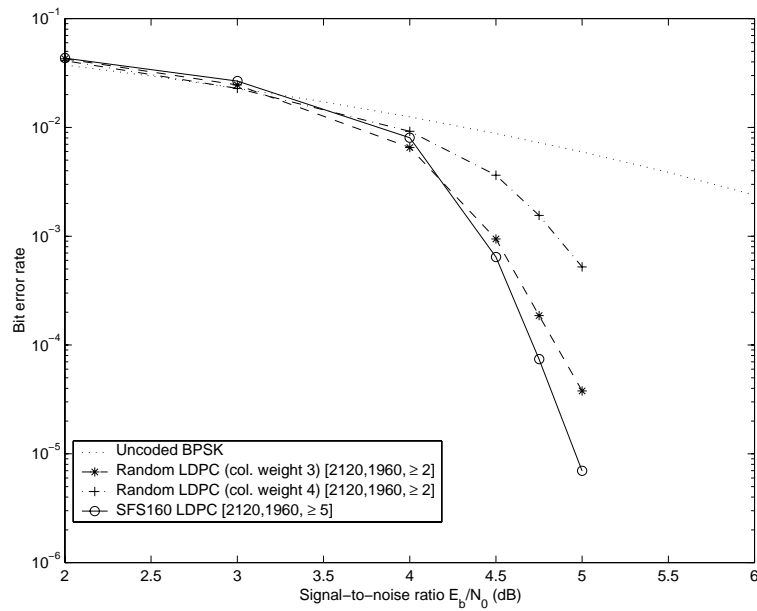


Figure 4.11: The decoding performance of the length-2120 SFS LDPC code in an AWGN channel using sum-product decoding with a maximum of 50 iterations.

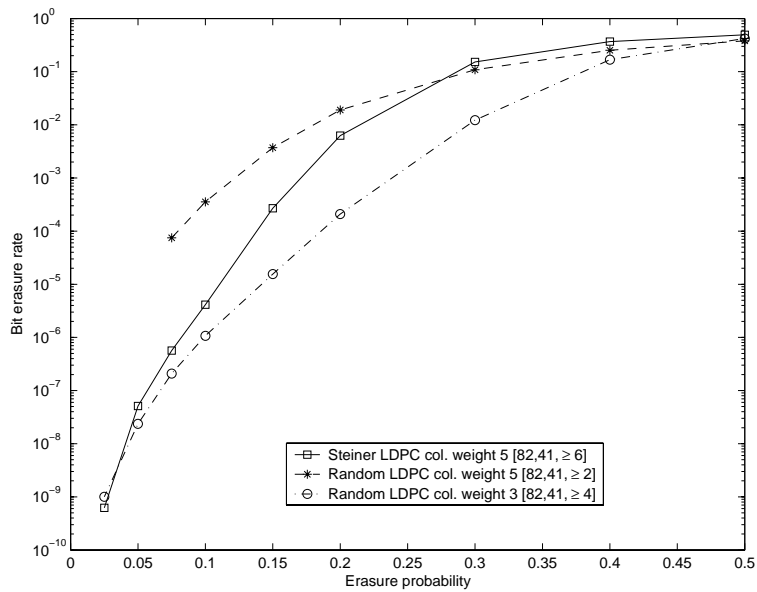


Figure 4.12: The decoding performance on a BEC of length-82, rate half, regular LDPC codes decoded using sum-product decoding.

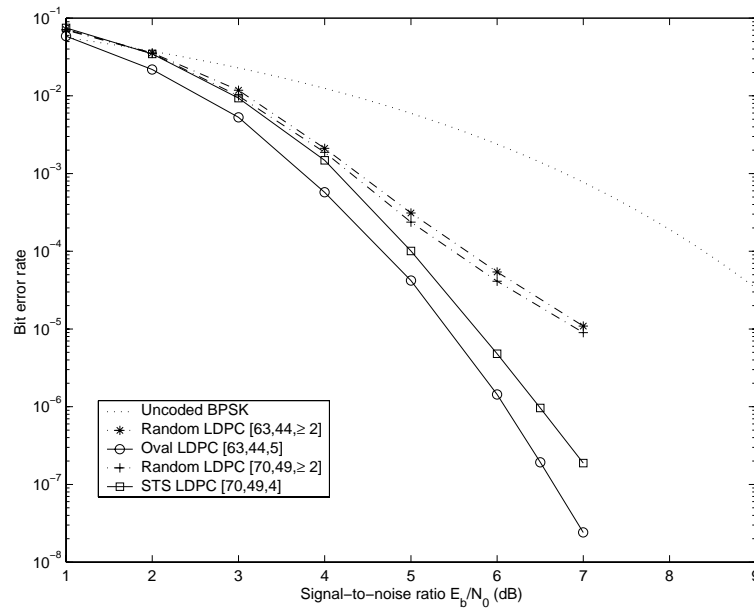


Figure 4.13: The decoding performance of oval and STS LDPC codes on an AWGN channel using sum-product decoding with a maximum of 50 iterations.

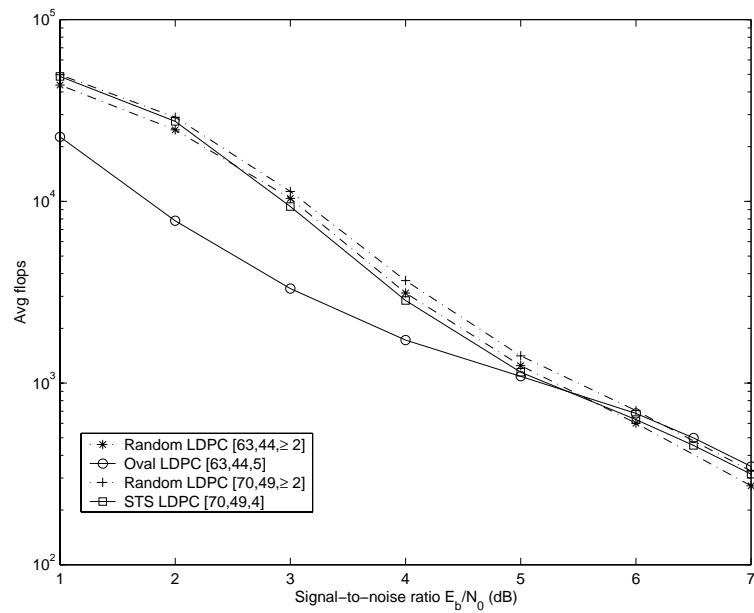


Figure 4.14: The average number of floating point multiplications required to decode a codeword for iteratively decoded LDPC codes on an AWGN channel with a maximum of 50 iterations.

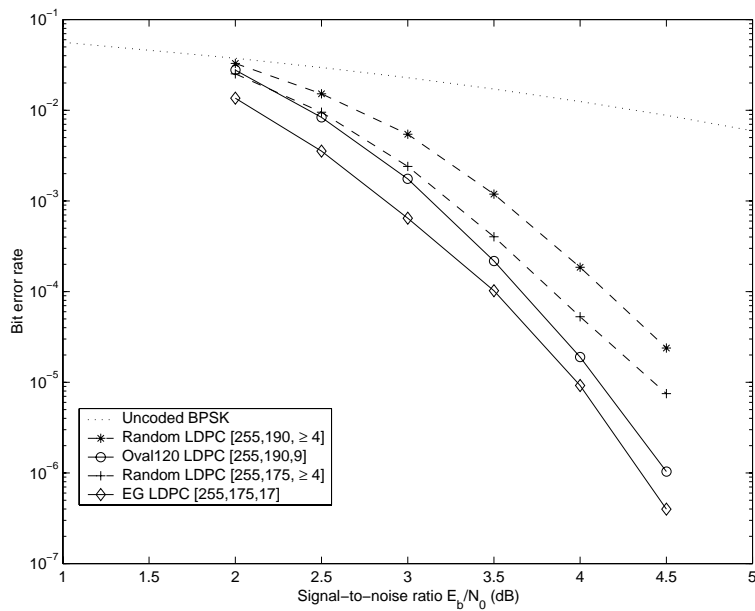


Figure 4.15: The decoding performance of length-255 LDPC codes on an AWGN channel using sum-product decoding with a maximum of 50 iterations.

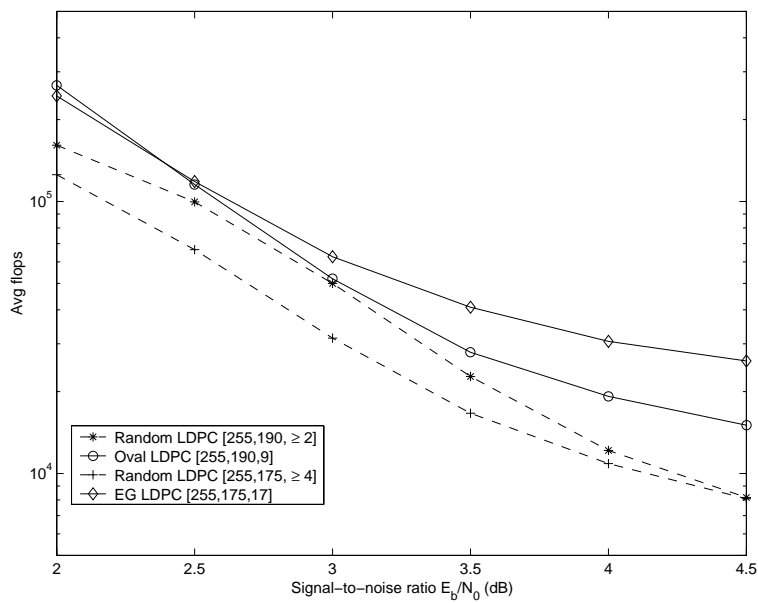


Figure 4.16: The average number of floating point multiplications required to decode a codeword for length-255 iteratively decoded LDPC codes on an AWGN channel with a maximum of 50 iterations.

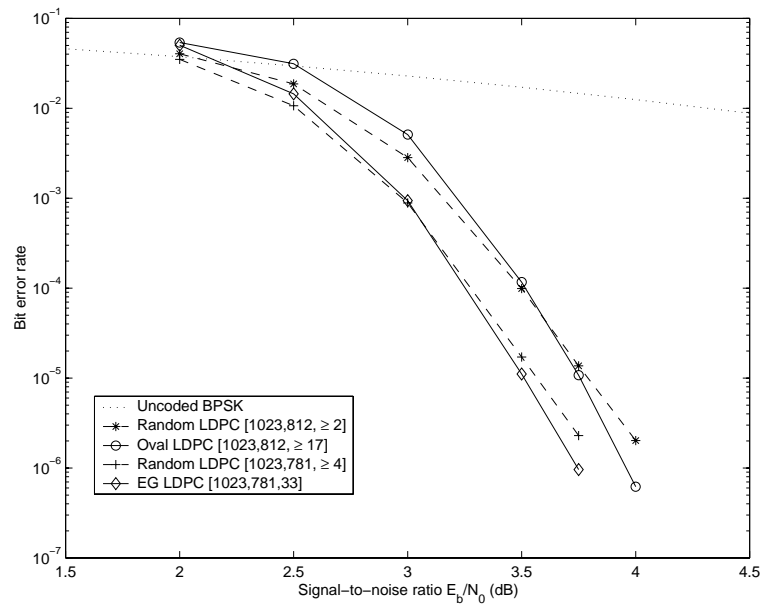


Figure 4.17: The decoding performance of length-1023 LDPC codes on an AWGN channel using sum-product decoding with a maximum of 1000 iterations.

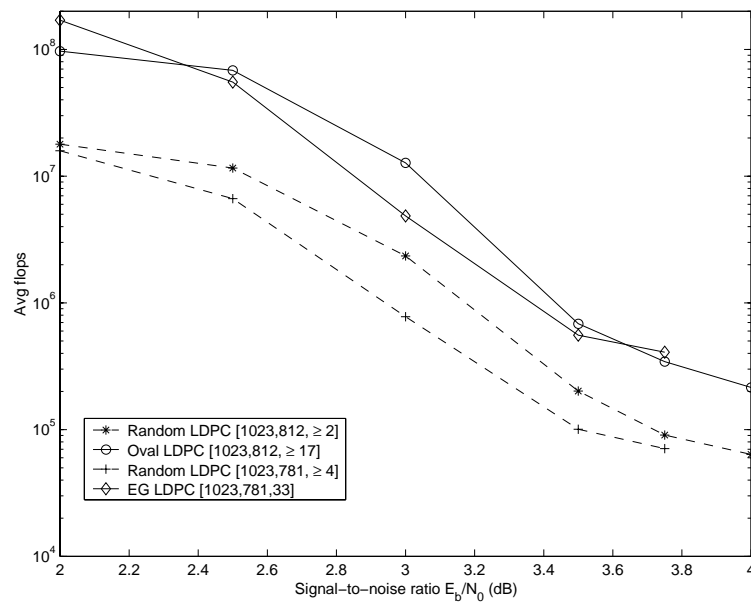


Figure 4.18: The average number of floating point multiplications required to decode a codeword for length-1023 iteratively decoded LDPC codes on an AWGN channel with a maximum of 1000 iterations.

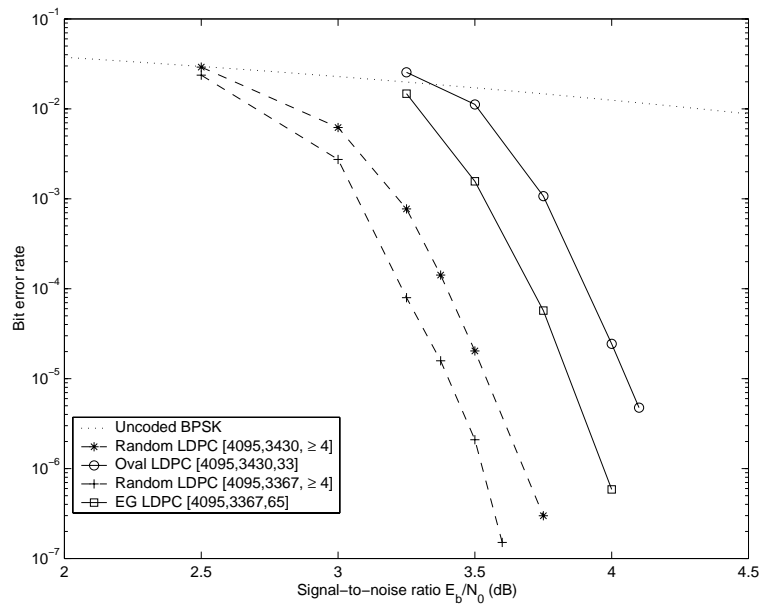


Figure 4.19: The decoding performance of the length-4095 LDPC codes on an AWGN channel using sum-product decoding with a maximum of 1000 iterations.

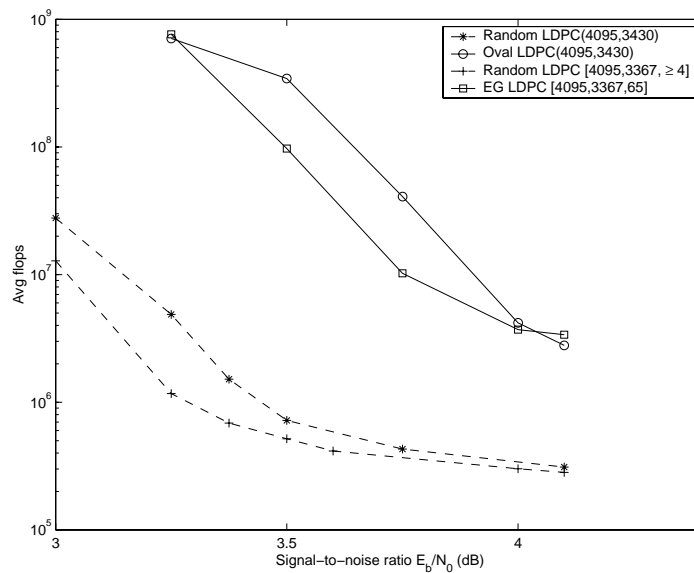


Figure 4.20: The average number of floating point multiplications required to decode a codeword for length-4095 iteratively decoded LDPC codes on an AWGN channel with a maximum of 1000 iterations.

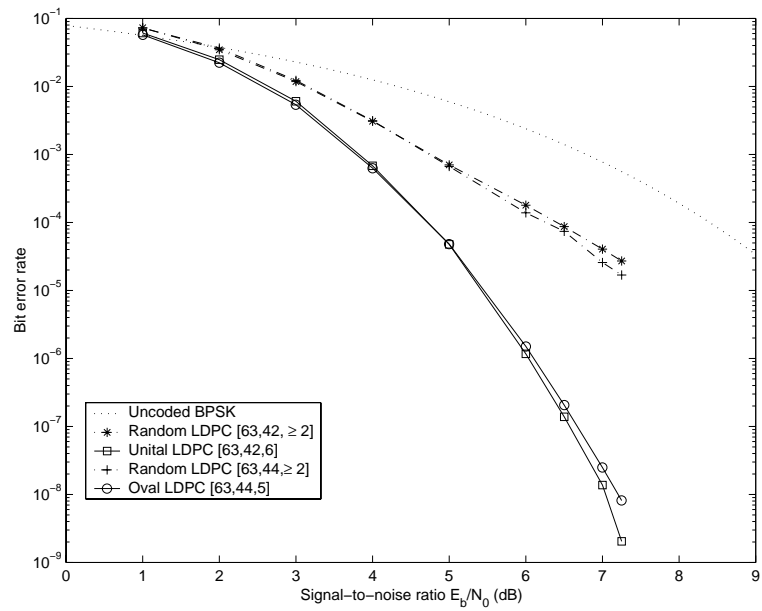


Figure 4.21: The decoding performance of the length-63 oval and unital LDPC codes on an AWGN channel using sum-product decoding with a maximum of 10 iterations.

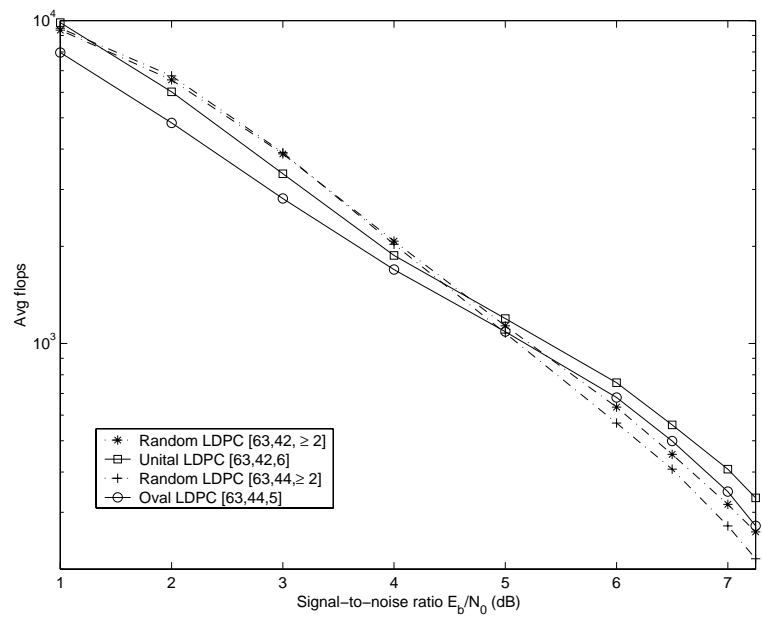


Figure 4.22: The average number of floating point multiplications required to decode a codeword for iteratively decoded LDPC codes on an AWGN channel with a maximum of 10 iterations.

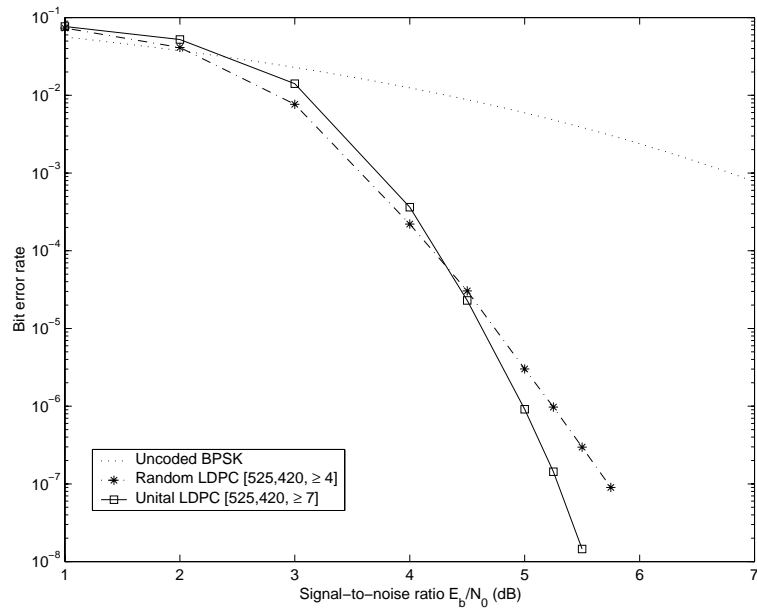


Figure 4.23: The decoding performance of the length-525 unital LDPC code on an AWGN channel using sum-product decoding with a maximum of 10 iterations.

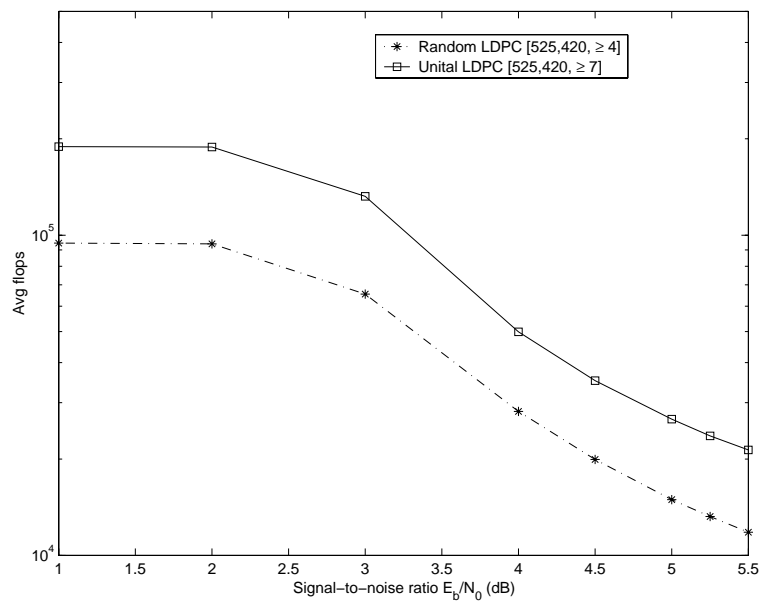


Figure 4.24: The average number of floating point multiplications required to decode a codeword for iteratively decoded LDPC codes on an AWGN channel with a maximum of 10 iterations.

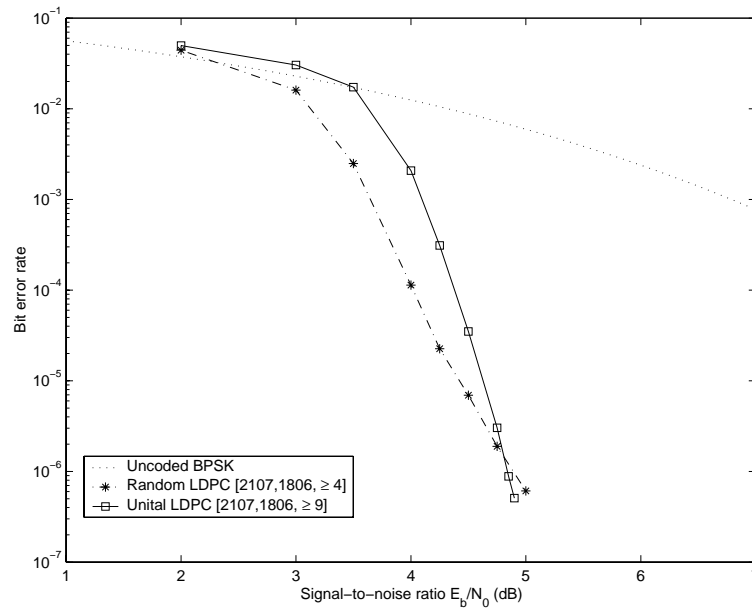


Figure 4.25: The decoding performance of the length-2107 unital LDPC code on an AWGN channel using sum-product decoding with a maximum of 10 iterations.

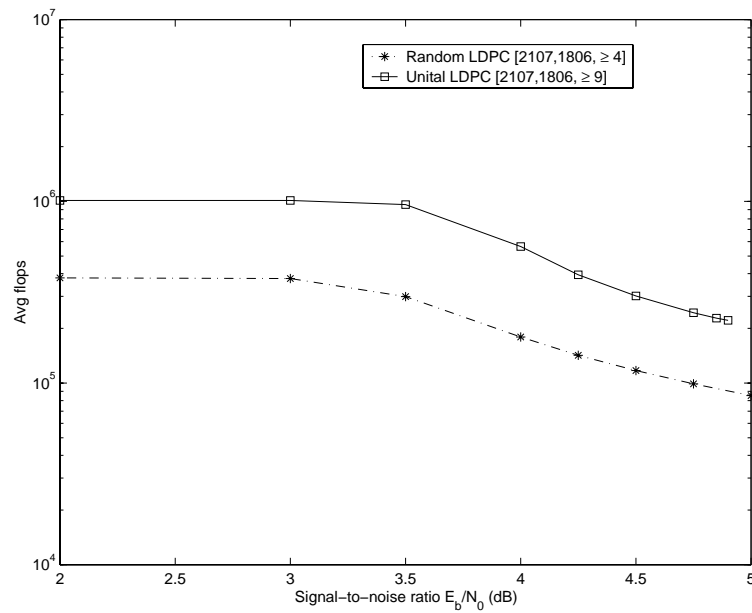


Figure 4.26: The average number of floating point multiplications required to decode a codeword for iteratively decoded LDPC codes on an AWGN channel with a maximum of 10 iterations.

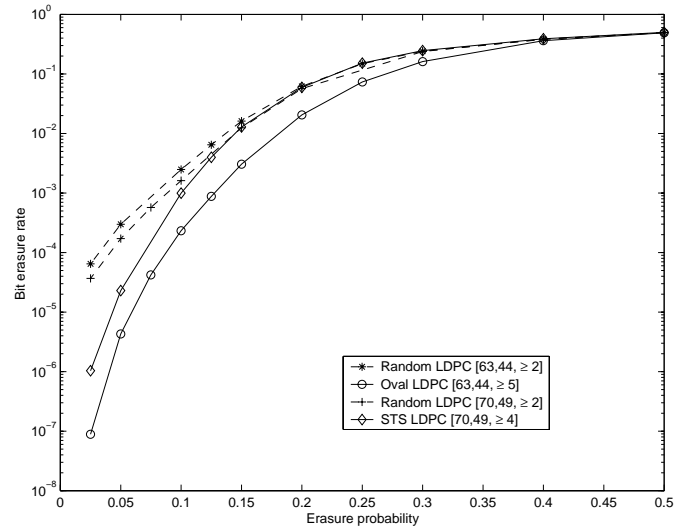


Figure 4.27: The erasure correction performance of short oval, STS and random LDPC codes on a binary erasure channel.

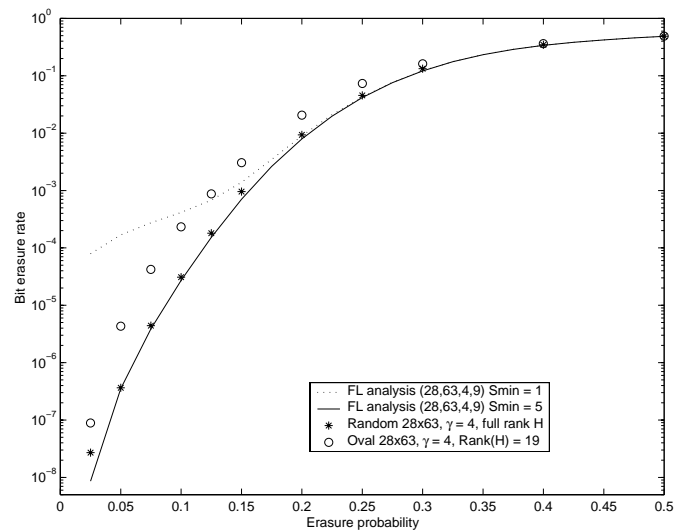


Figure 4.28: The erasure correction performance on a binary erasure channel of LDPC codes with $(4,9)$ -regular parity-check matrices of size 28×63 . The continuous curves show the average erasure correction performance of an ensemble while individual symbols give the simulated erasure correction performance of individual codes.

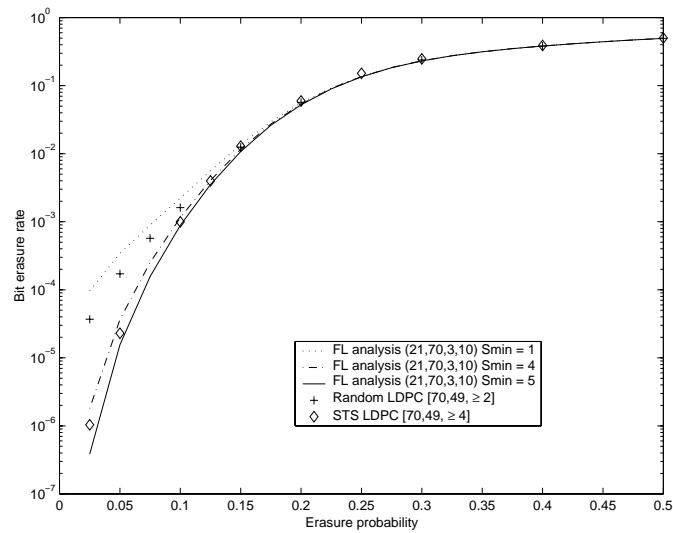


Figure 4.29: The erasure correction performance on a binary erasure channel of LDPC codes with $(3,10)$ -regular parity-check matrices of size 21×70 . The continuous curves show the average erasure correction performance of an ensemble while individual symbols give the simulated erasure correction performance of individual codes.

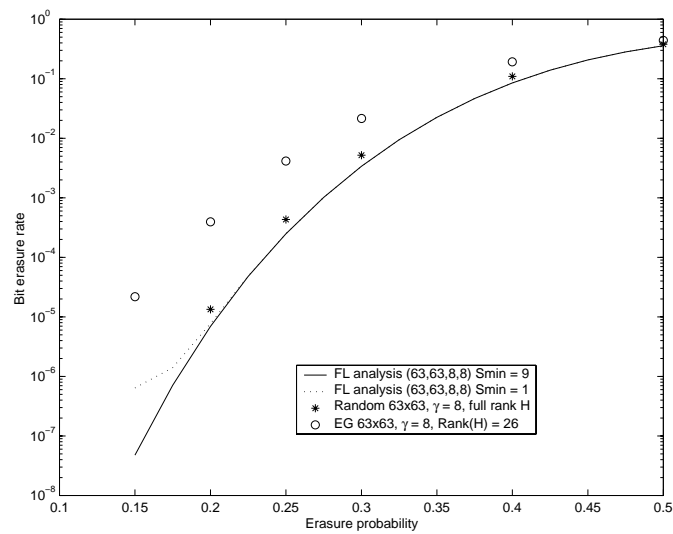


Figure 4.30: The erasure correction performance on a binary erasure channel of LDPC codes with $(8,8)$ -regular parity-check matrices of size 63×63 . The continuous curves show the average erasure correction performance of an ensemble while individual symbols give the simulated erasure correction performance of individual codes.

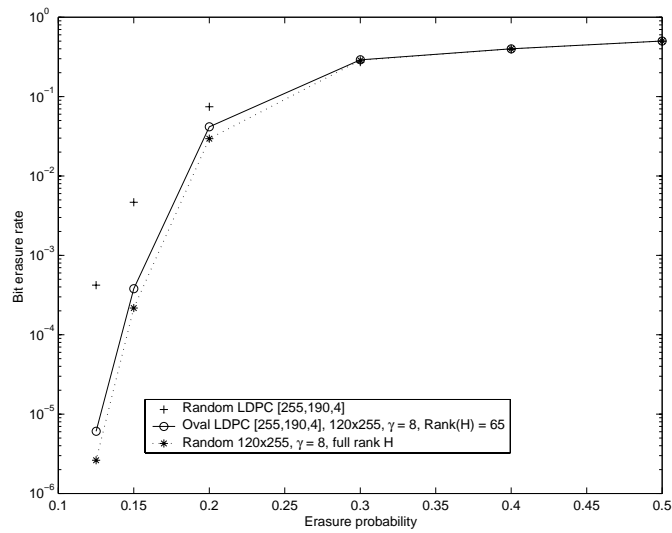


Figure 4.31: The erasure correction performance of length-255 oval and random LDPC codes on a binary erasure channel.

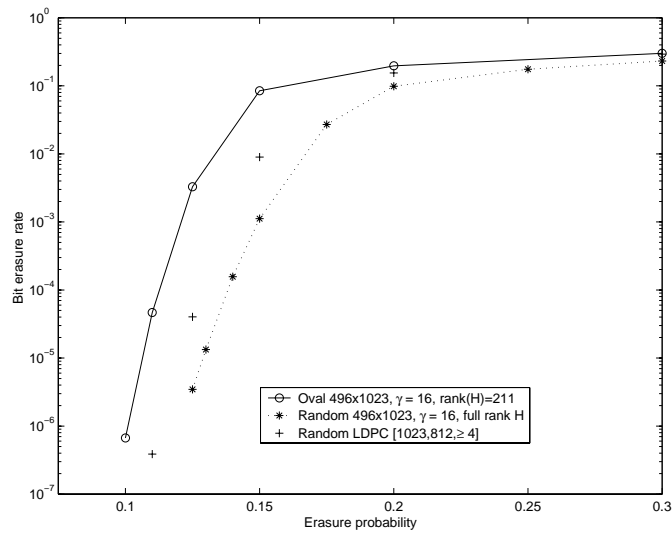


Figure 4.32: The erasure correction performance of length-1023 oval and random LDPC codes on a binary erasure channel.

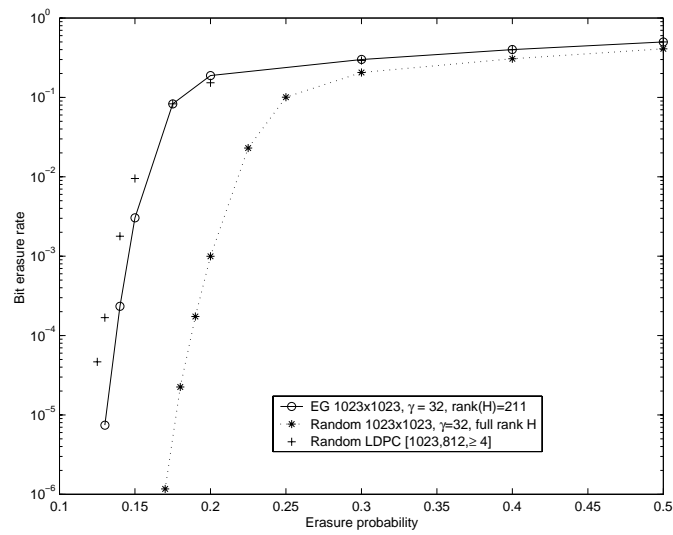


Figure 4.33: The erasure correction performance of length-1023 EG and random LDPC codes on a binary erasure channel.

A GENERALIZATION TO PARTIAL GEOMETRIES

In this chapter we consider partial geometries in a generalization to a large class of algebraic LDPC codes which contain many of the existing algebraic LDPC codes as special cases. The incidence matrices of partial geometries provide systematic constructions for regular LDPC codes with Tanner graphs free of 4-cycles and with deterministic code properties. The incidence structure of the partial geometries enables us to derive code properties such as minimum distance, girth, rate, and stopping set distribution for the whole class of codes.

5.1 Introduction

In the previous chapters promising LDPC codes were derived from Steiner 2-designs for most code lengths but for only a restricted set of (mostly high) code rates. The requirement that every pair of points occur in a block together ($\lambda = 1$) produces incidence graphs with the maximum number of bit nodes for a 4-cycle free graph with a given number of check nodes. For lower rate codes we require incidence structures with fewer bit nodes and hence we require designs with each pair of points in *at most* one block together but also with enough structure for control over the code properties.

Partial geometries represent a solution to this problem providing a deterministic method of constructing 4-cycle free bipartite graphs with less connectivity than the incidence graphs of Steiner 2-designs. In considering partial geometries we generalize our search to designs with similar properties to Steiner 2-designs, namely that each point occurs in a fixed number of blocks, each block in a fixed number of points and that the design incidence graphs are free of 4-cycles. The difference between partial geometries and Steiner 2-designs is that each pair of points in the partial geometry are not required to occur in a block together and thus partial geometries are not balanced incomplete block designs. The definition of partial geometries does however specify how closely a pair of points are connected depending on whether or not they are in a block together.

More specifically, a partial geometry, denoted $\text{pg}(s, t, \alpha)$, satisfies the following properties [19, p. 33]:

- P1.** Each point P is incident with $t + 1$ blocks and each block B is incident with $s + 1$ points.
- P2.** Any two distinct blocks have at most one point in common.
- P3.** For any non-incident point-block pair (P, B) the number of blocks incident with P and intersecting B equals some constant α .

It is this last property that separates the codes from partial geometries from the codes from Steiner 2-designs. Using it we can derive expressions for the girth, minimum distance and rate of the codes and, by adjusting α , increase or decrease the amount of connectivity between code bits and checks. By choosing the largest possible value of α we get the most connectivity and the resulting designs are in fact Steiner 2-designs, a special case of partial geometries. The subset of partial geometries with the smallest value of α , equal to 1, are the generalized quadrangles, which have also been considered as LDPC codes [118]. The four main classes of partial geometries are:

- a partial geometry with $\alpha = s + 1$ is a Steiner 2-design or $2-(v, s + 1, 1)$ design,
- a partial geometry with $\alpha = t$ is called a *net* or, dually with $\alpha = s$, a *transversal design* (TD),
- a partial geometry with $\alpha = 1$ is called a *generalized quadrangle* (GQ),
- if $1 < \alpha < \min\{s, t\}$ the partial geometry is *proper*,

We consider two classes of partial geometries in more detail, proper partial geometries in Section 5.3 and transversal designs in Section 5.4, but first we derive some results for the LDPC codes from all the partial geometries.

5.2 Codes from partial geometries

We can take the incidence matrix N of a partial geometry as the parity-check matrix H of an LDPC code which we denote by C^1 . The code C^1 has $m = |\mathcal{P}|$ parity-checks, length $n = |\mathcal{B}|$ and the parity-check matrix is $(s + 1, t + 1)$ -regular.

Lemma 5.2.1 *The error correction codes C^1 from all of the partial geometries have Tanner graphs free of 4-cycles.*

Proof. A 4-cycle requires two bit nodes to be connected to the same two check nodes (see Section 2.1.1) and thus that two columns of H contain ones in two places in common. By property **P2** of partial geometries, any two blocks have at most one point in common and so any two columns of N both have ones in at most one row. Thus the Tanner graph of the codes with $H = N$ are free of 4-cycles. \square

For any $\alpha > 1$ property **P3** guarantees the existence of a 6-cycle, so all partial geometries other than the generalized quadrangles, have a girth of 6. Further, the number of 6-cycles in a partial geometry code can be enumerated exactly.

Lemma 5.2.2 *The exact number of 6-cycles in the Tanner graph of a code from a partial geometry, $pg(s, t, \alpha)$, is*

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

Proof. The number of 6-cycles in H can be counted using only the properties **P1–P3** of the partial geometries. If we take a line l of the partial geometry there are $\binom{s+1}{2}$ different pairings of the points in l . Now, take one pair (P_1, P_2) of points in l . The point P_1 is incident in t lines other than line l , none of which contain the point P_2 . However, the point P_2 is connected to α of the points in each of these lines, one of which is P_1 . Thus the points P_1 and P_2 are connected in a cycle of size 6 through each of the t lines containing P_1 for each of the $\alpha - 1$ lines intersecting them. So, each pair of points (P_1, P_2) is involved in $t(\alpha - 1)$ 6-cycles. Given that there are $\binom{s+1}{2}$ pairs of points in a line l there are $t(\alpha - 1)\binom{s+1}{2}$ 6-cycles containing the points in l . There are n lines in total, and given a single 6-cycle includes 3 lines the result follows. \square

The generalized quadrangles have no 6-cycles and we can count the exact number of 8-cycles.

