Graph Modulation

A Pathway to Ultra-Efficient Communication and Storage for 6G Systems

A Technical Exploration of Advanced Modulation Schemes

William Matthews
University College
Supervised by Prof. J.P. Coon

May 2, 2019

Student Submission, University of Oxford
Engineering Science
Trinity 2019
Fourth Year Project

UNIVERSITY OF OXFORD

Acknowledgements

This project would not have been possible without the supervision of *Professor Justin P. Coon*, and for that I express my sincere gratitude. Professor Coon's communications lectures and advice on schemes with a faster growth rate than what I have been exposed to before has been both extremely useful and fascinating, I am eternally grateful for his time in teaching me these methods.

I would also like to extend my thanks my tutor at University College, *Professor Steve Collins*, who helped me maintain the motivation to keep going when I was struggling, and being an influential part of my education.

I am particularly thankful of the assistance given by *Daniel Mroz*, who helped me develop my mathematical skills and strive to be more mathematically correct in what I do. My gratitude is also extended to *Evan Chen* for his *An Infinitely Large Napkin* project, which has provided vital tools for this research.

Finally, I would like to thank my friends and family for their support and encouragement throughout my study.

For Charles.

Abstract

High-dimensional communications holds the potential to provide a high-throughput scheme which has a rate that scales *super-linearly* (considering bits per symbol). To benefit from the super-linear growth and obtain high data rates, combinatorics and more abstract methods need to be considered to obtain the extra bits of information. The key benefit of a super-linearly scaling scheme is once the scheme is known the solution to gaining a faster rate is merely achieved by increasing the number of channels.

The primary focus of this work is displaying how the rate of a scheme's capacity grows as the number of resources increase. Research showed that schemes exist which offer $\mathcal{O}(n\log(n))$ type growth, but nothing was found to offer $\mathcal{O}(n^{1+\epsilon})$ growth for some positive quantity ϵ.

Here, work is done to explore the super-linear growth rate problem, and a *contender* for *polynomial capacity growth*, $\mathcal{O}(n^{1+\epsilon})$ is found. This attempt at an original pathway to such a scheme is found through The Graph Modulation, by looking at the communications problem by considering the mathematical object of the graph. Various methods are employed to attempt to solve the problem, but *no ideal solution was found.*

Permutation Modulation, the first scheme to achieve super-linear growth (giving $\mathcal{O}(n\log(n))$), is benchmarked against a high lower bound for its space usage. Results showed that despite permutation modulation's non-optimal space usage, the rank decoder offers significant benefits in terms of decoding computation requirements and gives a practical solution to implementing permutation when n is driven to a large value.

Finally, the high lower bound used to analyse permutation modulation was found through an original codebook generator made specifically for this work, to space points evenly on a high-dimensional sphere. A codebook generator is constructed as the Thomson Modulation, which aims to use the available space within the power limits of a transmitter in an optimal manner by aiming for maximum minimum distance properties to be achieved. Results showed that the constructed codebook generator achieved better distance properties than QAM and was validated as a high lower bound for space usage.

FINAL HONOUR SCHOOL OF ENG

UNIVERSITY OF OXFORD

DECLARATION OF AUTHORSHIP

You should complete this certificate. It should be bound into your fourth year project report, immediately after your title page. Three copies of the report should be submitted to the Chairman of examiners for your Honour School, c/o Clerk of the Schools, examination Schools, High Street, Oxford.

Name (in capitals): ……

College (in capitals): …… **Supervisor:** ……

Title of project (in capitals): ……..……

Page count (excluding risk and COSHH assessments): ……

Please tick to confirm the following:

I have read and understood the University's disciplinary regulations concerning conduct in examinations and, in particular, the regulations on plagiarism (*The University Student Handbook. The Proctors' and Assessors' Memorandum, Section 8.8*; available at https://www.ox.ac.uk/students/academic/student-handbook) ☐

I have read and understood the Education Committee's information and guidance on academic good practice and plagiarism at https://www.ox.ac.uk/students/academic/guidance/skills. ☐

The project report I am submitting is entirely my own work except where otherwise indicated. ☐

It has not been submitted, either partially or in full, for another Honour School or qualification of this University (except where the Special Regulations for the subject permit this), or for a qualification at any other institution. ☐

I have clearly indicated the presence of all material I have quoted from other sources, including any diagrams, charts, tables or graphs. ☐

I have clearly indicated the presence of all paraphrased material with appropriate references. ☐

I have acknowledged appropriately any assistance I have received in addition to that provided by my supervisor. ☐

I have not copied from the work of any other candidate. ☐

I have not used the services of any agency providing specimen, model or ghostwritten work in the preparation of this project report. (See also section 2.4 of Statute XI on University Discipline under which members of the University are prohibited from providing material of this nature for candidates in examinations at this University or elsewhere: http://www.admin.ox.ac.uk/statutes/352-051a.shtml.) ☐

The project report does not exceed 50 pages (including all diagrams, photographs, references and appendices). ☐

I agree to retain an electronic copy of this work until the publication of my final examination result, except where submission in hand-written format is permitted. ☐

I agree to make any such electronic copy available to the examiners should it be necessary to confirm my word count or to check for plagiarism. ☐

Candidate's signature: …….. **Date:** ……..

Contents

Nomenclature

Communications Objects

ρ_i	Denotes a power
S	Denotes a *linear* SNR

Mathematical Objects

$\langle \dots \rangle$	Denotes a list
$\mathbb{C}$	The field of complex numbers
$\mathbb{F}_2$	The finite field with two elements
$\mathbb{R}$	The field of real numbers
$\mathcal{X}$	Mathcal denotes a *set* unless used for distributions or Big O notation
$\mathfrak{X}$	Mathfrak denotes a *space*
$\star$	Denotes an *optimal* object
$\{\dots\}$	Denotes a set
$X[i]$	Denotes the i'th element in a list

Operators

$\lvert\cdot\rvert$	Number of items in a set, list or array
$\lvert x\rvert$	Absolute value (for scalars)
$\mathcal{O}(\cdot)$	Description of limiting behaviour of a function

Statistical Objects

$\bar{\rho}_i$	Bars denote a mean, identical to the expectation
$\mathbb{E}(\cdot)$	Expectation
$\Phi(x)$	The CDF of the normalised Gaussian
σ	The standard deviation of a Gaussian
P_i	Denotes a probability

Chapter 1

Introduction

1.1 Communications Today

Before the appearance of the Internet and even the telephone, telecommunications brought about a clear mental image, the telegraph. The emergence of the telegraph in the 1830s disrupted perception of the scale of the world, the Pony Express (the fastest method to get a message across the USA) ceased operating as there was no way to compete with the far superior electrical signal.

Each development since the advent of telegraph (facsimile, voice and then data) has only further shrunk the planet and allowed for greater amounts of faster communication between people and machines.

We currently live in a period where mobile telephones are owned two thirds of humanity, with adoption increasing every year [1]. Data is moving around at an all-time high also due to the uptake of IOT devices, biometric data logging, larger files (higher-quality photos for instance) and more. Despite living in a time of high data transfer there still is a thirst for more, the migration to faster networks (4G in particular) will contribute to the predicted 50 exabytes of video that will be globally streamed in 2022, while under 5 exabytes were streamed throughout 2016 [1].

It is vital to continue innovating on the tools we have at our disposal in communications, as bandwidth over the air is finite and comes at a premium cost (O2 for instance paid £317.72M in April 2018 for 40MHz of the 3.4GHz spectrum) [2]. Fortunately the field of communications is far from young, with plenty of discoveries and optimisations that allow for interesting methods to be applied and explored to yield better, more efficient results for the bandwidth available.

1.2 Capacity Growth Rates

A full system's *bit rate* can be defined as:

$$\text{Bit Rate} = \text{Symbol Rate} \times \text{BPS} \times \text{Framing} \times \text{FEC} \tag{1.1}$$

Where FEC is Forward Error Correction, and Framing is to do with the clustering of bits in discrete packets, both terms have the unit of bits per bit, making them dimensionless. FEC and Framing are not important to this *physical layer* project so they are not considered, therefore the fundamentals that are left to work with are:

$$\text{Bit Rate} \propto \text{Symbol Rate} \times \text{BPS} \tag{1.2}$$

While increasing symbol rate is one potential method of gaining a higher Bit Rate, this project is focused on the BPS (Bits Per Symbol) term. If an increase in this factor can be achieved in a manner which is robust for its noise performance, future demands for high data rate communications can be satisfied while being efficient with bandwidth.

This project in particular explores *capacity growth rate* and how 'conventional' schemes' capacity grows as $\mathcal{O}(n)$ (n being the number of resources available, for this project n defines the number of *dimensions*, or 'half RF channels'), and then discusses other schemes offering capacities that grow faster than this.

This project explores and simulates some more advanced modulation schemes (Permutation, Rank and eluding to Graph) to see if the schemes can be applied in reality to attain higher data transfer rates. These newer, more advanced schemes use mathematical methods to beat the linear growth rate set out by existing and commonplace conventional schemes.

1.2.1 Shannon-Hartley theorem

The Shannon-Hartley theorem states the theoretical tightest upper bound on the capacity C in an Additive White Gaussian Noise (AWGN) channel with bandwidth B and linear SNR S can be described as the following:

$$C = B \log_2(1 + S) \tag{1.3}$$

Where the linear SNR S is defined as the mean signal power divided by the power of the noise

$$S = \frac{\bar{\rho}_{\text{signal}}}{\rho_{\text{noise}}} \tag{1.4}$$

This theorem sets a limit to the highest possible rate that can be achieved for a given scheme. An effort is made in this project to optimise schemes to reach their capacity limit with as low an SNR as possible. This is achieved by making the 'distance properties' of a scheme as 'good' as possible, how this is defined and achieved is discussed in this project.

1.3 Literature Review

Interest in capacity growth rates is an interesting step to take towards developing extremely high-capacity schemes. Slepian, in 1965 developed a scheme [3] named *Permutation Modulation* which has a capacity that grows faster than any 'conventional' scheme like QAM (one whose codebook is exhaustively defined and is only copied when adding another RF channel), and has a capacity that grows $\mathcal{O}(n\log(n))$.

A paper exists on the application and an overview of permutation modulation [4], which outlines the significant impact it has had on the progress in new communication schemes. One of the schemes highlighted in particular (Rank Modulation) is of interest as versions of it are beginning its use in computer memory, with practical examples given by A. Jiang in 2009 [5]. Using combinatorics allowed Slepian's scheme to develop and branch into multiple other schemes, such as Index Modulation [6], Parallel Combinatory Modulation [7] and more. The amount of recent effort on these schemes displays the interest schemes which offer super-linear capacity growth has, and their applications demonstrate that they have a place in the modern communications world.

Index modulation is currently not used in practise but offers the potential to 'do more with less' by using nulled carriers activated by a second signal constellation to gain more capacity [6]. Index modulation is in the permutation family, and is presently expected to become a part of 5G or 6G systems [6].

All the reviewed literature suggested that permutation-based schemes are going to become a mainstream part of communications in the near future, and the interest in super-linear capacity growth is something to focus effort on.

1.3.1 Gap Analysis and Aims

During the research performed *no analysis was found* on the effectiveness of space usage of Slepian's scheme. Slepian in his 1965 paper also offered ideas on how to generate optimal 'levels' for his permutation scheme, yet suggested there was room to improve the scheme for minimum error probability. No research was found on setting optimal 'levels' for permutation, so original research is offered on setting the 'levels' for his scheme, which has the potential to offer a higher power-limited capacity with lower signal to noise ratios than what is currently achieved.

Interest in a scheme that offers capacity to grow polynomially $\mathcal{O}(n^{1+\epsilon})$ (for some positive quantity

ϵ) is strongly of interest. A scheme growing polynomially would scale significantly faster than Permutation Modulation and satisfy the need for increased data rates for years to come. Despite extensive research *no scheme has been found* which scales as $\mathcal{O}(n^{1+\epsilon})$, and *no documents were found* which offer or suggest capacity scaling greater than Slepian's outlined scheme.

The information gathered from this literature review highlighted major areas that original research can be performed in at an accessible level.

Another area for research shown was the need for research on a scheme with polynomial scaling capacity. Although potentially a bit beyond the scope of this project, the literature exposed the lack of and need for a scheme that satisfies the polynomial growth requirement.

This project aims to analyse the fundamentals of super-linear capacity growth through the permutation modulation, as well as set optimal 'levels' for the permutation scheme.

The project then aims to analyse the effectiveness of space usage of the permutation scheme, when compared to a *high lower bound* of space usage when distance properties are used.

No literature was found on point spacing in volumes, so an aim was set to create an original codebook generator to optimally place points in a space, which gives a comparable high lower bound on space usage.

1.3.2 Report Structure

This report is structured to initially discuss the discrete communications model, and how this framework was validated and made for the high-dimensional communications problem.

The report then discusses distance-based schemes, in terms of commonly known schemes such as QAM and PSK, before showing efforts to develop an optimised scheme for the available space a transmitter has at its disposal.

A chapter on ordering based and heuristic schemes comes after which shows how super-linear capacity growth is achieved, and how the schemes used can be optimised to provide the lowest bit error rate for a given SNR. A practical implementation is given to show how the super-linear growth can be realistically exploited. In the same chapter some experiments are performed to try and find a scheme which grows more quickly by using the structure of the graph.

Chapter 2

Model and Methodology

2.1 Discrete Communications Model

2.1.1 Requirements

To test and validate various modulation schemes the mathematics describing performance can become extremely dense. To aid the speed of testing and development, a discrete model of the communications problem was developed in MATLAB®[1] to test ideas before a more in-depth mathematical formalism was applied.

The model (whose process flow is shown in figure 2.1) (and is identical to the Shannon-Weaver model of communication [8]) takes randomly generated binary data and modulates it. This signal is then sent to a channel where complex AWGN is added. This noisy signal is then fed to a demodulator and the received binary data are compared with the original data to obtain a Bit Error Rate (BER).

Figure 2.1: Process Flow of the discrete model

Type of System modelled

Communication can exist in many different systems, which operate in different spaces. Radio communication can use the complex space as both *in-phase* and *quadrature* components exist on a single channel.

Storage media (for example a DRAM cell) can exist in the positive real space, or even in the case where the number of electrons contained in a single DRAM capacitor can be counted, just the positive integers.

For the purposes of this project, the focus is on complex spaces, but the results can be generalised

[1] MATLAB® is a registered trademark of The MathWorks, Inc.

to work in real spaces. If a system has two independant real spaces, they can be considered as *pseudo-in-phase* and *pseudo-quadrature* components, making the results applicable both ways.

Justification of Discrete model

A discrete model was used in this project as when simulating a *system similar to real life* through modulation and demodulation with a high-frequency carrier, filter effects are experienced. These filter effects would mean the test is not performed in a true *non-bandlimited AWGN channel*. The discrete model was preferred as the filtered system would beat 'theoretical' performance.

In the interests of validating a model with theoretical results and allowing its extension onto modulation schemes such as Rank, Graph and more, the discrete model was preferred, as it allows for direct comaprison of any developed systems.

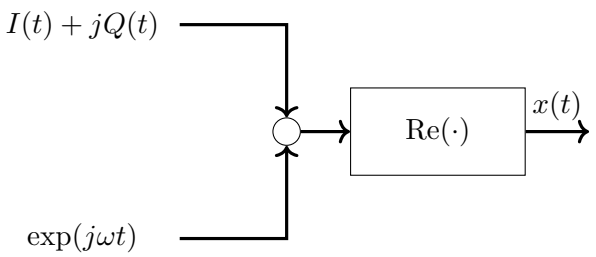


Figure 2.2: How In-phase and Quadrature components are transmitted in reality. Filters would be used to extract these components, so discrete complex values are transmitted instead (to preserve true AWGN for accurate results).

Data

The data in d and data out d' that this project considered is in binary. Binary exists in the Galois Field with two elements GF(2) which is written henceforth as the finite field with two elements $\mathbb{F}_2$. For a binary data array of length m, this is noted as the vector space $d \in \mathbb{F}_2^m$. For all experiments a *maximum entropy source* was considered, which implies each element of d having the statistics of being assigned a one as $d_i \sim \mathrm{Bern}\left(\frac{1}{2}\right)$.

Codebook

Work began by considering a (shared) codebook $\mathcal{C}$ which contains M symbols. A 'symbol' is a one-hot vector in n dimensional space which corresponds to a single value in binary. To restrict where elements of $\mathcal{C}$ can exist, a Euclidean space[2] was defined for which every element in $\mathcal{C}$ can exist in, $\mathcal{C} \subset \mathbb{R}^n$. The power limit for a symbol is an important feature in communications, so the *maximum power per n-dimensional symbol* was enforced for this project to be unity. This restriction shrinks the space of possible locations for symbols to be a n-ball (a hypervolume bounded by an $(n-1)$-sphere) with unit radius:

$$\mathfrak{B}^n = \{x \in \mathbb{R}^n : \|x\|_2 \leq 1\} \tag{2.1}$$

[2]Euclidean space is used as noise adds in a Euclidean manner across n channels

and the $(n-1)$-sphere having the definition:

$$\mathfrak{S}^{n-1} = \{x \in \mathbb{R}^n : \|x\|_2 = 1\} \tag{2.2}$$

where the L_2 norm is defined as

$$\|X_0\|_2 = \sqrt{\sum_{i=0}^{n-1} |X_0[i]|^2} \tag{2.3}$$

Each possible symbol is defined as $X_i \in \mathcal{C}$, such that $\mathcal{C} = \{X_0, X_1, \ldots, X_{M-1}\}$.

For a general APSK (Amplitude Phase Shift Keying) scheme, a codebook generator was constructed to generate M-QAM, M-PSK, M-ASK, M-PAM and OOK constellations. *Typically*, the symbols used by a APSK modulator are represented efficiently as *complex numbers*, $X_i \in \mathbb{C}$, or $X_i = a_i + jb_i$, where $j = \sqrt{-1}$, which gives a two-dimensional symbol. The complex number interpretation of a discrete signal extends to what is achievable in the practical case, *quadrature* and *in-phase* components of a sinusoid can be modulated to transmit and extract a complex number *on a single channel* as in figure 2.2 on the previous page.

In this report modulation schemes are explored which use symbols with a number of dimensions greater than two, $X_i \in \mathbb{R}^n$, so the decision was made to represent *all* symbols as lists[3], $X_i = \langle x_i^0, x_i^1, \ldots, x_i^{n-1} \rangle$ (for some integer n resembling the number of dimensions in the symbol). For the case of the *typically complex* symbols above they are packaged into a two-element list $X_0 = \langle a_i, b_i \rangle$. While this is just a notation detail, each of these n dimensions are sent in $\lceil \frac{n}{2} \rceil$ pairs as each channel has two orthogonal dimensions, which helps to increase rate. For example, the symbol in $\mathbb{R}^4$, $X_i = \langle 1, 2, 3, 4 \rangle$ would be sent over two channels as $X_{i,0} = 1 + j2$, and $X_{i,1} = 3 + j4$.

Under the alternative power constraint that each *channel* has an independant maximum power, the space a symbol can exist in is defined differently. A single complex channel (with quadrature and in-phase components as x_1, x_2) has a usable symbol region of the 2-ball, $\mathfrak{B}^2$. If two of these complex channels (the second having components x_3, x_4) are used in parallel and each has a two-ball of usable symbol region (power limits are independent to each channel), then the available space for a symbol to exist in is the *Cartesian product* of two 2-balls.

$$\mathfrak{D}^4 = \mathfrak{B}^2 \times \mathfrak{B}^2 = \left\{(x_1, x_2, x_3, x_4) \mid x_1^2 + x_2^2 \leq 1, x_3^2 + x_4^2 \leq 1\right\} \tag{2.4}$$

This object is known as the *duocylinder* and is analogous to the cylinder in three dimensions. Numerical integration[4] found $\mathfrak{D}^4$ to have 2.0 (to 5 s.f) times the hypervolume of $\mathfrak{B}^4$, which implies double

[3]Which are denoted by *angled brackets* ($\langle \cdot \rangle$), with the i'th element in a list X being $X[i]$

[4]Numerical integration was performed by inscribing a duocylinder and a 4-ball inside a 4-cube, placing uniformly dis-

the number of symbols can be fitted inside gaining an extra bit of information. $\mathfrak{D}^{2k}$ space is only mentioned in the *Graph Modulation* section of this report.

Modulator

The modulator for this model takes the binary data d and for a given number of bits (defined by the scheme's capacity) assigns a symbol, which is then passed to the channel.

Channel

The simulations used only consider an AWGN channel $y^t = x^t + z^t$, where y^t is the received signal on a single channel, x^t is the transmitted signal, and z^t is a stationary random variable, all at time t.

In the case of an RF channel, the complex plane $\mathbb{C}$ is used for transmission, meaning that $z^t \sim \mathcal{CN}\left(\mathbf{0}_{2,1}, \frac{\rho_{\text{noise}}}{\sqrt{2}}\mathbf{I}_2\right)$ where ρ_{noise} is the noise power in the AWGN channel, which can be found from $S = \frac{\bar{\rho}_{\text{signal}}}{\rho_{\text{noise}}}$, which is the SNR (signal to noise ratio) of the channel, and $\bar{\rho}_{\text{signal}}$ is the mean power of the modulated signal. $\bar{\rho}_{\text{signal}}$ for a specific scheme can be calculated by

$$\bar{\rho}_{\text{signal}} = \frac{1}{|\mathcal{C}|}\sum_{X_i \in \mathcal{C}} \|X_i\|_2^2 = \mathbb{E}(\|X_i\|_2^2) \tag{2.5}$$

In the case of a DRAM cell as mentioned earlier, only positive real values can be used, such that $z^t \sim \mathcal{N}(0, \rho_{\text{noise}})$. For this report the decision was made to explore the RF channel case, such that z^t is distributed as the complex normal distribution.

A received signal is represented by a list of elements for each dimension $Y^t = \langle y_1^t, y_2^t, \ldots, y_n^t \rangle$. In this discrete model it was assumed that it doesn't matter how Y^t is received, whether the transmitted elements were sequential or parallel, as ρ_{noise} for every channel is identical.

Demodulator

Since symbols are equally likely, *maximum likelihood regions* were defined which are regions $\mathfrak{R}_i \subset \mathbb{R}^n$ for all $i = 1, \ldots, M$, which do not intersect $\mathfrak{R}_i \cap \mathfrak{R}_k = \emptyset$ for all $i \neq k$. Under the case that a recieved symbol Y^t is assigned to the nearest neighbour $X_\star$,

$$X_\star^t = \arg\min_{X_i} \left\| X_i - Y^t \right\|_2 \tag{2.6}$$

Which implies the existence of distinct regions $\mathfrak{R}_k \subset \mathbb{R}^n$, where a detected Y_i may land in, mapping to a single symbol X_k. Each region $\mathfrak{R}_i$ is defined as all points that map to X_i. The analytic detail on the regions $\mathfrak{R}_i$ can be found by constructing a Voronoi Diagram (a tangible example in $\mathbb{R}^2$ can be found in figure 2.3 on the following page). Unfortunately Voronoi Diagrams for r points in d dimensions

tributed random points (such that all points lie in the 4-cube), and counting the proportion of random points inside each of the two spaces

require $\mathcal{O}\left(r^{\lceil \frac{d}{2} \rceil}\right)$ storage space to define, so it is not often feasible for high dimensional spaces. Maximum likelihood detection is the preferred method of detection, but some schemes (such as Rank Modulation) use scheme-specific methods of decoding in the interests of computation speed. These methods are explored and compared where applicable against the maximum likelihood benchmark later in the report.