Lemma 5.2.3 *The exact number of 8-cycles in the Tanner graph of a code from a partial geometry, $pg(s, t, 1)$, is*

$$N_8 = \frac{snt^2}{4} \binom{s+1}{2}.$$

Proof. The number of 8-cycles in H can be counted using again only the properties **P1–P3** of the partial geometries. If we take a line L of the partial geometry there are $\binom{s+1}{2}$ different pairings of the points in L . Now, take one pair (P_1, P_2) of points in L . The point P_1 is incident in t lines other than line L , none of which contain the point P_2 and we choose any one of them as L_1 . Similarly, the point P_2 is incident in t lines other than line L , none of which contain the point P_1 and again we choose any one of them as

L_2 . The lines L_1 and L_2 can not share a point for if they did that point would be in two lines intersecting L , contravening **P3**. However, all the points on L_1 must be incident in a line which intersects L_2 . For the point P_1 that line is L , while each of the other s points on L_1 are incident in s separate lines which intersects L_2 and each one of them forms an 8-cycle with L , L_1 and L_2 . So the pair of points (P_1, P_2) are incident in exactly $t^2 s$ 8-cycles. There are $\binom{s+1}{2}$ ways to choose a pair of points on each of n lines and each 8-cycle includes four lines and the result follows. \square

As the incidence graphs of partial geometries are free of 4-cycles, the size of the smallest stopping set is at least one greater than the minimum column weight of H [83] and thus the codes from partial geometries have minimum stopping set size

$$S_{\min} \geq s + 2. \quad (5.1)$$

Like the Steiner 2-designs the benefits of the deterministic construction of the partial geometries is that the storage requirements necessary to completely describe the LDPC codes are reduced. If storage is a significant issue it is possible to specify only the required value of s , t , and α and the entire code can be constructed on-line with some expenditure in terms of computational complexity using a known algebraic construction for that design. Alternatively, where the hard-wiring of the code's Tanner graph is employed as in [9] the exact regularity of the codes from partial geometries translates directly into regularity in the layout of an application-specific integrated circuit (ASIC). So as well as providing guaranteed code properties, LDPC codes from partial geometries are easier to implement than randomly constructed codes.

5.2.1 Minimum distance

The incidence matrices of partial geometries are column orthogonal (by definition **P2**) and so Massey's bound (Lemma 3.2.3) can be applied to lower bound the minimum distance of LDPC codes from partial geometries:

$$d \geq s + 2. \quad (5.2)$$

However, by using known properties of strongly regular graphs and Tanner's minimum distance bound we can derive, in a similar manner to [118], a minimum distance bound for codes from partial geometries which is tighter for some of the partial geometry codes.

In [104], Tanner presented the following bounds for the minimum distance d of a code with a regular parity-check matrix H provided that the multiplicity of the largest eigenvalue, μ_1 , of HH^T is 1. Let w_c be the column weight of H , w_r the row weight of H and μ_2 the second largest distinct eigenvalue of HH^T . Then from [104, Theorem 3.1] we have the *bit-oriented bound*:

$$d \geq \frac{n(2w_c - \mu_2)}{(\mu_1 - \mu_2)}, \quad (5.3)$$

and from [104, Theorem 4.1] the *parity-oriented bound* :

$$d \geq \frac{2n(2w_c + w_r - 2 - \mu_2)}{w_r(\mu_1 - \mu_2)}. \quad (5.4)$$

We shall use these bit and parity-oriented bounds, together with the properties of strongly regular graphs, to give lower bounds on d in terms of α , s and t for codes obtained from partial geometries. For this, we need expressions for the values and multiplicities of the eigenvalues of NN^T and we shall use two well known properties of strongly regular graphs presented in the following two Lemmas.

Lemma 5.2.4 [19, p. 21] *The adjacency matrix, A , of a strongly regular graph has three distinct real eigenvalues completely specified by the parameters of the graph.*

Proof. By definition the adjacency matrix of a strongly regular graph is symmetric, has a zero diagonal and column and row weights equal to n_1 thus:

$$AJ = JA = n_1J. \quad (5.5)$$

By (5.5), A has an eigenvalue, e_0 , with value n_1 and multiplicity 1 corresponding to the eigenvalue v of J .

Now if we consider the matrix A^2 we note that the (i, j) th entry of A^2 is the number of points connected to both of the points P_i and P_j so $A^2_{i,j}$ takes on one of three values

$$A^2_{i,j} = \begin{cases} n_1, & \text{if } i = j \\ p_1, & \text{if } P_i \text{ and } P_j \text{ are connected} \\ p_2, & \text{if } P_i \text{ and } P_j \text{ are not connected} \end{cases}$$

which can be expressed as:

$$A^2 = n_1I + p_1A + p_2(J - I - A). \quad (5.6)$$

By equations (5.5) and (5.6) I , J and A span a real algebra of dimension 3 which is commutative and consists of symmetric matrices, thus A has three eigenvalues [19]. We know already that A has an eigenvalue of n_1 and substituting into equation (5.6) gives the following relationship between the parameters of a strongly regular graph:

$$n_1n_1 = n_1 \cdot 1 + p_1n_1 + p_2(v - 1 - n_1)$$

and so

$$n_1(n_1 - p_1 - 1) = p_2(v - n_1 - 1).$$

Any other eigenvalue of J is zero, so the other eigenvalues of A , e_1, e_2 , satisfy:

$$\begin{aligned} e^2 &= n_1 + p_1 e - p_2(e + 1) \\ &= (n_1 - p_2) + e(p_1 - p_2) \end{aligned}$$

and so [19, p. 21]

$$e_1, e_2 = \frac{(p_1 - p_2) \pm \sqrt{(p_1 - p_2)^2 + 4(n_1 - p_2)}}{2},$$

and the lemma is proved. \square

It is also possible to determine the multiplicity of these eigenvalues in terms of the parameters of the strongly regular graphs. Denote by f_1 and f_2 the multiplicities of e_1 and e_2 . Since the multiplicity of e is 1:

$$v = 1 + f_1 + f_2.$$

The diagonal elements of A are all zero (2.18) and so

$$\text{Tr}(A) = n_1 + f_1 e_1 + f_2 e_2 = 0.$$

Combining gives:

$$f_1, f_2 = \frac{1}{2} \left[(v - 1) \pm \frac{(v - 1)(p_2 - p_1) - 2n_1}{\sqrt{(p_1 - p_2)^2 + 4(n_1 - p_2)}} \right].$$

Lemma 5.2.5 [16, p. 386] *For a partial geometry \mathcal{D} with incidence matrix N the adjacency matrix of the point graph of \mathcal{D} is $A = NN^T - (t + 1)I$.*

Proof. By the properties of partial geometries the matrix NN^T satisfies:

$$(NN^T)_{i,j} = \begin{cases} t + 1 & \text{if } i = j \\ 1 & \text{if } i \text{ and } j \text{ connected} \\ 0 & \text{if } i \text{ and } j \text{ not connected} \end{cases}$$

which is equivalent to the expression

$$NN^T = (t + 1)I + 1A + 0(J - A - I),$$

and the result follows. \square

From Lemma 5.2.4 and equation (2.19) the multiplicity of the largest eigenvalue of the adjacency matrix of a partial geometry is one, and the eigenvalues of A , written in the notation of partial geometries (2.19), are:

$$s(t + 1), \quad s - \alpha, \quad -(t + 1).$$

Class of partial geometry	Code length	Minimum distance bounds
Steiner 2-design	$\frac{(t+1)(st+s+1)}{s+1}$	$d \geq \max \left\{ \frac{(t+1)(2s+2-t)}{s+1}, \frac{2(2s+1)}{s+1} \right\}$
Net	$(s+1)(t+1)$	$d \geq \max \left\{ \frac{(t+1)(s+1)}{t}, \frac{2(s+t)}{t} \right\}$
Transversal design	$(t+1)^2$	$d \geq \max \left\{ \frac{(t+1)(2s-t+1)}{s}, 4 \right\}$
Generalized quadrangle	$(t+1)(st+1)$	$d \geq \max \{ (t+1)(s+2-t), 2(s+1) \}$
Proper partial geometry	$\frac{(t+1)(st+\alpha)}{\alpha}$	$d \geq \max \left\{ \frac{(t+1)(s+1-t+\alpha)}{\alpha}, \frac{2(s+\alpha)}{\alpha} \right\}$

Table 5.1: Minimum distance bounds for LDPC codes from partial geometries

From Lemma 5.2.5 it follows that if μ is an eigenvalue of A with multiplicity f then $\mu + (t+1)$ is an eigenvalue of NN^T with multiplicity f , and so for N the incidence matrix of a partial geometry, NN^T has eigenvalues

$$(s+1)(t+1), \quad s+t+1-\alpha, \quad 0 \quad (5.7)$$

with multiplicities

$$1, \quad \frac{st(s+1)(t+1)}{\alpha(s+t+1-\alpha)}, \quad \frac{s(s+1-\alpha)(st+\alpha)}{\alpha(s+t+1-\alpha)}. \quad (5.8)$$

Finally, we get an expression for the minimum distance:

Lemma 5.2.6 *The minimum distance of a code from a partial geometry, $pg(s, t, \alpha)$, satisfies*

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

Proof. With H , the parity-check matrix of an LDPC code, defined as the incidence matrix of a partial geometry, $pg(s, t, \alpha)$, we have $w_r = t+1$, $w_c = s+1$, $n = \frac{(t+1)(st+\alpha)}{\alpha}$. From Lemmas 5.2.4 and 5.2.5, HH^T has a largest eigenvalue $(s+1)(t+1)$ with multiplicity 1 and second largest eigenvalue $s+t+1-\alpha$. Substituting into equations (5.3) and (5.4) the result follows. \square

The minimum distance bounds for LDPC codes from each class are given in Table 5.1. Vontobel and Tanner considered LDPC codes based on generalized quadrangles and the minimum distance bounds in Table 5.1 for the generalized quadrangles were presented in [118].

The minimum distance bounds from Lemma 5.2.6 are weak for the Steiner 2-designs, nets and transversal designs; a better bound is provided by (5.2). However for the generalized quadrangle and proper partial geometry codes the bounds from Lemma 5.2.6

significantly improve on the bound in (5.2) to give minimum distances up to twice the column weight of H . As we saw in Chapter 4 increasing the LDPC code column weight in order to increase the code minimum distance can have a detrimental effect on code performance by increasing the code density, and so a minimum distance bound greater than $\gamma + 1$ is beneficial.

5.2.2 Linearly dependent rows in H

The generalized quadrangles [118] are described by incidence matrices which are significantly rank deficient and this may also be the case for other partial geometry designs. Thus we seek expressions of the 2-rank of H for the partial geometry codes as a function of the parameters of the partial geometry, not only to determine the rate of a given code without needing to construct it, but also as a means to select those parameters which lead to highly redundant parity-check matrices.

A simple upper bound on the 2-rank of a code is $\text{rank}(H)$, the number of non-zero eigenvalues of HH^T , which we know for partial geometry designs (5.7) and so we can upper bound the rank of H in terms of s , t and α :

$$\text{rank}_2(H) \leq \frac{st(s+1)(t+1)}{\alpha(t+s+1-\alpha)} + 1. \quad (5.9)$$

Next, for a lower bound on the 2-rank of H we use results from Brouwer's work on the p -rank of the adjacency matrices of strongly regular graphs [15].

Lemma 5.2.7 [15] *If A is the adjacency matrix of a strongly regular graph with eigenvalues k, r, u and multiplicities $1, f, g$ respectively, and the matrix M is defined as, $M = A + bJ + cI$, for some b and c , then M has eigenvalues $\theta_0 = k + bv + c$, $\theta_1 = r + c$, $\theta_2 = u + c$, with multiplicities $1, f, g$ respectively. Further,*

B1. If precisely one eigenvalue θ_i of M is $\equiv 0 \pmod p$ then $\text{rank}_p(M) = v - m_i$, where m_i is the multiplicity of that eigenvalue.

B2. For the case of two eigenvalues of $M \equiv 0 \pmod p$. If $\theta_0 \equiv \theta_1 \equiv 0 \pmod p$ and $\theta_2 \not\equiv 0 \pmod p$ then $\text{rank}_p(M) = g$ if $p|e$ and $\text{rank}_p(M) = g + 1$ otherwise. Similarly if $\theta_0 \equiv \theta_2 \equiv 0 \pmod p$ and $\theta_1 \not\equiv 0 \pmod p$ then $\text{rank}_p(M) = f$ if $p|e$ and $\text{rank}_p(M) = f + 1$ otherwise.

In the above, $e := \mu + b^2v + 2bk + b(\mu - \lambda)$, with $\lambda = r + u + \mu$ and $\mu = ru + k$.

For H the incidence matrix of a partial geometry we can define $M = HH^T = A + (t + 1)I$, and thus $k = s(t + 1)$, $r = s - \alpha$, $u = -(t + 1)$, $\theta_0 = (s + 1)(t + 1)$, $\theta_1 = s + t + 1 - \alpha$, $\theta_2 = 0$ and $e = \alpha(t + 1)$. Then Brouwer's results show that if $s + t + 1 - \alpha \equiv 1 \pmod{2}$, the 2-rank of HH^T is

$$\text{rank}_2(HH^T) = \frac{st(s + 1)(t + 1)}{\alpha(s + t + 1 - \alpha)} + 1$$

when $(s + 1)(t + 1)$ or $(t + 1)\alpha \equiv 1 \pmod{2}$, and

$$\text{rank}_2(HH^T) = \frac{st(s + 1)(t + 1)}{\alpha(s + t + 1 - \alpha)}$$

otherwise. As $\text{rank}_2(H) \geq \text{rank}_2(HH^T)$ we now have a lower bound on the 2-rank of H for certain choices of s, t and α , i.e. if $s + t + 1 - \alpha \equiv 1 \pmod{2}$

$$\text{rank}_2(H) \geq \frac{st(s + 1)(t + 1)}{\alpha(s + t + 1 - \alpha)}. \quad (5.10)$$

For Steiner 2-designs we have $\alpha = s + 1$ and the upper bound is full rank. For the projective geometry designs and the oval designs exact expressions for rank have been determined based on the geometric properties of these designs, see [60] and Section 4.4 respectively.

Using equation (5.10), we can see that the incidence matrix of a transversal design has at least s linearly dependent rows, and exactly that many for $t \equiv 0 \pmod{2}$, while for any partial geometry design we can choose s, t and α to guarantee at least

$$\frac{s(s + 1 - \alpha)(st + \alpha)}{\alpha(s + t + 1 - \alpha)} \quad (5.11)$$

linearly dependent rows in the parity-check matrix. So all partial geometries with $\alpha < s + 1$ produce codes with linearly dependent rows in their parity-check matrix. Thus only the codes from the Steiner 2-designs can be full 2-rank, although are not necessarily so.

Since we can bound the incidence matrix rank (5.9) we can bound the code dimension

$$k \geq t + \frac{st(t - \alpha)(t + 1)}{\alpha(s + t + 1 - \alpha)},$$

and the code rate is thus lower bounded by

$$R \geq \frac{st(t - \alpha)(t + 1) + \alpha t(s + t + 1 - \alpha)}{(s + t + 1 - \alpha)(t + 1)(st + \alpha)}.$$

5.3 Proper partial geometries

It would seem that the proper partial geometries are the class of partial geometries which provide the most flexibility as they are characterized by three independent parameters s, t , and α . In the context of LDPC codes this property translates to a flexible choice

Class	Conditions on s, t, α	$2\text{-rank}(H)$
Steiner 2-designs	t odd otherwise	$st + s \leq \text{rank}_2(H) \leq st + s + 1$ $\text{rank}_2(H) \leq st + s + 1$
Nets	s, t even s even, t odd otherwise	$\text{rank}_2(H) = st + s + 1$ $st + s \leq \text{rank}_2(H) = st + s + 1$ $\text{rank}_2(H) \leq st + s + 1$
Transversal designs	t even otherwise	$\text{rank}_2(H) = st + t + 1$ $\text{rank}_2(H) \leq st + t + 1$
Generalized quadrangles	t even, s odd s even, t odd otherwise	$\text{rank}_2(H) = \frac{st(s+1)(t+1)}{s+t} + 1$ $\frac{st(s+1)(t+1)}{s+t} \leq \text{rank}_2(H) \leq \frac{st(s+1)(t+1)}{s+t} + 1$ $\text{rank}_2(H) \leq \frac{st(s+1)(t+1)}{s+t} + 1$
Proper partial geometries	$s + t + 1 - \alpha$ odd s, t, α even t even, s, α odd otherwise	$\frac{st(s+1)(t+1)}{\alpha(s+t+1-\alpha)} \leq \text{rank}_2(H) \leq \frac{st(s+1)(t+1)}{\alpha(s+t+1-\alpha)} + 1$ $\text{rank}_2(H) = \frac{st(s+1)(t+1)}{\alpha(s+t+1-\alpha)} + 1$ $\text{rank}_2(H) = \frac{st(s+1)(t+1)}{\alpha(s+t+1-\alpha)} + 1$ $\text{rank}_2(H) \leq \frac{st(s+1)(t+1)}{\alpha(s+t+1-\alpha)} + 1$

Table 5.2: Bounds on the 2-rank of the parity-check matrix of LDPC codes from partial geometries

of the density and rank of H and the code length, minimum distance and rate. However, there is unfortunately only a small subset of the possible partial geometry parameters for which a partial geometry design is known to exist, and so our flexibility in choosing code parameters is constrained by design availability. In this section we present the known families of partial geometry designs and using one of these families demonstrate that some very good LDPC codes can be constructed.

The known constructions for proper partial geometries are given in Table 5.3. We present examples of the construction method for the first two classes in the table in the Appendix A.2. Table 5.4 shows the parameters of some LDPC codes from proper partial geometries.

5.3.1 Simulation results for proper partial geometry LDPC codes

In this section we compare the performance of LDPC codes from partial geometries with that of randomly constructed codes on the AWGN channel using sum-product de-

Construction	Parameter choice	(s, t, α)	Reference
Thas 1	$q = 2^h, d = 2^m, m < h$	$(q - d, q - \frac{q}{d}, q - \frac{q}{d} - d + 1)$	[107]
Thas 2	$q = 2^h, d = 2^m, m < h$	$(q - 1, (q + 1)(d - 1), d - 1)$	[107]
De Clerk et al.	$n \in \mathbb{Z}$	$(2^{2n-1} - 1, 2^{2n-1}, 2^{2n-2})$	[31]
Mathon	$q = 3^m, m \in \mathbb{Z}$	$(q - 1, 0.5(q^2 - 1), 0.5(q - 1))$	[76]
sporadic S1	-	(5,5,2)	[113]
sporadic S2	-	(4,17,2)	[47]

Table 5.3: Known constructions for proper partial geometry designs

coding. The simulation setup, and random code construction is the same as is outlined in Section 4.4.1.

Figs. 5.5–5.10 show the performance of LDPC codes derived from the proper partial geometries $\text{pg}(4,6,3)$, $\text{pg}(6,4,3)$, $\text{pg}(12,12,9)$, $\text{pg}(8,14,7)$ and $\text{pg}(24,28,21)$, compared to that of randomly constructed LDPC codes with the same rate and length but with column weight 3.

The $[63, 35, 10]$ LDPC code from the $\text{pg}(4,6,3)$ significantly outperforms the equivalent length and rate random code due to its good minimum distance and many linearly dependent rows, Fig. 5.1. The $\text{pg}(4,6,3)$ code has a column weight of 5 and so will have a slightly higher decoding complexity per iteration than the random code. However, as seen in Fig. 5.2, this is somewhat mitigated by the faster convergence of the decoding algorithm for the $\text{pg}(4,6,3)$ code. The dual, $\text{pg}(6,4,3)$ design, gives a $[45, 17, 10]$ code. Even though v is larger than b there are sufficiently many linearly dependent rows in the incidence matrix of the geometry that the resulting code has a reasonable rate. The large portion of linearly dependent rows in this code allows it to significantly outperform the equivalent length and rate random LDPC code as shown in Figs. 5.3 and 5.4. For this code the union bound is also shown, demonstrating that, as for the STS design, the algebraic structure of the partial geometry design does give a “better” code than the random code, in terms of maximum likelihood decoding performance, but at the same time this structure is carefully chosen to also give a better sum-product decoding performance.

The $[221, 139, \geq 14]$ LDPC code from the $\text{pg}(12,12,9)$ design significantly outperforms the equivalent length and rate random code due to its good minimum distance, 14, and many (139) linearly dependent rows. The $\text{pg}(12,12,9)$ code has a column weight of 13 and so will have a higher decoding complexity per iteration than the random code. However, as seen in Fig. 5.6, this is again somewhat mitigated by the faster convergence of the decoding algorithm for the $\text{pg}(12,12,9)$ code.

(s, t, α)	v	n	# linearly dept. rows	$[n, k, d]$	Source
(6,4,3)	63	45	35	[45, 17, 10]	Thas 1
(4,6,3)	45	63	17	[63, 35, 10]	Thas 1
(5,5,2)	81	81	≥ 30	[81, 30 – 31, ≥ 9]	S1
(7,8,4)	120	135	≥ 35	[135, 50 – 134, ≥ 9]	DeClerk et al.
(12,12,9)	221	221	139	[221, 139, ≥ 14]	Thas 1
(8,14,7)	153	255	71	[255, 173, ≥ 10]	Thas 1
(4,17,2)	175	630	≥ 21	[630, ≥ 476 , ≥ 6]	S2
(24,28,21)	825	957	581	[957, 713, ≥ 26]	Thas 1
(16,30,15)	561	1023	≥ 33	[1023, ≥ 495 , ≥ 18]	Thas 1
(26,27,18)	1080	1120	≥ 260	[1120, ≥ 300 , ≥ 28]	[108]
(7,27,3)	512	1792	≥ 70	[1792, ≥ 1350 , ≥ 9]	Thas 2
(31,32,16)	2016	2079	≥ 651	[1120, ≥ 714 , ≥ 33]	DeClerk et al.
(8,40,4)	729	3321	72 – 73	[3321, ≈ 2665 , ≥ 10]	Mathon
(15,51,3)	4096	13312	≥ 780	[13312, ≥ 9996 , ≥ 17]	Thas 2

Table 5.4: LDPC codes from proper partial geometries.

The $[255, 173, \geq 10]$ LDPC code from the $\text{pg}(8,14,7)$ also outperforms the equivalent length and rate random LDPC codes. However for this code the minimum distance, 10, and number of linearly dependent rows, 71, is slightly less and so the performance gain is not as marked as for the $\text{pg}(12,12,9)$ code. Also shown in Fig. 5.7 is the Euclidean geometry code which is the same length and (roughly) the same rate as the $\text{pg}(8,14,7)$ code. We can see that the EG code slightly outperforms the $\text{pg}(8,14,7)$ code. However the EG code does have twice as many non zero entries in its parity-check matrix and so its performance gain is accompanied by an increase in decoding complexity, Fig. 5.8. Note that in this figure the EG and oval codes appear not to have as large an increase in encoding complexity over the column weight 3 random code as would be expected by their large column weight. This is due to a faster convergence for these codes and hence a smaller average number of decoding iterations. If the number of floating point computations was instead measured for some fixed number of iterations the complexity would be linearly proportional to the column weight and a much larger difference in decoding complexity would be apparent in the figures. At very low signal-to-noise ratios, where the maximum number of iterations are used in most cases, the number of floating point operations required are exactly proportional to the column weights, with the number

of operations used to decode the EG code twice that required for the partial geometry code and over five times that required for the random code.

Lastly, Figs. 5.9 and 5.10 show the performance of the $[957, 713, \geq 26]$ LDPC code from the $pg(24, 28, 21)$ design. Due to the large column weight of this code a performance gain is only realised over the randomly constructed column weight 3 codes for very high signal-to-noise ratios. The increased density of H for the proper partial geometry code, $25/957$ as compared to $3/957$ for the random code, also gives a greater concentration of short cycles in the code outweighing the positive effects of the extra linearly dependent rows. Like the LDPC codes from Steiner 2-designs, the density of the LDPC parity-check matrix dominates the performance of the partial geometry codes at low signal-to-noise ratios with other properties such as minimum distance and girth playing a greater role as the signal-to-noise ratio is increased.

We wish then to consider partial geometry codes with small column weights which still involve linearly dependent rows in H . This leads us to focus on those partial geometries with column weights of 3, for which there is an infinite class in the transversal designs with $s = 2$.

5.4 Transversal designs

A transversal design is a partial geometry with $\alpha = s$ but like Steiner 2-designs they are also defined independently. We present this definition and their construction before deriving an alternative construction for transversal designs which are anti-Pasch.

A transversal design of order q , block size k , and index λ , denoted $TD_\lambda(k, q)$ is a triple $(\mathcal{V}, \mathcal{G}, \mathcal{B})$, where [3]

- T1.** \mathcal{V} is a set of kq elements,
- T2.** \mathcal{G} is a partition of \mathcal{V} into k classes (called groups) each of size q ,
- T3.** \mathcal{B} is a collection of k -subsets of \mathcal{V} (called blocks), and
- T4.** every unordered pair of elements from \mathcal{V} is either contained in exactly one group, or is contained in exactly λ blocks, but not both.

Lemma 5.4.1 *The blocks of the transversal design $TD_1(k, q)$ are the blocks of a $pg(r, k - 1, q - 1)$.*

Proof. By definition the blocks of the TD are size $s + 1 = k$. Next a point P must appear in a block with $(k - 1)q$ other points, the points it does not appear with in a group.

There are $k - 1$ points in every block with P , none of which can appear in two blocks (or P will appear in two blocks with the same point which is not allowed by **T4**), thus P is incident in $t + 1 = q$ blocks. It remains then to prove that each point not in a block B is incident in $s - 1$ blocks incident with B . Consider a block B in the transversal design. Firstly each point in B must come from a different group of the transversal design, if not the two points from the same group will appear in both a group and a block together and **T4** will not hold. Now consider any point P of the transversal design not in B . The point P will not appear in a block with the point of B in the same group as it (by **T4**) but (again by **T4**) it must appear in a block with every other point in B . Each of these blocks must be distinct or a pair of points will appear together in two blocks. Thus there are $s = k - 1$ blocks containing P which are incident with B as required. \square

The construction of transversal designs uses combinatorial structures called Latin squares. Indeed the existence of a $TD_1(s + 1, q)$, written $TD(s + 1, q)$, is equivalent to the existence of $s - 1$ mutually orthogonal Latin squares (MOLS) of order q [3]. The exact number of mutually orthogonal squares that exist for a given order q has not yet been resolved, however constructions exist for $s + 1$ MOLS of order q , for q a prime power and $s + 1 \leq q + 1$ [11]. First let

$$GF(q) = \{\phi_1, \phi_2, \dots, \phi_{q-1}, \phi_q\},$$

and define $q - 1$ MOLS by the $q \times q$ matrices M_1, \dots, M_{q-1} , taking the (i, j) th entry of M_l to be

$$M_l(i, j) = \phi_i \phi_l + \phi_j.$$

Then to construct a transversal design $TD(s + 1, q)$ [3], take a set of $s - 1$ mutually orthogonal squares of order q and construct a $(s + 1) \times q^2$ array O with one column for each of the positions (i, j) in the Latin squares. The first two rows of O label positions in the Latin squares, the first row giving the row number and the second the column number. In the third row are placed the corresponding entries of the first Latin square and so on, so that the l th row of O contains the entries of the $(l - 2)$ th Latin square. The result is that any two rows of O give in their vertical pairs, each ordered pair of points exactly once. Now add $(l - 1)q$ to each entry on the l th row of O and the columns of O are the blocks of the transversal design.

Example 5.4.1 *The $TD(4, 4)$ design requires two MOLS of order 4. With $GF(4) = \{(0, 0), (1, 0), (0, 1), (1, 1)\}$ written as $GF(4) = \{0, 1, 2, -\infty\}$, these MOLS are:*

$$M_1 = \begin{bmatrix} -\infty & 2 & 1 & 0 \\ 2 & -\infty & 0 & 1 \\ 1 & 0 & -\infty & 2 \\ 0 & 1 & 2 & -\infty \end{bmatrix}, \quad M_2 = \begin{bmatrix} 2 & -\infty & 0 & 1 \\ 1 & 0 & -\infty & 2 \\ -\infty & 2 & 1 & 0 \\ 0 & 1 & 2 & -\infty \end{bmatrix}.$$

Replacing the elements $\{0, 1, 2, -\infty\}$ with $\{1, 2, 3, 4\}$ gives the Latin squares:

$$M_1 = \begin{bmatrix} 4 & 3 & 2 & 1 \\ 3 & 4 & 1 & 2 \\ 2 & 1 & 4 & 3 \\ 1 & 2 & 3 & 4 \end{bmatrix}, \quad M_2 = \begin{bmatrix} 3 & 4 & 1 & 2 \\ 2 & 1 & 4 & 3 \\ 4 & 3 & 2 & 1 \\ 1 & 2 & 3 & 4 \end{bmatrix}.$$

The orthogonal array is then:

$$O = \begin{bmatrix} 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 3 & 3 & 3 & 3 & 4 & 4 & 4 & 4 \\ 1 & 2 & 3 & 4 & 1 & 2 & 3 & 4 & 1 & 2 & 3 & 4 & 1 & 2 & 3 & 4 \\ 4 & 3 & 2 & 1 & 3 & 4 & 1 & 2 & 2 & 1 & 4 & 3 & 1 & 2 & 3 & 4 \\ 3 & 4 & 1 & 2 & 2 & 1 & 4 & 3 & 4 & 3 & 2 & 1 & 1 & 2 & 3 & 4 \end{bmatrix},$$

and the blocks of the transversal design are given by the columns of

$$TD = \begin{bmatrix} 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 3 & 3 & 3 & 3 & 4 & 4 & 4 & 4 \\ 5 & 6 & 7 & 8 & 5 & 6 & 7 & 8 & 5 & 6 & 7 & 8 & 5 & 6 & 7 & 8 \\ 12 & 11 & 10 & 9 & 11 & 12 & 9 & 10 & 10 & 9 & 12 & 11 & 9 & 10 & 11 & 12 \\ 15 & 16 & 13 & 14 & 14 & 13 & 16 & 15 & 16 & 15 & 14 & 13 & 13 & 14 & 15 & 16 \end{bmatrix}.$$

From Section 5.2 we know that the LDPC codes with parity-check matrix the incidence matrix of a transversal design $TD(k, q)$ have at least s linearly dependent rows, point graphs free of 4-cycles, and a minimum distance of at least $s + 2$. Thus the LDPC codes derived from transversal designs, TD LDPC codes, have the following parameters:

Length:	$n = (t + 1)(st + s)/s = (t + 1)^2$
Number of parity bits:	$n - k \leq \frac{(s+1)(st+1)-1}{s}$
Rate:	$R \geq \frac{t(t-s+1)}{(t+1)^2}$
Minimum distance:	$d \geq s + 2$
Row weight of the parity-check matrix:	$t + 1$
Column weight of the parity-check matrix:	$s + 1$
Density:	$\frac{s+1}{(t+1)^2}$

For the special case of the column weight three designs, $s = 2$, we present a modification of the construction method in order to construct anti-Pasch transversal designs. In Section 3.3.2 we developed the role of Pasch configurations in Steiner triple systems and showed that by avoiding such configurations the minimum distance of the STS codes can be increased from 4 to 6 and the minimum stopping set size increased to 5. The same is true for the transversal designs.

Lemma 5.4.2 *Column weight 3 TD LDPC codes with no weight 4 codewords can be constructed from TD designs without Pasch configurations. These codes have a minimum distance of at least 6.*

Proof. The proof follows the line of reasoning of Lemma 3.3.1. The set of 4-line configurations which are possible in the blocks of a $TD(3, q^2)$ are those in Fig 3.5, the same as for the STS designs. This is because the same factors are limiting the arrangement of blocks in both STS and $TD(3, q)$ designs: three points per block, and no pair of points in more than one block together. Thus the only arrangement of 4 blocks in a transversal design which will result in a weight 4 codeword are the blocks of a Pasch configuration. Finally, five columns of an TD incidence matrix will contain 15 non-zero entries, and there is no way to divide 15 entries amongst a set of points so that each contains an even number. \square

Before presenting this construction for transversal designs we outline how a Pasch configuration can occur in a transversal design.

Lemma 5.4.3 *A Pasch configuration will occur in a $TD(3, q)$ constructed from a $q \times q$ Latin square L , if and only if a pair of columns in L contains any pair of points (a, b) in one row and the pair (b, a) in another.*

Proof. The columns of the transversal design are divided into q subsets of q columns such that each subset of q columns contain the same first point. Thus the columns of a Pasch configuration, if one exists, must come one pair from each of two subsets, say subsets i and j , so that two points in the set $1, 2, \dots, q$ both occur twice in the chosen blocks. Next, the second point of each block occurs in the same order in each subset, that is $q + 1, q + 2, \dots, 2q$ and so we pick blocks from the same two positions in each subset, say positions x and y so that two points in the set $q + 1, q + 2, \dots, 2q$ both occur twice in the chosen blocks. Thus if a Pasch configuration exists it will occur in the four blocks corresponding to the $((i - 1)q + x)$ th, $((i - 1)q + y)$ th, $((j - 1)q + x)$ th and $((j - 1)q + y)$ th blocks of the transversal design for any i, j, x , and $y \in \{1 \cdots q\}$. Now for the four blocks chosen above to be a Pasch configuration the final points in each block must contain the same pair of points twice. Since the final row is made up of the entries of the Latin square the four entries we are interested in are the elements in M which are the intersection of the columns x and y with the rows i and j . Thus a Pasch configuration will occur if and only if a pair of columns in the Latin square contains a pair of points (a, b) in one row and the pair (b, a) in another. \square

Example 5.4.2 *The pairs $(4, 3)$ and $(3, 4)$ occur in the first two columns of the Latin*

square M_1 of Example 5.4.1. Thus the transversal design constructed using M_1 :

$$TD = \begin{bmatrix} \mathbf{1} & \mathbf{1} & 1 & 1 & \mathbf{2} & \mathbf{2} & 2 & 2 & 3 & 3 & 3 & 3 & 4 & 4 & 4 & 4 \\ \mathbf{5} & \mathbf{6} & 7 & 8 & \mathbf{5} & \mathbf{6} & 7 & 8 & 5 & 6 & 7 & 8 & 5 & 6 & 7 & 8 \\ \mathbf{12} & \mathbf{11} & 10 & 9 & \mathbf{11} & \mathbf{12} & 9 & 10 & 10 & 9 & 12 & 11 & 9 & 10 & 11 & 12 \end{bmatrix}$$

has a Pasch configuration consisting of the 1st, 2nd 5th and 6th blocks through the points 1, 2, 5, 6, 12, and 11.

Construction 5.4.1 For an anti-Pasch $TD(3,q)$, q any positive integer, construct the $q \times q$ Latin square L such that:

$$L(i, j) = j + i - 1 \pmod{q}.$$

The orthogonal array is constructed as above with the first two rows of O labelling the positions in the Latin square and the third row holding the corresponding entries of L . Similarly, the transversal design is constructed from the orthogonal array by adding $(l-1)q$ to each entry on the l th row.

Lemma 5.4.4 Construction 5.4.1 produces transversal designs free of Pasch configurations.

Proof. As each row of the Latin square produced by Construction 5.4.1 is a cyclic shift of the first it is not possible for any pairs of points (a, b) and (b, a) to occur in the same pair of columns and so by Lemma 5.4.3 the result follows. \square

Example 5.4.3 The Latin square on 4 points produced by Construction 5.4.1 is

$$L = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \\ 3 & 4 & 1 & 2 \\ 4 & 1 & 2 & 3 \end{bmatrix},$$

and so the $TD(3,16)$ constructed with L ,

$$TD = \begin{bmatrix} 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 3 & 3 & 3 & 3 & 4 & 4 & 4 & 4 \\ 5 & 6 & 7 & 8 & 5 & 6 & 7 & 8 & 5 & 6 & 7 & 8 & 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 & 10 & 11 & 12 & 9 & 11 & 12 & 9 & 10 & 12 & 9 & 10 & 11 \end{bmatrix},$$

is anti-Pasch.

The lower bound on minimum distance for our new transversal designs is unfortunately tight. To prove this we show that each transversal design contains a 6-line configuration with all point degrees two. Such a configuration will give a weight 6 codeword in the code as in Lemma 3.2.4.

Lemma 5.4.5 *The transversal designs of Construction 5.4.1 with $q \geq 3$ contain 6-line configurations with all points of degree two.*

Proof. The proof is by construction. To find a 6-line configuration in a $\text{TD}(3, q)$ from Construction 5.4.1 we take the columns 1, 2, $q + 2$, $2q$, $2q + 1$ and $3q$. Since each set of q columns contain the same first point, columns 1 and 2 both contain point 1, columns $q + 2$ and $2q$ both contain point 2 and columns $2q + 1$ and $3q$ both contain point 3. Secondly, since each set of q columns contain the points $q + 1$ to $2q$ in that order, columns 1 and $q + 1$ both contain the point $q + 1$, columns 2 and $q + 2$ both contain the point $q + 2$, and columns $2q$ and $3q$ both contain the point $2q$. Lastly, the third points in each column are ordered from $2q + 1, 2q + 2, \dots, 3q$ in the first set of q columns, from $2q + 2, \dots, 3q, 2q + 1$ in the second set of q columns and so on. Thus columns 1 and $2q$ both contain point $2q + 1$, columns 2 and $3q$ both contain point $2q + 2$ and columns $q + 2$ and $2q + 1$ both contain the point $2q + 3$. Thus in these six columns there are nine points, 1, 2, 3, $q + 1$, $q + 2$, $2q$, $2q + 1$, $2q + 2$, and $2q + 3$ each occurring twice. \square

5.4.1 Simulation results for TD LDPC codes

In this section the performance of the TD LDPC codes on the AWGN channel, when decoded using the sum-product decoding algorithm [71], are compared to that of randomly generated codes of the same rate and length. The simulation setup, and random code construction is the same as detailed in Section 4.4.1.

Figs. 5.11–5.12 show the performance of a $(3, 16)$ -regular LDPC code from the transversal design $\text{pg}(2, 15, 2)$, compared with a randomly constructed code of the same rate and length. Also shown is the same length EG code compared to an equivalent length and rate random LDPC code. We can see that as for the EG LDPC code the TD LDPC code offers a significant decoding performance improvement over the same rate random code, but unlike the EG code, does this with a decrease rather than an increase in decoding complexity over the random code. The TD LDPC code thus offers a significant error correction improvement over an equivalent length random LDPC code without increasing the decoding complexity and also offers a much higher rate algebraic LDPC code than the existing EG and PG codes of the same length.

Figs. 5.13 and 5.14 show the performance of a regular LDPC code derived from the $\text{pg}(2, 24, 2)$ designs, compared with randomly constructed codes of the same rate and

length. Figs. 5.15 and 5.16 show the performance of a regular LDPC code derived from the $\text{pg}(2, 36, 2)$ design, compared with randomly constructed codes of the same rate and length. Finally, Figs. 5.17 and 5.18 show the performance of a regular LDPC code derived from the $\text{pg}(2, 47, 2)$ design, compared with randomly constructed codes of the same rate and length. We see that with transversal designs we can achieve high rate codes which significantly outperform randomly constructed LDPC codes and do this with a reduction in decoding complexity.

5.5 Discussion

The partial geometries provide a generalization to a large class of algebraic codes which contain many of the existing algebraic LDPC codes as special cases. The structure of geometries allowed us to present lower bounds for the minimum distance, minimum stopping set size and code rate and an expression for the exact number of the minimum weight cycles in all of these codes. The examination of partial geometries was motivated by the search for a wider range of algebraic LDPC codes and a greater amount of flexibility in choosing the code properties. Since each pair of points is contained in at most one block we can, by adjusting s , t and α , generate a much wider range of algebraic codes, provided that a construction is known for the chosen design.

Overall the performance trends observed in the LDPC codes from Steiner 2-designs are mirrored in the LDPC codes from partial geometries. Choosing partial geometry designs with large column weight and many linearly dependent parity-check equations results in excellent decoding performance for the smaller codes but for the larger codes the increased density of H , with increasing column weight, hinders their performance in channels with low signal-to-noise ratios.

Again, codes with column weight 3 represent the best performing codes over a wide range of code lengths and in this chapter we have presented a construction for column weight 3 codes which are both anti-Pasch and contain linearly dependent rows in H . In the anti-Pasch transversal designs we achieve LDPC codes which are not only deterministic but which significantly outperform the traditional randomly constructed LDPC codes of the same rate and length and do this with a reduced decoding complexity.

Overall, by relaxing the intersection requirements of Steiner 2-designs, to our requirements that a pair of points occur in *at most* one line together, a whole new set of LDPC codes have been constructed. However, although some very good LDPC codes have been produced, the limited range of proper partial geometries available hampered our flexibility in choosing α and hence varying the code connectivity. In the following chapter we increase our range of LDPC codes by relaxing even further the incidence requirements of bits and checks in our code, while still requiring that it be regular and free of 4-cycles.