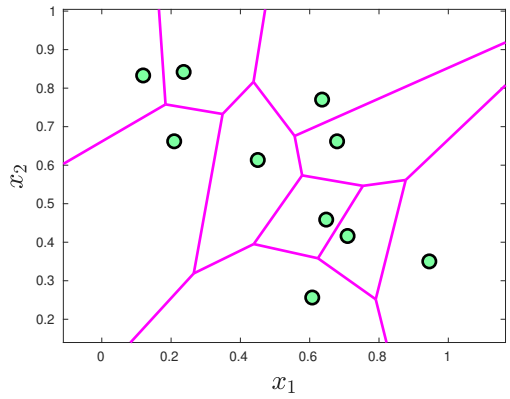


Figure 2.3: An example of a Voronoi Diagram with 10 randomly distributed points. Note the magenta boundaries of equidistance between green points, which define each individual region $\mathfrak{R}_i$.

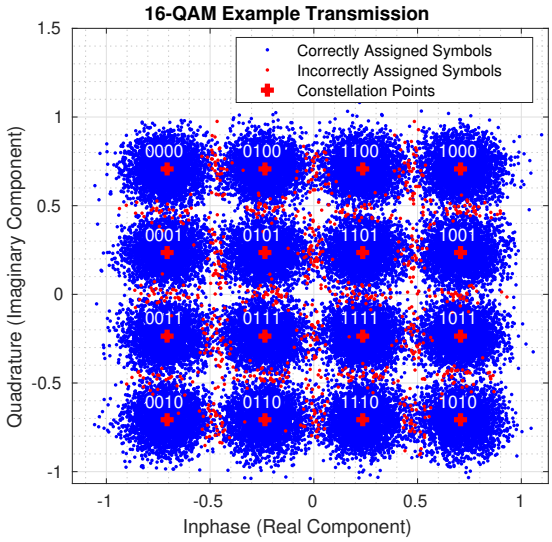


Figure 2.4: Example of a test run by the simulator, 16-QAM at 15dB SNR. Each point is a transmitted symbol, corrupted by the AWGN channel.

2.1.2 Governing Test Algorithm

Each 'test' of a modulation scheme has to be divided into separate parts:

1. A list of SNRs to test at is generated (linearly spaced in dB between appropriate ranges).
2. An experiment is run at each SNR in the list which lasts for 5×10^4 symbols, an experiment runs as follows:
   (a) Generate random data.
   (b) Encode the binary sequence as a list of symbols.
   (c) Corrupt symbols with an AWGN channel.
   (d) Decode the resulting list and convert to binary.
3. A comparison is made between expected and actual results and the outcome logged to a file.
4. Steps 2-3 are repeated 100 times for each SNR, giving a total length of 5×10^6 symbols per SNR value. An example of steps 2-3 is visible in figure 2.4, which shows how each transmission is observed by the demodulator and is detected, with errors highlighted and tracked.

5. Statistics are then performed on the results file, giving the scheme's measured performance.

The decision to repeat the test many times with random data is a Monte-Carlo method, in which many repeated random samples are taken, and the results analysed for accurate scheme performance.

2.1.3 Validation

To validate the discrete model computer program that was created, the constructed simulator was initially tested on BPSK. The encoder used the codebook $\mathcal{C} = \{-1, 1\}$, and the decoder used was maximum likelihood (nearest neighbor) as in equation (2.6) on page 9. BPSK was initially used as the bit error probability has a simple closed form [9, pg. 271]

$$P_{\text{bit error}} = Q\left(\sqrt{2S}\right) \tag{2.7}$$

Where $Q(x)$ is the Q-Function (the tail of the normalised Gaussian Distribution):

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty \exp\left(-\frac{u^2}{2}\right) \mathrm{d}u = 1 - \Phi(x) \tag{2.8}$$

This result can be derived by considering the problem graphically. BPSK was considered with the same $\mathcal{C}$ as noted above, which means that any error in the imaginary dimension does not change the assigned symbol. This problem therfore reduces to a 1D Gaussian in the real dimension (marginalising over the imaginary dimension), which can be drawn as following, looking at the probability of being correctly assigned to the transmitted symbol, -1:

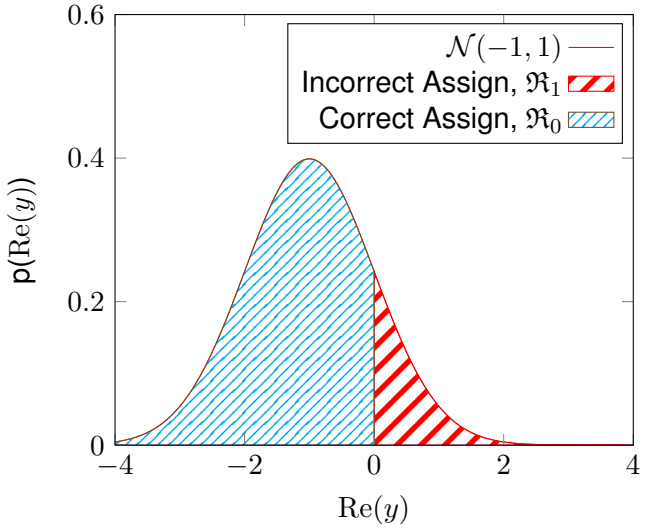


Figure 2.5: BPSK demodulator probability of correct assign sketch.

By drawing figure 2.5 the *tail of the Gaussian* is what is incorrectly assigned. To convert from a Gaussian with a generic variance to a normalised Gaussian the *z-score* is used, which is $z = \frac{y-\mu}{\sigma'}$.

The z-score of interest occurs is when $y - \mu = 1$, as this defines the boundary at which the two decision regions $\Re_0$ and $\Re_1$ meet when analysing the -1 symbol. It is worth noting that the power for the imaginary and real components is split in such a way that $\sigma^2_{\text{real}} + \sigma^2_{\text{imag}} = \sigma^2$, and since these components are *equal and BPSK marginalises over the imaginary dimension*, $\sigma^2_{\text{real}} = \frac{\sigma^2}{2}$.

Since σ^2 is the noise power, and the signal power *for this case* is unity, the SNR $S = \frac{1}{2\sigma^2_{\text{real}}}$. The result $\sigma_{\text{real}} = \sqrt{2S}^{-1}$ is then plugged into the z-score for $z\,(y - \mu = 1) = \sqrt{2S}$, which when used to calculate the probability of a bit error is $Q\left(\sqrt{2S}\right)$, which confirms the result claimed in equation (2.7) on the preceding page.

The BPSK simulation results in figure 2.6a on the next page closely matched the theoretical performance of the scheme, with the Q-function's prediction being within the 1σ bars set out. This result proved that the simulator produced correct results for a codebook with elements on the real line.

The next stage was to test with QPSK, which has a simple closed form for its SER (the analogy being that QPSK is a combination of two BPSK modulators) of [9, pg. 272]

$$P_{\text{symbol error}} = 1 - \left(1 - Q\left(\sqrt{2S}\right)\right)^2 \tag{2.9}$$

and has a BER of [10, pg. 172]

$$P_{\text{bit error}} \approx Q\left(\sqrt{S}\right) \tag{2.10}$$

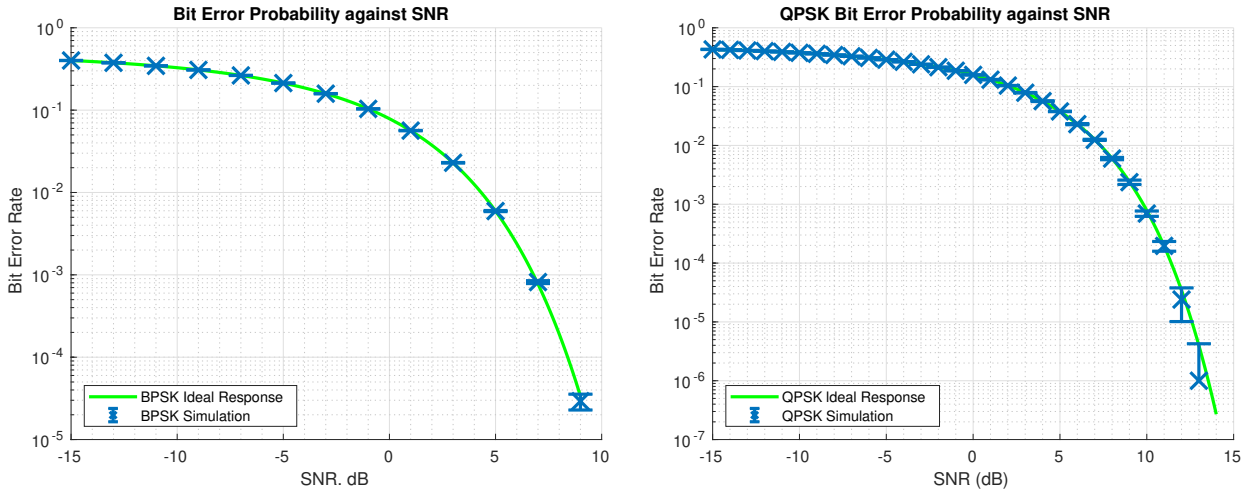
The simulation was run again for the QPSK case and the results again closely matched theory as can be seen on figure 2.6b on the following page. 27 out of 29 results were in a 1σ range, all were in a 2σ range. This result was taken to qualify the simulator for quadrature signals, and demonstrates that the noise is correctly calculated.

2.2 Limits of communication in an AWGN Channel

As mentioned earlier in equation (1.3) on page 3, there is a fundamental limit as to what the RF channel can provide in terms of capacity for a given SNR. Since the discrete model is used, the bandwidth B is set to unity as the 'sampling bandwidth'.

An interesting consequence of the high-dimensional nature of this project is that a number of independant dimensions greater than two is used in some of the schemes explored. Channel capacity is well known to be additive over independant channels [11], meaning that using $\frac{n}{2}$ independant channels used in an independant manner gives the same upper bound on performance as the channels used in a combined fashion.

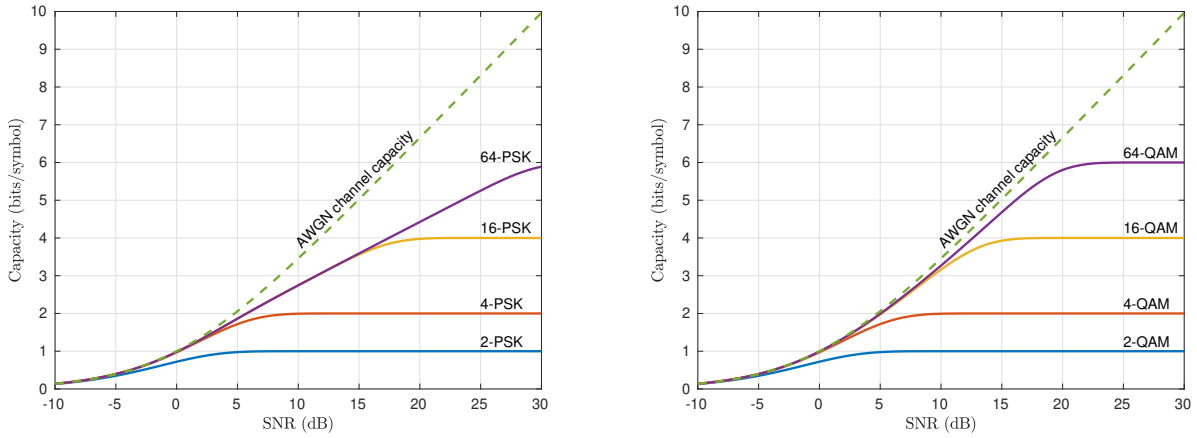
Nonetheless, reaching the capacity limit of a scheme with as low an SNR as possible is important



(a) BPSK results

(b) QPSK results

Figure 2.6: Validation simulation results (100 trials per SNR), error bars show the standard deviation for each test



(a) Mutual Information for PSK

(b) Mutual Information for QAM

Figure 2.7: Mutual information for 2-D modulation in an AWGN channel against SNR for QAM and PSK schemes

to have in communications, so effort was expended on optimising schemes such that their BER decreases to a low (BER $\leq 10^{-3}$) point with as low an SNR as possible. One way of visualising this is with a *mutual information plot* as shown in figure 2.7, where it is clear that QAM outperforms PSK in terms of reaching its maximum theoretical capacity sooner. Methods exist for calculating mutual information [12] (which were used to calculate the noted plots) but extending this method was outside the scope of this project, so BER curves (as in figure 2.6) were compared for a given capacity per channel.