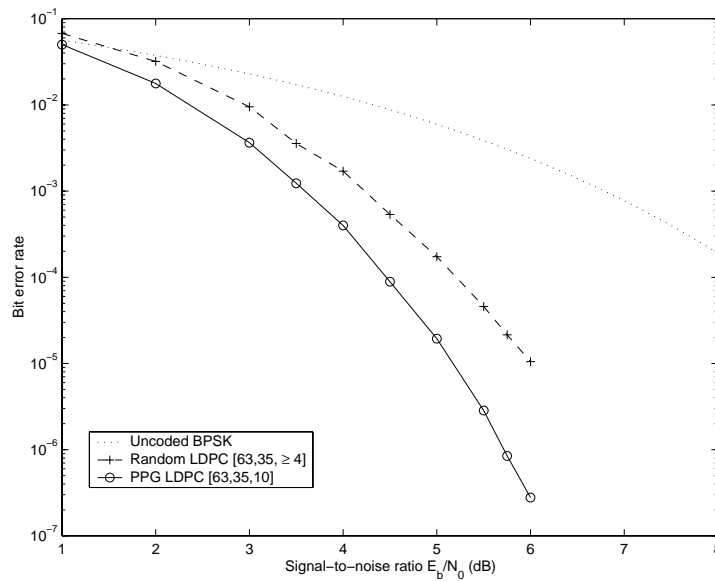


Figure 5.1: The decoding performance of a length-63 proper partial geometry code on an AWGN channel using sum-product decoding with a maximum of 10 iterations.

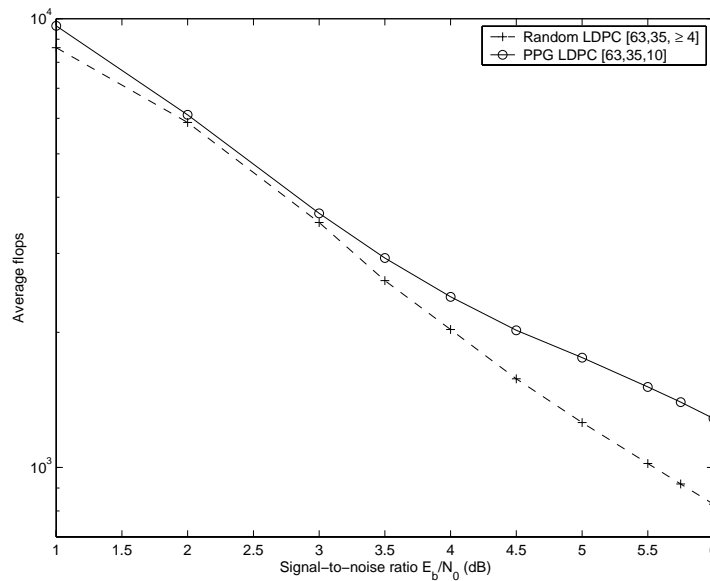


Figure 5.2: The average number of floating point multiplications required to decode a codeword for iteratively decoded low-density parity-check codes on an AWGN channel with a maximum of 10 iterations.

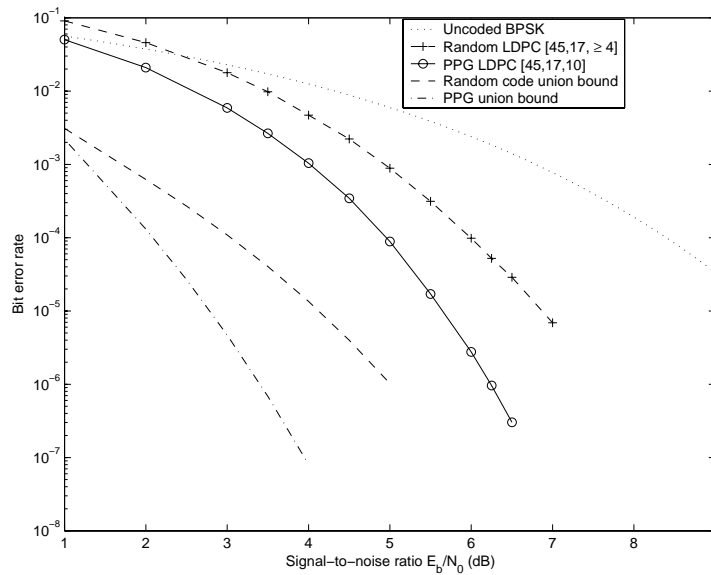


Figure 5.3: The decoding performance of a length 45 proper partial geometry code on an AWGN channel using sum-product decoding with a maximum of 10 iterations. Also shown is the union bound for both codes

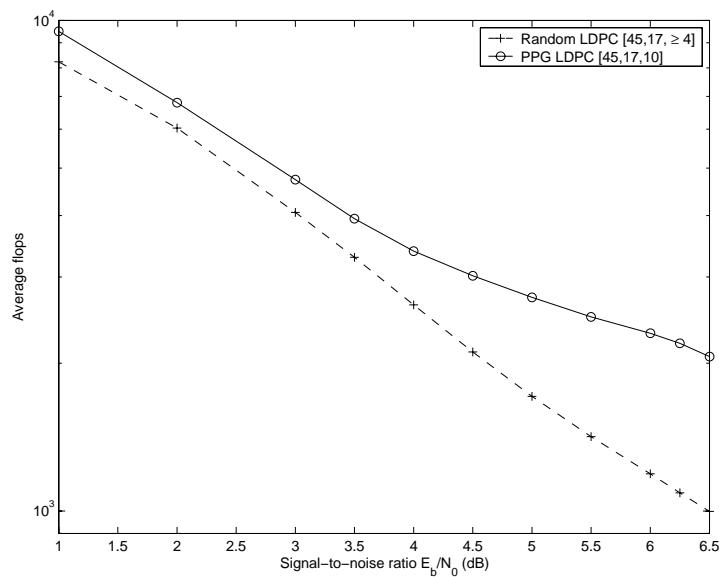


Figure 5.4: The average number of floating point multiplications required to decode a codeword for iteratively decoded low-density parity-check codes on an AWGN channel with a maximum of 10 iterations.

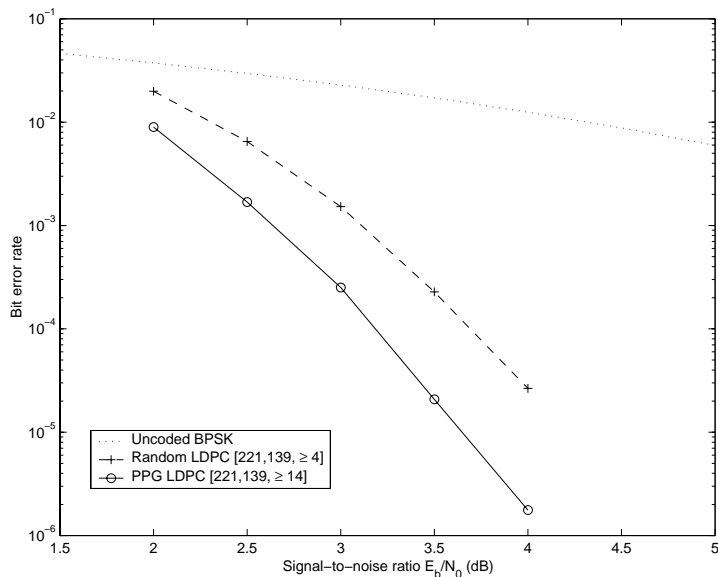


Figure 5.5: The decoding performance of a length-221 proper partial geometry code on an AWGN channel using sum-product decoding with a maximum of 200 iterations.

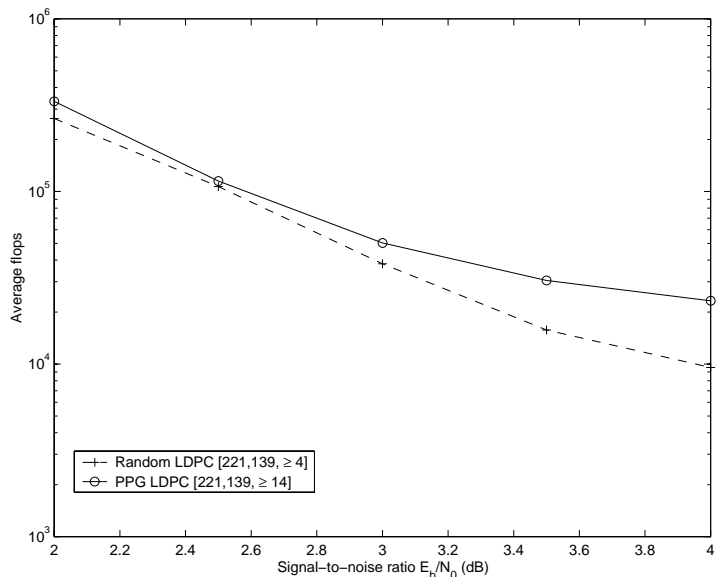


Figure 5.6: The average number of floating point multiplications required to decode a codeword for iteratively decoded low-density parity-check codes on an AWGN channel with a maximum of 200 iterations.

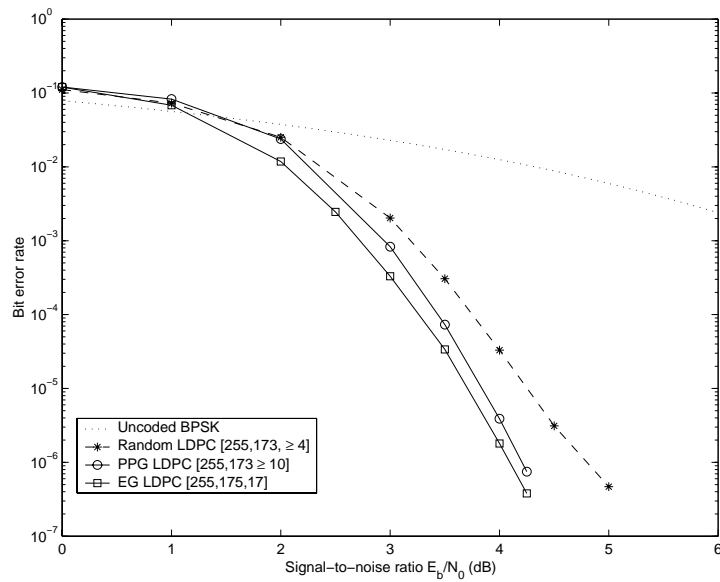


Figure 5.7: The decoding performance of length-255 proper partial geometry and EG codes on an AWGN channel using sum-product decoding with a maximum of 200 iterations.

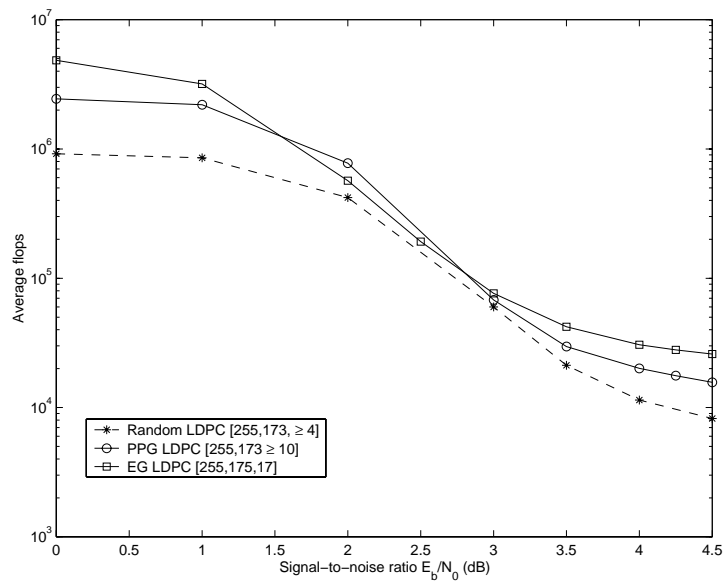


Figure 5.8: The average number of floating point multiplications required to decode a codeword for iteratively decoded low-density parity-check codes on an AWGN channel with a maximum of 200 iterations.

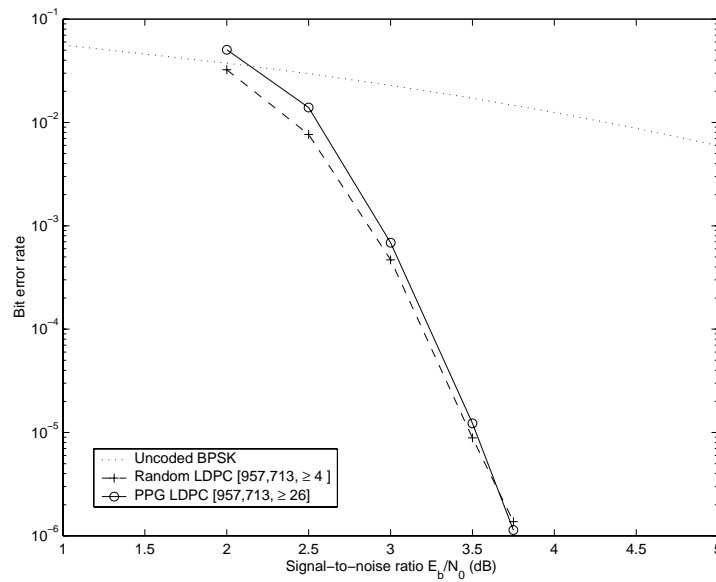


Figure 5.9: The decoding performance of a length-957 proper partial geometry code on an AWGN channel using sum-product decoding with a maximum of 1000 iterations.

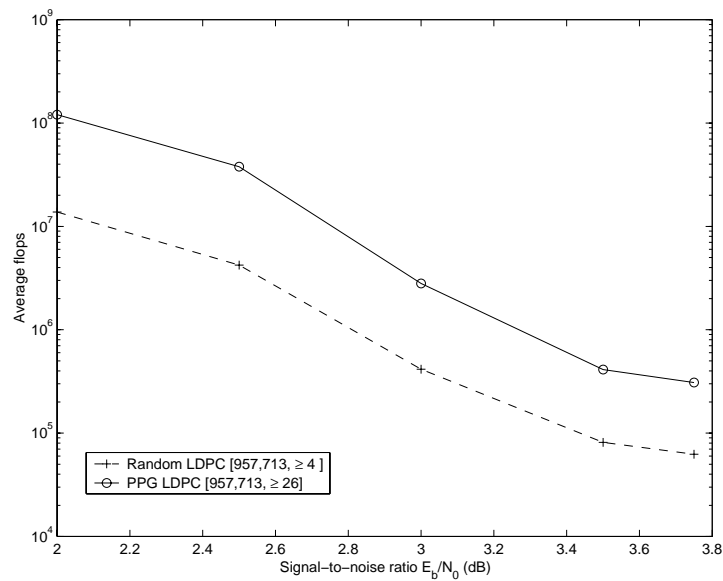


Figure 5.10: The average number of floating point multiplications required to decode a codeword for iteratively decoded low-density parity-check codes on an AWGN channel with a maximum of 1000 iterations.

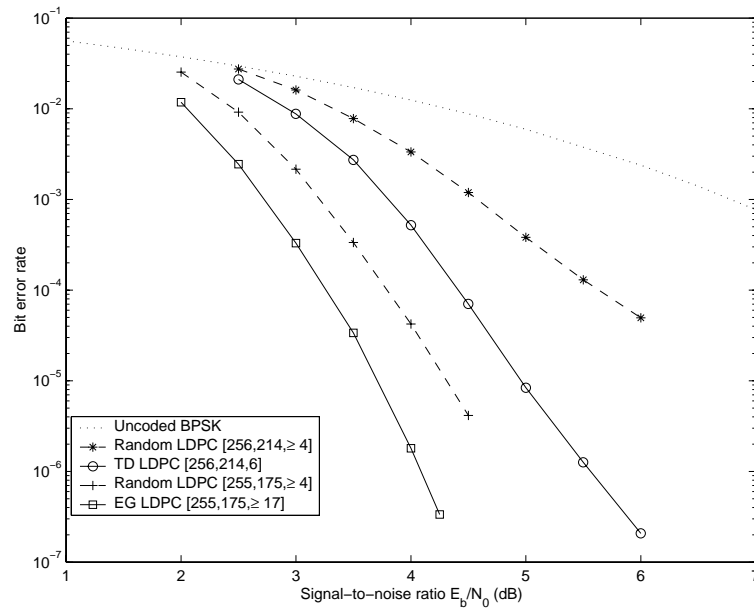


Figure 5.11: The decoding performance of length-256 LDPC codes on an AWGN channel using sum-product decoding with a maximum of 200 iterations.

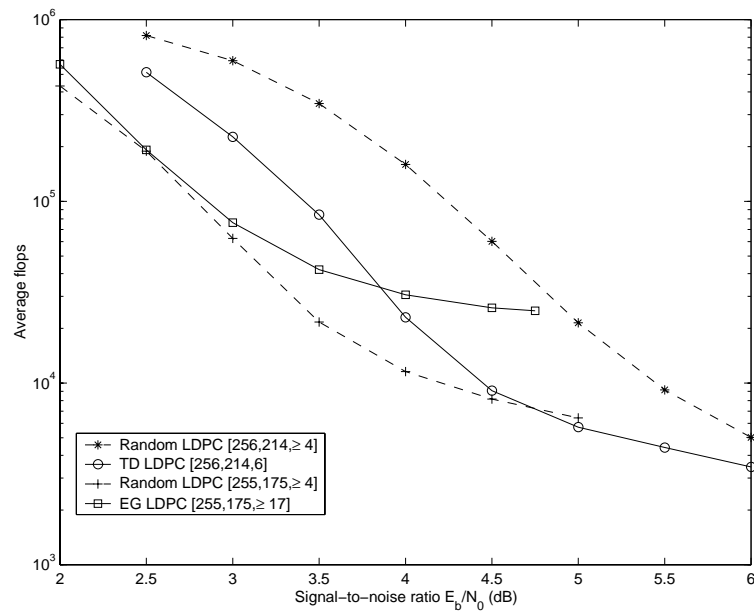


Figure 5.12: The average number of floating point multiplications required to decode a codeword for iteratively decoded low-density parity-check codes on an AWGN channel with a maximum of 200 iterations.

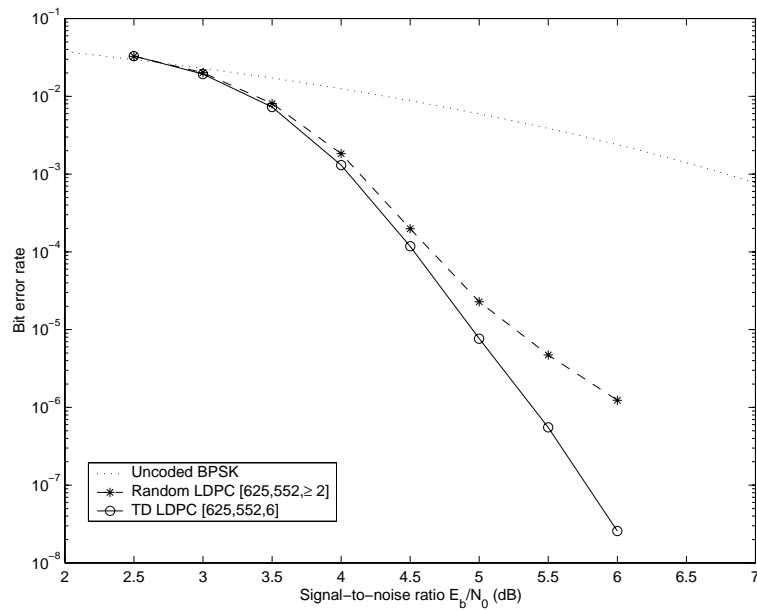


Figure 5.13: The decoding performance of LDPC codes on an AWGN channel using sum-product decoding with a maximum of 200 iterations

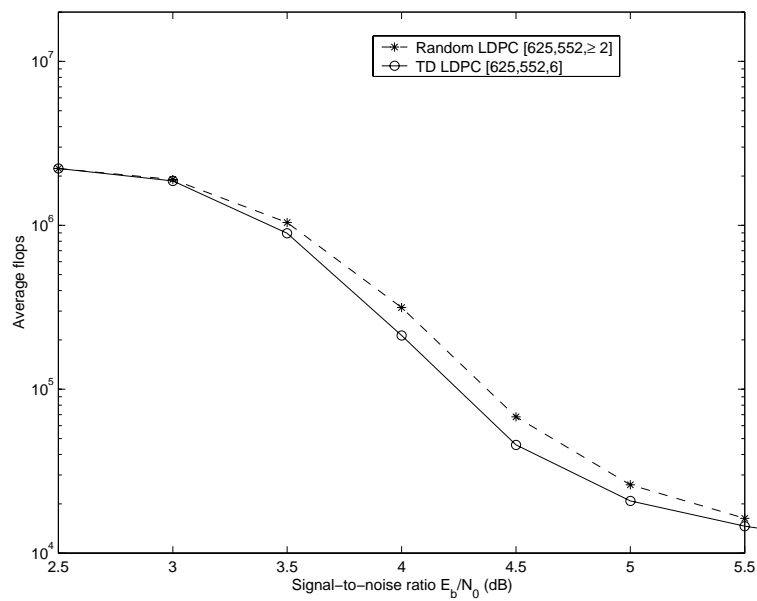


Figure 5.14: The average number of floating point multiplications required to decode a codeword for iteratively decoded low-density parity-check codes on an AWGN channel with a maximum of 200 iterations.

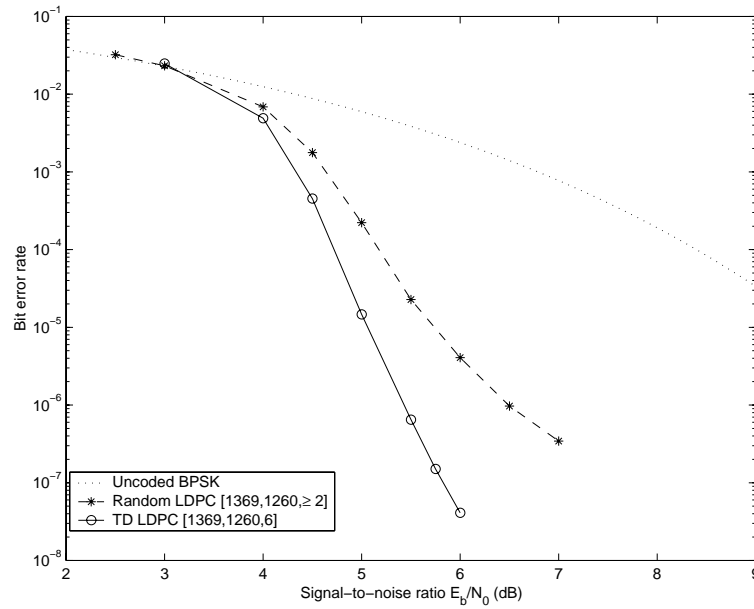


Figure 5.15: The decoding performance of length-1369 LDPC codes on an AWGN channel using sum-product decoding with a maximum of 1000 iterations

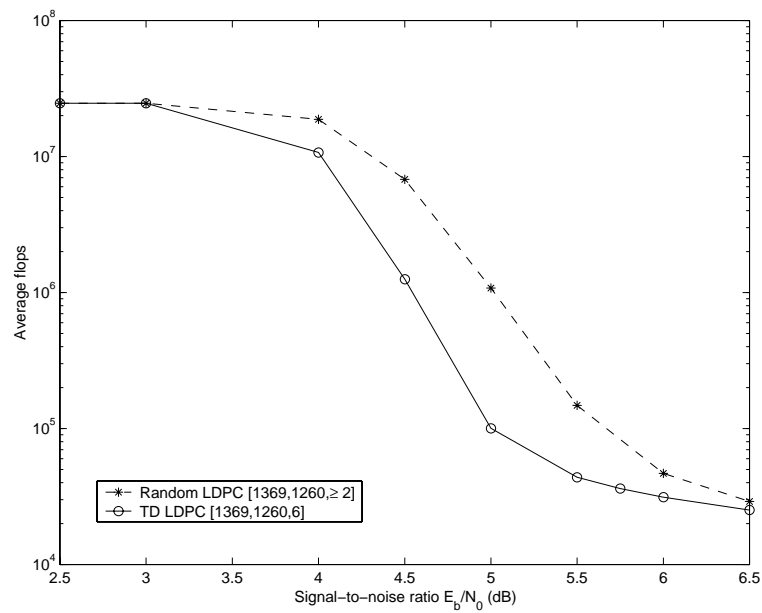


Figure 5.16: The average number of floating point multiplications required to decode a codeword for iteratively decoded low-density parity-check codes on an AWGN channel with a maximum of 1000 iterations.

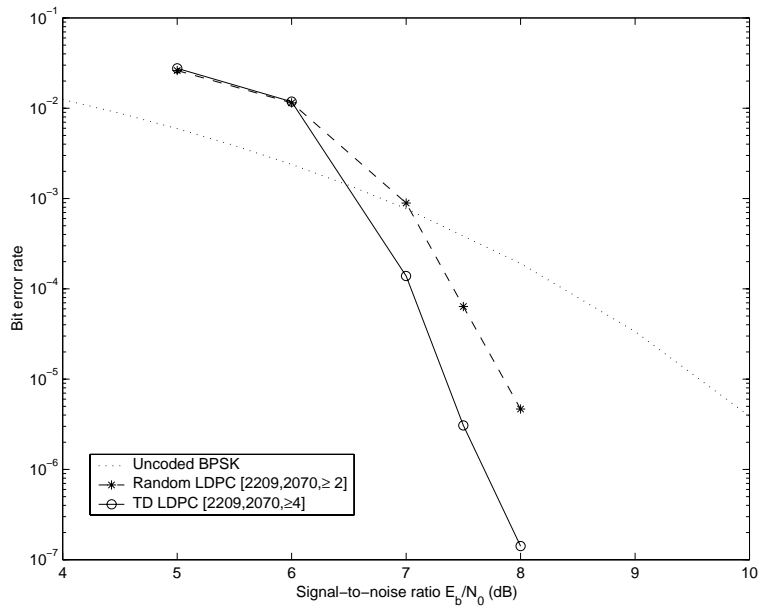


Figure 5.17: The decoding performance of length-2209 LDPC codes on an AWGN channel using sum-product decoding with a maximum of 1000 iterations

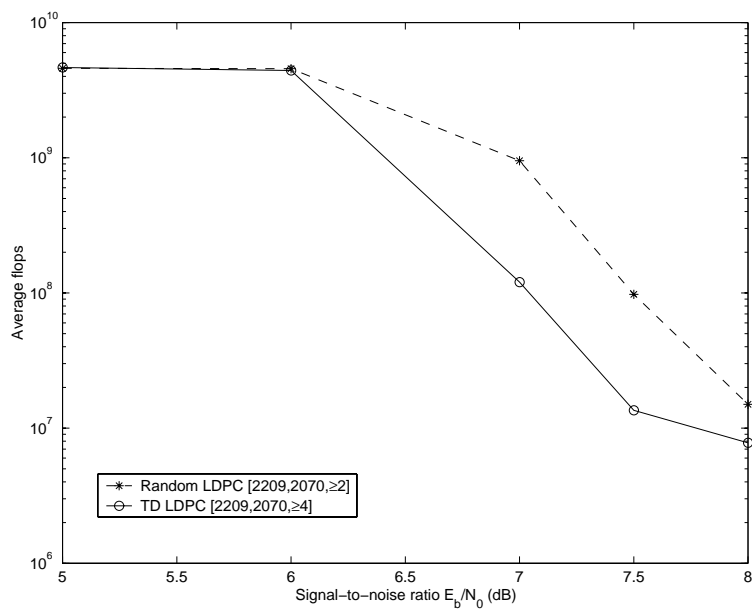


Figure 5.18: The average number of floating point multiplications required to decode a codeword for iteratively decoded low-density parity-check codes on an AWGN channel with a maximum of 1000 iterations.

CODE DESIGN USING RESOLVABILITY

In this chapter resolvable designs are employed to create LDPC codes with the parameter flexibility and decoding performance of the randomly constructed codes. By employing resolvability we can greatly extend the class of low-density parity-check codes that can be algebraically constructed. The resulting codes are $(3, \rho)$ -regular or $(4, \rho)$ -regular with Tanner graphs free of 4-cycles, for any value of ρ and for a flexible choice of code lengths. Further, cyclically resolvable cyclic Steiner 2-designs can be used to construct LDPC codes with simple encoding circuits.

6.1 Introduction

Our aim in this chapter is to design LDPC codes with the performance and flexibility of pseudo-random constructions, such as those of Gallager and MacKay and Neal, but with deterministic properties and construction. We saw in Chapters 3, 4 and 5 that Steiner 2-designs and partial geometries yield good LDPC codes which are regular and free of 4-cycles. However, the codes from these designs are high rate and there are only a limited range of code parameters for which they are available.

In particular, the LDPC codes from combinatorial designs with column weight 3, the Steiner triple systems and transversal designs, perform very well with iterative decoding. However for the STS LDPC codes (3.3)

$$n = \frac{v(v-1)}{6}, \quad \text{and} \quad R \geq 1 - \frac{6}{(v-1)},$$

while for the transversal designs

$$n = (t+1)^2, \quad \text{and} \quad R \geq 1 - \frac{3(t+1) - 2}{(t+1)^2}.$$

So as larger STS and TD designs are used the codes quickly become high rate. For STS LDPC codes longer than 150 the rate is already at least 0.8 (see Fig. 3.3).

A simplistic approach to obtaining a lower rate code is to choose a design with more blocks than the required code length and then to remove some columns of N . For each

pair of points in the omitted column the corresponding incidence is now zero and the matrix thus formed is no longer the incidence matrix of a 2-design. However, 4-cycles will still be avoided in the Tanner graph of the resulting code as removing columns (bit nodes) can not add cycles. Unfortunately, randomly removing columns from N results in a parity-check matrix with variable row weights, and can lead to rows with all entries zero, which is exactly the problem of the random constructions that we are trying to avoid. To retain regular codes we would like to be able to remove a group of columns of N in such a way that we reduce by one the weight of every row in the matrix. For this we propose in this chapter the use of designs which have the property of *resolvability*.

The concept of resolvability was introduced by Rev. Kirkman when he posed and solved the following problem:

Fifteen young ladies in a school walk out three abreast for seven days in succession: it is required to arrange them daily, so that no two shall walk twice abreast.

—Rev. T.P. Kirkman, *Lady's and Gentleman's Diary*, 1847.

If we think of girls as points and each column of three girls as a block, the solution to Kirkman's problem is a 2-(15, 3, 1) design. There are $v = 15$ girls, $\gamma = 3$ girls in each column, each pair of girls must appear together in a column once ($\lambda = 1$), each girl appears in $r = 7$ columns, one for each day of the week, and there are $b = 35$ columns of girls, five on each day of the week.

The extra property required of the solution to Kirkman's problem, which is not guaranteed by the structure of a Steiner triple system, is that every set of five blocks that constitute a resolution class must contain each point exactly once, since each girl must appear in precisely one of the columns each day. A design with this property is resolvable. Fig. 6.1 shows the arrangement of Kirkman's schoolgirls for four of the days of the week.

More formally, a design is resolvable if the blocks of the design can be arranged into r groups, called resolution classes, such that the v/γ blocks of each resolution class are disjoint, and each class contains every point precisely once. Steiner triple systems which are resolvable are called *Kirkman triple systems* (KTS).

Kirkman's problem for 15 schoolgirls can be asked of other numbers of v girls, where v is the number of points in a Steiner triple system. However, for the design to be resolvable there must be r groups of $v/3$ blocks, since there are $v/3$ blocks in each of r resolution classes, and so v must satisfy

$$v \equiv 0 \pmod{3}.$$

$$\left[\begin{array}{cccccccccccccccc} 1 & . & . & . & . & . & . & 1 & . & . & . & . & 1 & . & . & . & . & 1 & . & . & . & . & 1 & . & . & . & . & 1 \\ . & . & 1 & . & . & . & 1 & . & . & . & . & . & . & . & 1 & . & . & . & . & . & . & . & . & . & . & . & . & . & 1 \\ . & . & . & 1 & . & . & . & . & 1 & . & . & 1 & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & 1 & . \\ . & . & . & . & 1 & . & . & . & . & 1 & . & . & . & . & 1 & . & . & . & 1 & . & . & . & 1 & . & . & . & . & . & . & . \\ . & . & . & . & . & 1 & . & . & . & . & 1 & . & . & . & . & 1 & . & . & . & . & . & . & . & . & . & . & . & . & . & 1 & . \\ . & . & . & . & 1 & . & 1 & . \\ 1 & . & . & . & . & . & . & 1 & . & . & . & . & . & 1 & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & 1 & . & \dots \\ . & . & . & . & . & 1 & . & 1 & . & . & . & . & . & 1 & . & . & . & . & . & . & . & . & 1 & . & . & . & . & . & . & . & . & . & . \\ . & . & . & 1 & . & . & . & . & . & . & . & . & 1 & . & . & . & . & . & . & . & . & . & 1 & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & 1 & . & . & . & . & . & . & . & . & . & . & . & . & 1 & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & 1 & . \\ . & . \\ . & . \\ . & . \\ 1 & . & . & . & . & . & . & 1 & . & . & . & . & . & 1 & . & . & . & . & . & . & . & . & 1 & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \end{array} \right]$$

Figure 6.1: Part of the incident matrix of a Kirkman triple system on 15 points. For clarity only four of the seven resolution classes are shown.

We saw in Chapter 3 that if a 2 -($v, 3, 1$) design exists then v must satisfy

$$v \equiv 1, 3 \pmod{6}.$$

Thus Kirkman triple systems on v points, $\text{KTS}(v)$, are restricted to

$$v \equiv 3 \pmod{6}.$$

It wasn't until 1967 however that Ray-Chaudhuri and Wilson [84] showed this necessary condition is also sufficient so that Kirkman triple systems exist for any number of points $v \equiv 3 \pmod{6}$.

Resolvable designs provide the solution to our problem of lower rate, regular LDPC codes. As in the previous chapters we define the parity-check matrix of an LDPC code by the incidence matrix of a design. The number of blocks in the incidence matrix of a resolvable design, and hence the code length, can be reduced by v/γ blocks at a time while maintaining code regularity by simply removing all of the blocks in a resolution class together.

The following section presents LDPC codes from KTS designs before we focus on particular strategies for linear time encodable $(3, r)$ -regular LDPC codes from carefully chosen KTS designs in Section 6.3. Resolvability is not a property unique to Kirkman triple systems and a number of other families of resolvable Steiner 2-designs can be constructed. We look in particular at resolvable 2 -($v, 4, 1$) designs and resolvable oval designs in Section 6.4. We also demonstrate that many of the transversal designs we constructed in Construction 5.4.1 are resolvable and so produce good $(3, r)$ -regular LDPC codes with linearly dependent parity-checks.

6.2 LDPC codes from Kirkman triple systems

The properties of the codes from KTS designs depend on the particular construction used and so we present in the Appendix A.3 two construction methods for KTS designs. KTS designs are constructed using mixed difference systems, which were introduced in Definition 2.2.2. Briefly, the translates of all s of the k -subsets of a mixed difference system, with $\lambda = 1$, on point set $\mathcal{H} = \mathcal{G} \times Z_t$ ($|\mathcal{G}| = v$), form the blocks of a resolvable 2 - $(tv, sv, sk/t, k, 1)$ design [3].

Like the LDPC codes from STS designs, the incidence matrix of the KTS design is used to construct the parity-check matrix of the KTS code. The difference is that only a fraction of the columns in the incidence matrix are used to define H . As the KTS designs are a special case of the STS designs the results derived for STS designs hold for them also, and the KTS LDPC codes that retain all of the resolution classes have the same (construction independent) properties as all other STS LDPC codes. However retaining only a subset of the resolution classes of the design can change the minimum distance, girth and rank properties of the code.

We will denote by $\text{KTS}(v, \rho)$ an LDPC code constructed using the blocks of ρ resolution classes of a KTS design on v points. The number of resolution classes used in the parity-check matrix, $\rho \in \{4, 5, \dots, (v-1)/2\}$, determines the row weight of H , which is ρ , the code length $n = \frac{\rho v}{3}$, and the rate, $R \approx \frac{\rho-3}{\rho}$. For example, the entire incidence matrix of the KTS design in Example A.3.2 provides the parity check matrix for a $[35, 21, 4]$ LDPC code, while if just the first four resolution classes, shown in Fig. 6.1, are used for H the code has parameters $[20, 6, 6]$.

Alternatively, for a given code rate $R \approx \frac{\rho-3}{\rho}$, for ρ any integer, KTS LDPC codes exist for any block length $n \equiv \frac{3}{1-R} \pmod{\frac{6}{1-R}}$. Fig. 6.2 shows the available KTS LDPC codes up to lengths 1000. A large range of regular LDPC codes are made available by employing the resolution classes of KTS designs.

The number of non-zero entries in the parity-check matrix of a KTS code is $3n$ and so increases only linearly with n . The density of the parity-check matrix is thus inversely proportional to v , and independent of the choice of ρ :

$$\text{density}(H) = \frac{3}{v}.$$

The exact number of 6-cycles in the Tanner graph of a code from the complete KTS design is exactly the same as any other $\text{STS}(v)$. Removing columns can not add cycles and so the KTS codes are 4-cycle free. The number of 6-cycles in the KTS codes taking a subset of the blocks of the KTS design is upper bounded by (3.4):

$$N_6(\text{KTS}(v, \rho)) \leq \frac{v(v-1)(v-3)}{6}. \quad (6.1)$$

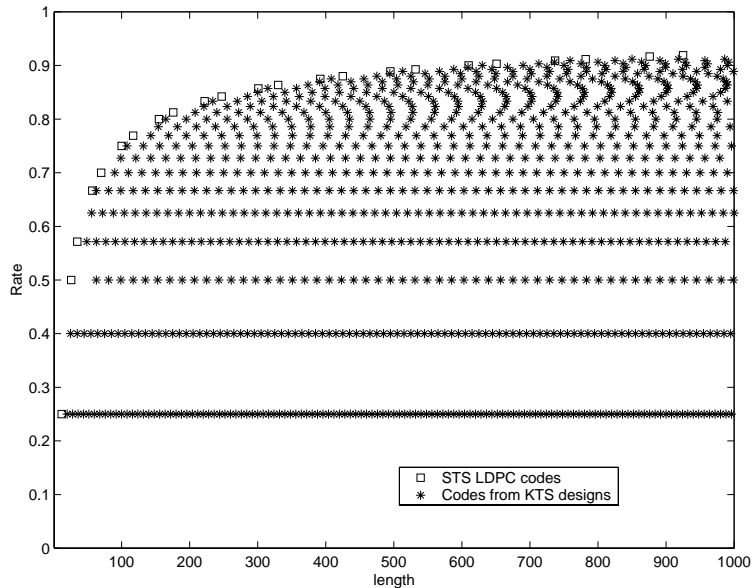


Figure 6.2: Rates and lengths of LDPC codes from Kirkman triple systems

For the KTS designs presented in the Appendix, we observe that Construction A.3.1 produces designs with full rank incidence matrices, while Construction A.3.2 produces designs with one linearly dependent row in their incidence matrix, corresponding to the point at ∞ . This row will remain linearly dependent in the KTS codes regardless of the selection of resolution classes. For all $v \leq 500$ corresponding to prime q , full rank (Construction A.3.1) and rank $v - 1$ (Construction A.3.2) codes have easily been constructed for any length greater than v . In fact the majority of possible selections of resolution classes produce codes of the maximum rank.

6.2.1 The minimum distance of KTS LDPC codes

The properties of Kirkman triple systems ensure that all columns in the parity-check matrix have weight 3, and that no two columns share more than one point. Thus each bit in the code is checked by γ orthogonal parity-check equations, and $d \geq 4$ for the KTS codes. As removing columns from a code can not decrease the number of parity-check equations orthogonal on each bit the minimum distance of the KTS codes with resolution classes removed is lower bounded by the minimum distance of the KTS codes with all resolution classes retained.

As for the STS codes, to obtain codes with minimum distance at least 6, we need to establish the existence of Kirkman triple systems which are also anti-Pasch. Fortunately, very recent constructions have been found for anti-Pasch KTS with $v \equiv 9 \pmod{18}$ [21].

When using KTS designs which are not anti-Pasch an option is to discard those resolution classes that involve a Pasch configuration when selecting resolution classes of the design to construct a KTS-LDPC code. For example, a rate- $\frac{1}{2}$ anti-Pasch LDPC code can be constructed from the KTS(21) in (A.3.1), by selecting the 1st, 2nd, 3rd, 4th, 8th and 9th resolution classes.

The upper bound on minimum distance of 10 for STS codes [72] does not apply to KTS codes with resolution classes removed as removing resolution classes will decrease the number of mitre and near codeword configurations.¹ Since the choice of resolution classes is not systematic a tight bound on minimum distance can not be derived. However, we have found that in general a successful method is to choose resolution classes which maximize Tanner's minimum distance bound [104].

6.2.2 The decoding performance of KTS LDPC codes

In this section the decoding performance of the KTS LDPC codes is compared with that of randomly constructed codes. The simulation setup, and random code construction is the same as detailed in Section 4.4.1. The density of the parity-check matrix is the same for both the KTS and random LDPC codes and so the number of operations per iteration for each code is equal. Thus the decoding complexity of the KTS and random LDPC codes is similar.

Fig. 6.10 shows the performance of rate- $\frac{1}{2}$ KTS and randomly generated LDPC codes. The KTS codes are (3, 6)-regular and the randomly generated LDPC codes have average row weights of 6, and constant column weight 3. Fig. 6.11 shows the performance of rate- $\frac{2}{3}$ KTS and randomly generated LDPC codes. The KTS codes are (3, 9)-regular and the randomly generated LDPC codes have row weights between 7 and 12, and constant column weight 3. Finally, Fig. 6.12 and Fig. 6.13 show the performance of rate ≈ 0.8 and rate ≈ 0.9 KTS and randomly constructed LDPC codes. The KTS codes are compared to randomly generated codes which have the same rate, codeword length and an equal number of non-zero entries in H .