Chapter 3

Distance-Based Schemes and Optimisation

This chapter covers schemes which have a capacity that grows $\mathcal{O}(n)$, with respect to number of resources. Conventional schemes (like QAM, PSK) are discussed to show that adding an extra channel only linearly increases capacity, before discussing other methods of generating a codebook to optimally fill the space available for transmission of a symbol. This chapter also acts as a primer before discussing schemes with a faster growth rate in the next chapter.

3.1 Conventional Schemes

3.1.1 Conventional Scheme Optimisation

For a given scheme in any APSK sense on a single complex channel, the problem of creating an *optimal codebook* can be defined as maximising the minimum distance between symbols.

$$\mathcal{C}^\star = \arg\left(\max_{\mathcal{C}\subset\mathfrak{B}^2}\left(\min_{i\neq j}\|X_i - X_j\|_2\right)\right) \tag{3.1}$$

Conventional schemes operate in $\mathbb{C}$ space, so by considering the number of dimensions n, one channel will have two dimensions.

Distance properties are very useful, as the point in space that a Y_i is measured in defines which region $\mathfrak{R}_i$ and therefore which X_i the symbol is assigned to. The point's location is perturbed by the noise on the Gaussian channel in a Euclidean sense (each orthogonal direction is added linearly), so it makes sense to maximise L_2 distance between symbols.

To demonstrate the importance of distance properties, QAM, PSK and PAM can be compared to show that the method of placing the one-hot code vectors can change the SNR at which the BER becomes sufficiently low.

3.1.2 Capacity Growth Rate

For a 'conventional' scheme, the growth rate in terms of bits per symbol grows linearly with the number of channels (where each channel is complex) provided the SNR is sufficiently high.

Each channel has M different messages which can be sent, therefore $\log_2(M)$ bits are sent *per channel*, so overall $\frac{n}{2}\log_2(M)$ bits are sent over $\frac{n}{2}$ channels in the ideal case.

3.1.3 Results

Results showed that despite having the same capacity schemes differed in their performance. An experiment was run for the case of $M = 16$ (shown in figure 3.1), the best performing scheme was QAM, and reached a BER 10^{-3} at 3.9 dB before PSK, and 17.9 dB before PAM.

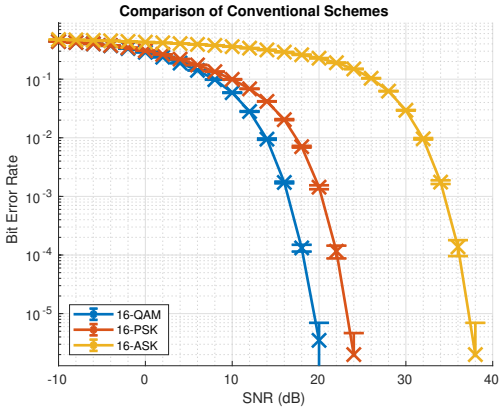


Figure 3.1: A comparison of the BER performance of various conventional schemes (QAM, PSK and PAM) with identical capacity.

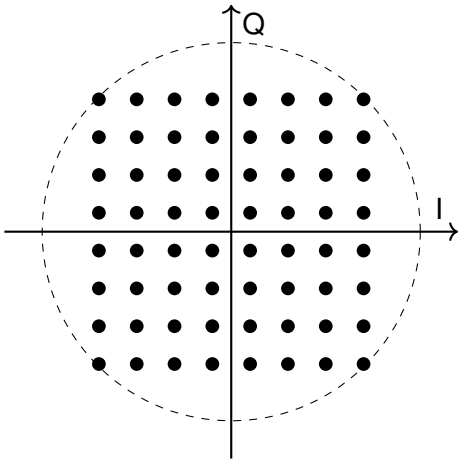


Figure 3.2: Square 64-QAM lattice. The dashed line resembles the maximum symbol energy for the scheme, which can be interpreted as the outer boundary on the 2-Ball $\mathfrak{B}^2$.

3.2 The Thomson Modulation

The Thomson modulation was originally created by the author of this report to try and find a method of realistically achieving maximum minimum distance properties given a space constraint.

The scheme relies primarily on a codebook generator that was created for the purpose of this work, with the focus to compare directly to permutation modulation later in this report in section 4.1 on page 26. The purpose of its construction was to analyse how well permutation codebook generation uses the available space $\mathfrak{S}^{n-1}$ (or even when considering the maximum symbol energy constraint for all possible space available in $\mathfrak{B}^n$).

A 'high lower bound' codebook generator was required which attempts to place equispaced points either on or inside the unit sphere, but this generalises to comparing to all possible schemes.

3.2.1 A High Lower Bound on Spacing of Elements of $\mathcal{C}$ in $\mathfrak{B}^n$ and $\mathfrak{S}^{n-1}$

If the QAM lattice in figure 3.2 on the preceding page is inspected, unused volume in the power limit can be seen which is available and *could be utilised to obtain better distance properties*.

The Thomson modulation has two variants in how it generates its codebooks:

1. The 'on the unit sphere' ($\mathcal{C} \subset \mathfrak{S}^{n-1}$) constraint is considered as it is used as a direct comparison to permutation modulation's 'equal symbol energy' constraint, which has symbols only on the surface of the unit n-sphere. This constraint is considered for the best possible usage of space on the $(n-1)$-sphere for maximum minimum distance, as specified in equation (3.2)

2. The 'inside the unit sphere' ($\mathcal{C} \subset \mathfrak{B}^n$) constraint is considered for the best possible usage of space in the n-ball for maximum minimum distance, as specified in equation (3.3)

$$\mathcal{C}_1^\star = \arg\left(\max_{\mathcal{C}\subset\mathfrak{S}^{n-1}}\left(\min_{i\neq j}\|X_i - X_j\|_2\right)\right) \tag{3.2}$$

$$\mathcal{C}_2^\star = \arg\left(\max_{\mathcal{C}\subset\mathfrak{B}^{n}}\left(\min_{i\neq j}\|X_i - X_j\|_2\right)\right) \tag{3.3}$$

Only a small modification to the algorithm was required to switch between generation of either variant, which is discussed in the next section.

High Lower Bound Generating Algorithm

The work by J.J. Thomson [13] and specifically his work on considering how electrons arrange themselves on a 2-sphere was initially used as a guide to develop an iterative method to spacing points. The objective of Thomson's problem is to determine the minimum electrostatic potential energy for k electrons contained to the surface of a unit sphere with a force given by coulomb's law [13]. This problem led to and inspired the development of an algorithm which works on any n-dimensional space where points are embedded in either $\mathfrak{B}^n$ or $\mathfrak{S}^{n-1}$.

The problems posed are similar, one is how to place k points optimally on $\mathfrak{S}^{n-1}$, and the other being placing k points optimally inside $\mathfrak{B}^n$.

To find a solution to this problem iteratively, each symbol was defined as a point $p_i \in \mathbb{R}^n$ with a vector in Euclidean space $p_i = \langle p_i^0, \ldots, p_i^{n-1}\rangle$, which has a velocity $v_i \in \mathbb{R}^n$ and acceleration $a_i \in \mathbb{R}^n$. Each point is assigned an identical mass m and a randomised starting position *in the n-ball or on the $(n-1)$-sphere depending on the variant*. From here a 'force' can be considered acting on each point, which is calculated by assuming each point behaves like an electron, and calculating the electrostatic repulsion force vector as

$$F_i = \sum_{r=1, r\neq i}^{k} \frac{f(p_r - p_i)}{\|p_i - p_r\|_2^3} \tag{3.4}$$

where f is a force multiplying constant. The 'force' on point i from a single other point acts in the direction of point r to i (normalised) and scales as the inverse of the Euclidean distance squared.

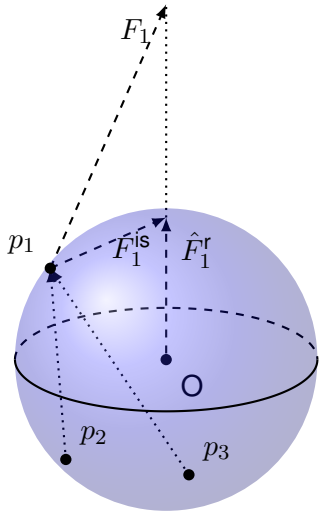


Figure 3.3: Diagram to aid in demonstrating how instantaneous forces on a single node p_1 are calculated from three points on a sphere, and then how the force's radial component is removed

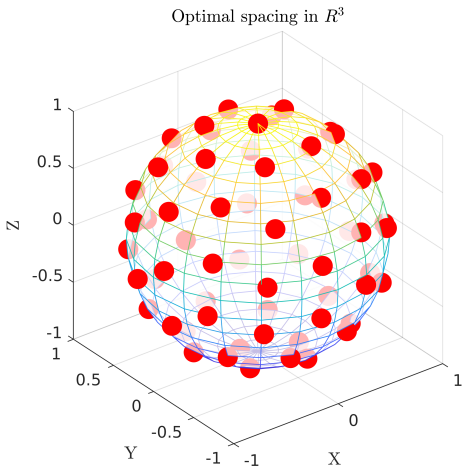


Figure 3.4: A Thomson Codebook with $M = 64$ in $\mathbb{R}^3$, using the '$\mathcal{C} \subset \mathfrak{S}^2$' constraint

After force calculation Euler's method is applied to update positions (given a simulation time increment Δt), velocities and accelerations *for all points* before the next round of force calculations.

$$a_i(t+1) = a_i(t) + \frac{F_i - Dv_i\|v_i(t)\|_2}{m} \tag{3.5a}$$

$$v_i(t+1) = v_i(t) + a_i(t) \cdot \Delta t \tag{3.5b}$$

$$p_i(t+1) = p_i(t) + v_i(t) \cdot \Delta t \tag{3.5c}$$

To force the symbol to be embedded inside the n-ball, $\mathfrak{B}^n$, the position vector p_i is normalised after calculation updating its position if its L_2 norm is greater than unity. If there is a need to force the symbol to be on the $(n-1)$-sphere, p_i is normalised after its position update no matter the magnitude, $\hat{p_i} = \frac{p_i}{\|p_i\|_2}$.

To prevent a velocity from winding up a small positive 'drag' term D was introduced to remove 'energy' from the system. Another method to prevent velocity windup was importantly *removing radial components of force when a point is on the boundary*. Removing the radial component is *critical* as otherwise the velocity will continue to wind up on account of the position being normalised continuously. This *could* be achieved by *taking the dot product of F_i with a tangent-space to the $(n-1)$-sphere at point p_i*. In practise this projection method is a difficult problem when considering high dimensions, so an approximation was created which considers the vectors as dimensionless

quantities (in terms of units). By considering F_i existing with its root at p_i as in figure 3.3 on the previous page, a *radial* unit vector can be constructed going towards $(p_i + F_i)$.

$$\hat{F}_i^{\mathsf{r}} = \frac{p_i + F_i}{\|p_i + F_i\|_2} \tag{3.6}$$

If p_i is subtracted from $\hat{F}_i^{\mathsf{r}}$, the *tangent-space projection approximation* is obtained as the *in-sphere force*, F_i^{is}.

$$F_i^{\mathsf{is}} = \hat{F}_i^{\mathsf{r}} - p_i \tag{3.7}$$

It is important to stress this method is only used for points on the boundary ($\|p_i\|_2 = 1$), which for the case of '$\mathcal{C} \subset \mathfrak{S}^{n-1}$' is every point. While the method's effects were not explored in significant depth, far fewer iterations were required until a 'solution' was found, but it was found to cause instability with too high a Δt or high a force multiplying constant f. Reducing the number of required iterations until convergence was extremely important for this algorithm as it has $\mathcal{O}(k^2 n)$ complexity.