The KTS LDPC codes perform as well as the randomly generated LDPC codes and can significantly outperform them for shorter code lengths where it is difficult to randomly construct LDPC codes without 4-cycles. When compared to random codes we see that KTS codes have met our aim (at least for small n) of constructing algebraic codes with the parameter flexibility and decoding performance of randomly constructed codes.

¹The relationship between these configurations and the code minimum distance is discussed in Section 3.3.2.

6.2.3 The implementation complexity of KTS LDPC codes

Due to the deterministic construction of resolution classes, the storage requirements necessary to completely describe the KTS codes are reduced. For a KTS code only the sets of the difference system are required (the translates can be constructed on-line), whereas for a random code the entire parity-check matrix must be stored. For a code from Construction A.3.1 this requires $(v - 3)/6$ sets of size 3 to be stored while codes from Construction A.3.2 require $v/3$ sets of size 3 to be stored. If storage is a significant issue it is possible to specify only the required m and primitive element θ (see constructions A.3.1 and A.3.2) and the entire code can be constructed on-line with some modest computational expense. Alternatively, where the hard-wiring of the codes Tanner graph is employed, as in [9], the regularity of the KTS codes translates directly into regularity in the layout of an integrated circuit.

The resolution classes of the KTS codes also offer a significant degree of flexibility when it comes to selecting code lengths and rates on-line. Once the sets of the difference family are stored, longer (and higher rate) codes can be achieved simply by adding another translate to the code which increases the number of message bits without changing the number of parity bits. The only information that needs to be communicated to completely specify the code in use is which resolution classes are employed.

The generator matrix of an LDPC code is typically constructed by applying Gaussian elimination to transform the parity-check matrix into upper triangular form and using back substitution to construct G . However, the new matrix is no longer sparse and the computational complexity is $O(n^2)$. If the parity-check matrix can be put into (near) upper-triangular form using just row and column swaps the matrix will still be sparse and (near-) linear encoding is possible [89]. The procedure for placing H into near upper triangular form in [89] is to randomly remove columns of H to be considered as belonging to message bits with the hope that enough of rows in the remaining sub-matrix have weight 1. Row and column permutations can be used to place those weight-1 rows in upper triangular form, the columns through the weight-1 rows are next removed from consideration, and the process is repeated with the remaining sub-matrix until no more weight one rows can be found [89].

For the KTS LDPC codes we observe that a gap of 3 can be obtained by randomly selecting $n - v + 3$ columns to be designated known. In all the KTS designs from Constructions A.3.1 and A.3.2 with $v \leq 57$ we have been able to produce a near upper triangular generator matrix with a gap of three. However we can not guarantee this for larger v . Instead we consider the automorphism classes of the KTS designs as a means of simple encoding in the following section.

6.3 Codes from cyclically resolvable cyclic designs

An automorphism of an STS design with point set \mathcal{P} and block set \mathcal{B} is a bijection $\alpha : \mathcal{P} \rightarrow \mathcal{P}$ such that:

$$B = \{x, y, z\} \in \mathcal{B} \iff B\alpha = \{x\alpha, y\alpha, z\alpha\} \in \mathcal{B}.$$

Thus α maps points to points such that the block set is retained.

Example 6.3.1 *The STS(9)*

$$\begin{aligned} \mathcal{B} = \{ & [1, 2, 3], [1, 4, 7], [1, 5, 9], [1, 6, 8], [4, 5, 6], [2, 5, 8], \\ & [2, 6, 7], [2, 4, 9], [7, 8, 9], [3, 6, 9], [3, 4, 8], [3, 5, 7] \}, \end{aligned}$$

has an automorphism

$$\alpha = (123)(456)(789).$$

An STS(v) is *cyclic* if it has an automorphism that is a permutation consisting of a single cycle of length v .

Example 6.3.2 *The STS(7)*

$$\mathcal{B} = [1, 2, 4], [2, 3, 5], [3, 4, 6], [4, 5, 7], [5, 6, 1], [6, 7, 2], [7, 1, 3],$$

is cyclic with an automorphism

$$\alpha = (1234567).$$

For a cyclic Steiner 2-design the set of points can be identified with Z_v , the set of integers $\{0, 1, 2, \dots, v-1\}$ modulo v . For a block $B = \{P_1, P_2, P_3\}$, in a cyclic STS design the block orbit containing B is defined by the set of distinct blocks

$$B + i = \{P_1 + i, P_2 + i, P_3 + i\} \pmod{v}$$

for $i \in Z_v$. If a block orbit has v blocks it is called *full* otherwise it is *short*. A cyclic STS(v) with $v \equiv 1 \pmod{6}$ has only full orbits while a cyclic STS(v) with $v \equiv 3 \pmod{6}$ has one short orbit.

If a resolvable design has a non-trivial automorphism of order v which preserves the resolution then it is cyclically resolvable. An STS which is cyclic with respect to an automorphism σ and which is also cyclically resolvable with respect to the same automorphism is called a *cyclically resolvable cyclic Steiner triple system* [44]. As KTS designs require

{1,4,16}	{8,11,2}	{15,18,9}	{19,20,3}	{5,6,10}	{12,13,17}	{0,7,14}
{2,5,17}	{9,12,13}	{16,19,10}	{20,0,4}	{6,7,11}	{13,14,18}	{1,8,15}
{3,6,18}	{10,13,4}	{17,20,11}	{0,1,5}	{7,8,12}	{14,15,19}	{2,9,16}
{4,7,19}	{11,14,5}	{18,0,12}	{1,2,6}	{8,9,13}	{15,16,20}	{3,10,17}
{5,8,20}	{12,15,6}	{19,1,13}	{2,3,7}	{9,10,14}	{16,17,0}	{4,11,18}
{6,9,0}	{13,16,7}	{20,2,14}	{3,4,8}	{10,11,15}	{17,18,1}	{5,12,19}
{7,10,1}	{14,17,8}	{0,3,15}	{4,5,9}	{11,12,16}	{18,19,2}	{6,13,20}
{1,11,9}	{4,14,12}	{7,17,15}	{10,20,18}	{13,2,0}	{16,5,3}	{19,8,6}
{2,12,10}	{5,15,13}	{8,18,16}	{11,0,19}	{14,3,1}	{17,6,4}	{20,9,7}
{3,13,11}	{6,16,14}	{9,19,17}	{12,1,20}	{15,4,2}	{18,7,5}	{0,10,8}

Figure 6.3: The cyclically resolvable cyclic Steiner triple system on 21 points given in [44, 79].

$v = 3 \pmod{6}$ the cyclically resolvable cyclic Steiner triple systems all have a short orbit of length $v/3$ containing the block [44]

$$\left\{ 0, \frac{v}{3}, \frac{2v}{3} \right\}.$$

In [79] cyclically resolvable cyclic Steiner triple systems, or $\text{CRCB}(v,3,1)$ designs, are classified into three types, depending on the position of the short orbit. In type $T1$ the short orbit is also a resolution class. In type $T2$ and $T3$ the blocks in the short orbit belong to separate resolution classes. The difference between the type $T2$ and $T3$ designs is the relationship between the resolution classes and block orbits. In the type $T2$ designs each resolution class not containing the short orbit is fully contained in one of the other orbits, whereas in type $T3$ designs the resolution classes not containing the short orbit have their blocks spread over different orbits. In the terminology of designs the three types are differentiated by the structure remaining when the regular short orbit is removed. The type $T1$ designs result in a cyclically resolvable group divisible design when the blocks of the short orbit is removed, the type $T2$ designs become cyclic semiframes and the type $T3$ designs reduce to cyclic quasiframes [79].

If a $\text{CRCB}(3p, 3, 1)$ of type $T2$ exists then [79]

$$p \equiv 1 \pmod{6}.$$

Example 6.3.3 *Fig 6.3 shows the blocks of a type $T2$ $\text{CRCB}(21,3,1)$ design. Each row of the figure is a resolution class, and a border surrounds the blocks in the same cyclic orbit.*

There can be more than one cyclically resolvable cyclic triple system on v points. For example there are two $\text{CRCB}(21,3,1)$ designs and 528 $\text{CRCB}(39,3,1)$ designs [64]. In this

thesis the designs given in [79] are employed to construct cyclically resolvable KTS LDPC codes.

6.3.1 Encoding LDPC codes from cyclically resolvable cyclic designs

The cyclic invariance of the LDPC codes from cyclically resolvable cyclic triple system can be used to provide simpler encoding, in a similar manner to quasi-cyclic error correction codes.

Suppose we have a quasi-cyclic code of the form in [55]:

$$H = [A_1, A_2, \dots, A_l], \quad (6.2)$$

where A_1, \dots, A_l are binary $v \times v$ circulant matrices. Then provided that one of the circulant matrices is invertible (say A_l) the parity-check matrix for this code can be constructed in systematic form by multiplying through by A_l^{-1} .

$$H_S = [A_l^{-1}A_1, A_l^{-1}A_2, \dots, A_l^{-1}A_l], \quad (6.3)$$

The final circulant is thus the identity I_v , and so H_S is in systematic form, and a generator matrix G can be easily constructed:

$$G = \begin{bmatrix} & (A_l^{-1}A_1)^T \\ I_{v(l-1)} & (A_l^{-1}A_2)^T \\ & (A_l^{-1}A_{l-1})^T \end{bmatrix}, \quad (6.4)$$

The result is a quasi-cyclic code of length vl and dimension $v(l-1)$. Linear-time encoding can be achieved using $(l-1)$ v -stage shift registers in much the same way as for cyclic codes but with separate length v shift registers for each circulant in G .

The algebra of $(v \times v)$ binary circulant matrices is isomorphic to the algebra of polynomials modulo $x^v - 1$ over $\text{GF}(2)$ [55]. A circulant matrix A is completely characterized by the polynomial $a(x) = a_0 + a_1x + \dots + a_{v-1}x^{v-1}$ with coefficients from its first row, and a code C of the form (6.2) is completely characterized by the polynomials $a_1(x), \dots, a_l(x)$. Polynomial transpose is defined as

$$a(x)^T = \sum_{i=0}^{n-1} a_i x^{n-i} \quad (x^n = 1).$$

For a binary code, length $n = vl$ and dimension $k = v(l-1)$, the k bit message $[i_0, i_1, \dots, i_{k-1}]$ is described by the polynomial $i(x) = i_0 + i_1x + \dots + i_{k-1}x^{k-1}$ and the codeword for this message is $c(x) = [i(x), p(x)]$, where $p(x)$ is given by

$$p(x) = \sum_{j=1}^{l-1} i_j(x) * (a_l^{-1}(x) * a_j(x))^T, \quad (6.5)$$

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

Figure 6.4: The cyclically resolvable cyclic Steiner triple system on 27 points given in [79].

$i_j(x)$ is the polynomial representation of the information bits $i_{v(j-1)}$ to i_{vj-1} ,

$$i_j(x) = i_{v(j-1)} + i_{v(j-1)+1}x + \cdots + i_{vj-1}x^{v-1}$$

and polynomial multiplication (*) is modulo $x^v - 1$.

While the LDPC codes from the cyclically resolvable cyclic Steiner triple systems are not quasi-cyclic they can be similarly encoded. For example, the type $T1$ CRCB design on 27 points in Fig. 6.4 has four full orbits with base blocks

$$\{0, 3, 16\}, \{2, 8, 6\}, \{1, 13, 23\}, \{1, 8, 9\},$$

and one short orbit

$$\{0, 9, 18\}.$$

For this code we have a 27×90 parity-check matrix

$$H = [A'_s, A_1, A_2, A_3, A_4].$$

H consists of four circulants

$$a_1(x) = 1 + x^3 + x^{16},$$

$$a_2(x) = x^2 + x^6 + x^8,$$

$$a_3(x) = x + x^8 + x^9,$$

$$a_4(x) = x + x^{13} + x^{23}.$$

and the matrix A'_s which is the blocks of the short orbit. The matrix A'_s can be thought of as the first 9 columns of the circulant A_s :

$$a_s(x) = 1 + x^9 + x^{18}.$$

The polynomial $a_4(x)$ is invertible with inverse given by,

$$a_4^{-1}(x) = x + x^4 + x^5 + x^6 + x^7 + x^9 + x^{12} + x^{13} + x^{15} + x^{17} + x^{20} + x^{21} + x^{23} + x^{24} + x^{25},$$

and so the parity-check matrix can be put into systematic form

$$H_S = [A_4^{-1}A'_s, A_4^{-1}A_1, A_4^{-1}A_2, A_4^{-1}A_3, I_{27}].$$

where $A_4^{-1}A'_s$ is a 27×9 matrix which is the first 9 columns of the matrix $A_4^{-1}A_s$. We thus have,

$$\begin{aligned} a_4^{-1}(x)a_s(x) &= 1 + x + x^3 + x^7 + x^8 + x^9 + x^{10} + x^{12} + x^{16} + x^{17} + x^{18} + x^{19} + x^{21} \\ &\quad + x^{25} + x^{26}, \\ a_4^{-1}(x)a_1(x) &= 1 + x + x^4 + x^5 + x^7 + x^9 + x^{11} + x^{13} + x^{15} + x^{18} + x^{19} + x^{22} + x^{23} \\ &\quad + x^{25} + x^{26}, \\ a_4^{-1}(x)a_2(x) &= x^4 + x^5 + x^7 + x^9 + x^{12} + x^{17} + x^{19} \\ &\quad + x^{22} + x^{26}, \\ a_4^{-1}(x)a_3(x) &= x^4 + x^6 + x^7 + x^{10} + x^{13} + x^{14} + x^{15} + x^{18} + x^{19} + x^{21} + x^{22} + x^{23} \\ &\quad + x^{24} + x^{25} + x^{26}. \end{aligned}$$

The generator matrix for this code is:

$$G = \begin{bmatrix} & & (A_4^{-1}A'_s)^T \\ & I_{90} & (A_4^{-1}A_1)^T \\ & & (A_4^{-1}A_2)^T \\ & & (A_4^{-1}A_3)^T \end{bmatrix}.$$

Using G in this form our code can be encoded using shift registers similarly to a quasi-cyclic code. Figure 6.5 shows an encoding circuit for this code. For the portion $A_4^{-1}A'_s$ of G only the first 9 registers in the shift register circuit are loaded with message bits, the remainder are initialized with zeros.

Note that although we use H_S to construct G we will use the original matrix, H to do our decoding. Both H and H_S are valid parity-check matrices for our code however H has the properties required for sum-product decoding.

When we choose a subset of the resolution classes of the cyclically resolvable cyclic Steiner triple system to construct our code the process is similar. For example, we take the LDPC code with the resolution class corresponding to the last row of Fig. 6.4 discarded. The parity-check matrix is regular and free of 4-cycles as required. The only difference to the full code is that the circulant A_3 is replaced with the matrix A'_3 which is the circulant A_3 with every third column removed.

$$H = [A'_s, A_1, A_2, A'_3, A_4].$$

Again H can be put into systematic form

$$H_S = [A_4^{-1}A'_s, A_4^{-1}A_1, A_4^{-1}A_2, A_4^{-1}A'_3, I_{27}],$$

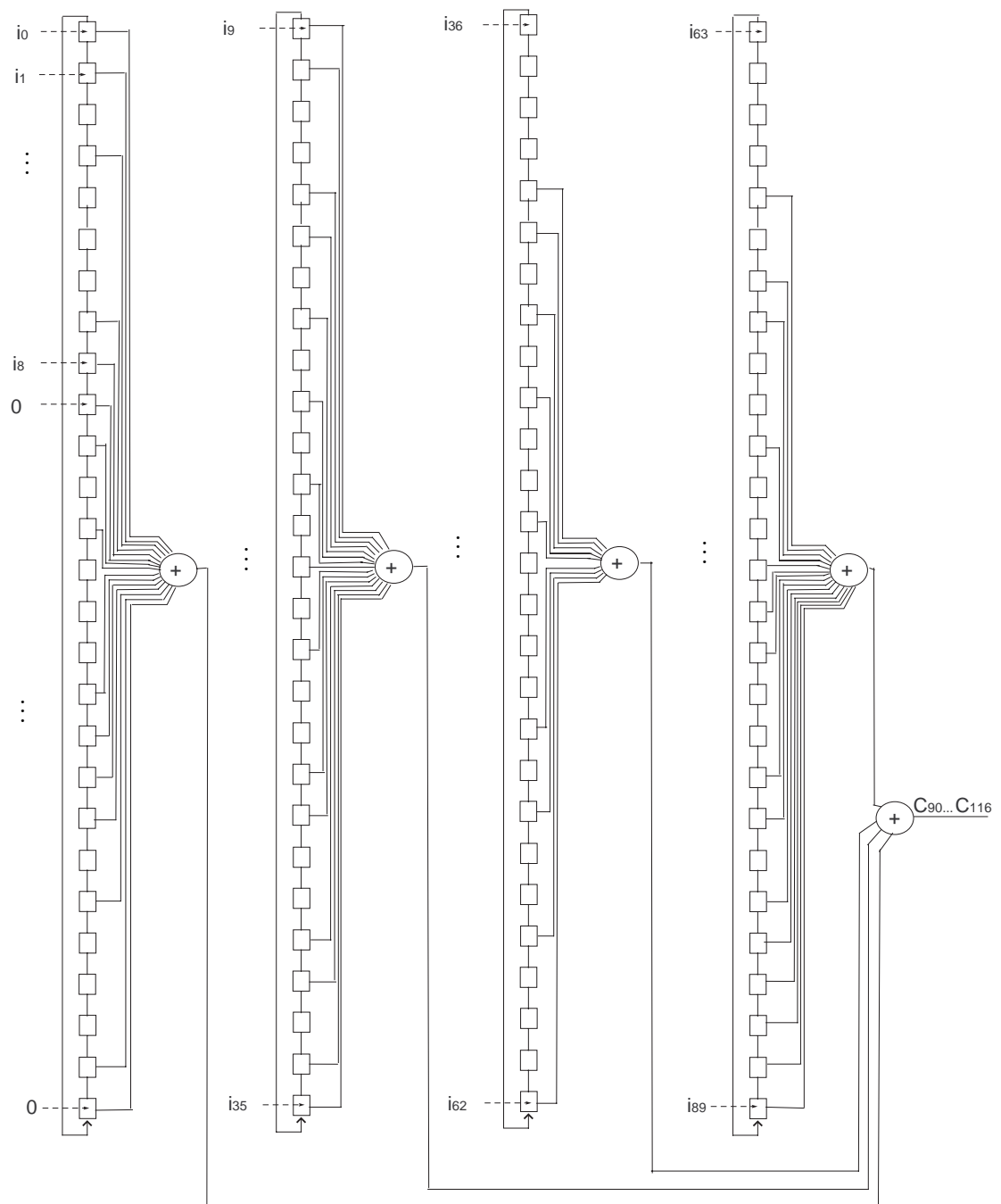


Figure 6.5: Encoding circuit for the $n = 117$, $k = 90$ LDPC code from the cyclically resolvable cyclic Steiner triple system on 27 points.

with generator matrix

$$G = \begin{bmatrix} & & (A_4^{-1}A'_s)^T \\ & I_{81} & (A_4^{-1}A_1)^T \\ & & (A_4^{-1}A_2)^T \\ & & (A_4^{-1}A'_3)^T \end{bmatrix}.$$

Of the KTS LDPC codes simulated in Section 6.2.2 the rate-1/2 length 258, rate-1/3 length 90, both rate-0.8, and the rate-0.9 length 567 KTS LDPC codes are from cyclically resolvable cyclic Steiner triple systems. The CRCB($v,3,1$) designs we have used are all from [79, Table 4.1].

There are many algebraic codes, such as the KTS LDPC codes of Constructions A.3.1 and A.3.2, with structure which are not cyclic. An option is to use the transform techniques of Tanner for group invariant codes which produce similar benefits as using the shift invariance of cyclic codes [101]. The structure of the KTS codes of Constructions A.3.1 and A.3.2 are reminiscent of that of quasi-cyclic codes. In a KTS code $v/3$ translates of a column of length v form one resolution class a number of which are concatenated together to form H , so that the number of code bits is thus a multiple of $v/3$. The translate operation in this case is slightly different from a cyclic shift as each group of $v/3$ points in the column are shifted modulo $v/3$. This is similar to the single parity orbit multiplicative-additive group invariant codes considered by Tanner in his work [101, Section VII] with the difference that the bit orbits have size $v/3$ instead of v . However, the condition that the bit orbits have size v can be relaxed as a subgroup of the additive cyclic group exists that leaves the bit nodes in that orbit fixed [101, p. 763]. This group can be factored out to leave a factor subgroup that generates the orbit. Where cyclically resolvable cyclic KTS designs do not exist with the parameters required, employing the KTS group invariance may be the best way to encode the KTS LDPC codes.

Alternatively, in the same way that resolution classes of a resolvable design can be selected to construct a regular LDPC code so too can the cyclic orbits of a cyclic STS design which is not resolvable. In this case the code would be quasi-cyclic with H completely described by circulants. Cyclic STS designs exist for all $v \equiv 1, 3 \pmod{6}$ except $v = 9$, with blocks the translates of a difference system in Z_v [3, Section 8.3]. Using a subsets of the blocks in the difference system to construct LDPC codes blocks must be removed v at a time which allows for codes of any rate

$$R = (l - 1)/l, \quad l \in \{2, 3, \dots, (v - 1)/6\}.$$

Quasi-cyclic codes from these designs are discussed in more detail in the following Chapter.

6.4 Resolvability and other designs

Kirkman triple systems are not the only combinatorial designs which are resolvable, and resolvable $2-(v, 4, 1)$, oval and transversal designs can also be constructed. We consider codes from these designs in this section.

Concurrently with our presentation of resolution classes of Steiner 2-designs [53] Kou et al. considered the uses of resolution classes, which they called “bundles”, for Euclidean geometry codes [67, 99]. The principal for EG codes is slightly different however as in that case codes defined by setting $H = N^T$ are considered. Thus removing resolution classes removes parity-checks while retaining the code length and so reduces both the number of linearly dependent rows and the column weight and so the code density and minimum distance is decreased.

6.4.1 LDPC codes from Kirkman quadruple systems

We saw in the previous chapter that although in general for full rank parity-check matrices, better LDPC codes are obtained when the column weight is 3 the exception is for very high rate codes where column weight 4 codes perform well also. Thus we consider next resolvable Steiner $2-(v, 4, 1)$ designs which we call *Kirkman quadruple systems* (KQS).²

If a $2-(v, 4, 1)$ design exists then $v \equiv 1, 4 \pmod{12}$. For resolvability there must be r groups of v/γ blocks and so it is also required that $v \equiv 0 \pmod{4}$. Thus KQS(v) designs can only exist for $v \equiv 4 \pmod{12}$ and in fact are proven to exist for all $v \equiv 4 \pmod{12}$ [49].

KQS designs can be constructed similarly to KTS designs. For example, let $q = 4m + 1$ be a prime power and θ a primitive element of $\text{GF}(q)$. The sets

$$A = \{0_1, 0_2, 0_3, \infty\}$$

$$B_{i,j} = \{\theta_j^i, \theta_j^{i+2m}, \theta_{j+1}^{i+m}, \theta_{j+1}^{i+3m}\}$$

for $0 \leq i \leq m - 1$ and $1 \leq j \leq 3 \pmod{3}$ form a mixed difference system which yields a resolvable $(3q + 1, 4, 1)$ design [3].

Like the KTS LDPC codes the LDPC codes from KQS designs benefit from a deterministic construction and so the storage requirements necessary to completely describe the code are reduced. For a KQS code only the $v/4$ sets of the difference system need to be stored, the translates can be constructed on-line. Alternatively it is possible to spec-

²Note that the KQS designs are not resolvable Steiner quadruple systems as (confusingly) Steiner quadruple systems is the name given to a set of weight 4 blocks with the property that any 3-subset of the point set is contained in a unique block, which is not a $2-(v, 4, 1)$ design

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

Figure 6.6: Incidence matrix of a resolvable oval design on 6 points.

ify only the required m and primitive element θ and the entire code can be constructed on-line with some expenditure in terms of computational complexity.

As for the KTS codes the resolution classes of the KQS codes also offer a significant degree of flexibility when it comes to selecting code lengths and rates on-line. Once the first set of the difference family are stored, longer higher rate codes can be achieved simply by adding another resolution class to the code which adds to the number of message bits without changing the number of parity bits. The only information that needs to be communicated to completely specify the code in use is the number of resolution classes employed.

There are also designs of block size 4 which are cyclically resolvable cyclic Steiner 2-designs. In [63] a construction is given for cyclically resolvable cyclic Steiner 2-designs with block size 4 on $4p$ points, where $p = 12t + 1$ and t odd. So linear-time encoding is also possible for KQS codes.

In Figs. 6.14–6.16 KQS codes are compared with randomly constructed LDPC codes from [73, 81]. Again, the simulation setup, and random code construction is the same as detailed in Section 4.4.1. For high rate codes the KQS LDPC codes significantly outperform randomly constructed LDPC codes.

6.4.2 LDPC codes from resolvable oval designs

As for the STS LDPC codes, as the length of the oval LDPC codes increases so too does the rate, and the oval LDPC codes are thus very high rate codes. However the oval designs are also resolvable and their resolution classes can be used to obtain low rate codes. A distinct resolution of the oval designs is defined by each point of the projective plane $\text{PG}(2, q)$ that is on the oval \mathcal{O} . Resolvable ovals are discussed, and an example presented in Appendix A.1.3. Fig. 6.6 gives the incidence matrix of the resolvable oval design on 6 points.

The columns of the incidence matrices of oval designs can be divided into $\gamma n/v$ resolution classes with v/γ columns per class and so we can generate a regular code with any

block length

$$\frac{v}{\gamma} \times l \quad \text{for} \quad l \in \{1, 2, \dots, \gamma \frac{n}{v}\}.$$

By using a fraction of the resolution classes of an oval design to define the LDPC codes we can achieve codes with low rates, and a wider range of code lengths.

An LDPC code formed from the resolution classes of an oval on $v = 2^m(2^m - 1)$ points will have at least $2^{2m-1} - 2^{m-1} - 3^m + 2^m$ linearly dependent rows in its parity-check matrix and a minimum distance lower bounded by $2^{m-1} + 1$. This is because removing columns from the incidence matrix can not increase its rank and nor can it decrease the number of orthogonal parity-check equations on each bit.

Example 6.4.1 *There are 9 resolution classes of the oval on 28 points with 7 blocks in each class. Selecting six of them produces a (4,6)-regular parity-check matrix H of length 42. The rank of H remains 19 and so a $[42,23,5]$ LDPC code is produced.*

Example 6.4.2 *The resolution classes of the oval on 120 points can be use to construct regular codes free of 4-cycles with the following parameters:*

$$[240, 175, 9], [225, 160, 9], [210, 145, 9], [195, 130, 9], [180, 115, 9], [165, 100, 9],$$

$$[150, 85, 9], [135, 70, 9], [120, 55, 9], [105, 44, 9], [90, 33, 9], [75, 22, 9], [60, 11, 9].$$

The number of linearly dependent parity-checks in each code is $120 - k + n$. A different selection of resolution classes can result in codes with the same length but with variations in rate, depending on how many of the existing linearly dependent parity-checks are made linearly independent by removing columns of N .

Fig. 6.18 shows the error correction performance of the (4,6)-regular $[42, 23, 5]$ code, designed by taking 6 of the resolution classes of the 2-(28, 4, 1) oval, on an AWGN channel compared to a randomly generated LDPC code with the same rate and length. The resolvable oval code significantly outperforms the random code due to the large portion of extra linearly dependent rows in the parity-check matrix of the code.

Fig. 6.17 shows the error correction performance of the (8,14)-regular $[210, 145, 9]$ code designed by taking 14 of the resolution classes of the 2-(120, 8, 1) oval on an AWGN channel compared to a randomly generated LDPC code with the same rate and length. Also shown is the same rate, but slightly longer, Euclidean geometry LDPC code [60]. The oval code shows a similar improvement, over the randomly constructed LDPC codes, as the EG LDPC code.

Removing the columns of the oval design does not change its density or minimum distance or reduce the number of linearly dependent parity-check constraints and so we see the same trends in performance for the resolvable oval codes as for the codes from the whole oval design. That is improved error correction performance compared to the random LDPC codes for short codes and poorer error correction performance compared to random LDPC codes at low signal-to-noise ratios as the code length and hence column weight is increased.

6.4.3 LDPC codes from resolvable transversal designs

In this section we show that many of the anti-Pasch transversal designs we constructed in Chapter 5 are resolvable. The resolution classes of these designs can be used to construct $(3,r)$ -regular LDPC codes with minimum distance 6 and minimum girth 6 in much the same way as the KTS designs.

Lemma 6.4.1 *The codes of Construction 5.4.1 for q odd are resolvable with the i th column of the j th resolution class given by the r th column of the transversal design where*

$$r(i, j) = (i - 1) * q + (j - 1 + i)(\text{mod } q). \quad (6.6)$$

Proof. The columns of the transversal design from Construction 5.4.1 are divided into q subsets of q columns with same first point in all the columns in the same subset and the second point in each column appearing in the same order $(q + 1, q + 2, \dots, 2q)$ in each subset. Thus for each resolution class the $(i - 1) * q$ term ensures that there will be one column from each subset of q columns in the transversal design and thus one of each point from the set $1, 2, \dots, q$. Next the $+i \pmod{q}$ term ensures that a column is chosen from each of the q positions in the column subsets and thus one of each point from the set $q + 1, q + 2, \dots, 2q$ has now been included in each resolution class. The term $+(j - 1) \pmod{q}$ ensures that each resolution class gets a different set of columns. Up to this point the argument above is true of any transversal design constructed from an orthogonal array. However for the allocation of the final set of points $2q + 1, 2q + 2, \dots, 3q$ the proof relies upon the construction method in Construction 5.4.1.

To construct the j th resolution class we are taking the j th column in the 1st set of q columns, the $(j + 1) \pmod{q}$ th column in the second set of q columns and the $(j + q) \pmod{q}$ th column from the last set of q columns. Now the third entry in the l th column of the m th set of q columns is $2q + (l + m - 1) \pmod{q}$ (see Construction 5.4.1). Thus the point from the $l = (j + i - 1) \pmod{q}$ th column in the $m = i$ th set is the point

$$P = 2q + (j + i - 1 + i - 1) \pmod{q} = 2q + (j - 2 + 2 * i) \pmod{q}.$$

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

Figure 6.7: The incidence matrix of the resolvable transversal design of Example 6.4.3.

Thus for a fixed j and $i = 1 \cdots q$ the set of points P will include one each of the set $2q + 1, 2q + 2, \dots, 3q$ if q is odd. \square

Example 6.4.3 *Starting with the TD ($q = 3$):*

$$TD = \begin{bmatrix} 1 & 1 & 1 & 2 & 2 & 2 & 3 & 3 & 3 \\ 4 & 5 & 6 & 4 & 5 & 6 & 4 & 5 & 6 \\ 7 & 8 & 9 & 8 & 9 & 7 & 9 & 7 & 8 \end{bmatrix},$$

we use equation (6.6) to place its columns into their resolution classes. For the 1st resolution class ($j = 1$) we take the columns 1, 5 and 9 for $i = 1, 2$ and 3 respectively. For the 2nd resolution class ($j = 2$) we take the columns 2, 6, and 7 for $i = 1, 2$ and 3 respectively and finally for the 3rd resolution class ($j = 3$) we take the columns 3, 4 and 8 for $i = 1, 2$ and 3 respectively. The transversal designs with columns in this order is:

$$TD = \begin{bmatrix} 1 & 2 & 3 & 1 & 2 & 3 & 1 & 2 & 3 \\ 4 & 5 & 6 & 5 & 6 & 4 & 6 & 4 & 5 \\ 7 & 9 & 8 & 8 & 7 & 9 & 9 & 8 & 7 \end{bmatrix}.$$

Fig. 6.7 shows the incidence matrix of this design.

As for the KTS codes the resolution classes of transversal designs can be used to derive regular codes with a large range of rates, dimensions and lengths. Fig. 6.8 shows the TD LDPC codes available from the resolution classes of codes constructed using Construction 5.4.1. In determining the code rates for Fig. 6.8 a parity-check matrix with 2 linearly dependent rows is assumed, i.e. we assume that removing columns will not produce added linearly dependent rows.

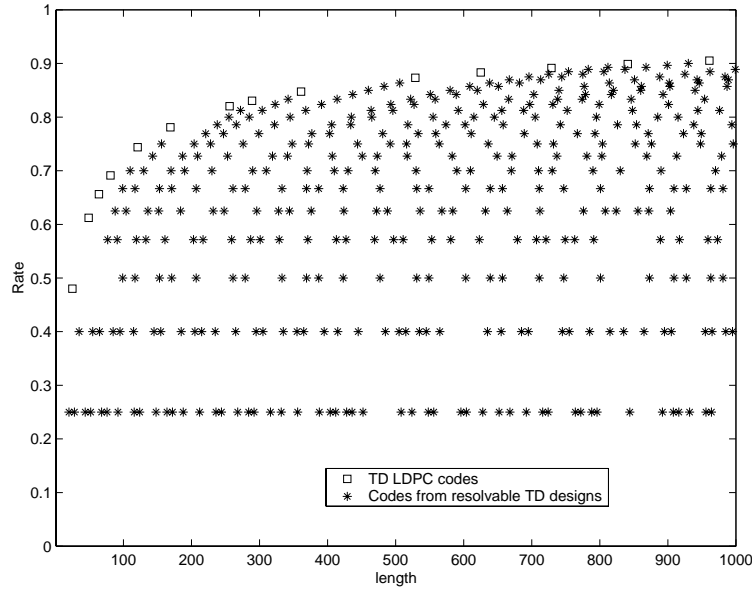


Figure 6.8: Rates and lengths of LDPC codes from resolvable transversal designs

In Figs. 6.19–6.24 LDPC codes from the resolution classes of anti-Pasch transversal designs are compared to randomly constructed LDPC codes from [73, 81]. The simulation setup, and random code construction is the same as detailed in Section 4.4.1. As for the TD LDPC codes the LDPC codes from resolvable TD designs perform very well with sum-product decoding.

6.5 Discussion

Our aim in this chapter has been to construct algebraic codes with the parameter flexibility and performance of the regular random LDPC codes. In terms of the choices of lengths and rates that are available for column weight 3, girth 6 LDPC codes this aim has been achieved. By employing the resolvability of combinatorial designs we can construct regular LDPC codes free of 4-cycles for a wide range of code lengths and rates. In particular, the length of the code can be chosen independently of the rate, and existence is proven for an infinite class of code lengths for each allowed code rate.

The Kirkman triple systems and resolvable transversal designs offer deterministic constructions for $(3, r)$ -regular codes free of 4-cycles even for those parameters (small n or high rate) for which this is difficult to achieve with random constructions. As well KQS designs offer a deterministic construction for $(4, r)$ -regular codes free of 4-cycles and the resolution classes of oval designs can be similarly employed to construct low rate LDPC codes with large column weight and a large number of linearly dependent rows in H .

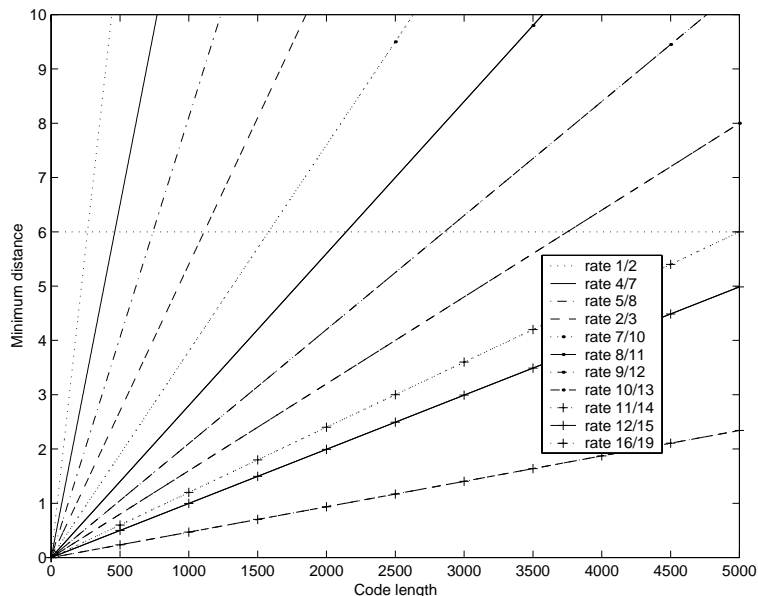


Figure 6.9: Minimum distance estimates for ensembles of LDPC codes with column weight 3. (A closeup of Fig. 2.5)

Furthermore, unlike for the random LDPC codes, the codes from KTS, KQS, resolvable oval, and resolvable transversal designs can offer significant implementation advantages due to the deterministic structure of their resolution classes and in some cases their simple encoding. However, the ad hoc manner of choosing resolution classes makes it more difficult to predict the structure and properties of the code and we can only loosely bound code properties such as minimum distance and girth in much the same way as for the random codes.

We can get a general idea of the rates and lengths for which the codes from anti-Pasch KTS and resolvable transversal designs would be expected to outperform randomly constructed codes. To do this we compare the known lower bound (of 6) for the minimum distance of these codes with the expected ensemble minimum distance of randomly constructed regular ensembles (as given in Fig. 2.5). Fig. 6.9 shows the expected ensemble minimum distance for a number of code rates and all lengths up to 5000. For example, randomly constructed rate- $\frac{1}{2}$ LDPC codes would be expected to achieve minimum distances better than 6 for code lengths as low as 250 while rate-0.8 LDPC codes are not expected to give better minimum distances than 6 until lengths over 5000.

As the choice of which resolution classes to employ is not deterministic different choices of resolution classes may result in codes with different properties and even different parameters. Like the randomly constructed codes trial and error is employed to find the best code. However unlike the randomly constructed codes, every LDPC code constructed us-

ing the resolution classes of the resolvable KTS, KQS, oval and TD designs will be regular and free of 4-cycles and lower bounds on minimum distance are guaranteed.

One particular subset of resolvable Steiner 2-designs, those which are cyclic and cyclically resolvable, provide the further advantage of simpler encoding circuits. We extend upon this approach in the following chapter, where we present LDPC codes with simple linear-time encoding.

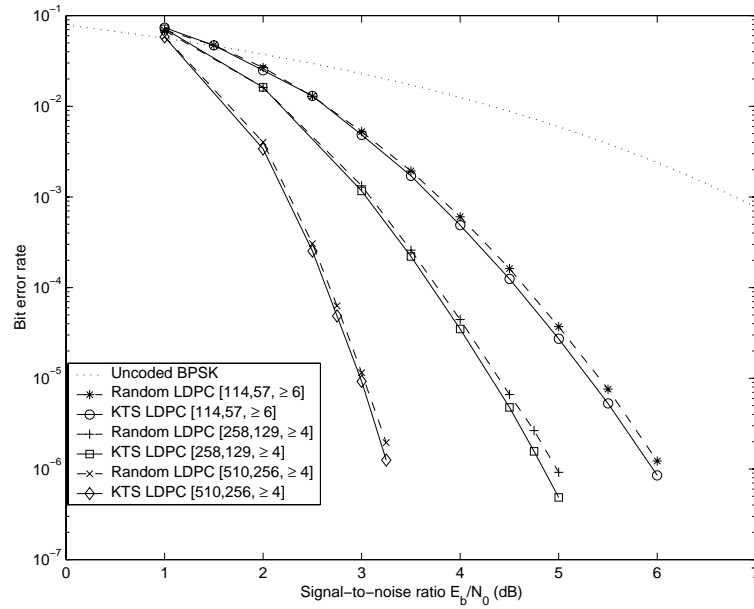


Figure 6.10: The decoding performance of KTS (3,6)-regular LDPC codes on an AWGN channel using sum-product decoding with a maximum of 10 iterations for the length-114 and 258 codes and a maximum of 500 iterations for the length-510 codes.

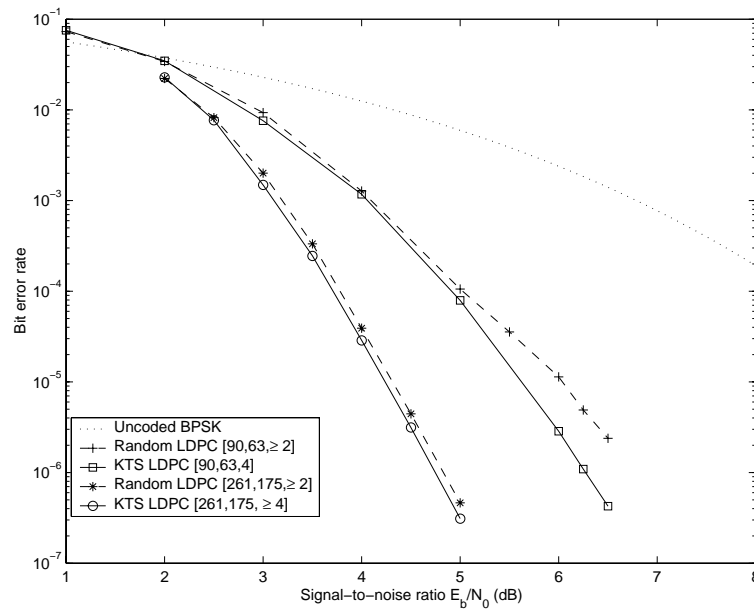


Figure 6.11: The decoding performance of KTS (3,9)-regular LDPC codes on an AWGN channel using sum-product decoding with a maximum of 10 iterations for the length-90 codes and a maximum of 50 iterations for the length-261 codes.