An *Energy Function* was also calculated for convenience which sums the total force on all points, with lower 'energies' being more well spaced and stable solutions.

By running the iterator, 'ideal' codebooks can be generated for n dimensional spaces, by testing for a steady state at each iteration (t), the test used is

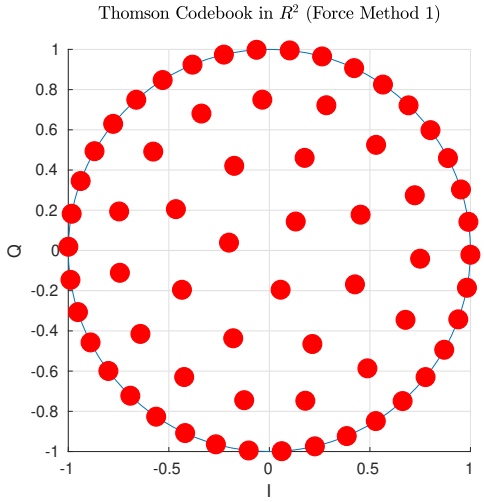
$$\sum_{r=1}^{k} \left\| x_i^t - x_i^{t-1} \right\|_2 < E_{\mathsf{max}} \tag{3.8}$$

When this condition is satisfied the simulation is considered to be solved, as the update is sufficiently small. The quantity E_{max} is defined as a small quantity, which was a multiple of the machine error (given by the `eps` function in MATLAB) in the simulations used.

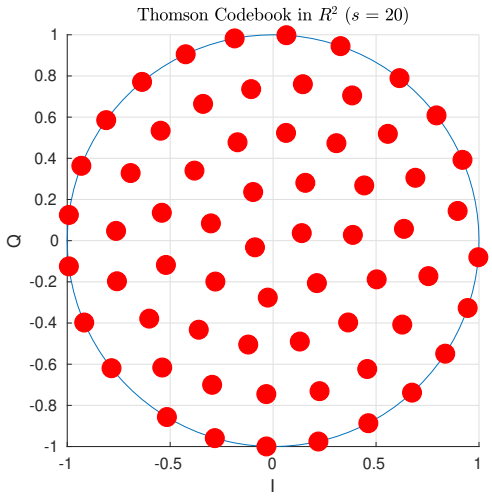
This codebook generating method proved successful for the '$\mathcal{C} \subset \mathfrak{S}^{n-1}$' constraint with an example codebook being visible in figure 3.4 on the preceding page. After some experimentation for the '$\mathcal{C} \subset \mathfrak{B}^n$' constraint, it was found the 'force function' in equation (3.4) on page 16 was inappropriately spacing points, and driving *all* the points to the outer surface of the ball. Under Thomson optimisation the minimum distances between points *should* be identical, yet this result did not follow when codebooks were initially generated. A new 'force function' was made for this scenario which was

$$F_i = \sum_{r=1, r\neq i}^{k} \frac{f(p_r - p_i)}{\|p_i - p_r\|_2^{s+1}} \tag{3.9}$$

The initiative behind this new force function was to increase the influence of the closest point to any given point, with the theory that driving s in the limit to infinity would give maximum minimum distance.



(a) A Thomson Codebook generated using Force equation (3.4) on page 16, giving uneven distances between points



(b) A Thomson Codebook generated using Force equation (3.9) on the previous page, with $s = 20$ and providing equal minimum distances

Figure 3.5: Example Thomson codebooks in $\mathbb{R}^2$, with $M = 64$ and using the '$\mathcal{C} \subset \mathfrak{B}^2$' constraint, showing how different Force Methods change the performance of the spacing. Minimum L_2 distances between points *should* be identical when optimising the maximum minimum distance constraint

Literature suggests that this is commonly known as Riesz s-Energy [14], and minimising it for $s \to \infty$ gives maximum minimum distance spacings for points in the ball. There was success in this endeavour, with figure 3.5b (which uses the said Riesz s-energy method) showing a large improvement of spacing over figure 3.5a (which uses the method outlined in equation (3.4) on page 16).

3.2.2 Validation

The Thomson codebook generator was validated for the case $\mathcal{C} \subset \mathfrak{S}^{n-1}$ using the Riesz s-energy method for $s = 20$, as when s was driven too high numerical instability occurred. The validation that was used was comparing Thomson codebooks in $\mathbb{R}^3$ to the vertices of Platonic solids (regular, convex polyhedra). Platonic solids are the *only* case in $\mathbb{R}^3$ where perfect spacing can be achieved, as each edge has the same length and the vertices lie on $\mathfrak{S}^2$. The Thomson codebooks for $k = 6, 12, 30$ were tested and found to adequately converged to the inscribed platonic solids, so the test was deemed a success. Higher dimensional polytopes exist which are 'Platonic', and when a selection were tested for $n = 4$ the results also proved to be successful. The Thomson codebook generator could not be validated for the $\mathcal{C} \subset \mathfrak{B}^n$ case as there is no known general solution, but minimum distances between points were measured, *and were all equal* for $k = 32, 64, 128$.

3.2.3 Scheme Performance

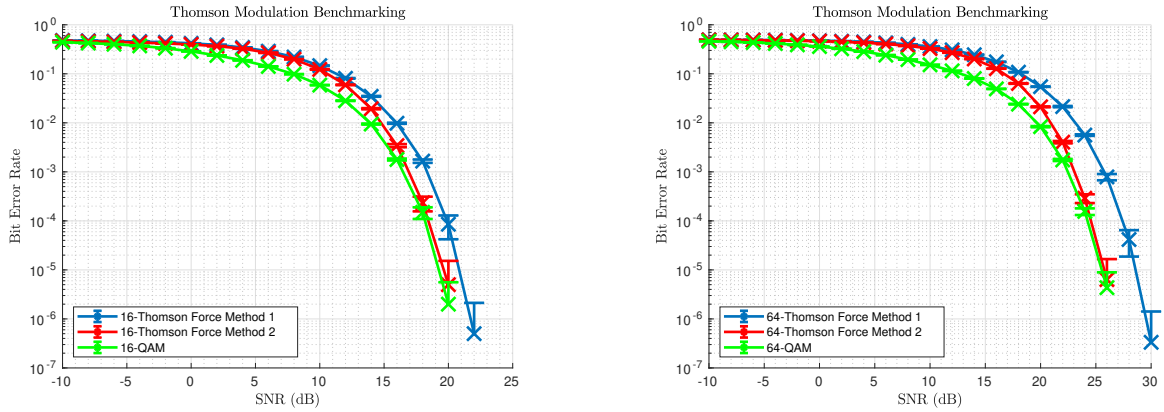
An immediate and direct comparison between $n = 2$, k-Thomson and M-QAM can be drawn, using the codebooks shown in figure 3.5 (with the $\mathcal{C} \subset \mathfrak{B}^2$ constraint). Experiments were run with $M = k = 64$ and $M = k = 16$, with results visible in figure 3.6 on the following page for both

	Min L_2 Distance	$\max\|X_i\|$	**Expected Power**	**PAPR**
16-QAM	0.4741	1	0.5556	1.80
16-Thomson	0.5230	1	0.7159	1.40
64-QAM	0.2020	1	0.4282	2.33
64-Thomson	0.2259	1	0.6155	1.62

Table 3.1: Comparing various features between Thomson and QAM codebooks, showing the Thomson scheme having a higher expected power, and also a lower PAPR when compared to QAM

tests showing the Thomson Modulation did not perform as well as QAM. The poorer performance of the $\mathcal{C} \subset \mathfrak{B}^2$ case when compared to QAM highlights that despite equal spacing being achieved (when using the second force method), the codebook's mean power is higher meaning a higher SNR is required for results similar to QAM. The results in table 3.1 provide information on some important features for the considered schemes. A lower peak to average power ratio for the Thomson schemes was found when directly compared to QAM, suggesting a more efficient use of available transmitter power. The results also indicate a better spacing between constellation points than QAM was achieved, suggesting for a given transmitter's available transmission space this method is more efficient.

The second 'force method' performed more strongly than the first, indicating that it allowed for the more even spacing of points in $\mathfrak{B}^2$. Minimum distances between points were found to be *equal to two significant figures*, indicating that this optimisation was appropriate, but the nearest *two neighbours* were not equal for all points.



(a) Scheme performance with $M = k = 16$

(b) Scheme performance with $M = k = 64$

Figure 3.6: Plots to show BER vs SNR on Thomson and QAM with an identical rate in $\mathbb{R}^2$, the two 'force methods' are shown to display the difference improved when the Riesz s-energy method was used

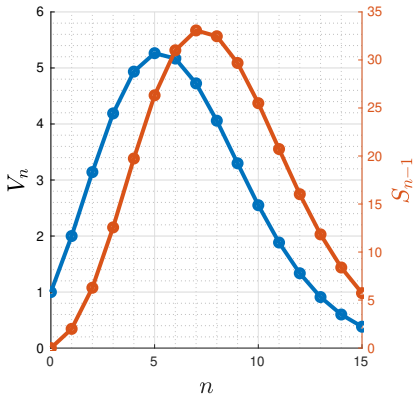
3.2.4 High-Dimensional Considerations and Future Work

Firstly, more work is required on the optimiser for the case where $\mathcal{C} \subset \mathfrak{B}^n$, there is little (if any) literature on equispacing points inside volumes and it is not known if the the codes are completely

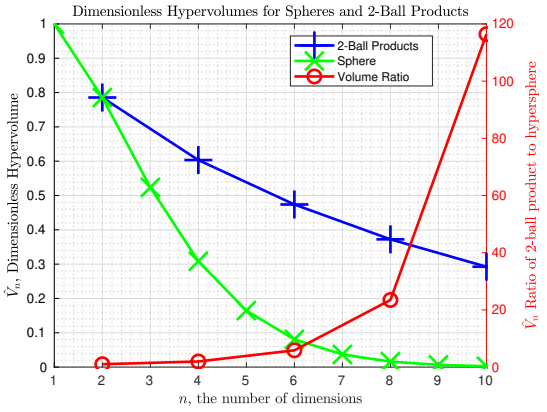
optimal for the space under the Riesz s-energy assumption. More work is also required in understanding how to properly define forces between points, as basic experimentation revealed that better results can be generated with different force functions. Perhaps a more sensible approach would be to consider *potential fields* around each point, and then for all the points to *flow* (update their position) in the direction that gives the greatest decrease in potential.

Different space constraints for the optimiser are also of interest to work with, such as for the case where the power limit is defined *per channel* rather than *per symbol*. Although this solves a slightly different problem, the space $\mathfrak{D}^n$ could be used for the maximum power per channel constraint.

The plot shown in figure 3.7a shows the magnitude of hypervolume of $\mathfrak{B}^n$ as well as the hypersurface area of $\mathfrak{S}^{n-1}$ with respect to dimension number n. The non-monotonic behaviour of the hypervolume and hypersurface area may seem like something which can be exploited, however once dimensions (in terms of units of length) are assigned to each dimension it is without meaning to compare between different n (an analogy being comparing a length to an area). A true comparison as shown in figure 3.7b is obtained by using a *dimensionless measure* of volume. The measure which was used to demonstrate was the ratio of $\mathfrak{B}^n$'s volume to a circumscribed n-cube. The experiments reveal a monotonic decline in hypervolume and hypersurface area available, which suggests high-dimensional generation of codebooks in $\mathfrak{D}$, $\mathfrak{B}$ and $\mathfrak{S}$ may be a poorly-placed focus.



(a) A diagram to demonstrate how hypervolume (V_n) of $\mathfrak{B}^n$ and hypersurface area (S_{n-1}) of $\mathfrak{S}^{n-1}$ change with dimension number n



(b) A diagram to demonstrate how *Dimensionless hypervolume* ($\hat{V}_n$) of $\mathfrak{B}^n$ and of $\mathfrak{D}^n$ change with dimension number n

Figure 3.7: Exploratory work into power limits for higher dimensions

3.3 Auto-Encoder Modulation

3.3.1 Justification

All the methods employed to solve the maximum minimum distance problem as expressed in equation (3.3) on page 16 use fairly standard algorithms, greedy in maximising minimum distance. It was believed there might be some interest in using machine learning methods to see if any better results can be achieved given certain constraints on transmitted symbol power.

A potential benefit is that given a large codebook (to the point where it isn't feasible to store all possible messages) all the different codes need not be stored and the detection can be performed by the network, without any need for high dimensional distance finding.

Auto-encoders are of specific interest as they are specifically designed to compress a high (m)-dimensional object (the original $\mathbb{F}_2^m$ data) into a low (n)-dimensional latent space (the transmitted symbol). The field of communications has seen auto-encoders before [15], however their use has not so far extended to any practical systems.

To obtain a high-level overview on how auto-encoders work, with reference to figure 3.8, binary data is fed into the 'Input Layer' and the network compresses this down to a n dimensional latent space (the Hidden Layer, the transmitted message). The n-dimensional latent space is clipped such that a value on each dimension can exist only within the bounds $[0, 1]$.

A Gaussian stationary random variable is added to each of the outputs of the encoder (g^{-1}) and the noisy n-dimensional latent space is then fed to the decoder (g). In the diagram the channel is shown as going from the layers with x to the layers with y. This is used when training to enforce good distance properties, and can be interpreted as a Gaussian channel in practise.

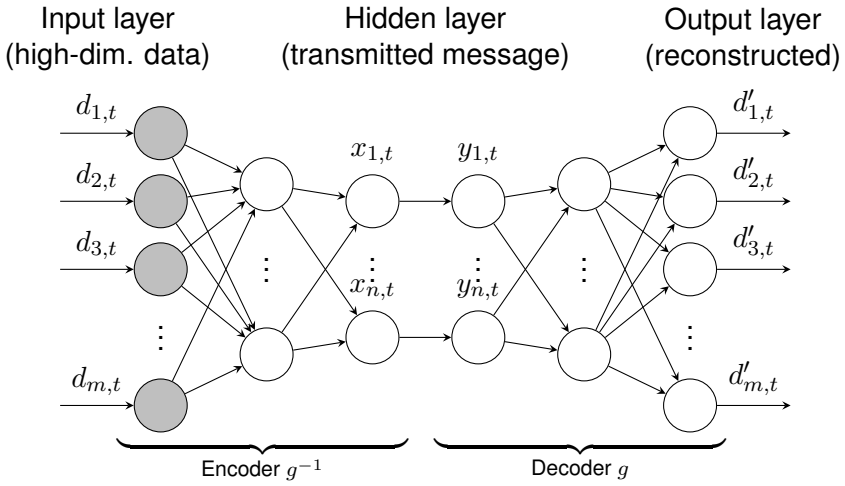


Figure 3.8: The auto-encoder structure used, showing how a single batch of data $d \in \mathbb{F}_2^m$ is processed at time t

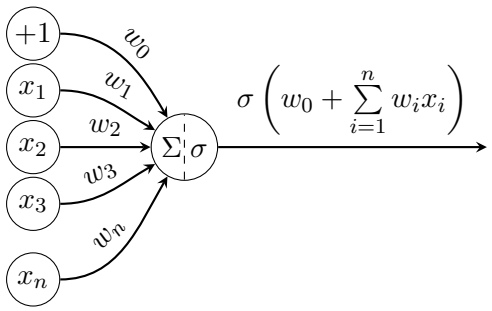


Figure 3.9: A detail view of each non-shaded neuron (with reference to figure 3.8 on the previous page), as used in experiments

3.3.2 Implementation Details and Results

An experiment was run to test if an auto-encoder can naturally find a better scheme than what has been discovered or proposed.

The input vector d was four bits of binary data ($d_{\text{in}} \in \mathbb{F}_2^4$) in a two-dimensional latent space, with limits in the range $[-1, 1]$. The net used was symmetric (the number of layers in g equalling the number of layers in g^{-1}), with five layers for each the encoder and decoder.

The decision was made for each layer to work as a sum and and a sigmoid operator as shown in figure 3.9, where the sigmoid operator is defined as:

$$\sigma(x) = \frac{1}{1 + \mathrm{e}^{-x}} \tag{3.10}$$

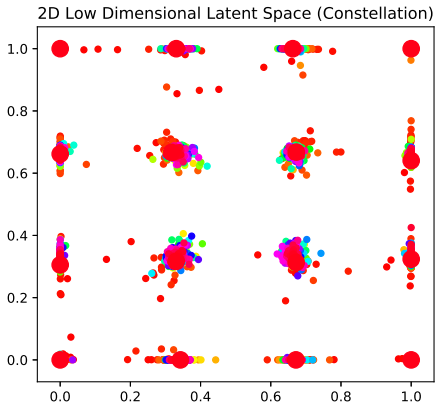
Sigmoid was chosen as the activation function for the neurons as ReLU (Rectified Linear Units) and other functions could not be found to converge within the limited time and hardware available for the project.

The channel's noise power (the variance of the Gaussian random variable) was initialised at a very low value (1×10^{-6}) and gradually crept up until the net couldn't converge.

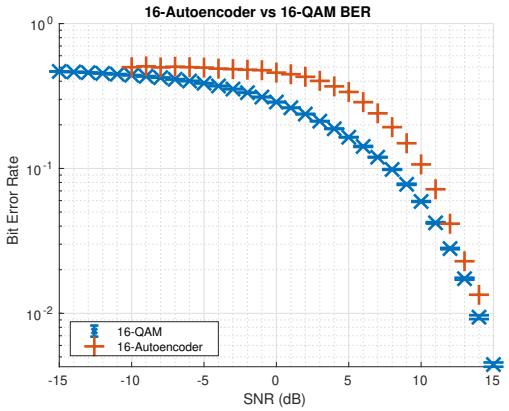
The auto-encoder was trained on random binary data (each bit being distributed as a Bernoulli distribution, $\mathrm{Bern}\left(\frac{1}{2}\right)$) and a squared-error loss function was used to calculate gradients.

A *Dropout* was used at the end of each layer, which has a probability of turning off a neuron's output. Dropout is used to aid nets in converging, as some neurons may end up being 'stuck on' or off which is undesirable. The dropout has a probability of being on represented by the random variable $D' \sim \mathrm{Bern}\,(\gamma)$, the decision was made to set $\gamma = 0.99$ as it was sufficient to let the net converge without hindering training speed.

The results of training the net for sufficiently long on the constraints noted above found *eventual convergence to QAM*, as visible in figure 3.10a on the next page. Some effort was put into *normalising the output vector* such that it lands on the $(n-1)$-sphere (as in the Thomson Modulation), but sufficient



(a) Constellation variation through training (red to violet, with large red points as the final result). *Note the convergence to 16-QAM. In this diagram the x-axis is in-phase, y-axis is quadrature.*



(b) Results of a trial run compared directly to QAM with an identical rate ($M = 16$). Error bars are not available due to how batches of samples were processed in *TensorFlow*.

Figure 3.10: Best results achieved from running the neural net in *Tensorflow*. 5 dense $\mathbb{R}^{64\times1}$ fully layers each with sigmoidal activation and dropout (0.99) on *both* g and g^{-1}, trained on 400K examples at 8dB SNR on channel, learning rate of 0.002, batch size of 512.

compute power was unavailable to make the net converge, so the search was halted here.

Results were available for a codebook of size $M = 16$ in a two dimensional space so a comparison to 16-QAM was created. By running the neural net's 'channel' layer at different SNRs and performing a threshold on the output of the net, the results can be directly compared. The thresholding operation (in Python) is expressed as:

```
bitsOut = [1 if decodeOut_ > 0.5 else 0 for decodeOut_ in decodeOut]
```

Worse performance was observed with the auto-encoder, with the bit error rate decreasing at a faster rate but at higher SNRs than maximum-likelihood QAM. The faster rate of BER decrease on the auto-encoder scheme slowed and converged with the results for QAM on higher SNRs. This may be due to the auto-encoder not discovering *Gray coding*, which, when observed the auto-encoder failed to do correctly. The auto-encoder is not a perfect system as it was not trained for very long (only four hundred thousand examples) and had a very simple architecture of a multi-layer perceptron.

Potential benefits still exist in higher dimensions where M can be left to grow large, and a maximum-likelihood decoder is as a result not required to search a large space, conceivably offering power savings by just performing a few layers of tensor operations.

3.3.3 Challenges

Auto-encoders do come with drawbacks, for large M the auto-encoder needs to see each of the 2^M combinations *at least once* and more for accurate models. Train times therefore increase exponentially, suggesting auto-encoders are not a viable solution to the problem of communications

due to prohibitively large training effort.

A potential highlighted improvement is since (in RF) communications operates in a complex space, it would also be ideal to have complex-valued neural networks, yet current networks only exist in real space [15]. This factor also brings into question what neural net structure (in terms of layers, loss functions, activation etc) should be used, as there is no clear answer as to what the optimal structure is for the problem of communications.

Typical advanced schemes use feedback, which is a memory effect which until recently was not able to be incorporated into neural nets due to the lack of it being differentiable [16].

An open question in communications is what SNR S to train the network at, research and experiments have shown that training at low SNR does not allow for the discovery of structure vital for high SNR communication [15]. Training across many SNRs severely effects training times, so the problem of the 'optimal' SNR to train at remains open.

3.4 Summary on Distance Based Schemes

The work on distance based schemes revealed that little in terms of high-dimensional work had been applied in the field. Conventional schemes were reviewed and offered simple to describe forms but came with the drawback of not using all the space the transmitter has available to it.

The Thomson scheme was made for this report to rectify the problem of not using all available space, and an improvement in the distance properties of constellation points on the space was observed when the optimisation was applied. Despite the improvement of the distance properties, the average power for the scheme increased, which meant that for a given SNR the scheme *appeared* to perform slightly worse than the conventional QAM scheme.

Auto-encoders were also utilised to try and find a structure that maximises distance properties, but the computational effort required grew quickly and work was discontinued due to project time limits.

Overall the analysis showed a $\mathcal{O}(n)$ type growth of the capacity for conventional schemes. The detailed analysis on how the Thomson scheme scales with dimensions was not performed due to being outside the scope of the project.

Chapter 4

Heuristic & Ordering-Based Schemes

This chapter discusses schemes which have a capacity that grows faster than $\mathcal{O}(n)$, with respect to number of resources. Ordering based schemes (like Permutation and Rank Modulation) are discussed and reviewed to show how the benefit of adding an extra resource can provide the reward of a super-linear scaling of $\mathcal{O}(n\log(n))$. Heuristic schemes are explored and probed in the form of attempts to find The Graph Modulation, a scheme which could scale as $\mathcal{O}\left(n^{1+\epsilon}\right)$ for some positive quantity ϵ.

4.1 Permutation Modulation

4.1.1 Definition [3]

In 1965, David Slepian outlined a class of codes and decoders for describing information, in a system called *Permutation Modulation*.

The system relies on *permuting elements of a generating word* to transmit different messages. What this means is that if there exists a generating code word, for instance (ignoring any constraints which may be set out later on $\mathcal{C}$ for the sake of example) $X_0 = \langle 1, 2, 3\rangle$, the elements can be permuted such that all $3! = 6$ possible permutations can be generated in the codebook:
$\mathcal{C} = \{\langle 1, 2, 3\rangle, \langle 1, 3, 2\rangle, \langle 2, 1, 3\rangle, \langle 2, 3, 1\rangle, \langle 3, 1, 2\rangle, \langle 3, 2, 1\rangle\}$

Each of the three elements of a code word would be transmitted on their own *separate dimension*, so two complex channels would be required in this scenario. The 'permutation of the generating word' feature has an interesting consequence in that all messages have identical energy, the maximum that can be transmitted.