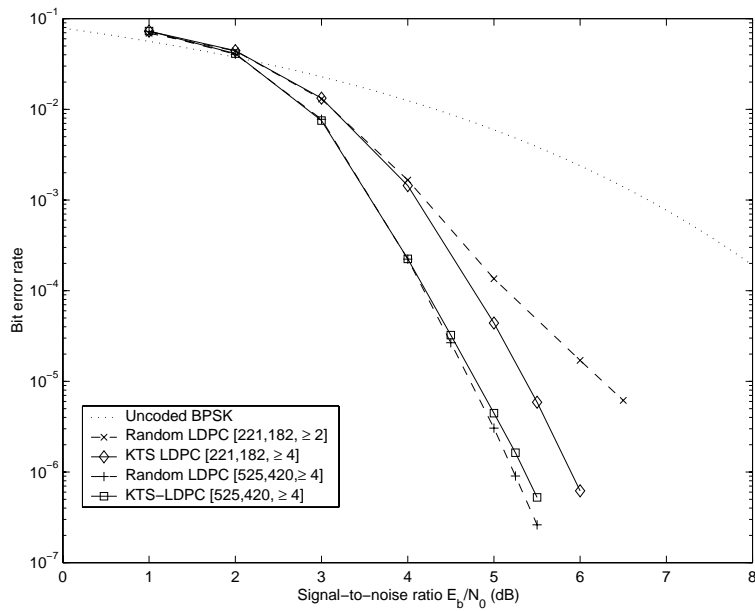


Figure 6.12: The decoding performance of KTS (3,15)-regular LDPC codes on an AWGN channel using sum-product decoding with a maximum of 10 iterations.

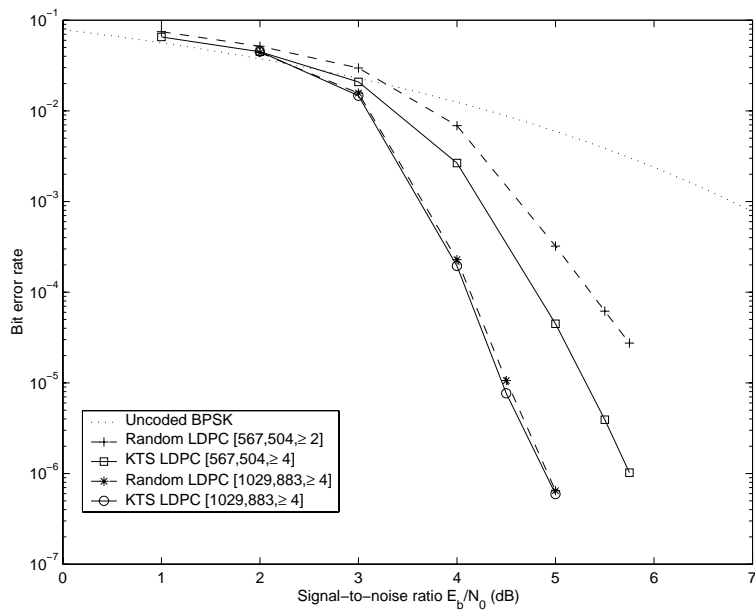


Figure 6.13: The decoding performance of KTS (3,30)-regular LDPC codes on an AWGN channel using sum-product decoding with a maximum of 10 iterations for the length-567 codes and a maximum of 50 iterations for the length-1029 codes.

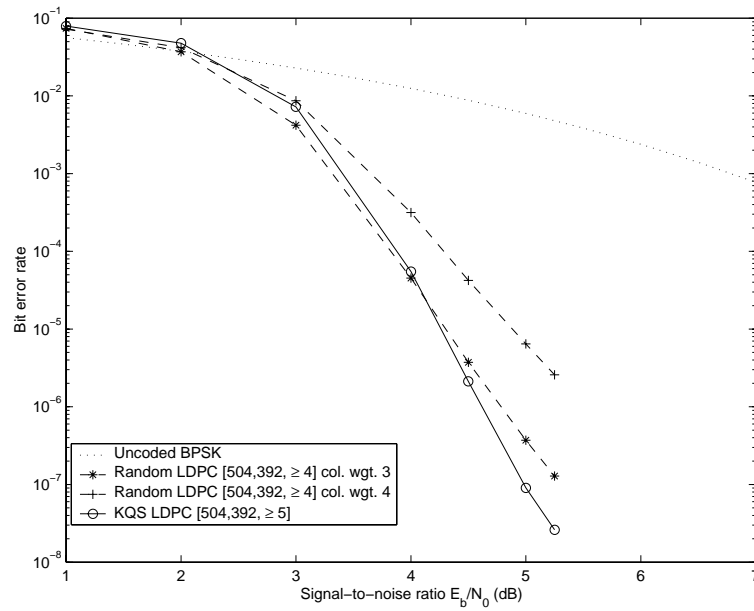


Figure 6.14: The decoding performance of a length-504 KQS LDPC code on an AWGN channel using sum-product decoding with a maximum of 50 iterations

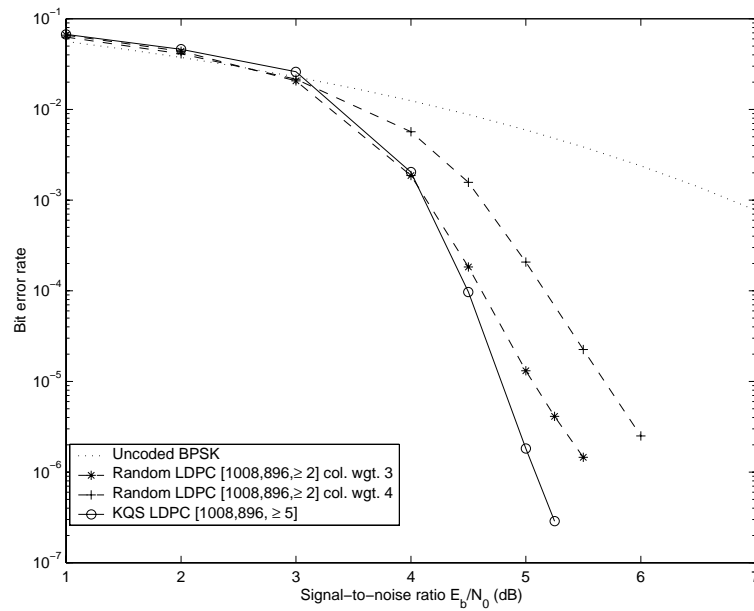


Figure 6.15: The decoding performance of a length-1008 KQS LDPC code on an AWGN channel using sum-product decoding with a maximum of 50 iterations

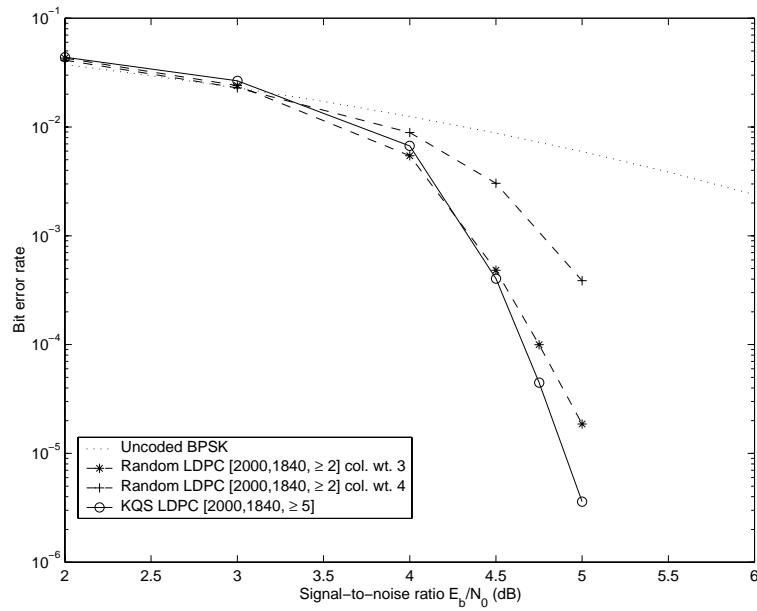


Figure 6.16: The decoding performance of a length-2000 KQS LDPC code on an AWGN channel using sum-product decoding with a maximum of 50 iterations

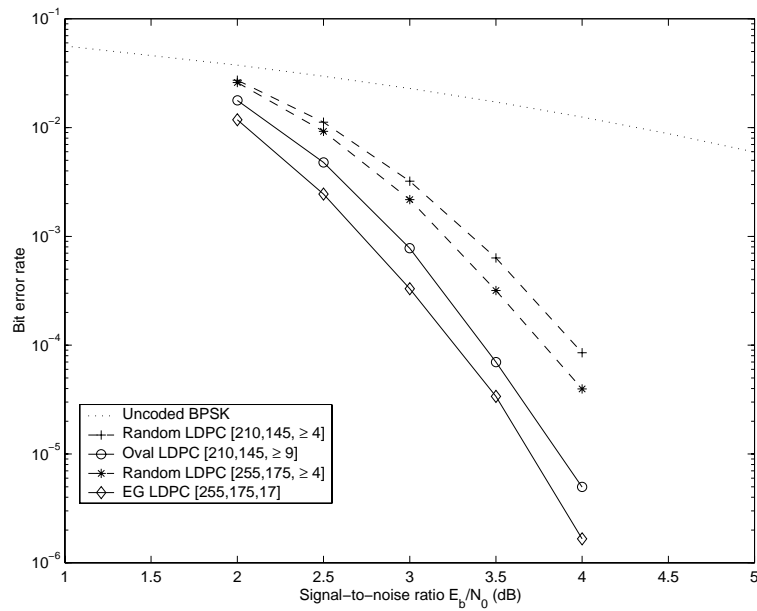


Figure 6.17: The decoding performance of a resolvable oval code on an AWGN channel using sum-product decoding with a maximum of 200 iterations

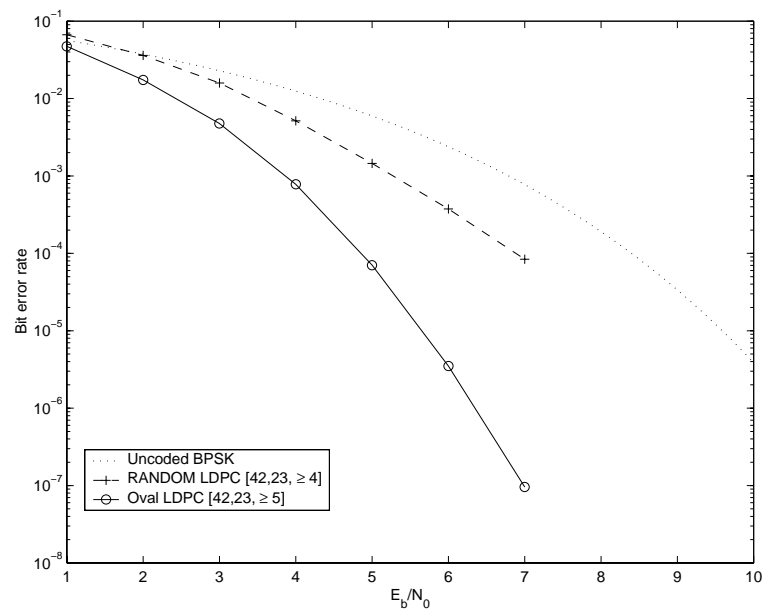


Figure 6.18: The decoding performance of a resolvable oval code on an AWGN channel using sum-product decoding with a maximum of 10 iterations

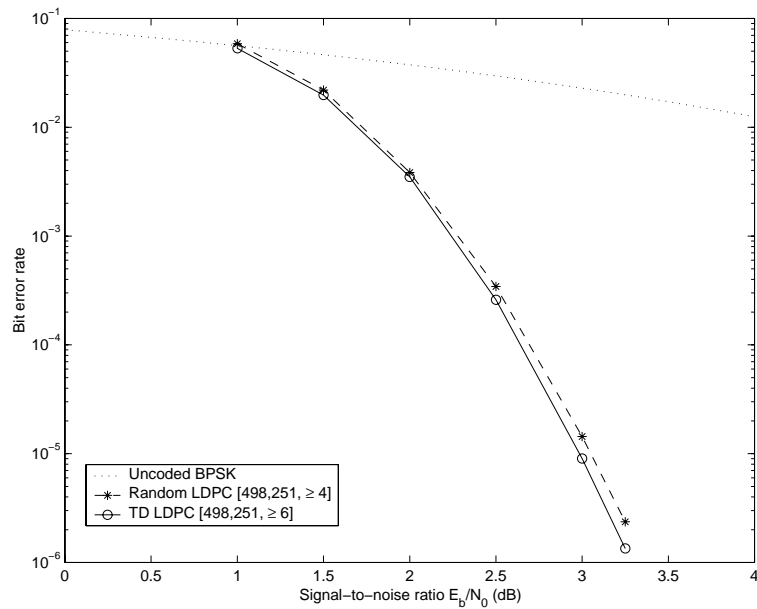


Figure 6.19: The decoding performance of a length-498 resolvable TD LDPC code on an AWGN channel using sum-product decoding with a maximum of 500 iterations

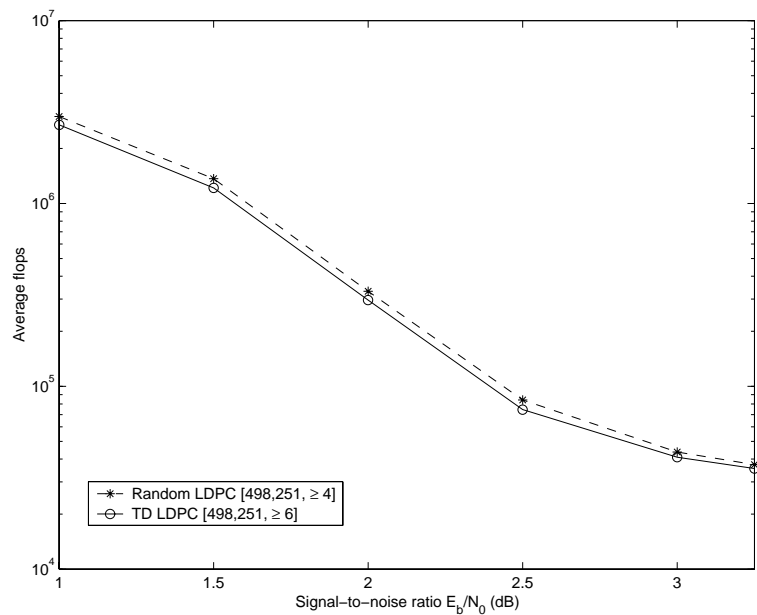


Figure 6.20: The average number of floating point multiplications required to decode a codeword for iteratively decoded low-density parity-check codes on an AWGN channel with a maximum of 500 iterations.

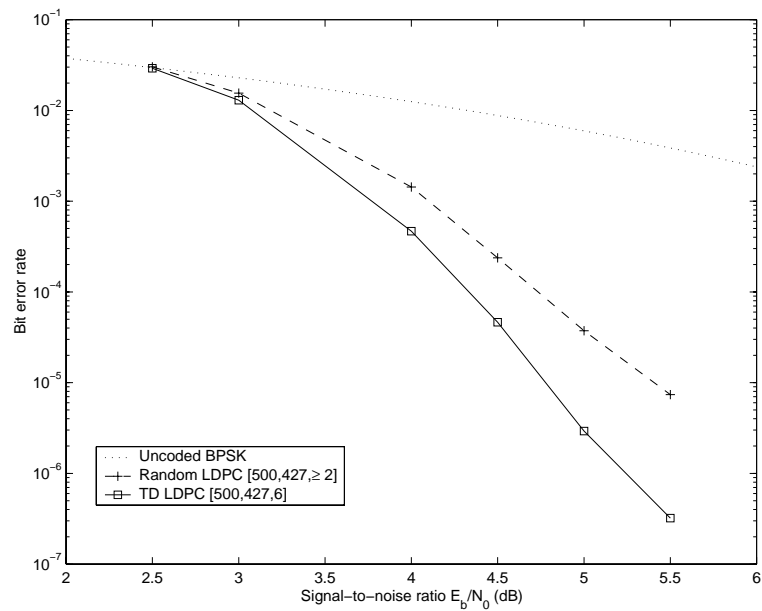


Figure 6.21: The decoding performance of a length-500 resolvable TD LDPC code on an AWGN channel using sum-product decoding with a maximum of 100 iterations

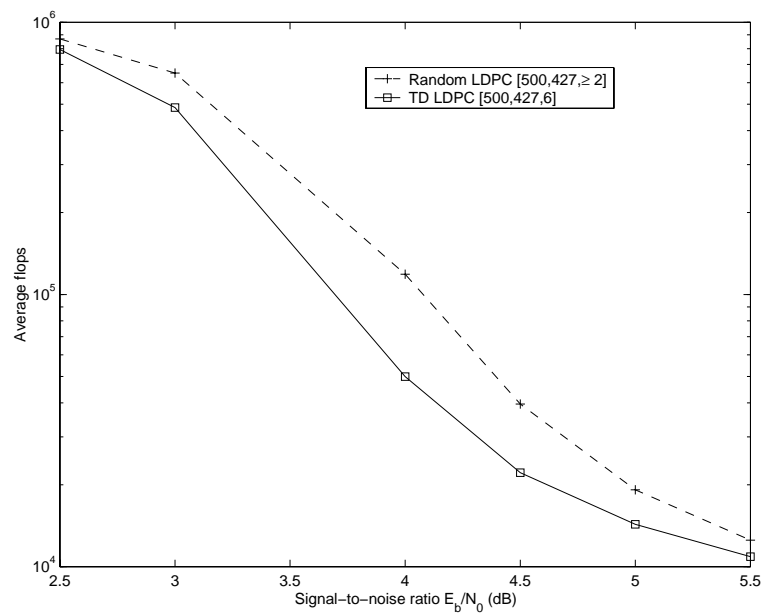


Figure 6.22: The average number of floating point multiplications required to decode a codeword for iteratively decoded low-density parity-check codes on an AWGN channel with a maximum of 100 iterations.

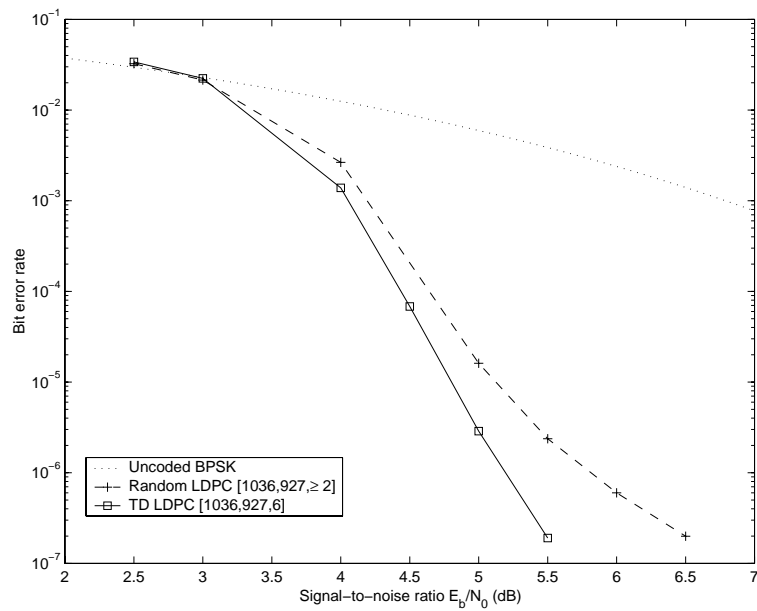


Figure 6.23: The decoding performance of a length-1036 resolvable TD LDPC code on an AWGN channel using sum-product decoding with a maximum of 1000 iterations

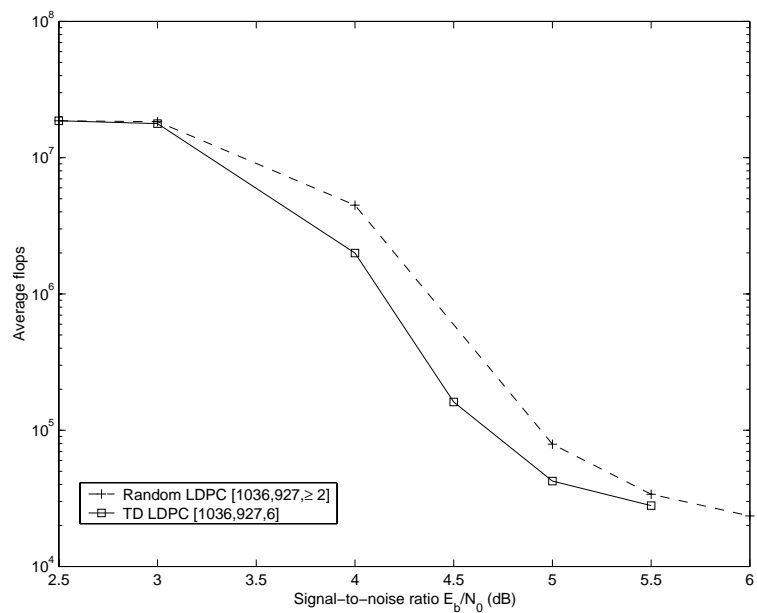


Figure 6.24: The average number of floating point multiplications required to decode a codeword for iteratively decoded low-density parity-check codes on an AWGN channel with a maximum of 1000 iterations.

CYCLIC AND QUASI-CYCLIC LDPC CODES

In this chapter we construct algebraic LDPC codes with low encoding complexity by designing codes which have the graph-based properties necessary to perform well with the sum-product decoding algorithm but which are also cyclic or quasi-cyclic. We make extensive use of combinatorial structures such as difference sets and difference families to design the new codes. The translates of these sets provide us with the incidence structures necessary for systematic constructions of regular LDPC codes with Tanner graphs free of 4-cycles and with deterministic code properties. Although the main focus is regular codes we also extend our construction of quasi-cyclic codes to produce irregular LDPC codes with simple linear-time encoding circuits.

7.1 Introduction

One serious shortcoming of LDPC codes is their potentially high *encoding* complexity, which is in general quadratic in the block length. Finding computationally efficient encoders is therefore important for LDPC codes to be considered as serious contenders for competing with turbo codes in future applications of forward error correction. Several approaches have been suggested, including the manipulation of the parity-check matrix to establish that while the complexity is, strictly speaking, quadratic, the actual number of encoding operations grows essentially linearly with block length [89]. This is successful for some irregular LDPC codes whose degree distributions have been optimized to allow transmission near to capacity [89], however the proof is non-constructive.

A more traditional approach to the encoding complexity problem is to employ cyclic, or quasi-cyclic, codes as encoding can be achieved in linear time using simple feedback shift registers. In [103, 105, 102] Tanner considered binary quasi-cyclic codes, or more generally, group invariant codes, which can be analyzed using his generalized transform methods [101].

Many of the Euclidean and projective geometry codes [60] (see Section 3.2) are also cyclic codes. The projective geometry designs from the points and lines of a $PG(2, q)$, which provide the cyclic projective geometry codes in [70, 60] (also called difference-set cyclic codes), have a combinatorial interpretation as the symmetric designs derived from difference sets. These projective geometry designs however represent only small subset of the difference structures routinely considered in combinatorics. The original construction of the PG codes was made in the context of one-step majority-logic decoding and so it seems plausible that further difference structures suitable for constructing codes for the sum-product decoding paradigm were not considered in this context. In this chapter we construct further cyclic, and quasi-cyclic, LDPC codes made possible by considering a wider range of difference structures to construct new codes.

7.2 Cyclic LDPC codes

A linear code C of length n is cyclic if it is invariant under cyclic shifts, i.e. whenever $c = (c_0, c_1, \dots, c_{n-1})$ is a codeword then so is $(c_{n-1}, c_0, c_1, \dots, c_{n-2})$ [65]. We present here a brief introduction to cyclic codes before designing cyclic LDPC codes in Section 7.2.1. See [65] or [120] for a thorough introduction to cyclic codes and their representation as ideals.

A cyclic code can be thought of as an ideal in a polynomial ring. Associate with each code vector $c = (c_0, c_1, \dots, c_{n-1})$, a polynomial

$$c(x) = c_0 + c_1x + \dots + c_{n-1}x^{n-1}.$$

The code C is represented by the set of polynomials $c(x)$, for all $c \in C$. The ring

$$R_n = F[x] \text{ mod } (x^n + 1)$$

consists of the residue classes of all polynomials with coefficients from the field F , modulo $x^n + 1$. Thus every polynomial with degree less than n is an element of the ring R_n . An ideal I in a ring is a linear subspace of R_n such that for all polynomials $c(x) \in I$ the polynomials $r(x)c(x)$ are also in I for all $r(x) \in R_n$. This is equivalent to the requirement that for all $c(x) \in I$, $xc(x) \in I$. Since $xc(x)$ is the polynomial equivalent of a cyclic shift, the set of code polynomials of a cyclic code of length n is an ideal in the ring R_n .

For every ideal, there is a unique monic polynomial $g(x) \in I$ of minimum degree which divides $x^n + 1$, such that $g(x)$ generates the ideal. Thus any $c(x) \in I$ can be written uniquely as

$$c(x) = f(x)g(x), \quad f(x) \in R_n$$

such that if $g(x)$ has degree r , $f(x)$ must have degree less than $n-r$. In coding terminology the set of polynomials $c(x)$, for every possible $f(x)$ with coefficients from F and degree less

than $n - r$, is the codeword set of the length n code with generator $g(x)$. The dimension of the code, k , is thus $n - r$. The code is binary if the field $F = \text{GF}(2)$. Since $g(x)$ divides $x^n + 1$, a parity-check polynomial $h(x)$ can be found

$$h(x) = \frac{x^n + 1}{g(x)}, \quad (7.1)$$

such that

$$c(x)h(x) = f(x)g(x)h(x) \equiv 0 \pmod{x^n + 1}, \quad (7.2)$$

since by definition $g(x)h(x) \equiv 0 \pmod{x^n + 1}$.

In matrix notation a generator matrix for C is

$$G = \begin{bmatrix} g_0 & g_1 & \cdots & g_{n-k} & 0 & \cdots & 0 \\ 0 & g_0 & g_1 & \cdots & g_{n-k} & 0 & \cdots & 0 \\ \vdots & & \ddots & \ddots & \ddots & \ddots & & \vdots \\ 0 & \cdots & & 0 & g_0 & g_1 & \cdots & g_{n-k} \end{bmatrix} \quad (7.3)$$

and a parity-check matrix for C is

$$H = \begin{bmatrix} 0 & \cdots & 0 & h_k & \cdots & h_1 & h_0 \\ 0 & \cdots & 0 & h_k & \cdots & h_1 & h_0 & 0 \\ \vdots & & & & & & & \vdots \\ h_k & \cdots & h_1 & h_0 & 0 & \cdots & 0 \end{bmatrix}. \quad (7.4)$$

It is worth noting that both G and H constructed in this way are full rank with k rows in G and $n - k$ rows in H . However, more than one valid generator and parity-check matrix exist for a given code, and for LDPC codes it can in fact be beneficial to consider parity-check matrices which are not full rank. More importantly though, for a good LDPC code H must be sparse and, ideally, free of 4-cycles. Thus polynomials $g(x)$ and $h(x)$ are required such that $h(x)$ describes a matrix which is both sparse and free of 4-cycles.

Since the generator polynomial of a cyclic code of length n must be a factor of $x^n - 1$ [65], by factorization of $x^n - 1$ into irreducible polynomials over a field F it is possible to determine all cyclic codes over F of length n . The parity polynomial is then $x^n - 1/g(x)$ for each $g(x)$. Unfortunately, the parity-check matrices constructed in this way will, with very high probability, be dense and there is no way to control the graph-based properties of the code. For this reason we consider cyclic codes designed by first specifying a parity-check polynomial which has the required properties for LDPC codes and then working backwards to determine $g(x)$ for our code.

7.2.1 Defining the parity-check polynomial

For a code to be used with sum-product decoding it is necessary that its parity-check matrix, H , be sparse and desirable that H have few small cycles or stopping sets. To

achieve this our codes are designed by specifying H rather than G . For a cyclic code this is done by choosing the polynomial

$$\Theta(x) = \theta_0 + \theta_1x + \cdots + \theta_{n-1}x^{n-1},$$

and then H_Θ is the matrix with rows of n cyclic shifts of the vector of the coefficients of $\Theta(x)$:

$$H_\Theta = \begin{bmatrix} \Theta_{n-1} & \Theta_{n-2} & \cdots & \Theta_2 & \Theta_1 & \Theta_0 \\ \Theta_{n-2} & \Theta_{n-3} & \cdots & \Theta_1 & \Theta_0 & \Theta_{n-1} \\ \vdots & & & & & \vdots \\ \Theta_0 & \Theta_{n-1} & \cdots & \Theta_3 & \Theta_2 & \Theta_1 \end{bmatrix}. \quad (7.5)$$

To obtain the properties needed for cyclic LDPC codes we consider polynomials $\Theta(x)$ defined using cyclic difference sets. The translates of these sets provide us with the incidence structures necessary for systematic constructions of cyclic LDPC codes with Tanner graphs free of 4-cycles and with deterministic code properties.

Construction 7.2.1 *Select a size γ_0 subset B of a $(n, \gamma, 1)$ cyclic difference set D , $B = \{b_1, b_2, \dots, b_{\gamma_0}\}$ such that the polynomial $h(x)$,*

$$h(x) = \text{GCD}(\Theta(x), x^n + 1), \quad \Theta(x) = \sum_{i=1}^{\gamma_0} x^{b_i},$$

has dimension $\ll n$. The binary LDPC code then has the circulant parity-check matrix in (7.5).

The polynomials $h(x)$ and $\Theta(x)$ describe the same ideal, and hence the same code, so a generator polynomial for the cyclic LDPC code described by H_Θ is given by

$$g(x) = (x^n + 1)/h(x),$$

and a generator matrix for the code can be constructed as in equation (7.7).

Theorem 7.2.1 *The matrix H_Θ described by Construction 7.2.1 is a regular square matrix described by a Tanner graph which is 4-cycle free.*

Proof. There are n points in the difference set, hence n coefficients of $\Theta(x)$. All n translates of B are used and so H_Θ has n rows and n columns. Each translate of B has weight γ_0 , and since all v translates of B are used each point will be included γ_0 times, thus the row weight $r = \gamma_0$, and regularity is also proved. Next, consider the codes defined using the entire difference set. The proof that these codes are 4-cycle free is similar to the proof that the translates of a (v, γ, λ) difference set are the blocks of a Steiner 2-design

(see e.g. [3]). For the elements $a, b \in \mathcal{G}$, we have that $a = d_i + (a - d_i)$ for each i , so a occurs in the translates $D + (a - d_i)$. Similarly, the element b occurs in the translates $D + (b - d_i)$. Thus a and b occur together in a translate $D + h$, $h \in \mathcal{G}$ precisely when $h = a - d_i = b - d_j$ for some i, j . But $a - d_i = b - d_j \Leftrightarrow a - b = d_i - d_j$ and $a - b$ is an element of \mathcal{G} . By definition there are λ pairs (d_i, d_j) for which the differences $d_i - d_j$ give an element of \mathcal{G} and so the elements a, b occur together in exactly λ translates. With $\lambda = 1$ each pair of elements occur in exactly one translate, and hence one column of H_Θ , together so the matrix H_Θ has its incidence graph free of 4-cycles since a 4-cycle requires two points to be in two columns together. For the matrices H_Θ defined using only a subset of D we are in effect removing points from the matrix and this can not add 4-cycles. \square

For H_Θ to describe a non-trivial code the rank of H_Θ must be less than n . Equivalently, the dimension of the ideal generated by $\Theta(x)$ in the ring R_n must be less than n . The dimension of an ideal in R_n is the number of zeros of $x^n + 1$ which are not zeros of $\Theta(x)$, and can be found by considering the polynomial $h(x)$ which is the greatest common divisor (GCD) of $\Theta(x)$ and $x^n + 1$. The dimension of the ideal is then equal to n minus the degree of $h(x)$. So to determine the rate of code with parity-check matrix H_Θ requires the dimension of the polynomial $h(x) = \text{GCD}(\Theta(x), x^n + 1)$. To construct new cyclic LDPC codes we have used the program Magma [14] to search for subsets of the difference set which give $h(x)$ with the required degree.

For the special case where the entire difference set is employed in Construction 7.2.1, H_Θ is the incidence matrix of a cyclic Steiner 2-design, as the translates of a $(v, \gamma, 1)$ difference set are the blocks of a Steiner 2- $(v, \gamma, 1)$ design [3]. Difference sets defined on any group isomorphic to Z_v produce cyclic incidence matrices since $D + j \pmod{v}$, $j \in \{0, 1, 2, \dots, v - 1\}$ is a cyclic shift of D . These sets, called *perfect* difference sets, exist for all $\gamma = p^s + 1$, p a prime and s an integer [95]. The designs from difference sets with $\gamma = 2^s + 1$ are the projective geometry designs originally proposed for use as the parity-check matrix of difference-set cyclic codes by Weldon [119] and at the same time in the form of projective geometries by Rudolph [91] and more recently as LDPC codes by Lucas et al. [70].

Unfortunately difference-set cyclic designs are not very common and so few cyclic codes are produced, there being four codes having length less than 1000, namely $n = 7, 21, 73$ and 273. Further, the column weight of these codes increases with n as does the rate which approaches 1 as n increases and, as we saw in Chapters 4 and 5, this hinders the code performance at low signal-to-noise ratios. When designing LDPC codes it is not necessary to limit our attention to the difference sets with $p = 2$ or indeed to employ the entire difference set. By choosing a subset B of a difference set we produce a wide range of cyclic LDPC codes with different rates and column lengths.

Example 7.2.1 *The search of 3-subsets of the (21,5,1) perfect difference set, $D = \{3, 6, 7, 12, 14\}$, shows that the subset $B = \{3, 6, 12\}$ gives an ideal described by:*

$$\Theta(x) = x^3 + x^6 + x^{12},$$

in the ring $R_n = F[x] \pmod{x^{21} + 1}$ with the required properties. The greatest common divisor of $\Theta(x)$ and $x^{21} + 1$ is $h(x) = 1 + x^3 + x^9$ and so the dimension of the ideal is 12. Thus $\Theta(x)$ describes a [21,9,4] cyclic LDPC code with generator polynomial

$$g(x) = (x^n + 1)/h(x) = 1 + x^3 + x^6 + x^{12}.$$

Example 7.2.2 *The translates of the (273,17,1) perfect difference set*

$$D = \{1, 2, 4, 8, 16, 32, 64, 91, 117, 128, 137, 182, 195, 205, 234, 239, 256\},$$

form the blocks of the projective geometry design on 273 points. This design gives the cyclic [273,191,18] LDPC code in [70]. The subset

$$B = \{2, 8, 32, 117, 128, 137, 182, 195, 234, 239\}$$

of D gives a (10,10)-regular [273,124,11] LDPC code.

The difference-set cyclic, or projective geometry, codes include only those difference sets with $\gamma = 2^s + 1$ as the rank of the circulant matrices are derived over the field $\text{GF}(p)$ [45] and so the codes constructed in [119, 91] are non-binary for $p \neq 2$. However a binary incidence matrix exists for every difference set and so binary cyclic LDPC codes can be produced using Construction 7.2.1 for every set with $\gamma = p^s + 1$. The codes from these sets have fewer linearly dependent checks and so are lower rate codes.

Example 7.2.3 *The (57,8,1) difference set $\{1, 6, 7, 9, 19, 37, 38, 42\}$ has $\gamma = 7^1 + 1$. The translates of the subset $B = \{1, 6, 9, 37\}$ of this set produces a rate 1/5 cyclic LDPC [57,12,5] code with generator polynomial*

$$\begin{aligned} g(x) &= (x^{57} + 1)/(x + x^6 + x^9 + x^{37}) \\ &= 1 + x^3 + x^6 + x^9 + x^{12} + x^{15} + x^{18} + x^{21} + x^{24} + x^{27}x^{30} + x^{33} + x^{36} \\ &\quad + x^{39} + x^{42} + x^{45} + x^{48} + x^{51} + x^{54}. \end{aligned}$$

The incidence structures constructed from the translates of an entire difference set are Steiner 2-designs and so difference sets play a prominent role in the field of combinatorics.

However, this property is not necessary for our application. We consider a second source of sets, to produce cyclic LDPC codes, from a generalization of cyclic difference sets called *difference triangles*. Difference triangles relax the conditions required of difference sets, that is that every difference occurs exactly once, to the requirement that each difference occurs at most once.

Definition 7.2.1 [26, p. 312] *An (m, γ) -difference triangle is a set $\mathcal{X} = \{X_1, \dots, X_m\}$ where for $1 \leq i \leq m$, each $X_i = \{a_{i,0}, a_{i,1}, \dots, a_{i,\gamma}\}$, is a set of γ integers, such that the differences*

$$a_{i,l} - a_{i,j} \quad 1 \leq i \leq m, \quad 0 \leq j \neq l \leq \gamma,$$

are all distinct and non-zero.

Structures closely related to difference triangles are *difference packings* which in addition allow for short orbits:

Definition 7.2.2 [26, p. 313] *An m -DP(v, γ) difference packing is a set $\mathcal{X} = \{X_1, \dots, X_m\}$ where for $1 \leq i \leq m$, $X_i = \{b_{i,1}, b_{i,2}, \dots, b_{i,\gamma}\}$ and for $1 \leq i, i' \leq m$, $1 \leq j \neq l \leq \gamma$ and $1 \leq j' \neq l' \leq \gamma$ then $b_{i,l} - b_{i,j} \equiv b_{i',l'} - b_{i',j'}$ only if $i' = i$, $l' = l$, and $j' = j$.*

For every prime power q there is a 1-DP($q^2 - 1, q$) [26]. For μ a primitive element of $F(q^2)$ and u a integer not a multiple of $q + 1$, define $d_1 = 0$, and $\mu^{d_k+1} = 1 + \mu^{u+k(q+1)}$ for $k = 0, \dots, q - 2$. The set $\{d_1, \dots, d_m\}$ is then a 1-DP($q^2 - 1, 1$). Together with the short base block $\infty \cup \{a(q + 1) : 0 \leq a \leq q - 2\}$ the result is the affine plane AG(2, q). The binary finite Euclidean geometry codes presented in [60] are 1-DP($q^2 - 1, q$) difference packings with $q = 2^s$ and are closely related to affine planes.

Difference triangles and packings with $m = 1$ provide a useful source of starting sets for Construction 7.2.1.

Example 7.2.4 *The 1-DP(15,3) $X_1 = \{1, 3, 4, 12\}$ produces a [15,7,5] cyclic LDPC code with parity-polynomial, $\Theta(x) = x + x^3 + x^4 + x^{12}$, and generated by:*

$$g(x) = x + x^4 + x^6 + x^7 + x^8.$$

Example 7.2.5 *The incidence matrix of the 1-(63,8) difference packing*

$$X_1 = \{0, 22, 32, 43, 48, 56, 60, 62\}$$

is used to construct the parity-check matrix of the [63,37,9] Euclidean geometry code. Using Construction 7.2.1 the following cyclic LDPC codes are constructed:

$$[63, 31, 7], \quad \Theta(x) = 1 + x^{32} + x^{48} + x^{56} + x^{60} + x^{62},$$

$$[63, 21, 6], \quad \Theta(x) = x^{22} + x^{32} + x^{43} + x^{56} + x^{62},$$

$$[63, 31, 5], \quad \Theta(x) = x^{32} + x^{43} + x^{56} + x^{62},$$

$$[63, 15, 4], \quad \Theta(x) = x^{32} + x^{56} + x^{62}.$$

Example 7.2.6 *The set $X_1 = \{0, 52, 60, 71, 72, 127, 155, 169, 176, 192, 201, 214, 216, 219, 245, 249\}$, with $v = 255$ is used to construct the parity-check matrix of the $[255, 175, 17]$ Euclidean geometry code. Using Construction 7.2.1 the following cyclic LDPC codes are constructed:*

$$[255, 127, 9], \quad \Theta(x) = x^{52} + x^{71} + x^{127} + x^{155} + x^{169} + x^{177} + x^{214} + x^{245},$$

$$[255, 63, 5], \quad \Theta(x) = x^{52} + x^{127} + x^{169} + x^{214}.$$

The advantage of choosing a subset of the difference set to construct the LDPC code is not only a wider range of choice when it comes to selecting the rate of a cyclic LDPC code but also a degree of flexibility when it comes to selecting the column weight, and hence density, of the resulting parity-check matrix.

Subsequent to the publication of our work on cyclic LDPC codes [54] the preprint [94] was obtained in which a similar method of searching for cyclic LDPC codes was independently proposed. In [94] cyclic LDPC codes are constructed by searching over all possible ideals in R_n and checking each for 4-cycles using the idempotents of cyclic codes. Construction 7.2.1 is a less systematic shortcut to this method, by considering only ideals constructed using a subset of a difference set or packing 4-cycle free codes are automatically guaranteed. A number of the codes in Examples 7.2.4, 7.2.5 and 7.2.6 are also presented in [94].

7.2.2 The impact of cyclic parity-checks on decoding performance

In this section we consider the decoding performance of the new cyclic LDPC codes presented in Section 7.2.1. First we consider their performance on the binary erasure channel in order to investigate the role of the structure of the cyclic codes influences their decoding performance, before their performance on the AWGN channel is considered. The simulation setup, and random code construction, is the same as described in Section 4.4.1.

The new cyclic codes of Examples 7.2.2–7.2.6 would be expected to perform well on the BEC as they have good minimum distances, at least $\gamma + 1$, a girth of 6, no stopping sets up to size γ , and many linearly dependent rows in H . However, we see that as longer cyclic codes are considered their performance, relative to randomly constructed LDPC codes with the same rate and length, is significantly poorer. As in Section 4.6 we compare the cyclic codes both to randomly constructed codes with the same rate and length as well

as to the ensemble average for all codes with the same number of parity-checks (although not necessarily the same rate).

Fig. 7.3 shows the performance of the $[63,31,7]$ cyclic code of Example 7.2.5. The $[63,31,7]$ code is from the ensemble of all codes with length 63, with 63 parity-checks, column and row weights 6, and minimum stopping set size $S_{\min} = 7$. In Fig. 7.3, finite length analysis is used to show that codes from this ensemble do on average give excellent erasure correction performances. However, the simulated performance of the cyclic $[63,31,7]$ code is significantly worse than the $(63,63,6,6)$ ensemble average. The question then is whether this difference is due to the cyclic structure of the $[63,31,7]$ code or, as for the LDPC codes from Steiner 2-designs, due to the fact that so many of the parity-check equations in the code are linearly dependent. To answer this question we simulate the performance of two codes with full rank 63×63 parity-check matrices, one randomly constructed and the other cyclic. The cyclic code is the cyclic shift of a subset, from the same difference triangle as the $[63,31,7]$ code, chosen so that a full rank H_{Θ} is produced.

As would be expected the randomly constructed binary 63×63 , $(6,6)$ -regular matrix performs close to the ensemble average. Interestingly, so too does the cyclically constructed binary matrix. Thus, contrary to our supposition in [54], the cyclic relationship between parity-checks does not seem to hinder the decoding performance of cyclic codes using the sum-product algorithm at these lengths. In a similar manner to the combinatorial codes of Chapter 4, the difference in performance between the cyclic $[63,31,7]$ code and the $(63,63,6,6)$ ensemble average is attributed to the linear dependence of half its parity-checks. Even so, the cyclic $[63,31,7]$ code outperforms the equivalent length and rate random code since it comes from an ensemble with 63 instead of 32 checks and because it has a larger smallest stopping set size, 7 instead of 4.

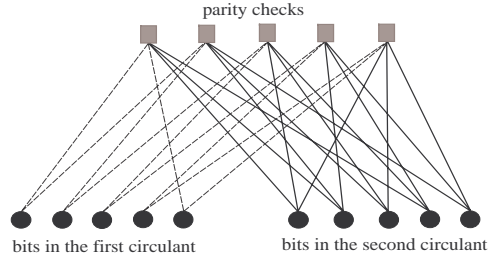
Fig. 7.5 depicts the simulated performance and finite length analysis of length 255 $(4,4)$ -regular codes. Again the full rank, random and cyclic, parity-check matrices from the ensemble show the best error correction performance while the cyclic $[255,63,5]$ code from the ensemble performs significantly worse than the ensemble average due to the many linearly dependent rows in its parity-check matrix. Significantly, in this case the equal length and rate random code outperforms the cyclic code at lower erasure probabilities as shown in Fig. 7.6. Also shown in this figure is the performance of this code with the linearly dependent rows removed from its parity-check matrix. Although the linearly dependent parity-checks do not give the same decoding advantage as linearly dependent checks, they are not altogether superfluous.

As can be seen in Figs. 7.7–7.12 the relative performance of the cyclic LDPC codes on the binary erasure channel is reflected in their performance on the AWGN channel. For the small codes the cyclic LDPC codes can provide a decoding performance advantage over randomly constructed codes but not for the longer LDPC codes. The limitation of

$$H = \left[\begin{array}{cccc|cccc} 1 & 1 & & & 1 & & 1 & & 1 \\ & & 1 & 1 & & & & & & \\ & & & & 1 & 1 & & & & \\ & & & & & & 1 & 1 & & \\ 1 & & & & & & & & 1 & 1 \end{array} \right]$$

a) Parity-check matrix with two circulants

$$G = \left[\begin{array}{cccc|cccc} 1 & & & & 1 & & 1 & & \\ & 1 & & & & 1 & & & 1 \\ & & 1 & & & & 1 & & \\ & & & 1 & & & & 1 & \\ & & & & 1 & & & & 1 \end{array} \right]$$



b) Generator matrix in systematic form c) Tanner graph representation

Figure 7.1: A rate- $\frac{1}{2}$ quasi-cyclic code from circulants

these cyclic LDPC codes is the requirement that the parity-check matrix of the code be square. There are two repercussions of this limitation; firstly, it requires a large portion of linearly dependent rows in H which makes higher rate codes very difficult to find, and secondly, the column weight and row weight of H are the same which limits the number of bits in each parity-check equation.

7.3 Quasi-cyclic LDPC codes with H a row of circulants

An alternative to cyclic codes with encoding complexity almost as low are *quasi-cyclic* codes, first presented in [111] and [55]. Recall from Chapter 6 that a code is quasi-cyclic if for any cyclic shift of a codeword by c places the resulting word is also a codeword, and so a cyclic code is a quasi-cyclic code with $c = 1$.

In Chapter 6, KTS designs which were invariant under cyclic shifts were considered. Here we focus in particular on generating binary quasi-cyclic codes described by a parity-check matrix

$$H = [A_1, A_2, \dots, A_l], \quad (7.6)$$

where A_1, \dots, A_l are binary $v \times v$ circulant matrices.

Provided that one of the circulant matrices is invertible (say A_l) the generator matrix

for the code can be constructed in systematic form

$$G = \begin{bmatrix} & & (A_l^{-1}A_1)^T \\ & I_{v(l-1)} & (A_l^{-1}A_2)^T \\ & & \vdots \\ & & (A_l^{-1}A_{l-1})^T \end{bmatrix}, \quad (7.7)$$

resulting in a quasi-cyclic code of length vl and dimension $v(l-1)$. As one of the circulant matrices must be invertible, the construction of the generator matrix in this way necessitates a full rank H . Encoding can be achieved with linear-time complexity using $(l-1)$ v -stage shift registers in much the same way as for cyclic codes.

As described in Chapter 6, a circulant matrix A is completely characterized by the polynomial $a(x) = a_0 + a_1x + a_{v-1}x^{v-1}$ with coefficients from its first row, and a code C of the form (7.6) is completely characterized by the polynomials $a_1(x), \dots, a_l(x)$. For a binary $[n, k]$ code, length $n = vl$ and dimension $k = v(l-1)$, the k bit message $[i_0, i_2, \dots, i_{k-1}]$ is described by the polynomial $i(x) = i_0 + i_1x + \dots + i_{k-1}x^{k-1}$ and the codeword for this message is $c(x) = [i(x), p(x)]$, where $p(x)$ is given by

$$p(x) = \sum_{j=1}^{l-1} i_j(x) * (a_l^{-1}(x) * a_j(x))^T, \quad (7.8)$$

$i_j(x)$ is the polynomial representation of the information bits $i_{v(j-1)}$ to i_{vj-1} ,

$$i_j(x) = i_{v(j-1)} + i_{v(j-1)+1}x + \dots + i_{vj-1}x^{v-1}$$

and polynomial multiplication $(*)$ is modulo $x^v - 1$.

Example 7.3.1 A rate- $\frac{1}{2}$ quasi-cyclic code with $v = 5$, is made up of a first circulant described by $a_1(x) = 1 + x$, and a second circulant described by $a_2(x) = 1 + x^2 + x^4$. The second circulant is invertible

$$a_2^{-1}(x) = x^2 + x^3 + x^4,$$

and so the generator matrix contains a 5×5 identity matrix and the 5×5 matrix described by the polynomial

$$(a_2^{-1}(x) * a_1(x))^T = (1 + x^2)^T = 1 + x^3.$$

Fig. 7.1 shows the parity-check matrix, generator matrix and Tanner graph of this code.

For this small example the code is neither sparse nor 4-cycle free, both of which we require for LDPC codes. In the following subsection difference families are described

and it is shown that they can be used to construct the quasi-cyclic matrices we need to produce LDPC codes which are 4-cycle free. Our construction is similar to the original construction for quasi-cyclic codes by Townsend and Weldon [111], where H is also defined by row of circulants, but with H now composed so as to be sparse and regular.

7.3.1 Quasi-cyclic codes from difference families

A difference family is an arrangement of a group of v elements into not necessarily disjoint subsets of equal size which meet certain difference requirements. More precisely:

Definition 7.3.1 [3] *The t γ -element subsets, called base blocks, of an Abelian group \mathcal{G} , D_1, \dots, D_t with $D_i = \{d_{i,1}, d_{i,2}, \dots, d_{i,\gamma}\}$ form a (v, γ, λ) difference family if the differences $d_{i,x} - d_{i,y}$, ($i = 1, \dots, t$; $x, y = 1, \dots, \gamma, x \neq y$) give each non-zero element of \mathcal{G} exactly λ times.*

If the Abelian group is Z_v each translate is a cyclic shift and the difference family is a cyclic difference family.

Example 7.3.2 *The subsets $D_1 = \{1, 2, 5\}$, $D_2 = \{1, 3, 9\}$ of Z_{13} form a $(13, 3, 1)$ difference family with differences:*

$$\begin{aligned} \text{From } D_1 : \quad & 2 - 1 = 1, \quad 1 - 2 = 12, \quad 5 - 1 = 4, \\ & 1 - 5 = 9, \quad 5 - 2 = 3, \quad 2 - 5 = 10, \\ \text{From } D_2 : \quad & 3 - 1 = 2, \quad 1 - 3 = 11, \quad 9 - 1 = 8, \\ & 1 - 9 = 5, \quad 9 - 3 = 6, \quad 3 - 9 = 7. \end{aligned}$$

In this work we are interested in difference families with $\lambda = 1$ which, as we will see in the following, allows the design of codes free of 4-cycles. The existence of $(v, 3, 1)$ difference families has long been proven for all $v \equiv 1 \pmod{6}$, v a prime power [82]. Existence results for $(v, 4, 1)$ and $(v, 5, 1)$ difference families, $v \equiv 1 \pmod{12}$ and $v \equiv 1 \pmod{20}$ respectively, have been proven for all v a prime power [24] and these results are extended to $(v, 6, 1)$ difference families in [23], and $(v, 7, 1)$ difference families in [22]. We propose constructions for both regular and irregular quasi-cyclic codes using these difference families.

Construction 7.3.1 (Regular codes): *To construct a length vl rate $\frac{l-1}{l}$ regular quasi-cyclic code $H = [a_1(x), a_2(x), \dots, a_l(x)]$ with column weight γ , take l of the base blocks of a $(v, \gamma, 1)$ difference family, and define the j th circulant of H as the transpose of the*

circulant formed from the j th base block in the difference family as follows:

$$a_j(x) = x^{d_{j,1}} + x^{d_{j,2}} + \dots + x^{d_{j,\gamma}}.$$

For an irregular quasi-cyclic code we define the column weight distribution of a length vl rate $\frac{l-1}{l}$ code as the vector $W = [w_1, w_2, \dots, w_l]$, where w_j is the column weight of the columns in the j th circulant. Denote by w_{\max} the maximum column weight of H ,

$$w_{\max} = \max\{w_1, w_2, \dots, w_l\}.$$

Construction 7.3.2 (*Irregular codes*): To construct a length vl rate $\frac{l-1}{l}$ irregular quasi-cyclic code, $H = [a_1(x), a_2(x), \dots, a_l(x)]$, with weight distribution $W = [w_1, w_2, \dots, w_l]$, take l base blocks D_1, \dots, D_l of a $(v, \gamma, 1)$ difference family with $\gamma \geq w_{\max}$, such that $a_j(x)$ is defined, using w_j of the elements of D_j , as

$$a_j(x) = x^{d_{j,1}} + x^{d_{j,2}} + \dots + x^{d_{j,w_j}}.$$

The j th circulant of H is then the transpose of the circulant described by $a_j(x)$.

The choice of which elements in the base block to use for a circulant is arbitrary, and in fact a single base block can be used to construct two circulants provided that the element set chosen for each are disjoint. The row weight, ρ , of the parity-check matrix is constant, and given by

$$\rho = \sum_{i=1}^l w_i. \quad (7.9)$$

Constructions 7.3.1 and 7.3.2 can also be applied to a single difference set. If the set is a Singer difference set the codes constructed will be the same as those obtained in [60] by column splitting the projective geometry codes with rotating column weight distribution. However, in the following we apply Constructions 7.3.1 and 7.3.2 to difference families which gives us a more flexible choice of code parameters.

7.3.2 The girth of quasi-cyclic LDPC codes from difference sets

To demonstrate that the quasi-cyclic codes are free of 4-cycles requires a well known result of difference families:

Lemma 7.3.1 [3] *A pair of elements from Z_v occur together exactly λ times in the set of translates of every base block in a (v, γ, λ) difference family.*

Proof. The set of translates required are the sets $D_i + d = \{d_{i,1} + d, d_{i,2} + d, \dots, d_{i,\gamma} + d\}, \forall d \in Z_v, i = 1, \dots, t$. Now, consider two elements $a, b \in Z_v$, $a = d_{i,j} + (a - d_{i,j})$ so a occurs in the translates $D_i + (a - d_{i,j})$. Similarly, b occurs in the translates $D_i + (b - d_{i,k})$. Thus a, b occur together in a translate $D_i + d$ precisely when $d = a - d_{i,j} = b - d_{i,k}$ for some j, k . Now, $a - d_{i,j} = b - d_{i,k} \Leftrightarrow a - b = d_{i,j} - d_{i,k}$ which by the definition of difference families occurs exactly λ times and the result follows. \square

Lemma 7.3.2 *The codes of Constructions 7.3.1 and 7.3.2 have Tanner graphs free of 4-cycles.*

Proof. Follows from the choice of $\lambda = 1$. First consider the regular case. Each column of $H = [a_1(x), a_2(x), \dots, a_l(x)]$ is a translate of one of the sets D_j in the difference family. To show that there can be no four cycles in H we need to show that no two columns of H can have a non-zero entry in the same two rows, which is equivalent to requiring that two elements of Z_v can occur together in at most one of all the translates of the base blocks in the difference family. Since two elements occur together in exactly λ translates, we need only choose $\lambda = 1$ to avoid 4-cycles. The argument follows naturally to the irregular construction. Considering only w_j of the elements in a given base block of the difference family in effect removes elements from the blocks in the set of translates, and 4-cycles can not be added by removing entries from H . \square

The avoidance of 4-cycles in the quasi-cyclic LDPC codes guarantees a minimum distance of at least $\gamma_0 + 1$ by Massey's minimum distance bound if each bit in the code is checked by at least γ_0 parity checks. Unfortunately, for the quasi-cyclic codes of rate-1/2, the number of ones in a row of the parity-check matrix is an upper bound on the minimum distance of the code [103]. So we have $d \leq \gamma_0 + \gamma_1$, where γ_0 and γ_1 are the weights of the two circulants in a rate half quasi-cyclic code.

Since 4-cycles are avoided in H the girth of the quasi-cyclic codes from difference families is at least 6. Recently in [105] Tanner showed that all codes with parity-check matrices from circulants with column weight ≥ 3 must contain a 6-cycle. What may also be significant is the number of cycles in the code of length 6, and the structure of the circulant matrices allows us to say something about the number of these cycles, denoted by N_6 .

Suppose that the cycle occurs through positions $(i_1, j_1), (i_1, j_2), (i_2, j_2), (i_2, j_3), (i_3, j_3), (i_3, j_1)$ in a circulant matrix A with $\gamma \geq 3$. Then, as A is circulant, there is also a 6-cycle through the positions $(i_1 + 1, j_1 + 1), (i_1 + 1, j_2 + 1), (i_2 + 1, j_2 + 1), (i_2 + 1, j_3 + 1), (i_3 + 1, j_3 + 1), (i_3 + 1, j_1 + 1)$ in A , and so on for $\{+1, +2, \dots, +(v-1)\}$ addition modulo v . So that a circulant matrix that has a 6-cycle must have at least v such cycles. If there is another 6-cycle, not yet counted, through the points in any column of A it must also

occur in $v - 1$ cyclic shifts by the same argument.

Lemma 7.3.3 *For a quasi-cyclic matrix H made up of the circulants from l sets of a $(v, \gamma, 1)$ difference family,*

$$N_6 \in \{v, 2v, 3v, \dots, \binom{\gamma}{2}(l\gamma - 1)(\gamma - 1)\frac{vl}{3}\}$$

Proof. Choose a pair of checks, a and b which check a bit c . Each check includes $l\gamma - 1$ bits other than c . Denote by A the set of bits other than c included in check a and by B the set of bits other than c included in check b . The $2(l\gamma - 1)$ bits $A \cup B$ must be distinct otherwise a and b will both check the same two points, which is not allowed by definition of a $(v, \gamma, 1)$ difference family. A 6-cycle through a and b requires a third check to include both a point in A and a point in B . This can happen a maximum of $(l\gamma - 1)(\gamma - 1)$ times, if the set of $(l\gamma - 1)(\gamma - 1)$ checks on the points in A , which are not a , are also the checks on the points in B , and we use each of the $(l\gamma - 1)(\gamma - 1)$ checks on the points in A and B , other than a and b , once in a 6-cycle. A check can not be used twice as it can not appear in more than one column with either a or b . Thus the pair of checks a and b can be involved in a maximum of $(l\gamma - 1)(\gamma - 1)$ 6-cycles without also forming 4-cycles. There are $\binom{\gamma}{2}$ pairs of checks in a column of H and vl columns in H , however we have counted each 6-cycle three times - once for each pair of checks, thus the maximum number of 6-cycles is

$$\binom{\gamma}{2}(l\gamma - 1)(\gamma - 1)vl/3.$$

As above, for each 6-cycle through the points in a column of A there are a total of v 6-cycles in the circulant and so the number of 6-cycles must be a multiple of v . \square

Note that the matrix H formed by taking the maximum possible number of circulants from a $(v, \gamma, 1)$ difference family is the incidence matrix of a cyclic Steiner 2 -($v, \gamma, 1$) design. If H is from the entire $(v, \gamma, 1)$ difference family the maximum number of 6-cycles are guaranteed by Lemma 3.2.1 (with $l = r/\gamma$) since every pair of points occurs exactly once in the translates of a difference set.

Now if the quasi-cyclic code is irregular, Lemma 7.3.3 will need to be modified to account for the different column weights.

Lemma 7.3.4 *For a quasi-cyclic matrix H , with weight distribution $W = [w_1, w_2, \dots, w_l]$, made up of l circulants from sets of a $(v, \gamma, 1)$ difference family, the number of 6-cycles satisfies:*

$$N_6 \leq \frac{v}{3} \sum_{k=1}^l \binom{w_k}{2} \left[\left(\sum_{m=1}^l w_m(w_m - 1) \right) - (w_k - 1) \right].$$

Proof. The proof is similar to the proof of Lemma 7.3.3, the only difference being the way in which the number of bits and checks are counted. Each check includes $\sum_{k=1}^l w_k$ points but each of those points are included in a different number of checks depending on the circulant weight w_l . Thus we need to count the number of checks on the points in A one circulant at a time. In a circulant of weight w_l there are w_l points in a and $(w_l - 1)$ checks other than a through each of these points. The exception is the circulant containing point c for which we do not count the checks on c . Thus the number of checks on the points in A is:

$$\sum_{m=1}^l [w_m(w_m - 1)] - (w_k - 1),$$

where w_k is the weight of the circulant containing point c . Finally we need only count the cycles through one column in each circulant and multiply by v and the result follows. \square

7.3.3 Regular quasi-cyclic LDPC codes

The incidence matrices of the quasi-cyclic LDPC codes are required to be full rank as one of the circulants is required to be invertible. Thus, the best performances for regular quasi-cyclic LDPC codes, over most signal-to-noise ratios, will be for those codes with column weight 3, and so we are interested in cyclic $(v, 3, 1)$ difference families.

To construct a cyclic $(v, 3, 1)$ difference family for all $v \equiv 1 \pmod{6}$ requires m triples $\{q, b, c\}$ which partition the set $\{1, \dots, 3m\}$ such that $a+b = c$ or $a+b+c = 0 \pmod{6m+1}$ (see [3]). Then the triples $\{0, a, a+b\}$ form a $(6m+1, 3, 1)$ difference system. The differences arising from the triple $\{0, a, a+b\}$ are $\pm a, \pm b, \pm(a+b)$ that is $\pm a, \pm b, \pm c$. So altogether the differences are just the non-zero elements $\pm 1, \dots, \pm 3m$ of Z_{6m+1} . For $m \equiv 0$ or $1 \pmod{4}$ the triples are Skolem triple system of order m and for $m \equiv 2$ or $3 \pmod{4}$ O'Keefe triple systems will provide the required triples [3].

Quasi-cyclic codes can also be constructed for all $v \equiv 3 \pmod{6}$ using a construction due to Rosa. Although not strictly difference families the modified difference systems from Rosa triples provide exactly the properties required to make up the circulants of a quasi-cyclic code. A Rosa triple system with $v = 6m+3$ is the partition of $\{1, 2, \dots, 2m; 2m+2, \dots, 3m+1\}$ or $\{1, 2, \dots, 2m; 2m+2, \dots, 3m; 3m+2\}$ into m triples $\{i, a_i, i+a_i\}$. The differences obtained from a Rosa triple system are precisely all the non-zero elements of Z_{6m+3} other than $2m+1$ and $4m+2$. Take also the first $2m+1$ translates of $\{0, 2m+1, 4m+2\}$ and the remaining differences are produced [3]. Since every difference occurs at most once in the translate of the modified difference families, Lemma 7.3.2 holds also for quasi-cyclic codes constructed from these sets.

As in Section 7.2.1, we do not need to restrict our attention solely to difference families.

Difference triangles and packings with $m > 1$ are also useful for Constructions 7.3.1 and 7.3.2.

The choice of which sets in the difference family or packing to select is not deterministic. As for the choice of resolution classes, we choose the combination of sets which which maximize Tanner's minimum distance bound [104].

Example 7.3.3 For $v = 15$, $m = 2$ the two Rosa triples $\{1, 3, 4\}$ and $\{2, 6, 8\}$ partition $\{1, 2, 3, 4, 6, 8, \}$. The set of translates of each triple make a 15×15 circulant, the second of which is invertible. Thus these two circulants construct a quasi-cyclic rate- $\frac{1}{2}$ $[50, 25, 4]$ LDPC code.

Example 7.3.4 Choosing four sets $\{0, 35, 62\}$, $\{0, 46, 60\}$, $\{0, 37, 59\}$, and $\{0, 39, 55\}$ from a $(63, 3, 1)$ difference family produces a quasi-cyclic $[252, 189, 4]$ LDPC code with parity-check matrix comprised of four circulants:

$$\begin{aligned} a_1(x) &= 1 + x^{35} + x^{62}, \\ a_2(x) &= 1 + x^{46} + x^{60}, \\ a_3(x) &= 1 + x^{37} + x^{59}, \\ a_4(x) &= 1 + x^{39} + x^{55}. \end{aligned}$$

Unlike for the cyclic codes, we would expect that the decoding performance of the regular quasi-cyclic LDPC codes would be similar to that of the randomly constructed LDPC codes for short-to-medium length codes. Both the quasi-cyclic and random constructions of regular LDPC codes lead to codes with full rank parity-check matrices and with the same column weights, hence (with the exception of the rate half codes) similar minimum distances. Simulation results show that this is indeed the case. In Figs. 7.13–7.15 quasi-cyclic LDPC codes are compared with randomly generated LDPC codes created using the construction method from [71, 81]. The simulation setup, and random code construction is the same as is outlined in Section 4.3. Fig. 7.13 shows the performance of the quasi-cyclic LDPC code in Example 7.3.4. The quasi-cyclic code in Fig. 7.14 is constructed from five base blocks of the $(103, 3, 1)$ difference family and the quasi-cyclic code in Fig. 7.15 is constructed from base blocks of the $(181, 3, 1)$ difference family.

For codes of rate-1/2 the quasi-cyclic codes do not perform as well as the randomly constructed codes, probably due to the minimum distance limitation shown by Tanner [103].¹ However, for rates-2/3 and above the performance of the quasi-cyclic codes is comparable to that of the random codes of the same rate, codeword length and density. Thus

¹That is not to say that good rate half quasi-cyclic LDPC codes cannot be constructed, but this may

we have regular LDPC codes with parameters and performance similar to the randomly constructed LDPC codes but with the structure required for simple linear-time encoding circuits.

7.3.4 Irregular quasi-cyclic LDPC codes

Since randomly constructed irregular LDPC codes outperform the regular ones it is natural to consider if a similar advantage can be obtained for quasi-cyclic LDPC codes with irregularly weighted circulants. While optimized irregular codes produce excellent performance with reasonable decoding complexity, they are significantly affected by the computational complexity of the *encoding* algorithm. While in the case of regular codes a number of good algebraic constructions have been presented, less consideration has been given to structured irregular codes. The same encoding efficiency will be achieved for the quasi-cyclic codes regardless of whether each circulant has the same weight or not.

For the irregular codes we begin with larger block size difference families and then use varying size subsets of some of the base blocks in the family. The existence of $(v, \gamma, 1)$ difference families is less well known for families with $\gamma \geq 3$ but there have been a number of recent results in this area [121, 17, 24]. Infinite families are known for γ up to 7 and sporadic examples obtained for larger γ .

Example 7.3.5 *The $(65, 5, 1)$ cyclic difference family has base blocks $\{0, 1, 3, 31, 45\}$, $\{0, 4, 10, 19, 57\}$ and $\{0, 5, 16, 41, 48\}$. A length 260 rate- $\frac{3}{4}$ irregular quasi-cyclic code, with weight distribution $W = [5, 4, 3, 2]$ is constructed from the circulants:*

$$\begin{aligned} a_1(x) &= 1 + x + x^3 + x^{31} + x^{45}, \\ a_2(x) &= 1 + x^4 + x^{10} + x^{19}, \\ a_3(x) &= x^{16} + x^{41} + x^{48}, \\ a_4(x) &= 1 + x^5. \end{aligned}$$

The weight 2 circulants in the irregular codes in particular need to be chosen carefully. The guarantee of no 4-cycles is not enough for weight two circulants because a cycle of size 6 in weight 2 columns will guarantee a codeword of weight three and a stopping set of size 3.

require using a matrix, rather than a row, of circulants, as is done using group structured LDPC codes in [106], or by column and row splitting Euclidean and projective geometry codes as in [60]. In fact this latter technique may also be successfully extended to difference families which would need only row splitting to produce quasi-cyclic LDPC codes made up of a matrix of circulants. However, it is not yet clear how to construct circuits to encode these circulant matrix quasi-cyclic codes.

For the length 606 and 707 codes regular column weight 3 codes can't be produced from the (101,5,1)-difference family as there are not enough sets, so we have chosen regular quasi-cyclic codes from $(v,3,1)$ difference sets with similar length. The performance of the quasi-cyclic code of Example 7.3.5 is shown in Fig. 7.16. The performance of the quasi-cyclic codes in Example 7.3.6, are shown in Figs. 7.17 – 7.20. The decoding performance of the quasi-cyclic codes demonstrates that there is a modest performance gain to be made over the regular quasi-cyclic codes by using irregular quasi-cyclic codes.

There is a great deal of flexibility in choosing circulants via Construction 7.3.2. The weight distribution can be varied depending on the channel noise level and no matter what distribution is chosen quasi-cyclic 4-cycle free codes are guaranteed.

The irregular quasi-cyclic LDPC codes offer the same encoding advantages of the regular quasi-cyclic LDPC codes. That the circulants are different weights makes no difference to the process of finding G , with the only constraint again the requirement that one circulant be invertible.

Example 7.3.7 *Fig 7.2 shows an encoding circuit for a rate 1/2 quasi-cyclic LDPC code, from the (21,5,1) difference set $D = \{3, 6, 7, 12, 14\}$, with circulants:*

$$\begin{aligned} a_1(x) &= x^6 + x^7, \\ a_2(x) &= x^3 + x^{12} + x^{14}, \\ a_2^{-1}(x) &= 1 + x^3 + x^4 + x^6 + x^8 + x^9 + x^{11} + x^{12} + x^{13} + x^{14} + x^{18}, \\ a_2^{-1}(x)a_1(x) &= 1 + x^2 + x^3 + x^5 + x^6 + x^8 + x^{10} + x^{11} + x^{12} + x^{13} + x^{15} + x^{16} + x^{20}. \end{aligned}$$

The message bits $k_0 \dots k_{20}$ are loaded into the shift register circuit to produce codeword bit c_{21} , and at each subsequent shift of the entries in the register codeword bits c_{22} to c_{41} are produced.

7.4 Discussion

The aim of this chapter has been to design codes which have the properties necessary to perform well with the sum-product decoding algorithm but which are also cyclic or quasi-cyclic. The most significant benefit of constructing cyclic and quasi-cyclic codes is that they can be encoded simply with shift register circuits. As well, the storage requirements necessary to completely describe the code are reduced, requiring only that the base blocks of the difference family are specified; the translates can be constructed on-line.

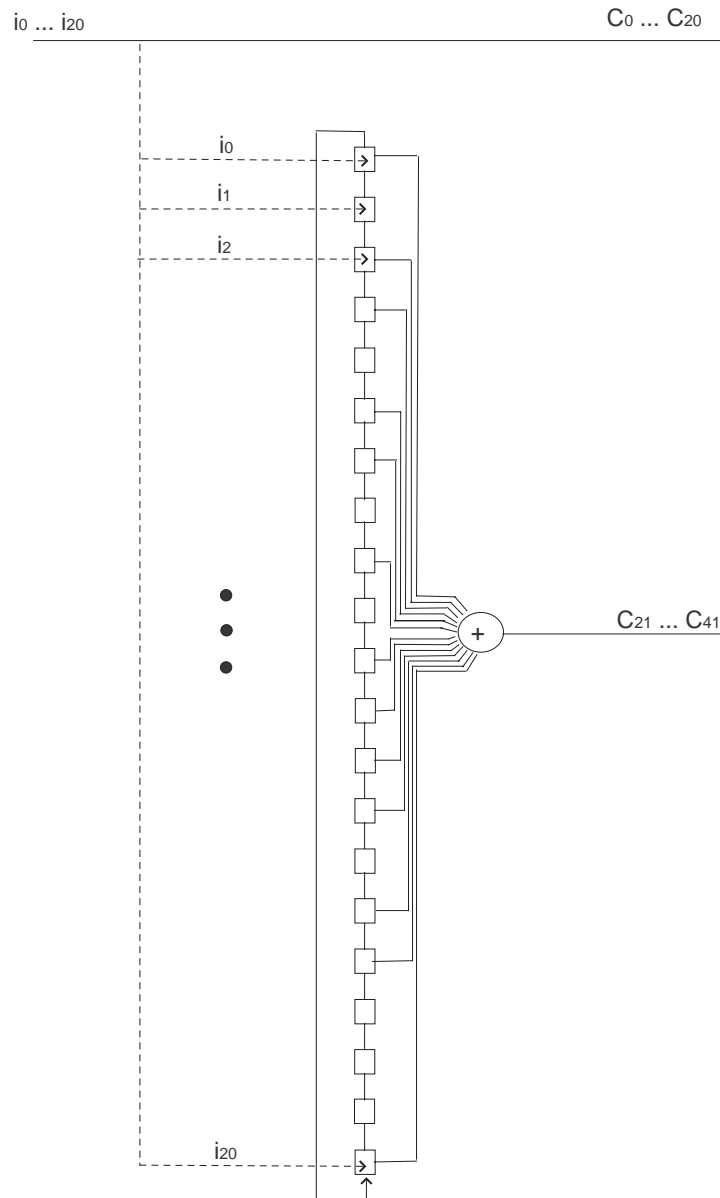


Figure 7.2: Encoding circuit for the $n = 42$, $k = 21$ irregular quasi-cyclic LDPC code of Example 7.3.7.

The cyclic construction of LDPC codes presented here is restricted by the need for many linearly dependent rows in the parity-check matrix of the code to obtain cyclic codes having non-trivial rate. However, this construction does produce some very good short LDPC codes which perform better than the equivalent length and rate randomly constructed codes. The codes from the entire difference set or packing, which are in most cases the finite geometry codes, give the highest rate codes and so are likely to be the best codes unless implementation complexity is a significant issue.

The limited range of cyclic LDPC codes available motivated our search for quasi-cyclic LDPC codes which also have simple encoding circuits but which offer greater flexibility for designing the codes. Unlike for the cyclic codes, the properties and performance of the regular quasi-cyclic LDPC codes are similar to that of the randomly constructed LDPC codes. Like randomly constructed codes the quasi-cyclic codes have full rank parity-check matrices with the same column and row weights and hence (with the possible exception of the rate half codes) similar minimum distances.

Motivated by the success of the irregular random LDPC codes we also considered quasi-cyclic codes with non-uniform column weight. The decoding performance of the irregular quasi-cyclic codes demonstrates that there is a modest performance gain to be made over the regular quasi-cyclic codes. Likewise, the irregular quasi-cyclic LDPC codes show an improved decoding performance over the standard randomly constructed LDPC codes. While it is not expected that the codes presented will outperform randomly constructed optimized irregular codes, since the column weight distribution of the quasi-cyclic codes is restricted by the need for circulants, they do offer significant benefits in terms of low complexity encoder implementation.

Similarly to the codes from resolvable designs, the choice of which subsets of the difference family to use is not systematic, which makes it more difficult to predict the structure and properties of the quasi-cyclic codes and we can only loosely bound code properties such as minimum distance and girth in much the same way as for the random codes. However unlike the randomly constructed codes, every LDPC code constructed using subsets of a difference family will be quasi-cyclic and free of 4-cycles and lower bounds on minimum distance are guaranteed.

As for the codes from resolvable designs there is also the possibility of changing the code rate and length on line by selecting circulants dynamically. Alternatively, for applications on channels with varying noise levels, the option exists to maintain the code length and rate but to vary the weight of the circulants that are chosen. For high noise channels smaller weight circulants produce the best results and reduce the lowest decoding complexity while in low noise environments a large portion of high column weight circulants improves the bit error rate.

Overall, the aims of this chapter have largely been met and a large class of cyclic and quasi-cyclic LDPC codes have been presented, which both perform well when iteratively decoded with the sum-product decoding algorithm, and which have simple linear-time encoding circuits. The quasi-cyclic codes in particular offer algebraically constructed codes with similar performance to the traditional random LDPC codes but with the added advantage of a cyclic structure offering substantial implementation advantages over random codes.

Since the presence of circulants with row weight greater than 2 upper bound the girth of the graph to 6 [105], the cyclic and quasi-cyclic codes presented in this chapter can not give the logarithmic increase of girth with length known to be achievable for randomly constructed codes. Thus for large enough lengths these structured codes will give poorer decoding performances, with sum-product decoding, than an average randomly constructed code. Given this limitation, future work focusing on more general quasi-cyclic codes, where a column weight of 3 or more does not require the presence of a circulant of weight 3 or more, and encoding circuits for them, may well prove beneficial.

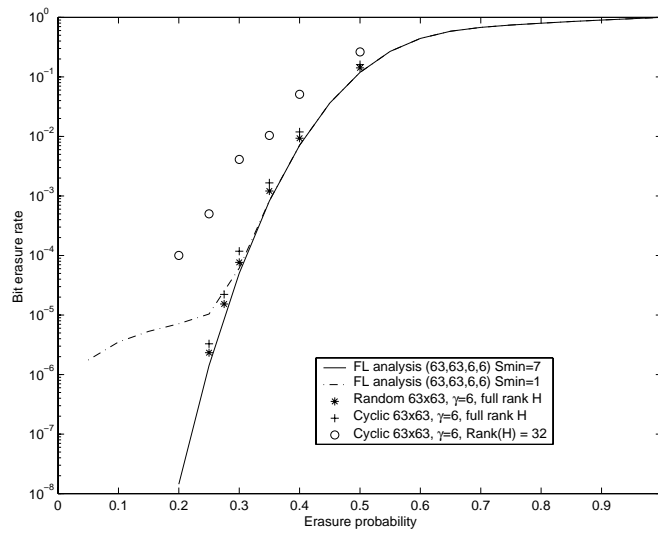


Figure 7.3: The erasure correction performance on a binary erasure channel of LDPC codes with (6,6)-regular parity-check matrices of size 63×63 . The continuous curves show the average erasure correction performance of an ensemble while individual symbols give the simulated erasure correction performance of individual codes.

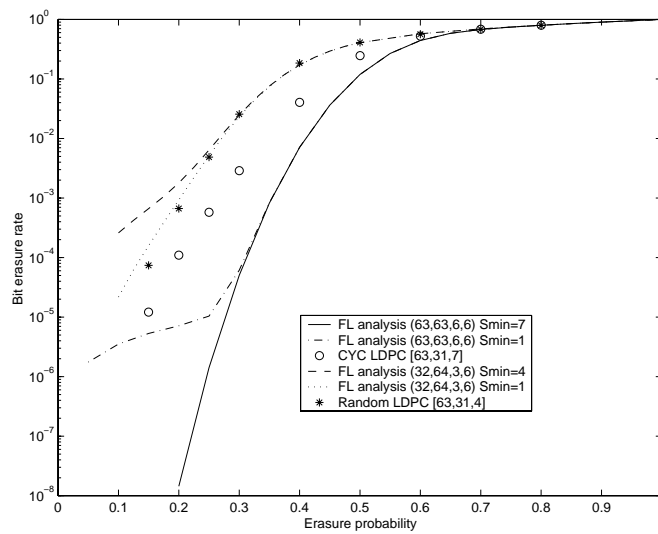


Figure 7.4: The erasure correction performance of length 63 rate-1/2 codes on a binary erasure channel. Continuous curves show the average erasure correction performance of an ensemble while individual points give the simulated erasure correction performance of individual codes.

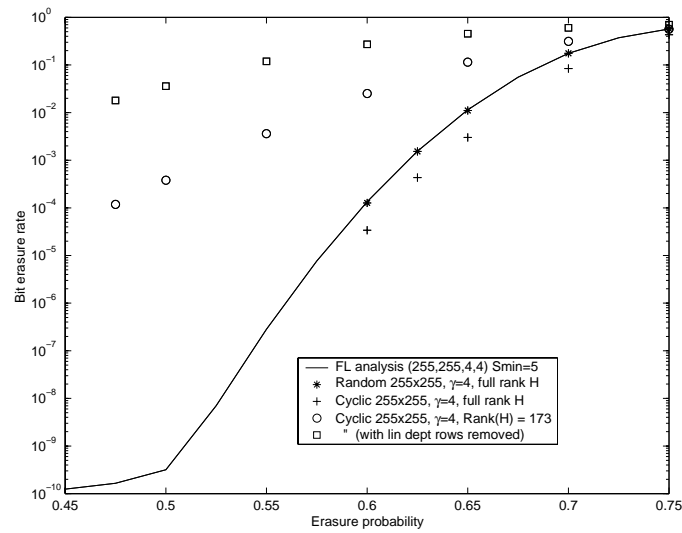


Figure 7.5: The erasure correction performance on a binary erasure channel of LDPC codes with $(4,4)$ -regular parity-check matrices of size 255×255 . The continuous curves show the average erasure correction performance of an ensemble while individual symbols give the simulated erasure correction performance of individual codes.

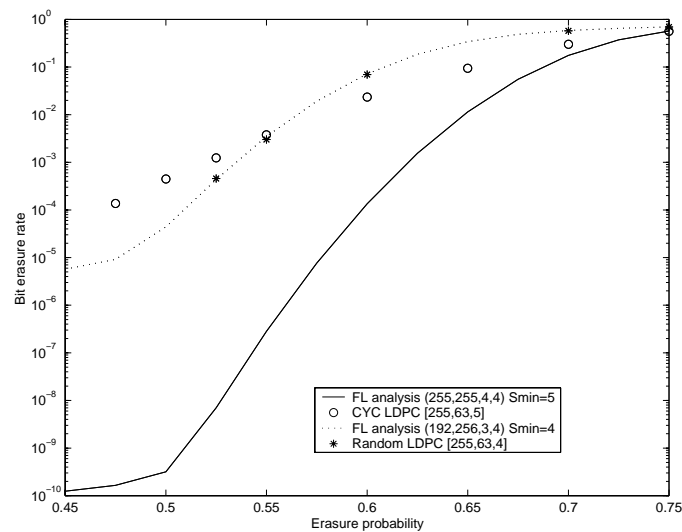


Figure 7.6: The erasure correction performance of length-255 rate-1/4 codes on a binary erasure channel. Continuous curves show the average erasure correction performance of an ensemble while individual points give the simulated erasure correction performance of individual codes.