If this is now generalised to having a generating code word $X_0 = \langle \mu_0, \mu_1, \ldots, \mu_{n-1}\rangle$ for n dimensions, choosing the appropriate values for each μ_i becomes an interesting problem, as there are $n!$ different list permutations. Before working on choosing optimal μ_i values, Slepian outlined some variants on the scheme which change the way μ_i values are chosen.

A general feature of Slepian's scheme is how he extends the generating code to have *repeating blocks of each value*, where a symbol may look like:

$$X_0 = \left\langle \overbrace{\mu_0, \ldots, \mu_0}^{m_0}, \overbrace{\mu_1, \ldots, \mu_1}^{m_1}, \ldots, \overbrace{\mu_{k-1}, \ldots, \mu_{k-1}}^{m_{k-1}} \right\rangle \tag{4.1}$$

The repeating nature reduces maximum rate per channel, but it provides some robustness to noise through *repetition coding*. Optimising an integer on the number of repetitions (m_i values) is an extremely difficult problem however and was outside the scope of this project, so *multiplicity 1 codes* were explored, which is the case where all $m_i = 1$.

Slepian outlined two different variants to the Permutation scheme, which set out some rules on how the μ_i elements are used.

Variant 1 Coding

Variant 1 of Permutation Modulation allows for each μ_i to be *any real number* such that when considered as a vector lands inside $\mathfrak{S}^{n-1}$, the surface of the $(n-1)$-sphere.

$$\sum_{k=0}^{n-1} |\mu_k| = 1 \tag{4.2}$$

The generating list is permuted only, and the appropriate μ is sent on each channel. The absolute sum of all the μ elements is equal to unity as the assumption of unity is the total maximum power of the transmission scheme still holds. An example constellation can be viewed in figure 4.1a on page 29.

An interesting observation on Variant 1 codes in two and three dimensional space is that since no sign flipping occurs all the points were found to exist on a $(n-1)$-dimensional hyperplane, a subspace of $\mathbb{R}^n$. An especially interesting finding was the optimally spaced code hyperplane passes through the origin. Although Slepian proved that under Permutation modulation Variant 1 that optimal codes satisfy [3]

$$\sum_{i=1}^{k} m_i \mu_i = 0 \tag{4.3}$$

a more intuitive and accessable proof was constructed to support the hyperplane claim.

Claim 1. *The optimal codes (permuted vectors) for a Variant 1 codebook lie on a hyperplane defined by $J_{1,n} X_i = 0$ where J is a vector of ones.*

Proof of Claim 1. The affine hyperplane can be defined as the set of all points that satisfy

$$a_0 x_0 + a_1 x_1 + \cdots + a_{n-1} x_{n-1} = c \tag{4.4}$$

for some constant c.

If the permuted elements are fed into this equation in lieu of x

$$a_0\mu_0 + a_1\mu_1 + \cdots + a_{n-1}\mu_{n-1} = c \tag{4.5}$$

and $a_i = 1$ for all $i \in 1, 2, \ldots$, then the ordering of the μ values does not change the constant, thus proving that permuted vectors satisfy this constraint and therefore all the permuted vectors lie on the hyperplane $\mathfrak{H}$:

$$\mathfrak{H} = \left\{ x \in \mathbb{R}^n \,\middle|\, \sum_{i=0}^{n} x_i = \sum_{i=0}^{n} \mu_i = c \right\} \tag{4.6}$$

Recalling that the permuted vectors are constrained to be on $\mathfrak{S}^{n-1}$, the space of valid points is written as $\mathfrak{V} = \mathfrak{S}^{n-1} \cap \mathfrak{H}$, $\mathfrak{V}$ follows as a $(n-2)$-sphere with radius $r = \sqrt{1-c^2}$ when in the nonempty region of $-1 \leq c \leq 1$.

When codes are optimally spaced, the maximum L_2 distance and therefore arc length between them is used, which occurs when the radius of $\mathfrak{V}$ is maximised. Maximum radius therefore occurs which is when c is zero, or when $\mathfrak{H}$ intersects the origin. □

Variant 2 Coding

Variant 2 of Permutation Modulation is similar but it only allows for each μ_i to be *any real positive number* such that when considered as a vector lands inside $\mathfrak{S}^{n-1}$, (the same constraint as in equation (4.2) on the previous page). Variant 2 however operates with another constraint:

$$0 \leq \mu_0 \leq \mu_1 \leq \cdots \leq \mu_{n-1} \tag{4.7}$$

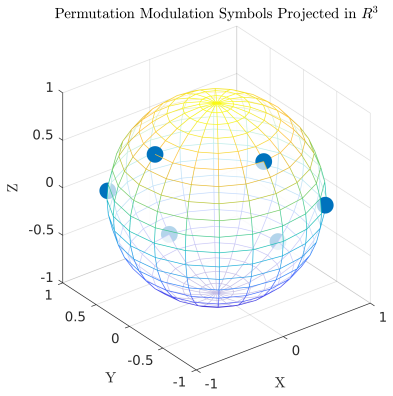
This constraint allows for an interesting method that can be used to encode more messages.

Variant 2 allows for *sign flipping* of each element, re-using the example from earlier where $X_0 = \langle 1, 2, 3 \rangle$ (a valid Variant 2 code), $\mathcal{C}$ grows to include permutations where -1 is used in lieu of 1, -2 in place of 2 and so on for all elements.

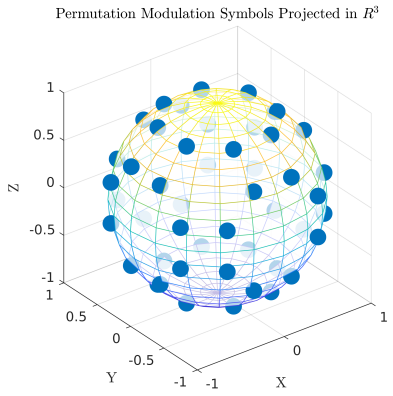
As sending the negative of a μ_i is just a phase effect, identical power per symbol is preserved and similarly to Variant 1, the absolute sum of all the μ elements is equal to unity. An example constellation can be viewed in figure 4.1b on the following page.

4.1.2 Capacity Growth Rate

By considering the result of a linear growth rate from section 3.1.2 on page 15, this scheme offers a significant advantage. For Variant 1 (no sign flipping), the system has n dimensions, which gives n different 'messages' which can be sent from *each dimension*. When considering all n dimensions



(a) 6 Permuted Sign-Flip Points on 2-Sphere (Variant 1)



(b) 48 Permuted Sign-Flip Points on 2-Sphere (Variant 2)

Figure 4.1: Diagrams to convey what optimal permutation codebooks appear as in $\mathbb{R}^3$, each axis is a separate communications dimension

used together in permutation with the constraint that each μ can be transmitted only once, $\log_2(n!)$ bits of transmission are present.

The *Stirling Approximation* can be used to simplify this capacity expression, which is defined as follows:

$$\log_2(n!) = n\log_2(n) - n\log_2(\mathrm{e}) + \mathcal{O}(\log_2(n)) \tag{4.8}$$

When the *Stirling Approximation* is used, the capacity for Permutation Variant 1 can be shown to approximate

$$C_1(n) \approx n(\log_2(n) - \log_2(\mathrm{e})) \tag{4.9}$$

which grows bounded as $\mathcal{O}(n\log(n))$.

For Variant 2 (sign flipping), the system has n dimensions, which provides $2n$ different 'messages' which can be sent from each channel when the sign flip is considered, but when considering all n dimensions used together in permutation, the system therefore has $\log_2(2^n \cdot n!)$ bits of transmission. When the *Stirling Approximation* is applied again for this scheme the result is:

$$C_2(n) \approx n(\log_2(n) - \log_2(\mathrm{e}) + 1) \tag{4.10}$$

which also grows bounded as $\mathcal{O}(n\log(n))$, as $\log_2(\mathrm{e}) > 1$.

Another, perhaps more intuitive way of looking at Variant 2 codes is to consider the permutation and the sign as separate entities. The signs of the n elements can represent a n-bit long word, and the absolute value of the elements resembling the permutation.

These results give *fractions of a bit*, which does not have a clear-cut solution to using them. These *spare messages* may be useful for error detection (perhaps in check-sums), but for this project the

set was merely cleaved to only include possible messages (the first $2^{\lfloor \log_2(|\mathcal{C}|) \rfloor}$). For example, for the Variant 1 code example from earlier with a generating list $X_0 = \langle 1, 2, 3 \rangle$, the last two elements are culled from the list such that only $2^{\lfloor \log_2(3!) \rfloor = 4}$ elements remain.

4.1.3 Binary to Permutation and Back

To describe a permutation, a method using *Lehmer codes* [17] can be used to systematically describe the ordering and thus convert from binary to a permutation, and back. This method is advantageous to have as it prevents the need for the modulator and demodulator to store a large lookup table.

The *default list* is assumed to be X_0[1], which is ordered. Given a permuted list of n elements, the first element of the default list can end up in n different positions, so it can be assigned a number between 0 and $n-1$. The next number has $n-1$ possible locations, so it is assigned a number between 0 and $n-2$.

A worked example is given with the following permuted list: $X_\diamond = \langle \mu_2, \mu_0, \mu_4, \mu_1, \mu_3 \rangle$, and also given the default list $X_0 = \langle \mu_0, \mu_1, \mu_2, \mu_3, \mu_4 \rangle$. μ_0 is in the second position of $X_\diamond$, so it is assigned index 1. μ_1 is in the fourth position of $X_\diamond$, but since one element is assigned it's in the third *free* position, so it is assigned index 2. μ_2 is in the first *free* position of $X_\diamond$ so it is assigned index 0. μ_3 and μ_4 are hence given indices 1 and 0.

The index sequence (Lehmer code) $X_\diamond$, which is $X_{\diamond,\text{Lehmer}} = \langle 1, 2, 0, 1, 0 \rangle$ is gained from this permutation, but it still has to be converted to binary. The *factoradic system* allows this conversion to take place, which is a *mixed-radix* numeral system. The conversion on the example works as follows:

$$N_{\text{perm}} = 1 \times 4! + 2 \times 3! + 0 \times 2! + 1 \times 1! + 0 \times 0! = 37 \tag{4.11}$$

From which an integer (37) is obtained from the permutation $X_\diamond$. The general formula is:

$$N_{\text{perm}} = \sum_{r=0}^{n-1} X_{\text{Lehmer}}[n-r-1] \times r! \tag{4.12}$$

To go in the direction of a number to a permutation, the following algorithm can be performed to generate a Lehmer code:

1. N_{perm} is saved as variable K, and n is saved as variable N
2. $R \leftarrow \lfloor \frac{K}{N!} \rfloor$ is calculated, and this integer given is the $(n-N)^{\text{th}}$ value in the Lehmer code
3. K is updated to remove the assigned portion $K \leftarrow K - R \times N!$

[1] For Variant 1 this list has different signs, but for Variant 2 the list is entirely positive, with sign flipping occurring by mapping n bits to signify if the sign is flipped. This is just an implementation detail.

4. N is decremented by one and the process repeats at step 2, while $N > 0$

Progressing from a Lehmer code to a permutation is trivial, as it is a mapping pointing which indices are permuted to which other index.