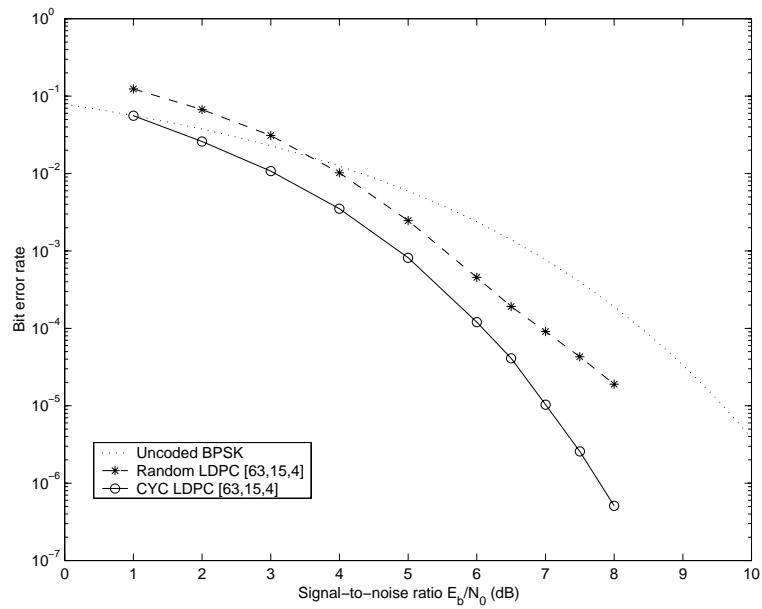


Figure 7.7: The decoding performance of LDPC codes on an AWGN channel using sum-product decoding with a maximum of 10 iterations.

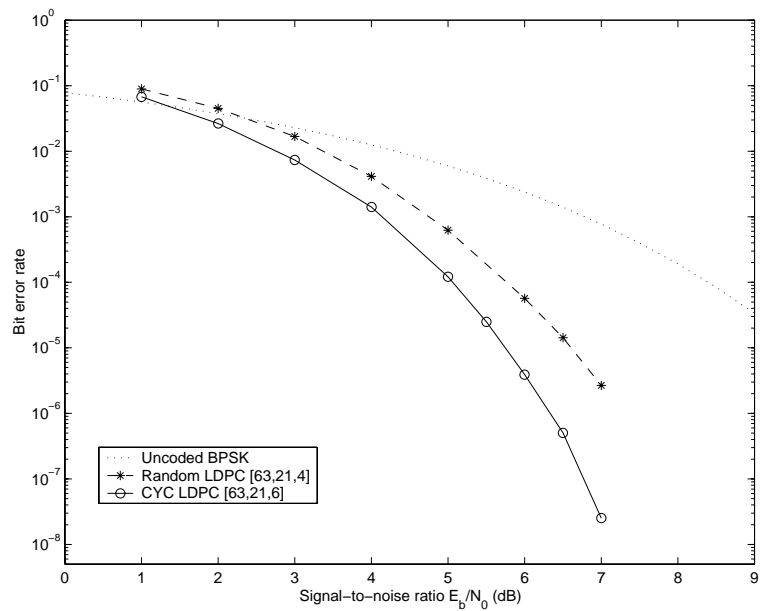


Figure 7.8: The decoding performance of LDPC codes on an AWGN channel using sum-product decoding with a maximum of 10 iterations.

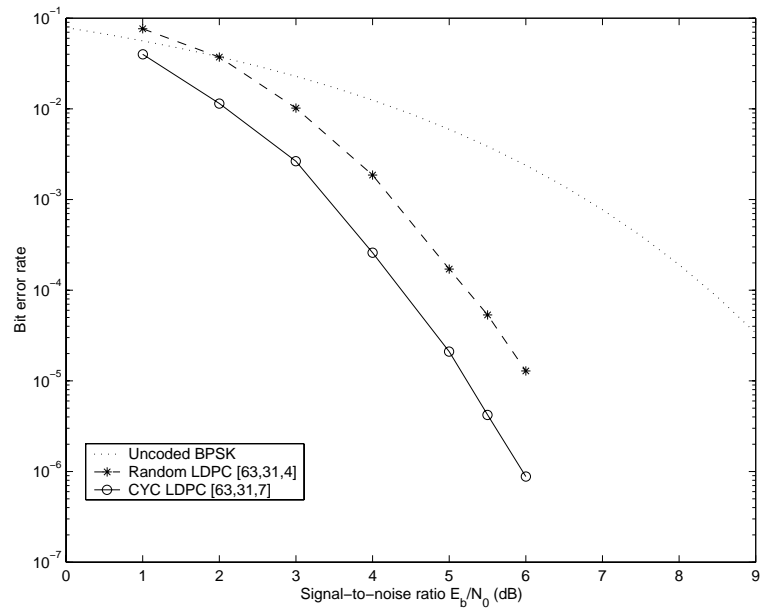


Figure 7.9: The decoding performance of LDPC codes on an AWGN channel using sum-product decoding with a maximum of 10 iterations.

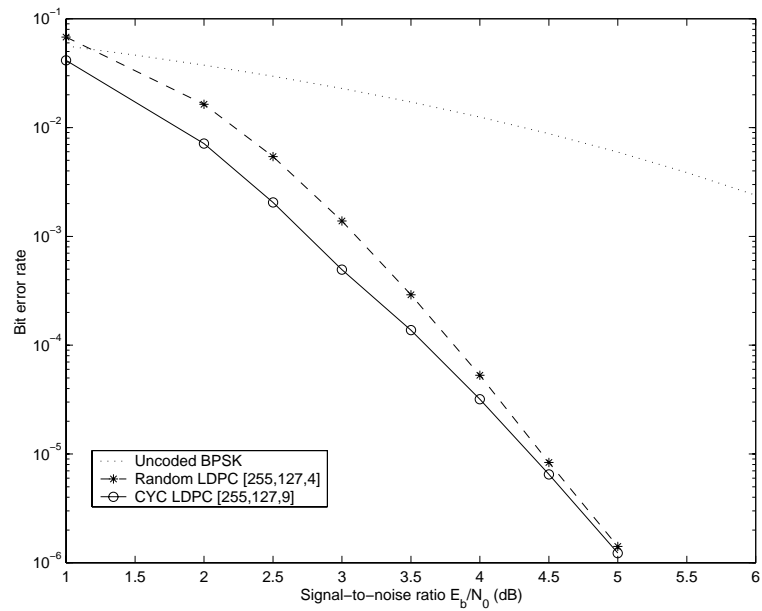


Figure 7.10: The decoding performance of LDPC codes on an AWGN channel using sum-product decoding with a maximum of 10 iterations.

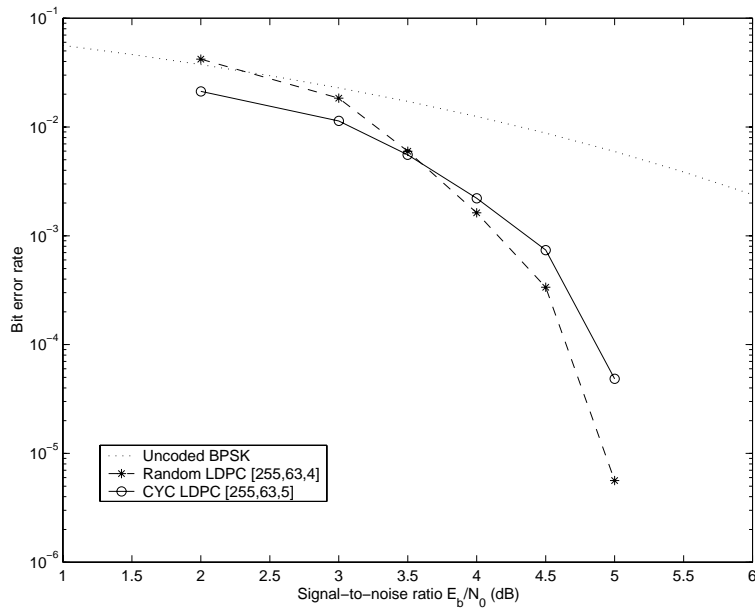


Figure 7.11: The decoding performance of LDPC codes on an AWGN channel using sum-product decoding with a maximum of 10 iterations.

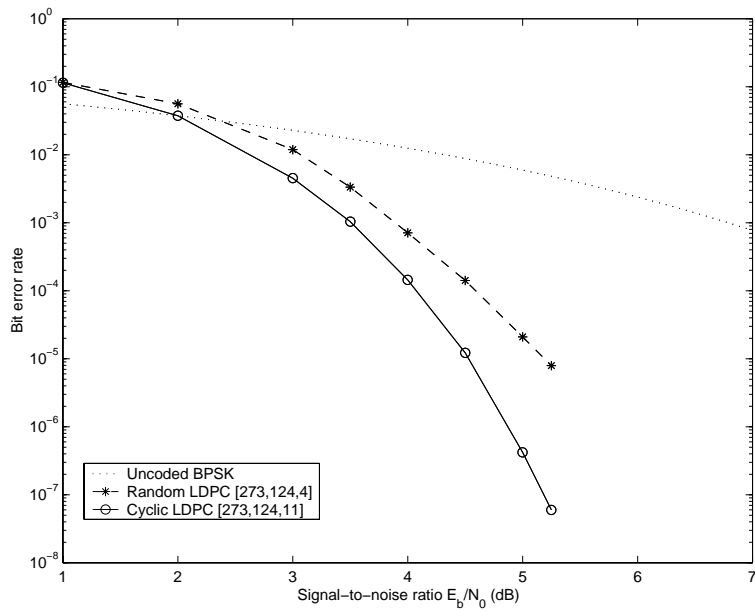


Figure 7.12: The decoding performance of LDPC codes on an AWGN channel using sum-product decoding with a maximum of 10 iterations.

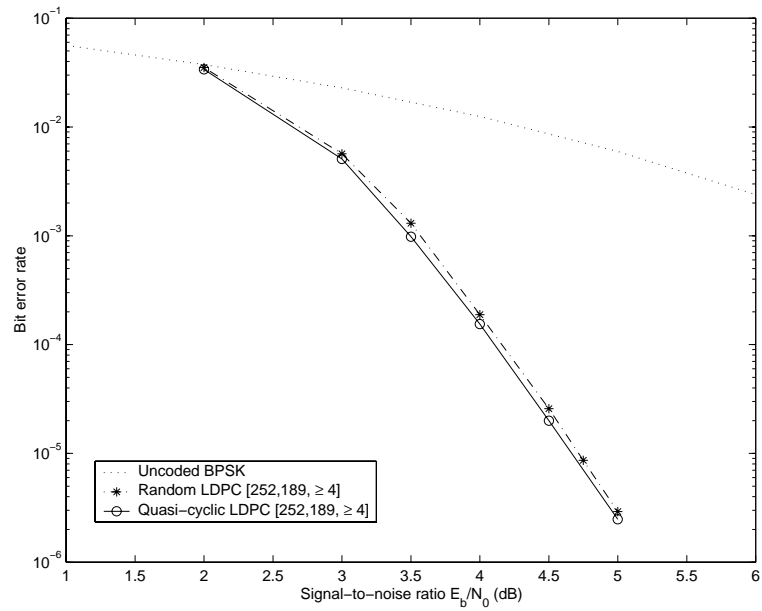


Figure 7.13: The decoding performance of LDPC codes on an AWGN channel using sum-product decoding with a maximum of 50 iterations.

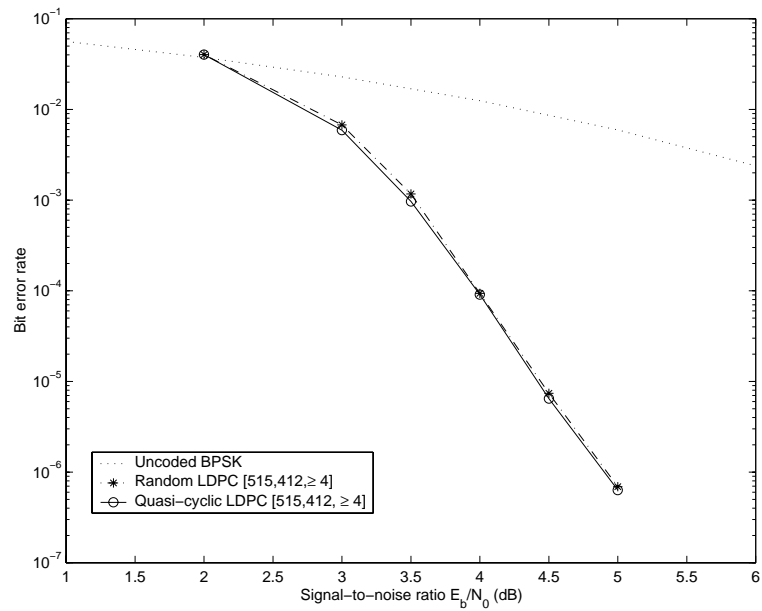


Figure 7.14: The decoding performance of LDPC codes on an AWGN channel using sum-product decoding with a maximum of 50 iterations.

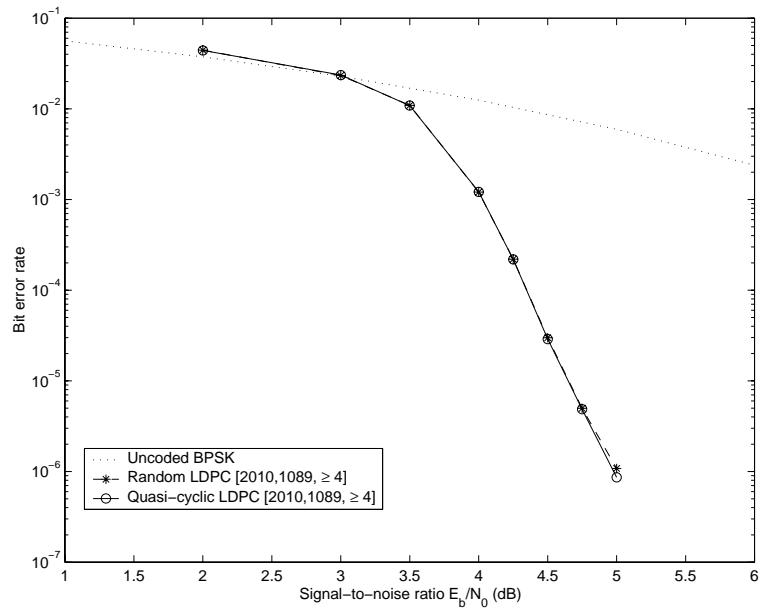


Figure 7.15: The decoding performance of LDPC codes on an AWGN channel using sum-product decoding with a maximum of 50 iterations.

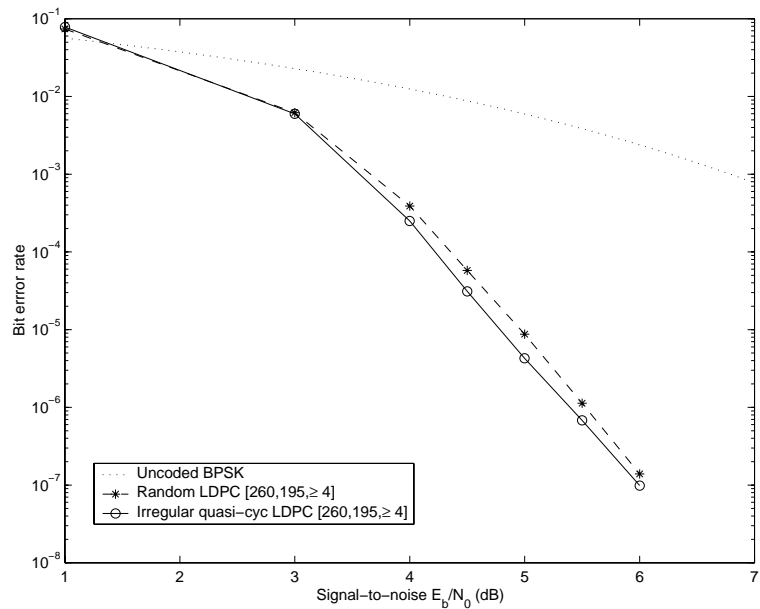


Figure 7.16: The decoding performance of LDPC codes on an AWGN channel using sum-product decoding with a maximum of 10 iterations.

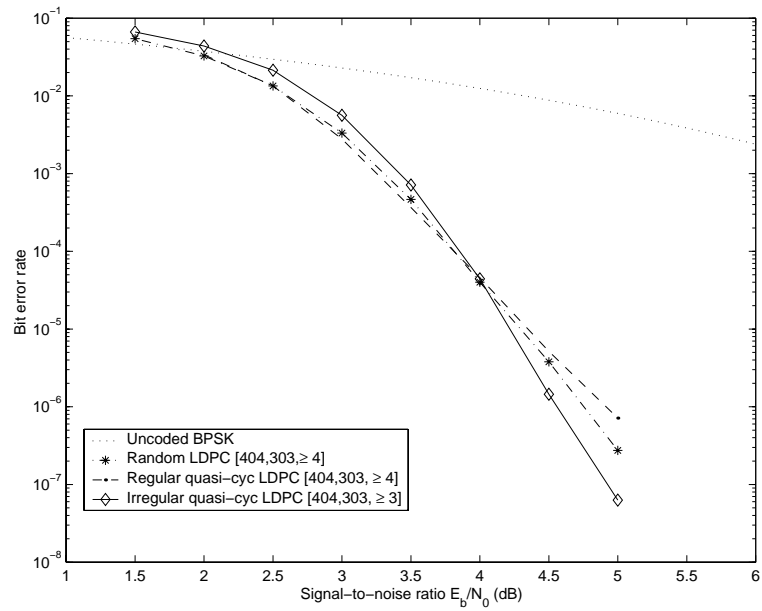


Figure 7.17: The decoding performance of LDPC codes on an AWGN channel using sum-product decoding with a maximum of 50 iterations.

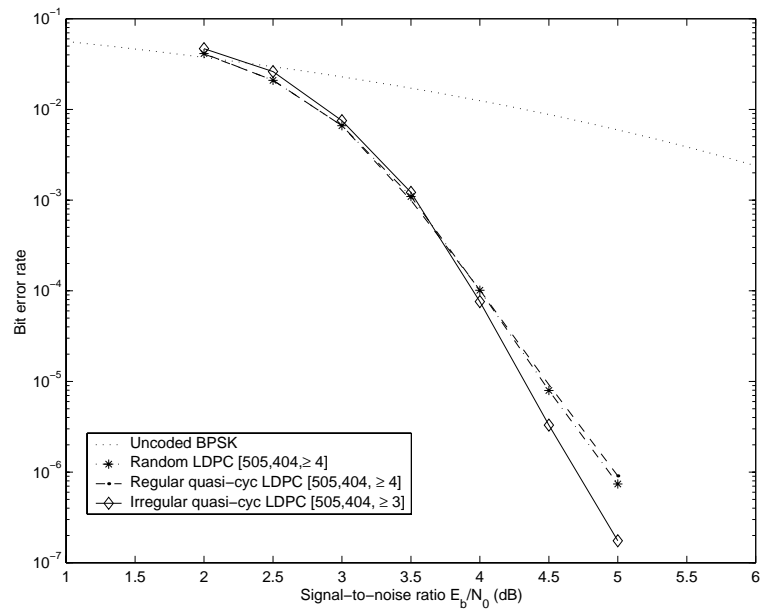


Figure 7.18: The decoding performance of LDPC codes on an AWGN channel using sum-product decoding with a maximum of 50 iterations.

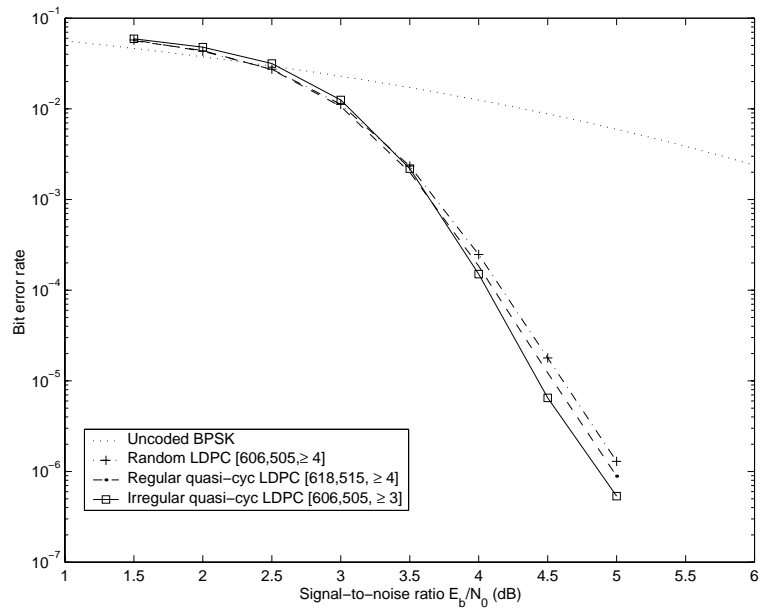


Figure 7.19: The decoding performance of LDPC codes on an AWGN channel using sum-product decoding with a maximum of 50 iterations.

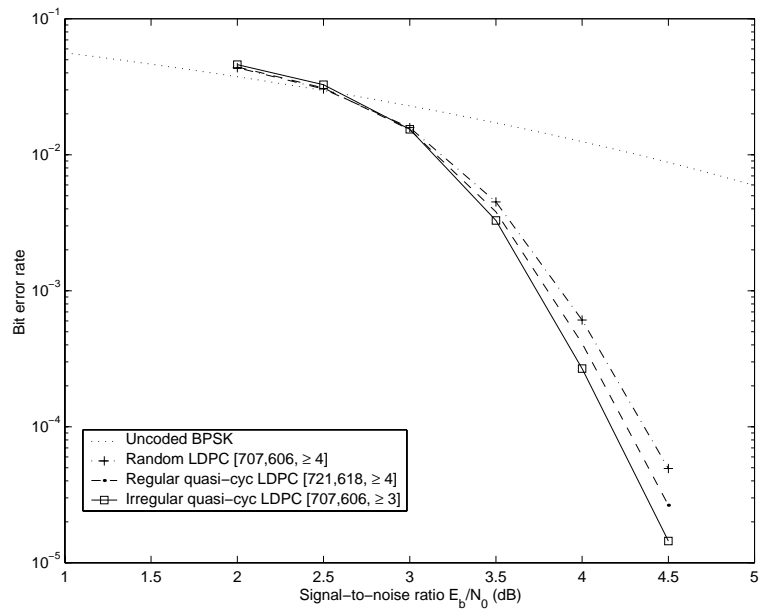


Figure 7.20: The decoding performance of LDPC codes on an AWGN channel using sum-product decoding with a maximum of 50 iterations.

CONCLUDING REMARKS

This thesis has applied combinatorial theory to the problem of designing algebraic LDPC codes. Drawing widely from existing incidence structures we have presented new families of algebraic LDPC codes with definite benefits over the traditional randomly constructed codes, particularly where small to medium length codes are concerned. The benefits of considering codes from combinatorial designs are guaranteed code properties, parameter flexibility, and ease of implementation including reduced storage requirements and reduced encoding complexity. Designs also offer codes with guaranteed girth, in all cases at least 6, and provide desirable properties not easily achieved with random constructions, such as rank deficient parity-check matrices.

In the first part of this thesis we were concerned with LDPC codes which are completely deterministic, and we construct in Chapters 3 to 5 a wide range of codes from Steiner 2-designs and partial geometries. Steiner 2-designs offer very high rate LDPC codes, most likely the highest rate 4-cycle free codes possible for a given length and column weight. Generalizing to partial geometries enabled us to unify many of the existing algebraic LDPC codes under one general class, and to derive lower bounds for the minimum distance, minimum stopping set size and code rate, and expressions for the exact number of minimum weight cycles in all of these codes.

There are two main types of algebraic codes from designs, those with full (or close to full) rank parity-check matrices, and those with a significant proportion of linearly dependent rows in their parity-check matrices. The codes from designs with (approximately) full rank incidence matrices have properties and decoding performances similar to randomly constructed regular LDPC codes. In most cases the algebraic codes from designs show decoding improvements over the random codes due to the guaranteed removal of 4-cycles. This is particularly evident for codes, such as those with high rate, where this is difficult to do using random constructions. We saw that longer column weights allow for better decoding performance at high SNR values but hinder performance at low SNR values.

For the LDPC codes from Steiner 2-designs and partial geometries with rank deficient

incidence matrices, the linearly dependent rows in H can provide a significant benefit in decoding performance. However, this is complicated by the accompanying increase in column weight for these codes. The performance gain of the new unital, oval, and proper partial geometry codes with larger length is typically realised only at high signal-to-noise ratios. However, for the small codes the LDPC codes from designs with rank deficient incidence matrices represent an excellent alternative to randomly constructed LDPC codes as they significantly improve upon their decoding performance as well as offer a deterministic construction, regularity and guaranteed girth.

As the codes from the Steiner 2-designs and partial geometries have Tanner graphs with girth 6 (or 8 in the case of the generalized quadrangles), they do not show the logarithmic relationship between girth and length known to be achievable for sufficiently long random codes. Thus, for long enough codes randomly constructed codes would be expected to outperform codes from these designs. However, as we saw for the Steiner triple systems and transversal designs, this length may be long enough so that codes from designs represent the best choice for all practical applications requiring high rate LDPC codes.

In a compromise between completely deterministic codes and parameter flexibility, we employed design resolvability in the second part of this thesis to construct regular 4-cycle free LDPC codes for a much wider range of code rates and lengths. The resolvable designs allow us to construct LDPC codes with similar properties to the randomly constructed codes. The non-systematic selection of resolution classes for these codes means that while some code properties can be guaranteed the codes are not fully deterministic and less can be said about their properties than for the codes from full designs. However, this also means that the codes from resolvable designs are not limited by the tight upper bound on girth facing the codes from the full designs.

The last part of this thesis is aimed at addressing the high encoding complexity associated with LDPC codes. To produce cyclic and quasi-cyclic error correction codes, which are good LDPC codes, we made extensive use of combinatorial structures such as difference sets, difference families and mixed difference families to design the new codes. The translates of these sets provide us with the incidence structures necessary for systematic constructions of regular LDPC codes with Tanner graphs free of 4-cycles and with deterministic code properties.

The quasi-cyclic codes in particular have similar properties and performances to the randomly constructed LDPC codes and represent a large family of flexible 4-cycle free, regular LDPC codes with simple linear-time encoding circuits and deterministic construction. Furthermore, the irregular circulants offer a flexible choice of weight distribution which can be alternated depending on the channel. For high noise channels smaller weight circulants produce the best results and with the lowest decoding complexity while in low

noise environments a large portion of high column weight circulants improves the bit error rate.

In general, the majority of the proven performance properties of LDPC codes have been derived for ensembles of finite or infinite length codes with a given degree distribution. While codes from a given ensemble are generally constructed randomly there is no reason to suppose that a random code from a given ensemble is any better than an algebraically constructed code from the same ensemble. In fact, as we have seen for a number of codes in this thesis, algebraic codes can allow access to better ensembles (with better minimum distance or stopping set distribution) than practically available with random constructions. The random constructions do have the advantage of flexibility as, until recently, very few of the known algebraic codes have the properties required for sum-product decoding. This thesis has addressed this shortage by extending the range of LDPC codes which can be constructed algebraically and showing that many of these algebraic codes can outperform the available randomly constructed codes with the same parameters.

8.1 Future directions

The one area where we were unable to find incidence structures with the properties we desired were long designs with low block weight and significantly rank deficient incidence matrices. The small unital, oval and proper partial geometry designs demonstrate that the combination of low block weight and many linearly dependent rows produces excellent LDPC codes. However we were unable to present constructions for rank deficient incidence structures with small column weights at much longer lengths. A partial result was achieved with the transversal designs for which an infinite class of column weight 3 and 4 incidence structures exist for an infinite range of lengths but for which only $\gamma - 1$ linearly dependent rows are guaranteed, where γ is the column weight. Finding long codes with a large number of linearly dependent parity-checks but also with small column weights will also facilitate a better understanding of the effect of rank deficient parity-checks on the sum-product decoding algorithm.

A second area of future interest is the combinatorial construction of non-binary LDPC codes. Non-binary codes can provide performance improvements over random LDPC codes [30], and may also be better suited to some communications channels. Designs are naturally described by binary matrices, representing incidence, and sparse non-binary matrices are not normally associated with this field. However, recent interest in non-binary designs such as generalized Bhaskar Rao designs [92] may provide useful structures for non-binary LDPC codes.

Finally, in this thesis we have limited our consideration to codes described by incidence

graphs with simple even parity-check constraints. It is highly likely that better codes can be produced using subcode constraints [100] as has been done very successfully for codes from generalized quadrangles in [118, 117].

CONSTRUCTION OF DESIGNS

For completeness we present the construction methods used for the designs we require for our algebraic codes.

A.1 Steiner 2-designs

Constructions for three Steiner 2-designs are presented: projective geometries, ovals and unitals.

A.1.1 Projective geometries

In this section we present projective geometries which are used in the construction of both oval and unital designs. Further details on projective geometries can be found in [3, 4]; our treatment of projective geometries follows Anderson [3].

Consider the set S of triples $\mathbf{x} = (x, y, z)$ of elements of the finite field $\text{GF}(q)$, where (x, y, z) are not all zero. S has $q^3 - 1$ members, but we identify triples \mathbf{x} and \mathbf{y} if $\mathbf{x} = \kappa\mathbf{y}$ for some non-zero element $\kappa \in \text{GF}(q)$, and say that \mathbf{x} and \mathbf{y} are equivalent. Denote the equivalence class of \mathbf{x} by $[\mathbf{x}]$. Each equivalence class has $q - 1$ members, corresponding to the $q - 1$ possible non-zero values of κ , and so there are $(q^3 - 1)/(q - 1) = q^2 + q + 1$ different classes $[\mathbf{x}]$, which we take as the points of $PG(2, q)$.

Next define the blocks (or *lines*) as follows: If $\alpha = (\alpha_0, \alpha_1, \alpha_2)$ is a triple of elements of $\text{GF}(q)$, not all zero, define the line $[\alpha]$ to be the set of all points such that $\alpha_0x + \alpha_1y + \alpha_2z = 0$. By an argument similar to the one for points, there are $q^2 + q + 1$ blocks. To see that there are $q + 1$ points on each line, consider the line $[\alpha]$ where $\alpha = (\alpha_0, \alpha_1, \alpha_2)$. Not all the α_i are zero, so suppose for example that $\alpha_1 \neq 0$. Then, if $[\mathbf{x}]$ is on $[\alpha]$, x_1 is uniquely determined by x_0 and x_2 , where x_0 and x_2 cannot both be zero. There are $q^2 - 1$ choices of x_0 and x_2 , so there are $q^2 - 1$ vectors $x \neq 0$ satisfying $\alpha_0x + \alpha_1y + \alpha_2z = 0$, and hence there are $(q^2 - 1)/(q - 1) = q + 1$ distinct points $[\mathbf{x}]$ on $[\alpha]$.

As an example, we construct the finite projective plane $\text{PG}(2,2^2)$, which is also a 2 - $(21, 5, 1)$ design. Here we use the field $\text{GF}(4)$, which can be thought of as the elements $\{0, 1, \alpha, \alpha + 1\}$, where $\alpha^2 = \alpha + 1$. Writing β in place of $\alpha + 1$, so that $\alpha\beta = \alpha(\alpha + 1) = \alpha^2 + \alpha = 1$ and $\beta^2 = (\alpha + 1)^2 = \alpha^2 + 1 = \alpha$, and omitting brackets and commas, the 21 points can be written as

$$\begin{aligned} &100, \quad 010, \quad 001, \quad 1\alpha\alpha, \quad 1\beta0, \quad 01\beta, \quad 1\alpha1, \quad 101, \quad 10\alpha, \quad 1\beta\alpha, \quad 1\beta1, \\ &10\beta, \quad 11\alpha, \quad 1\beta\beta, \quad 110, \quad 011, \quad 1\alpha0, \quad 01\alpha, \quad 1\alpha\beta, \quad 11\beta, \quad 111. \end{aligned}$$

Note that there are 21 points, and not $4^3 - 1 = 63$, since we identify points that differ only by a scalar multiple. Thus, for example, 01β and $0\alpha1$ define the same point since $\alpha(0, 1, \beta) = (0, \alpha, 1)$. Similarly, there are 21 lines in $\text{PG}(2,2^2)$:

$$\begin{aligned} [100] : & \quad 010 \quad 001 \quad 01\beta \quad 011 \quad 01\alpha, \\ [010] : & \quad 100 \quad 001 \quad 101 \quad 10\alpha \quad 10\beta, \\ [001] : & \quad 100 \quad 010 \quad 1\beta0 \quad 110 \quad 1\alpha0, \\ [110] : & \quad 001 \quad 11\alpha \quad 110 \quad 11\beta \quad 111, \\ [011] : & \quad 100 \quad 1\alpha\alpha \quad 1\beta\beta \quad 011 \quad 111, \\ [101] : & \quad 010 \quad 1\alpha1 \quad 101 \quad 1\beta1 \quad 111, \\ [1\alpha0] : & \quad 001 \quad 1\beta0 \quad 1\beta\alpha \quad 1\beta1 \quad 1\beta\beta, \\ [01\alpha] : & \quad 100 \quad 01\beta \quad 1\alpha1 \quad 1\beta\alpha \quad 11\beta, \\ [10\alpha] : & \quad 010 \quad 10\beta \quad 1\beta\beta \quad 1\alpha\beta \quad 11\beta, \\ [1\beta0] : & \quad 001 \quad 1\alpha\alpha \quad 1\alpha1 \quad 1\alpha0 \quad 1\alpha\beta, \\ [01\beta] : & \quad 100 \quad 1\beta1 \quad 11\alpha \quad 01\alpha \quad 1\alpha\beta, \\ [10\beta] : & \quad 010 \quad 1\alpha\alpha \quad 10\alpha \quad 1\beta\alpha \quad 11\alpha, \\ [11\alpha] : & \quad 1\alpha\alpha \quad 01\beta \quad 1\beta1 \quad 10\beta \quad 110, \\ [11\beta] : & \quad 1\alpha1 \quad 10\alpha \quad 1\beta\beta \quad 110 \quad 01\alpha, \\ [1\alpha1] : & \quad 1\alpha\alpha \quad 1\beta0 \quad 101 \quad 01\alpha \quad 11\beta, \\ [1\beta1] : & \quad 01\beta \quad 101 \quad 11\alpha \quad 1\beta\beta \quad 1\alpha0, \\ [\alpha11] : & \quad 10\alpha \quad 011 \quad 1\beta1 \quad 1\alpha0 \quad 11\beta, \\ [\beta11] : & \quad 1\beta0 \quad 1\alpha1 \quad 10\beta \quad 11\alpha \quad 011, \\ [1\alpha\beta] : & \quad 1\beta0 \quad 01\beta \quad 10\alpha \quad 1\alpha\beta \quad 111, \\ [1\beta\alpha] : & \quad 1\beta\alpha \quad 10\beta \quad 1\alpha0 \quad 01\alpha \quad 111, \\ [111] : & \quad 101 \quad 1\beta\alpha \quad 110 \quad 011 \quad 1\alpha\beta, \end{aligned}$$

In the above, the five points identified as lying on the line $[1\alpha\beta]$ are, by definition, those points (x, y, z) which satisfy the equation $x + \alpha y + \beta z = 0$.

A.1.2 Ovals

This material on oval designs is essentially Bose and Shrikhande's original presentation [13], using the terminology of Assmus and Key [4, 58]. Oval designs are constructed

from projective planes, and we use the projective plane constructed in Section A.1.1 in our example.

An *oval* in a projective plane of even order q is a set of $q + 2$ points that meet each line of the plane in 0 or 2 points. To construct an oval requires a non-degenerate conic C on a projective plane $\text{PG}(2, 2^m)$, e.g. the conic

$$xz = y^2.$$

There are $q + 1$ points P_1, P_2, \dots, P_{q+1} on this conic, where $q = 2^m$. Through any point $P_i = (x', y', z')$ on the conic there pass $q + 1$ lines, q of which meet the conic in the other q points on the conic, and the remaining line $z'x + x'z = 0$ meets the conic in the single point P_i , and is therefore tangent to C . The $q + 1$ tangents to the conic all pass through the point $P_0 = (0, 1, 0)$ which is called the *nucleus* (also *pole*, or *knot*) of the conic. The $q^2 + q + 1$ lines of the plane may be divided into three classes:

- (a) the $q(q + 1)/2$ *secants*, each of which meets the conic in 2 points, neither of which is P_0 ;
- (b) the $q + 1$ *tangents*, each of which meets the conic in one point and passes through P_0 ; and
- (c) the $q(q - 1)/2$ *exterior lines* which do not meet the conic and hence do not pass through P_0 .

A regular oval \mathcal{O} in the projective plane $\text{PG}(2, q)$ is now formed by taking the $q + 1$ points P_1, P_2, \dots, P_{q+1} on the conic, together with the nucleus P_0 . The $q^2 - 1$ points of the plane other than P_0 and the points of the conic are called *retained points*.

Oval designs can now be defined: let Π be a projective plane of even order q , and let \mathcal{O} be an oval of Π . The *oval design* $W(\Pi, \mathcal{O})$ is the incidence structure having for points the lines of Π exterior to \mathcal{O} , and for blocks the points of Π not on the oval \mathcal{O} , namely the retained points. Incidence is given by the incidence in Π ; that is, in an oval design, a point is considered to belong to a block if the corresponding line and point in $\text{PG}(2, 2^m)$ are incident. It is easy to show that oval designs are Steiner systems with parameters $2 - (q(q - 1)/2, q/2, 1)$; see [4, Chapter 8]. The family of oval designs constructed in this way have the following parameters:

$$v = q(q - 1)/2, \quad b = q^2 - 1, \quad \rho = q + 1, \quad \gamma = q/2, \quad \lambda = 1,$$

where $q = 2^m$.

To complete the example above, we take the conic $xz = y^2$ defined on points (x, y, z) of the projective plane $\text{PG}(2, 2^2)$ from Section A.1.1. There are $q + 1 = 5$ points P_1, P_2, \dots, P_5

exterior lines	retained points				
[11 α]	1 $\alpha\alpha$	01 β	1 β 1	10 β	110
[11 β]	1 α 1	10 α	1 $\beta\beta$	110	01 α
[1 α 1]	1 $\alpha\alpha$	1 β 0	101	01 α	11 β
[1 β 1]	01 β	101	11 α	1 $\beta\beta$	1 α 0
[α 11]	10 α	011	1 β 1	1 α 0	11 β
[β 11]	1 β 0	1 α 1	10 β	11 α	011

Table A.1: The exterior lines and retained points of an oval design

on this conic:

$$100, \quad 001, \quad 1\alpha\beta, \quad 1\beta\alpha, \quad 111,$$

and these points, together with the nucleus $P_0 = 010$, form an oval \mathcal{O} . The $q^2 + q + 1 = 21$ lines of the plane are divided into three classes:

(a) the $q(q + 1)/2 = 10$ secants:

$$\begin{aligned} & [010], \quad [011], \quad [110], \quad [01\alpha], \quad [1\alpha0], \\ & [01\beta], \quad [1\alpha\beta], \quad [111], \quad [1\beta\alpha], \quad [1\beta0] \end{aligned}$$

(b) the $q + 1 = 5$ tangents:

$$[100], \quad [001], \quad [101], \quad [10\alpha], \quad [10\beta]$$

(c) the $q(q - 1)/2 = 6$ exterior lines:

$$[11\alpha], \quad [11\beta], \quad [1\alpha1], \quad [1\beta1], \quad [\alpha11], \quad [\beta11].$$

To construct the oval design we take the points exterior to the oval as blocks and the non exterior lines as points. The set of exterior lines and their corresponding retained points are shown in Table A.1. We take as points the 6 lines of $\text{PG}(2,2^2)$ that are exterior to the oval, and as blocks the set of $q^2 + q + 1 - (q + 2) = q^2 - 1 = 15$ retained points. Thus if we take the 6 exterior lines in the order listed above, and the 15 retained points in the following order:

$$\begin{aligned} & 010, \quad 1\alpha\alpha, \quad 1\beta0, \quad 01\beta, \quad 1\alpha1, \quad 101, \quad 10\alpha, \quad 1\beta1, \\ & 10\beta, \quad 11\alpha, \quad 1\beta\beta, \quad 110, \quad 011, \quad 1\alpha0, \quad 01\alpha, \quad 11\beta. \end{aligned}$$

we obtain the incidence matrix of the $2-(6, 2, 1)$ oval design,

secant/tangent	retained points on secant/tangent
[01 α]	01 β 1 α 1 11 β
[010]	101 10 α 10 β
[01 β]	1 β 1 11 α 01 α
[011]	1 $\alpha\alpha$ 1 $\beta\beta$ 011
[001]	1 β 0 110 1 α 0

Table A.2: Construction of the resolution classes of an oval design

$$N = \begin{bmatrix} \cdot & \cdot & 1 & \cdot & \cdot & 1 & 1 & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & 1 & \cdot \\ 1 & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & 1 & 1 & \cdot & \cdot & \cdot & 1 & \cdot \\ \cdot & 1 & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & 1 & 1 & \cdot & \cdot \\ \cdot & 1 & \cdot & \cdot & 1 & \cdot & 1 & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & 1 \\ 1 & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & 1 & \cdot & \cdot & 1 & \cdot & 1 & \cdot & \cdot \\ \cdot & \cdot & 1 & 1 & \cdot & \cdot & \cdot & 1 & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & 1 \end{bmatrix}.$$

The first column, for example, reflects that the point 1 $\alpha\alpha$ is incident with the two lines [11 α], [1 α 1] exterior to the oval.

A.1.3 Resolvable ovals

Oval designs are resolvable with a distinct resolution defined by each point P on \mathcal{O} : the $q - 1$ blocks corresponding to the retained points on a secant or tangent through P form a single parallel class, and the full set of $q + 1$ parallel classes forms the resolution [58].

Choosing point $P = 100$ on \mathcal{O} leads to a resolvable oval design as follows: $q + 1 = 5$ members of the set of secants and tangents listed above intersect point P , and each such line contains $q - 1 = 3$ retained points. The set of secant/tangent lines and their corresponding retained points are shown in Table A.2.

We now take as points of the oval design the 6 exterior lines, and as blocks the set of $q^2 + q + 1 - (q + 2) = (sq + 1)(q - 1) = 15$ retained points, taken from Table A.2 in the natural (left-to-right, top-to-bottom) order. Each row of Table A.2 therefore constitutes a resolution class of the design, and we obtain the incidence matrix of the 2-(6, 2, 1) oval

design:

$$N = \begin{bmatrix} 1 & . & . & 1 & . & . & . & . & 1 & . & 1 & . & 1 & . & . \\ . & . & 1 & . & . & 1 & . & . & 1 & . & . & 1 & . & . & 1 \\ . & 1 & . & 1 & . & . & 1 & . & . & . & . & 1 & . & 1 & . \\ . & . & 1 & . & 1 & . & . & 1 & . & . & 1 & . & . & 1 & . \\ . & 1 & . & . & . & 1 & . & 1 & . & 1 & . & . & 1 & . & . \\ 1 & . & . & . & 1 & . & 1 & . & . & 1 & . & . & . & . & 1 \end{bmatrix}.$$

The first column, for example, reflects that the retained point 01β is incident with the two lines $[11\alpha]$, $[1\beta 1]$ exterior to the oval.

A.1.4 Unitals

The unitals presented here are derived from Hermitian unitals constructed using the method of [58]. Unital designs are constructed from projective planes, and we use the projective plane constructed in Section A.1.1 in our example.

An *unitary polarity* in a projective plane of even order $q = m^2$ is a set of points of the plane with cardinality $m^3 + 1$ and the property that every line of the plane meets the set in 1 or $m + 1$ points.

For the projective plane presented in Section A.1.1, the set of points

$$110, \quad 011, \quad 1\alpha 0, \quad 01\alpha, \quad 1\beta 0, \quad 01\beta, \quad 101, \quad 10\alpha, \quad 10\beta$$

form a unitary polarity. The lines

$$[11\beta], \quad [010], \quad [1\beta 1], \quad [\beta\alpha 1], \quad [1\alpha 1], \quad [111],$$

$$[001], \quad [\alpha 11], \quad [100], \quad [\beta 11], \quad [11\alpha], \quad [1\alpha\beta],$$

all contain 3 of the points in the unitary polarity and all other lines contain one of these points.

A *unital design* or *unital* has as points the point set of a unitary polarity and for blocks those lines in the projective plane that meet the point set of the unitary polarity in $m + 1$ points [4]. The points and blocks of the design retain the incidence of the points and lines of the geometry. Thus a unital design is a *Steiner 2-design* with parameters $2-(m^3 + 1, m + 1, 1)$. That is, a unital design consists of $b = m^2(m^3 + 1)/(m + 1)$ subsets, called blocks, of a set of $v = m^3 + 1$ points with the property that every point is contained in $r = m^2$ blocks, every block contains $\gamma = m + 1$ points and every pair of points is contained in exactly one block together.

The unitary polarity defined above on the plane $\text{PG}(2^2)$ produces a 2 - $(9, 3, 1)$ unital design with incidence matrix is

$$N = \begin{bmatrix} 1 & . & . & . & . & 1 & 1 & . & . & . & 1 & . \\ . & . & . & . & . & 1 & . & 1 & 1 & 1 & . & . \\ . & . & 1 & 1 & . & . & 1 & 1 & . & . & . & . \\ 1 & . & . & 1 & 1 & . & . & . & 1 & . & . & . \\ . & . & . & . & 1 & . & 1 & . & . & 1 & . & 1 \\ . & . & 1 & . & . & . & . & . & 1 & . & 1 & 1 \\ . & 1 & 1 & . & 1 & 1 & . & . & . & . & . & . \\ 1 & 1 & . & . & . & . & . & 1 & . & . & . & 1 \\ . & 1 & . & 1 & . & . & . & . & . & 1 & 1 & . \end{bmatrix}$$

A.2 Proper partial geometries

The known constructions for proper partial geometries are given in Table 5.3. We present the construction for the first class in the table which are due to Thas [107]. These designs are constructed from projective planes and we use the projective plane constructed in Section A.1.1 in our example.

An (m, k) -arc in a projective plane of order q is a set of m points, no k of which are collinear. The arc is perfect if

$$m = (q + 1)(k - 1) + 1.$$

To construct an $((q + 1)(k - 1) + 1, k)$ -arc, let

$$f(y, z) = ay^2 + byz + cz^2$$

be an irreducible quadratic over $\text{GF}(q)$ and let H be any sub group of the additive group of $\text{GF}(q)$ with order d . In the affine plane, $\text{AG}(2, q)$, embedded in $\text{PG}(2, q)$, define

$$A := \{(y, z) : f(y, z) \in H\}.$$

Any affine line meets A in 0 or d points and A is a perfect $((q + 1)(d - 1) + 1, d)$ -arc.

The partial geometries $\text{pg}(s, t, \alpha)$ of this construction have the parameters

$$s = q - d, \quad t = q(d - 1), \quad \alpha = (q - 1)(d - 1)/d,$$

and require an $((q + 1)(d - 1) + 1, d)$ -arc defined in a projective plane of order q . Such an arc exists for all $q = 2^h$, $d = 2^m$, for h, m , any integers so long as $h > m$. The points of the partial geometry are the points of $\text{PG}(2, q)$ that are not contained in the arc and the lines of the partial geometry are the lines of $\text{PG}(2, q)$ that are incident with k points of the arc where the incidence of is that of $\text{PG}(2, q)$.

As an example we construct the $\text{pg}(2,2,1)$. First we find an $\text{EG}(2,2^2)$ embedded in the $\text{PG}(2,2^2)$, by removing one line from $\text{PG}(2,2^2)$ and all the points through it. By omitting all points on the line $x = 0$ from $\text{PG}(2,2^2)$ we see that all remaining points have $x = 1$ (see Section A.1.1) and so we can represent the points of $\text{EG}(2,2^2)$ by the shortened (y, z) . Thus $\text{EG}(2,2^2)$ contains the points:

$$\begin{array}{cccccccc} 00, & \alpha\alpha, & \beta0, & \alpha1, & 01, & 0\alpha, & \beta\alpha, & \beta1, \\ 0\beta, & 1\alpha, & \beta\beta, & 10, & \alpha0, & \alpha\beta, & 1\beta, & 11, \end{array}$$

and has lines which are all the lines of $\text{PG}(2,2^2)$ except $[100]$. For example the line $[1\alpha\beta]$ from $\text{PG}(2,2^2)$, is

$$[1\alpha\beta] : \beta0 \quad 0\alpha \quad \alpha\beta \quad 11$$

in $\text{EG}(2,2^2)$. Next we choose $H = \{0,1\}$, and use the irreducible quadratic

$$f(y, z) = \alpha y^2 + yz + z^2.$$

Then the points (y, z) in $\text{EG}(2,2^2)$ for which $f(y, z) = 0$ or 1 , give us our arc:

$$A := 00, \quad \alpha\alpha, \quad 01, \quad \beta\alpha, \quad \beta1, \quad \alpha0.$$

Finally the points of the partial geometry are all the points in $\text{PG}(2,2^2)$ other than those in A :

$$\begin{array}{cccccccc} 010, & 001, & 1\beta0, & 01\beta, & 1\alpha1, & 10\alpha, & 10\beta, \\ 11\alpha, & 1\beta\beta, & 110, & 011, & 01\alpha, & 1\alpha\beta, & 11\beta, & 111, \end{array}$$

and the lines of the partial geometry are the lines of the $\text{PG}(2,2^2)$ which contain two points of A . It is easy to check the set of lines in $\text{PG}(2,2^2)$ to find the subset which intersect A twice, they are:

$$\begin{array}{cccccccc} [010], & [001], & [011], & [101], & [1\alpha1], & [01\alpha], & [1\beta0], \\ [01\beta], & [10\beta], & [11\alpha], & [1\alpha1], & [1\beta1], & [\alpha11], & [1\beta\alpha], & [111], \end{array}$$

The point set is $\mathcal{H} = GF(q) \times Z_3$ and the mixed difference system consists of the sets:

$$\begin{aligned} A &= \{0_1, 0_2, 0_3\}, \\ B_{i,j} &= \{\theta_j^i, \theta_j^{i+2m}, \theta_j^{i+4m}\}, \quad 1 \leq i \leq m \\ C_{i,j} &= \{\theta_j^{i+m}, \theta_{j+1}^{i+3m}, \theta_{j+2}^{i+5m}\}, \quad 1 \leq i \leq m \\ D_{i,j} &= \{\theta_j^i, \theta_{j+1}^{i+2m}, \theta_{j+2}^{i+4m}\}, \quad 1 \leq i \leq m \end{aligned}$$

for $1 \leq j \leq 3$, where θ_j^i is $(\theta^i, j) \in \mathcal{H}$. The sets $A, B_{i,j}$ and $C_{i,j}$ of the mixed difference system make up the blocks of one resolution class of a design, and each translate of these sets gives a further resolution class. Next, each set $D_{i,j}$ with its translates give a resolution class; so we obtain a total of $9m + 1$ resolution classes and we have a $KTS(3q)$.

Example A.3.1 Take $\mathcal{H} = GF(7) \times Z_3$, $m = 1$, $q = 7$ and $v = 21$. Choose $\theta = 3$ and the mixed difference system is:

$$\begin{aligned} A &= \{0_1, 0_2, 0_3\}, \\ B &= \{3_1, 6_1, 5_1\}, \quad \{3_2, 6_2, 5_2\}, \quad \{3_3, 6_3, 5_3\}, \\ C &= \{2_1, 4_2, 1_3\}, \quad \{2_2, 4_3, 1_1\}, \quad \{2_3, 4_1, 1_2\}, \\ D &= \{3_1, 3_2, 3_3\}, \quad \{6_2, 6_3, 6_1\}, \quad \{5_3, 5_1, 5_2\}. \end{aligned} \tag{A.1}$$

The sets A , B , and C make up the blocks of the first resolution class of the design and the 6 translates of these sets make up the blocks of the next 6 resolution classes. The blocks in the second resolution class (the translate of A , B and C with $g = 1$) are:

$$\begin{aligned} \{1_1, 1_2, 1_3\}, \quad \{4_1, 0_1, 6_1\}, \quad \{4_2, 0_2, 6_2\}, \quad \{4_3, 0_3, 6_3\}, \\ \{3_1, 5_2, 2_3\}, \quad \{3_2, 5_3, 2_1\}, \quad \{3_3, 5_1, 2_2\}. \end{aligned}$$

Next, the translates of each block in D make up a resolution class, for the first block this class is:

$$\begin{aligned} \{3_1, 3_2, 3_3\}, \quad \{4_1, 4_2, 4_3\}, \quad \{5_1, 5_2, 5_3\}, \quad \{6_1, 6_2, 6_3\}, \\ \{0_1, 0_2, 0_3\}, \quad \{1_1, 1_2, 1_3\}, \quad \{2_1, 2_2, 2_3\}. \end{aligned}$$

Altogether there are 10 resolution classes, each with 7 blocks, to give the $KTS(21,70,10,3,1)$ -design.

Construction A.3.2 [3, Theorem 9.1.5] Let $q = 6m + 1$ be a prime power, m an integer and take $\mathcal{H} = GF(q) \times Z_t \cup \infty$. Choose θ , a primitive element of $GF(q)$, so that $\theta^{6m} = 1$

and $\theta^{3m} = -1$, and choose an integer u , so that $\theta^m + 1 = 2\theta^u$. Then the sets:

$$\begin{aligned} A &= \{0_1, 0_2, \infty\}, \\ B_i &= \{\theta_2^{i+u+m}, \theta_2^{i+u+3m}, \theta_2^{i+u+5m}\}, \quad 0 \leq i \leq m-1, \\ C_i &= \{\theta_1^i, \theta_1^{i+m}, \theta_2^{i+u}\}, \quad 0 \leq i \leq m-1, \\ D_i &= \{\theta_2^{i+2m+u}, \theta_1^{i+2m}, \theta_1^{i+3m}\}, \quad 0 \leq i \leq m-1, \\ E_i &= \{\theta_2^{i+4m+u}, \theta_1^{i+4m}, \theta_1^{i+5m}\}, \quad 0 \leq i \leq m-1, \end{aligned}$$

form a mixed difference system in \mathcal{H} . The sets of the mixed difference system partition the $2q + 1$ elements of \mathcal{H} and make up the first resolution class. Each translate of the sets give a further resolution class and we have a $KTS(2q + 1)$ design. Note that when forming the translates $g + \infty = \infty$.

Example A.3.2 Take $\mathcal{H} = GF(7) \times Z_2 \cup \infty$ and $m = 1$, $q = 7$ and $v = 15$. Choose $\theta = 3$ and $u = 2$ and the required mixed difference system is:

$$\begin{aligned} A &= \{0_1, 0_2, \infty\}, & B &= \{3_2, 5_2, 6_2\}, \\ C &= \{1_1, 3_1, 2_2\}, & D &= \{2_1, 6_1, 4_2\}, & E &= \{4_1, 5_1, 1_2\}. \end{aligned} \tag{A.2}$$

These sets make up the six blocks of the first resolution class of the $KTS(15)$, and each successive resolution class is obtained by forming translates of these sets. The first translate is

$$\{1_1, 1_2, \infty\}, \{4_2, 6_2, 0_2\}, \{2_1, 4_1, 3_2\}, \{3_1, 0_1, 5_2\}, \{5_1, 6_1, 2_2\},$$

If we take the ordering of the points to be $\{0_1, \dots, 6_1, 0_2, \dots, 6_2, \infty\}$ the first 20 columns of N for this $KTS(15,35,3,7,1)$ -design are shown in Fig. 6.1.

GLOSSARY

Most frequently used symbols

$[n, k, d]$	block code: length n , dimension k and minimum distance d , 9
$2-(v, b, r, \gamma, \lambda)$ or $2-(v, \gamma, \lambda)$	Steiner 2-design, 23
A	adjacency matrix, 27
α	partial geometry parameter, 27
B	block, 23
\mathcal{B}	set of blocks, 23
b	number of blocks, 23
C	error correction code, 12
c	codeword, 10
$c(x)$	polynomial representation of a codeword, 152
\mathcal{D}	combinatorial design, 23
d	minimum Hamming distance, 9
F	finite field, 26
γ	design block size, 23
\mathcal{G}	Abelian group, 25
g	an element of \mathcal{G} , 25
G	generator matrix, 10
$GF(q)$	Galois field with q elements, 9
$g(x)$	generator polynomial, 152
H	parity-check matrix, 10
$h(x)$	parity-check polynomial, 153
I	ideal, 152
\mathcal{I}	incidence relation, 23
J	all ones matrix, 97

k	code dimension, 9
λ	t -design parameter, 23
m	number of parity-checks, 12
N	point-by-block incidence matrix, 23
n	code length, 9
n_1	strongly regular graph parameter, 28
N_6	number of 6-cycles, 36
N_8	number of 8-cycles, 95
O	orthogonal array, 106
\mathcal{O}	oval design, 65
\oplus	modulo-2 addition, 10
P	point, 23
\mathcal{P}	set of points, 23
p_1	strongly regular graph parameter, 28
p_2	strongly regular graph parameter, 28
P_i^{ext}	extrinsic probability, 12
$\text{PG}(m, q)$	projective geometry, 26
$\text{pg}(s, t, \alpha)$	partial geometry, 27
P_i^{int}	a priori probability, 12
R	code rate, 9
ρ	number of resolution classes in a KTS code, 124
r	point degree of a design, 23
$\text{rank}_q(H)$	rank of H over $\text{GF}(q)$, 11
R_n	ring of polynomials degree n , 152
S_{\min}	minimum stopping set size, 20
s	partial geometry parameter, 27
θ	primitive element, 127
t	partial geometry parameter, 27
V	vector space, 26
v	number of points, 23
w_c	column weight of H , 11
w_r	row weight of H , 11
x	transmitted codeword, 13

y	received vector, 13
Z_v	cyclic group, 25
z	bit-wise hard decision of the received vector, 13

Abbreviations

APP	a posteriori probability
AWGN	additive white Gaussian noise
BER	bit error rate
BEC	binary erasure channel
KTS	Kirkman triple system
KQS	Kirkman quadruple system
LDPC	low-density parity-check
LLR	log-likelihood ratio
MOLS	mutually orthogonal Latin squares
PG	partial geometry
SNR	signal-to-noise ratio
STS	Steiner triple system
TD	transversal design

BIBLIOGRAPHY

- [1] Special issue on Codes on Graphs and Iterative Algorithms. *IEEE Trans. Inform. Theory*, vol. 47, no. 2, February 2001.
- [2] B. Ammar, B. Honary, Y. Kou, and S. Lin. Construction of low density parity check codes: A combinatorial design approach. In *Proc. International Symposium on Information Theory (ISIT'2002)*, page 311, Lausanne, Switzerland, June 30 – July 5 2002.
- [3] I. Anderson. *Combinatorial Designs: Construction Methods*. Mathematics and its Applications. Ellis Horwood, Chichester, 1990.
- [4] E. F. Assmus, Jr. and J. D. Key. *Designs and their Codes*, volume 103 of *Cambridge Tracts in Mathematics*. Cambridge University Press, Cambridge, U.K., 1993.
- [5] E. F. Assmus, Jr. and H. F. Mattson, Jr. New 5-designs. *J. Combin. Theory*, 6:122–151, 1969.
- [6] C. Berrou and A. Glavieux. Near optimum error correcting coding and decoding: Turbo codes. *IEEE Trans. Commun.*, 44(10):1261–1271, October 1996.
- [7] C. Berrou, A. Glavieux, and P. Thitimajshima. Near Shannon limit error-correcting coding and decoding: Turbo-codes. In *Proc. IEEE Int. Conf. on Communications (ICC'93)*, pages 1064–1070, Geneva, Switzerland, May 1993.
- [8] V. Bhagavatula, S. Hongwei, and L. Jingfeng. Low density parity check (LDPC) codes for optical data storage. In *Proc. International Symposium on Optical Memory and Optical Data Storage, Topical Meeting*, pages 371–373, 7-11 July 2002.
- [9] A. J. Blanksby and C. J. Howland. A 690-mW 1-Gb/s 1024-b, rate-1/2 low-density parity-check code decoder. *IEEE J. Solid-State Circuits*, 37(3):404–412, March 2002.
- [10] J. W. Bond, S. Hui, and H. Schmidt. Constructing low-density parity-check codes with circulant matrices. In *Proc. IEEE Information Theory Workshop (ITW1999)*, page 52, Metsovo, Greece, June 27 – July 1 1999.
- [11] R. C. Bose. On the application of the properties of Galois fields to the construction of hyper-Graeco-Latin squares. *Sankhya*, 3:323–338, 1938.

-
- [12] R. C. Bose. Strongly regular graphs, partial geometries and partially balanced designs. *Pacific J. Math.*, 13:389–419, 1963.
- [13] R. C. Bose and S. S. Shrikhande. On the construction of sets of mutually orthogonal Latin squares and the falsity of a conjecture of Euler. *Trans. Amer. Math. Soc.*, 95:191–209, 1960.
- [14] W. Bosma, J. Cannon, and C. Playoust. The Magma algebra system I: The user language. *J. Symbolic Computation*, 24(3/4):235–265, September 1997.
- [15] A. E. Brouwer and C. A. van Eijl. On the p -rank of strongly regular graphs. *Algebra and Combinatorics*, 1:72–82, April 1992.
- [16] F. Buekenhout. *Handbook of Incidence Geometry*. North-Holland, Amsterdam, The Netherlands, 1995.
- [17] M. Buratti. Constructions of $(q, k, 1)$ difference families with q a prime power and $k = 4, 5$. *Discrete Mathematics*, 138(1-3):169–175, 1995.
- [18] J. W. Byers, M. Luby, and M. Mitzenmacher. A digital fountain approach to asynchronous reliable multicast. *IEEE J. Selected Areas Commun.*, 20(8):1528–1540, October 2002.
- [19] P. J. Cameron and J. H. van Lint. *Graphs, Codes and Designs*. London Mathematical Society Lecture Note Series, No. 43. Cambridge University Press, Cambridge, 1980.
- [20] L. L. Carpenter. Oval designs in desarguesian projective planes. *Designs, Codes and Cryptography*, 9(1):51–59, August 1996.
- [21] Y. M. Chee, C. J. Colbourn, and A. C. H. Ling. Asymptotically optimal erasure-resilient codes for large disk arrays. *Discrete Appl. Math.*, 102(1–2):3–36, May 2000.
- [22] K. Chen, R. Wei, and L. Zhu. Existence of $(q, 7, 1)$ difference families with q a prime power. *J. Comb. Designs*, 10(2):126–138, 2002.
- [23] K. Chen and L. Zhu. Existence of $(q, 6, 1)$ difference families with q a prime power. *Designs, Codes and Cryptography*, 15(2):167–173, April 1998.
- [24] K. Chen and L. Zhu. Existence of $(q, k, 1)$ difference families with q a prime power and $k = 4, 5$. *J. Comb. Designs*, 7(1):21–30, 1999.
- [25] S.-Y. Chung, G. D. Forney, Jr., T. J. Richardson, and R. L. Urbanke. On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit. *IEEE Commun. Letters*, 5(2):58–60, February 2001.

- [26] C. J. Colbourn and J. Dinitz (Eds.). *The CRC Handbook of Combinatorial Designs*. CRC Press, Boca Raton, 1996.
- [27] C. J. Colbourn, E. Mendelsohn, A. Rosa, and J. Siran. Anti-mitre Steiner triple systems. *Graphs Combin.*, 10:215–224, 1994.
- [28] C. J. Colbourn and A. Rosa. *Triple Systems*. Oxford University Press, 1999.
- [29] P. Danziger, E. Mendelsohn, M. J. Grannell, and T. S. Griggs. Five-line configurations in Steiner triple systems. *Utilitas Mathematica*, 49:153–159, 1996.
- [30] M. C. Davey and D. J. C. MacKay. Low density parity check codes over $\text{GF}(q)$. *IEEE Commun. Letters*, 2(6):165–167, June 1998.
- [31] F. De Clerk, R. H. Dye, and J. A. Thas. An infinite class of partial geometries associated with the hyperbolic quadratic in $\text{PG}(4n - 1, 2)$. *European J. Combin.*, 1:323–326, 1980.
- [32] P. Delsarte. A geometric approach to a class of cyclic codes. *J. Combin. Theory*, 6:340–358, 1969.
- [33] P. Delsarte. On cyclic codes that are invariant under the general linear group. *IEEE Trans. Inform. Theory*, 16:760–769, 1970.
- [34] C. Di, D. Proietti, I. E. Telatar, T. J. Richardson, and R. L. Urbanke. Finite-length analysis of low-density parity-check codes on the binary erasure channel. *IEEE Trans. Inform. Theory*, 48(6):1570–1579, June 2002.
- [35] C. Di, T. J. Richardson, and R. L. Urbanke. Weight distributions: How deviant can you be? In *Proc. International Symposium on Information Theory (ISIT'2001)*, page 50, Washington, DC, June 24–29 2001.
- [36] E. Eleftheriou and S. Ölçer. Low-density parity-check codes for digital subscriber lines. *Proc. IEEE Int. Conf. on Communications (ICC'2002)*, 3:1752–1757, 2002.
- [37] P. Elias. Error-free coding. *IRE Trans. Inform. Theory*, PGIT-4:29–37, Sept 1954.
- [38] T. Etzion, A. Trachtenberg, and A. Vardy. Which codes have cycle-free Tanner graphs? *IEEE Trans. Inform. Theory*, 45(6):2173–2181, September 1999.
- [39] J. L. Fan. Array codes as low-density parity-check codes. *Proc. 2nd Int. Symp. on Turbo Codes, Brest, France*, pages 543–546, 4–7 September 2000.
- [40] J. L. Fan. *Constrained Coding and Soft Iterative Decoding*. The Kluwer International Series in Engineering and Computer Science. Kluwer Academic Publishers, 2001.
- [41] M. P. C. Fossorier. Iterative reliability-based decoding of low-density parity check codes. *IEEE J. Selected Areas Commun.*, 19(5):908–917, May 2001.

- [42] R. G. Gallager. Low-density parity-check codes. *IRE Trans. Inform. Theory*, IT-8(1):21–28, January 1962.
- [43] R. G. Gallager. *Low-Density Parity-Check Codes*. MIT Press, Cambridge, MA, 1963.
- [44] M. Genma, M. Mishima, and M. Jimbo. Cyclic resolvability of cyclic Steiner 2-designs. *J. Combin. Designs*, 5(3):177–187, May 1997.
- [45] R. L. Graham and F. J. MacWilliams. On the number of information symbols in difference-set cyclic codes. *Bell Sys. Tech. J.*, 7:1057–1070, September 1966.
- [46] M. J. Grannell, T. S. Griggs, and C. A. Whitehead. The resolution of the anti-Pasch conjecture. *J. Combin. Designs*, 8(4):300–309, July 2000.
- [47] W. H. Haemers. A new partial geometry constructed from the Hoffman-Singleton graph. In P. J. Cameron, J. W. P. Hirschfeld, and D. R. Hughes, editors, *Finite Geometries and Designs, London Mathematical Society Lecture Note Series*, volume 49, pages 119–127. Cambridge University Press, Cambridge, 1981.
- [48] H. Hanani. On quadruple systems. *Canadian Journal of Mathematics*, 12:145–157, 1960.
- [49] H. Hanani, D. K. Ray-Chaudri, and R. M. Wilson. On resolvable designs. *Discrete Math.*, 3:343–357, 1972.
- [50] F. C. Holroyd, K. A. S. Quinn, C. Rowley, and B. S. (Eds) Webb. *Combinatorial designs and their applications*, volume 403 of *Chapman & Hall/CRC Research Notes in Mathematics*. Chapman & Hall, London, UK, 1999.
- [51] P. Horak, N. K. C. Phillips, W. D. Wallis, and J. L. Yucas. Counting frequencies of configurations in Steiner triple systems. *Ars Combinatoria*, 46:65–75, 1997.
- [52] D. Hösli, E. Svensson, and D. Arnold. High-rate low-density parity-check codes: Construction and application. *Proc. 2nd Int. Symp. on Turbo Codes, Brest, France*, pages 447–450, 4–7 September 2000.
- [53] S. J. Johnson and S. R. Weller. Regular low-density parity-check codes from combinatorial designs. In *Proc. IEEE Information Theory Workshop (ITW2002)*, pages 90–92, Cairns, Australia, September 2001.
- [54] S. J. Johnson and S. R. Weller. Can cyclic codes be useful low-density parity-check codes? In *Proc. Australian Communications Theory Workshop (AusCTW'03)*, pages 81–86, Melbourne, Australia, 5–7 February 2003.
- [55] M. Karlin. New binary coding results by circulants. *IEEE Trans. Inform. Theory*, IT-15(1):81–92, 1969.

- [56] K. Karplus and H. Krit. A semi-systolic decoder for the PDSC-73 error-correcting code. *Discrete Applied Math*, 33(1–3):109–128, November 1991.
- [57] P. Kaski and P. R. J. Östergård. The Steiner triple systems of order 19. To appear, *Mathematics of Computation*, available at <http://tltpc54.hut.fi>.
- [58] J. D. Key. Some applications of Magma in designs and codes: Oval designs, Hermitian unitals and generalized Reed–Muller codes. *J. Symbolic Computation*, 31(1/2):37–53, January/February 2001.
- [59] Y. Kou, S. Lin, and M. P. C. Fossorier. Low-density parity-check codes: Construction based on finite geometries. In *Proc. IEEE Globecom Conf.*, pages 825–829, San Francisco, CA, November 2000.
- [60] Y. Kou, S. Lin, and M. P. C. Fossorier. Low-density parity-check codes based on finite geometries: A rediscovery and new results. *IEEE Trans. Inform. Theory*, 47(7):2711–2736, November 2001.
- [61] Y. Kou, J. Xu, H. Tang, S. Lin, and K. Abdel-Ghaffar. On circulant low density parity check codes. In *Proc. International Symposium on Information Theory (ISIT'2002)*, page 200, Lausanne, Switzerland, June 30 – July 5 2002.
- [62] J. Lafferty and D. Rockmore. Codes and iterative decoding on algebraic expander graphs. In *Proc. International Symposium on Information Theory and its applications (ISITA2000)*, Hawaii, USA, November 5–8 2000.
- [63] C. W. H. Lam and Y. Miao. On cyclically resolvable cyclic Steiner 2-designs. *J. Comb. Theory Ser. A*, 85:194–207, 1999.
- [64] C. W. H. Lam and Y. Miao. Cyclically resolvable cyclic Steiner triple systems of order 21 and 39. *Discrete Mathematics*, 219:173–185, 2000.
- [65] S. Lin and D. Costello, Jr. *Error Control Coding: Fundamentals and Applications*. Prentice-Hall Series in Computer Applications in Electrical Engineering. Prentice-Hall, Inc., Englewood Cliffs, N. J. 07632, 1983.
- [66] S. Lin, H. Tang, and Y. Kou. On a class of finite geometry low density parity check codes. In *Proc. International Symposium on Information Theory (ISIT'2001)*, page 2, Washington, DC., June 24 – 29 2001.
- [67] S. Lin, H. Tang, Y. Kou, J. Xu, and K. Abdel-Ghaffar. Codes on finite geometries. In *Proc. IEEE Information Theory Workshop (ITW2002)*, pages 14–16, Cairns, Australia, September 2001.
- [68] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman. Efficient erasure correcting codes. *IEEE Trans. Inform. Theory*, 47(2):569–584, February 2001.

- [69] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman. Improved low-density parity-check codes using irregular graphs. *IEEE Trans. Inform. Theory*, 47(2):585–598, February 2001.
- [70] R. Lucas, M. P. C. Fossorier, Y. Kou, and S. Lin. Iterative decoding of one-step majority logic decodable codes based on belief propagation. *IEEE Trans. Commun.*, 48(6):931–937, June 2000.
- [71] D. J. C. MacKay. Good error-correcting codes based on very sparse matrices. *IEEE Trans. Inform. Theory*, 45(2):399–431, March 1999.
- [72] D. J. C. MacKay and M. C. Davey. Evaluation of Gallager codes for short block length and high rate applications. In B. Marcus and J. Rosenthal, editors, *Codes, Systems and Graphical Models; volume 123 of IMA Volumes in Mathematics and its Applications*, pages 113–130. Springer-Verlag, New York, 2000. Available from (<http://wol.ra.phy.cam.ac.uk/mackay/CodesRegular.html>).
- [73] D. J. C. MacKay and R. M. Neal. Near Shannon limit performance of low density parity check codes. *Electron. Lett.*, 32(18):1645–1646, March 1996. Reprinted *Electron. Lett.*, vol. 33(6), pp. 457–458, March 1997.
- [74] G. A. Margulis. Explicit constructions for graphs without short cycles and low density codes. *Combinatorica*, 2(1):71–78, 1982.
- [75] J. L. Massey. *Threshold Decoding*. M.I.T. Press, Cambridge, Massachusetts, 1963.
- [76] R. A. Mathon. A new family of partial geometries. *Geometriae Dedicata*, 73:11–19, 1998.
- [77] R. J. McEliece, D. J. C. MacKay, and J.-F. Cheng. Turbo decoding as an instance of Pearl’s “belief propagation” algorithm. *IEEE J. Selected Areas Commun.*, 16(2):140–152, February 1998.
- [78] J. A. McGowan and R. C. Williamson. Removing loops from LDPC codes. In *Proc. Australian Communications Theory Workshop (AusCTW’03)*, pages 140–143, Melbourne, Australia, 5–7 February 2003.
- [79] M. Mishima and M. Jimbo. Some types of cyclically resolvable cyclic Steiner 2-designs. *Congressus Numerantium*, 123:193–203, 1997.
- [80] T. Mittelholzer. Construction of Steiner systems and high-rate LDPC codes. June 2000.
- [81] R. M. Neal. (www.cs.toronto.edu/radford/homepage.html).
- [82] E. Netto. Zur theorie der triplesysteme. *Math Ann.*, 42:143–152, 1893.

- [83] A. Orlitsky, R. L. Urbanke, K. Viswanathan, and J. Zhang. Stopping sets and the girth of Tanner graphs. In *Proc. International Symposium on Information Theory (ISIT'2002)*, page 2, Lausanne, Switzerland, June 30–July 5 2002.
- [84] D. K. Ray-Chaudhuri and R. M. Wilson. Solution of Kirkmans schoolgirl problem. *Proc. Symp. Math.*, 19:187–203, 1971.
- [85] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke. Design of capacity-approaching irregular low-density parity-check codes. *IEEE Trans. Inform. Theory*, 47(2):619–637, February 2001.
- [86] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke. Finite-length analysis of various low-density parity-check ensembles for the binary erasure channel. In *Proc. International Symposium on Information Theory (ISIT'2002)*, page 1, Lausanne, Switzerland, June 30 – July 5 2002.
- [87] T. J. Richardson and R. L. Urbanke. Finite-length density evolution and the distribution of the number of iterations on the binary erasure channel. Unpublished manuscript, available at <http://lthcwww.epfl.ch/papers/RiU02.ps>.
- [88] T. J. Richardson and R. L. Urbanke. The capacity of low-density parity-check codes under message-passing decoding. *IEEE Trans. Inform. Theory*, 47(2):599–618, February 2001.
- [89] T. J. Richardson and R. L. Urbanke. Efficient encoding of low-density parity-check codes. *IEEE Trans. Inform. Theory*, 47(2):638–656, February 2001.
- [90] J. Rosenthal and P. O. Vontobel. Constructions of LDPC codes using Ramanujan graphs and ideas from Margulis. In *Proc. 38th Annual Allerton Conf. on Communications, Control and Computing*, Allerton House, Monticello, IL, U. S. A., October 4–6 2000.
- [91] L. D. Rudolph. A class of majority logic decodable codes. *IEEE Trans. Inform. Theory*, IT-13(2):305–307, April 1967.
- [92] J. Seberry. Bose’s method of differences applied to construct Bhaskar Rao designs. *J. Statist. Plann. Inference*, 73:215–224, 1998.
- [93] C. E. Shannon. A mathematical theory of communication. *Bell Sys. Tech. J.*, 27:379–423, 623–656, July–Oct. 1948.
- [94] T. Shibuya and K. Sakaniwa. Factor graphs for cyclic codes with no cycles of length four. To appear in *IEICE Trans. on Fundamentals*, available at <http://www.nime.ac.jp/tshibuya/publication/data/cycle.pdf>.

- [95] J. Singer. A theorem in finite projective geometry and some applications to number theory. *Trans. Amer. Math. Soc.*, 43:377–385, 1938.
- [96] M. Sipser and D. A. Spielman. Expander codes. *IEEE Trans. Inform. Theory*, 42(6):1710–1722, November 1996.
- [97] V. Sorokine, F. R. Kschischang, and S. Pasupathy. Gallager codes for CDMA applications—Part I: Generalizations, constructions, and performance bounds. *IEEE Trans. Commun.*, 48(10):1660–1668, October 2000.
- [98] D. R. Stinson and Y. J. Wei. Some results on quadrilaterals in Steiner triple systems. *Discrete Math.*, 105:207–219, 1992.
- [99] H. Tang, J. Xu, Y. Kou, S. Lin, and K. Abdel-Ghaffar. On algebraic construction of Gallager low density parity check codes. In *Proc. International Symposium on Information Theory (ISIT'2002)*, page 482, Lausanne, Switzerland, June 30 – July 5 2002.
- [100] R. M. Tanner. A recursive approach to low complexity codes. *IEEE Trans. Inform. Theory*, IT-27(5):533–547, September 1981.
- [101] R. M. Tanner. A transform theory for a class of group-invariant codes. *IEEE Trans. Inform. Theory*, 34(4):752–775, July 1988.
- [102] R. M. Tanner. Codes with sparse graphs: Transform analysis and constructions. In *Proc. International Symposium on Information Theory (ISIT'1998)*, page 116, Cambridge, MA, USA, August 16–21 1998.
- [103] R. M. Tanner. Transforming quasi-cyclic codes with sparse graphs. unpublished, January 2000.
- [104] R. M. Tanner. Minimum-distance bounds by graph analysis. *IEEE Trans. Inform. Theory*, 47(2):808–821, February 2001.
- [105] R. M. Tanner. Spectral graphs for quasi-cyclic LDPC codes. In *Proc. International Symposium on Information Theory (ISIT'2001)*, page 226, Washington, DC, June 24–29 2001.
- [106] R. M. Tanner, Deepak Sridhara, and Tom Fuja. A class of group-structured LDPC codes. In *Sixth International Symposium on Communication Theory and Applications (ISCTA'2001)*, Ambleside, England, July 15–20 2001.
- [107] J. A. Thas. Construction of partial geometries. *Simon Stevin*, 46:95–98, 1973.
- [108] J. A. Thas. Some results on quadratics and a new class of partial geometries. *Simon Stevin*, 55:129–139, 1981.

- [109] V. D. Tonchev. Quasi-symmetric designs, codes, quadrics, and hyperplane sections. *Geometriae Dedicata*, 48:295–308, 1993.
- [110] V. D. Tonchev and R. S. Weishaar. Steiner triple systems of order 15 and their codes. *J. Statist. Plann. Inference*, 58:207–216, 1997.
- [111] R. L. Townsend and E. J. Weldon. Self-orthogonal quasi-cyclic codes. *IEEE Trans. Inform. Theory*, IT-13(2):183–195, April 1967.
- [112] R. L. Urbanke. *LdpcOpt*: <http://lthcwww.epfl.ch/research/ldpcopt/>.
- [113] J. H. Van Lint and A. Schrijver. Construction of strongly regular graphs, two weight codes and partial geometries by finite fields. *Combinatorica*, 1:63–73, 1981.
- [114] J. H. van Lint and R. M. Wilson. *A Course in Combinatorics*. Cambridge University Press, Cambridge, 1992.
- [115] B. Vasic. Structured iteratively decodable codes based on Steiner systems and their application in magnetic recording. In *Proc. IEEE Globecom Conf.*, pages 2954–2960, San Antonio, TX, November 2001.
- [116] B. Vasic. High-rate low-density parity check codes based on anti-Pasch affine geometries. *Proc. IEEE Int. Conf. on Communications (ICC'2002)*, 3:1332–1336, April 28 – May 2 2002.
- [117] P. Vontobel. *Algebraic coding for iterative decoding*. PhD thesis, Swiss Federal Institute of Technology, Zurich, 2003.
- [118] P. O. Vontobel and R. M. Tanner. Construction of codes based on finite generalized quadrangles for iterative decoding. In *Proc. International Symposium on Information Theory (ISIT'2001)*, page 223, Washington, DC, June 24–29 2001.
- [119] E. J. Weldon. Difference-set cyclic codes. *Bell Sys. Tech. J.*, 7:1045–1055, September 1966.
- [120] S. B. Wicker. *Error Control Systems for Digital Communication and Storage*. Prentice Hall, Upper Saddle River, NJ 07458, 1995.
- [121] R. M. Wilson. Cyclotomy and difference families in elementary Abelian groups. *J. Number Theory*, 4:17–47, 1972.
- [122] J. Xu, Y. Kou, S. Lin, and K. Abdel-Ghaffar. A general class of LDPC finite geometry codes and their performance. In *Proc. International Symposium on Information Theory (ISIT'2002)*, page 309, Lausanne, Switzerland, June 30 – July 5 2002.