4.1.4 Demodulation

Slepian gave a useful solution to the problem of demodulation in the original permutation modulation paper [3]. Rather than considering decision boundaries and the L_2 norm (a maximum-likelihood detector, as in equation (2.6) on page 9) for deciding which symbol was transmitted, the *dot product* can be used to assign a received symbol *which is also maximum-likelihood*.

$$X_\star^t = \arg\left(\max_{X_i} X_i^\top Y^t\right) \tag{4.13}$$

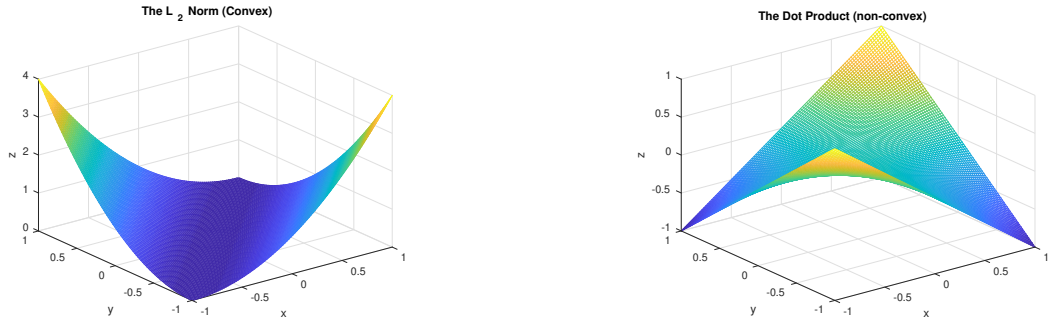
In this project, this method was used after comparing to nearest-neighbour (which gave identical results), as it offered significant speed improvements in the simulator.

Rank Modulation

Rank modulation is a different decoding method that can be used to decide which permutation was sent. Rank modulation uses a nonlinear sort operation to decode the noisy signal. In MATLAB the `sort` function can be used to obtain the indices as follows

```
[~,orders] = sort(nd_signal,2);
```

These orders are immediately used with the algorithms as noted in section 4.1.3 on the previous page. This method offers significant computation speed advantages, a single operation solves the entire system rather than performing a large number of dot products and picking the maximum. A consequence of only detecting the order of the received values means that rank modulation is constrained to Variant 1 codes, unless a *sign detector* can be used. A sign detector was utilised for this project, so both Variant 1 and Variant 2 codes were compared.



(a) The L_2 norm, a *convex* function, $z = (x - y)^2$ (b) The dot product, a *non-convex* function, $z = xy$

Figure 4.2: Comparison of functions that were attempted for the optimisation of Permutation X_0

4.1.5 Generating List Optimisation

The same rule on conventional schemes as in equation (3.2) on page 16 can be applied to choosing the best elements of the generating list to maximise minimum distance when permuted. This is an important feature to optimise, as maximising the minimum distance between permuted vectors gives optimal robustness against noise.

A function was made (`permscore(x0)`) which when given a generating list X_0 returns a 'utility score' of the code. The score used was the minimum distance between vectors as an L_2 norm, the L_2 norm was chosen as it is a *convex function*, as visible in figure 4.2a on the previous page. Some work was considered on *minimising the maximum dot product*, as the dot product is *much* faster to calculate, but this method was rejected and invalid as the dot product is a *non-convex function* as visible in figure 4.2b on the preceding page.

Convexity was an important feature to have in determining a 'score' for each input X_0. An optimiser was made which optimises the elements (μ_i) of X_0 using the *Nealer-Mead Simplex Algorithm* [18], given the constraints set about in section 4.1.1 on page 26. The problem the optimiser solved was fundamentally

$$X_0^\star = \arg\left(\max_{X_0 \in \mathfrak{S}^{n-1}} \texttt{permscore}\left(X_0\right)\right) \tag{4.14}$$

The problem could not be immediately optimised due to the rate at which the codebook's size grows. *For the case of Variant 2 codes* the complexity of `permscore(x0)` was originally found to be $\mathcal{O}\big((2^n n!)^2\big)$, which was deemed very poor. For instance, for five dimensions in Variant 2, $2^5 \times 5!$ (3840) vectors must be generated and then the minimum distance found between all of them, giving 14745600 distances that must be calculated *and* compared. The solution to allowing this problem to be solved more efficiently is to observe the symmetry of the permuted vectors. For instance on the 2-Sphere (as in figure 4.1b on page 29), two orthants were considered as this means the minimum distance between points in the orthant as well as distance to the nearest points outside of the orthant can be considered. There are $n!$ points in each orthant (visible in figure 4.1b on page 29 as the hexagon of points), so once the simplification is applied $2n!$ points only need to be considered. When expanded to higher dimensions, more orthants need to be analysed. There also exists symmetry in the way the L_2 norm works, it is commutative, so only half the number of distances need to be calculated. Applying these simplifications improves the complexity of `permscore(x0)` to $\mathcal{O}\big((n!)^2\big)$. *For the case of Variant 1 codes* all points need to be considered, the optimisation considering calculating 'half' the L_2 norms was also still valid meaning comparatively few L_2 norms needed to be calculated.

A Heuristic Result

After running many optimisations, work was performed on finding a formula which closely approximates the optimally found generating list for Variant 2 due to its much slower optimisation, and allows for a close starting point for fast convergence to the optimal generating list. The approximated ideal levels for Variant 2 were

$$X_0'[i] = (i+1)\left(1 + \frac{n+1}{9}\right) - \frac{n+1}{10}, \qquad i \in 0, 1, \dots \tag{4.15a}$$

$$X_0 = \frac{X_0'}{\|X_0'\|_2} \tag{4.15b}$$

Where n is the number of dimensions considered for the permutation scheme

4.1.6 'Optimal Code' Performance and Random Codes

To benchmark the optimal generating list X_0, experiments were performed against randomly generated codes. A *Uniform Distribution* was used $A \sim \text{unif}(-1, 1)$ to obtain a list of n random samples $\langle a_0, a_1, \dots, a_{n-1} \rangle$ giving one random sample for each dimension. Using the n random samples, the generating code X_0 is constructed as

$$X_0'[i] = a_i, \qquad i \in 0, 1, \dots \tag{4.16a}$$

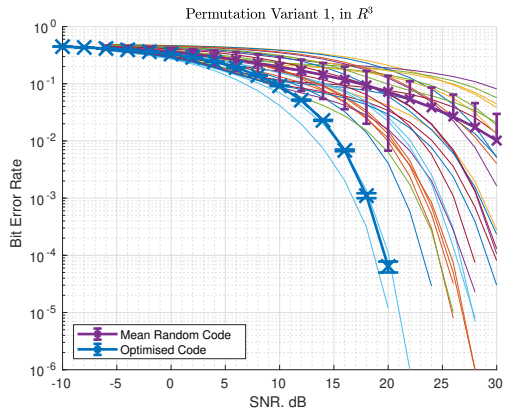
$$X_0 = \frac{X_0'}{\|X_0'\|_2} \tag{4.16b}$$

Results indicated (as shown in figure 4.3 on the following page) that the optimised code performed extremely strongly, and in fact was the best contender on all *Variant 2* scheme tests. There exists a better codebook for Variant 1 in $\mathbb{R}^3$ as there are six elements in the permuted codebook, yet only $2^{\lfloor \log_2(6) \rfloor} = 4$ codes being usable. With two codes being removed arranging the codes to be in the form of a square gives better distance properties than a hexagon. Optimally removing elements was not explored in this project, hence the reason a better code existed in the $n = 3$ case on Variant 1.

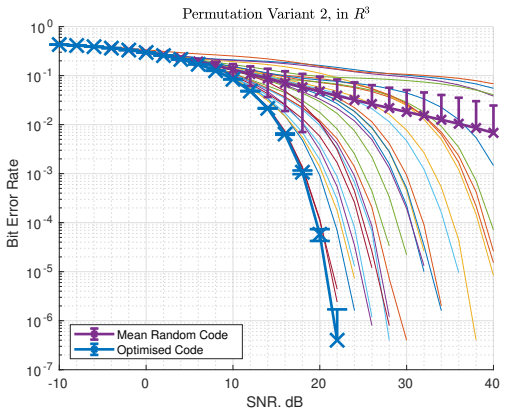
Overall the *Nealer-Mead Simplex* optimisation method proved an outstanding success, due to the ability to make the problem convex and optimise for a solution. The performance of the optimised code against the random code benchmark comes to light as n grows, the gap between the optimised and random codes grow in turn, with a clear example shown in figure 4.3d on the next page.

Comparison to QAM and Thomson

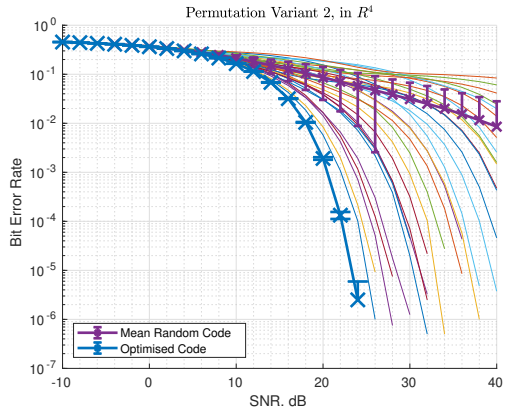
Permutation on 4 dimensions (two complex channels) Variant 2 was selected due to higher rates and better point distribution over $\mathfrak{S}^{n-1}$. This permutation choice gives $\lfloor \log_2(2^4 4!) \rfloor = 8$ bits over two complex channels, or four bits per channel which gives an identical rate to conventional 16-QAM.



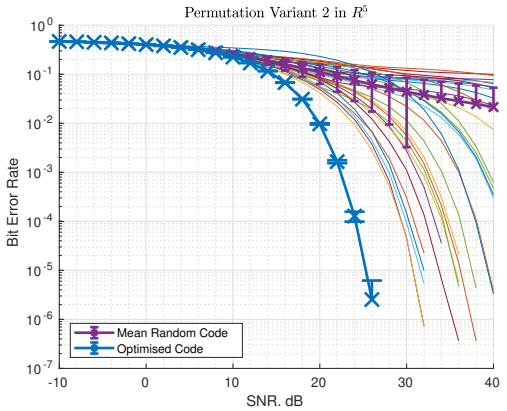
(a) Performance of Permutation in $\mathbb{R}^3$, Variant 1



(b) Performance of Permutation in $\mathbb{R}^3$, Variant 2



(c) Performance of Permutation in $\mathbb{R}^4$, Variant 2



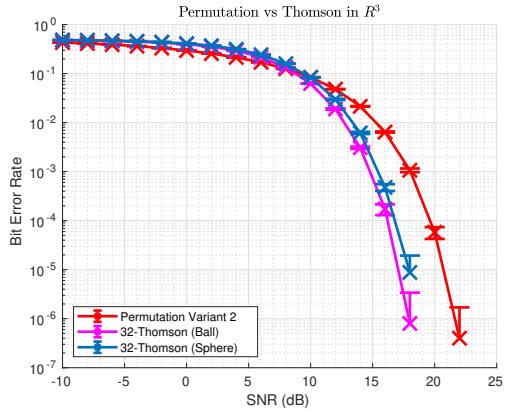
(d) Performance of Permutation in $\mathbb{R}^5$, Variant 2

Figure 4.3: Performance of optimised vs *random codes*, optimised codes were never beaten by random codes in Variant 2

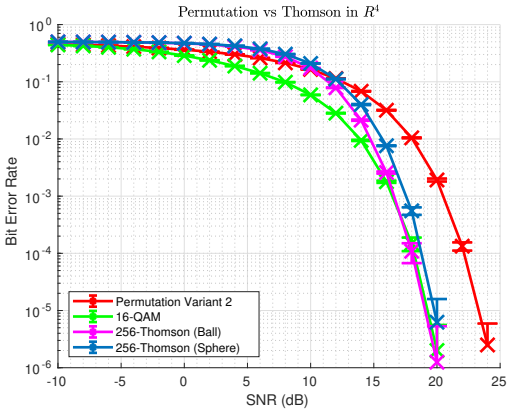
The experimental results (visible in figure 4.4b on the following page) showed Thomson modulation offered significant performance increases over Permutation, which suggests that the Permutation scheme uses space poorly. QAM however was the best performer of all the schemes, but not significantly better than Thomson owing to its faster roll-off in BER. Experimental results in $\mathbb{R}^3$ (visible in figure 4.4a on the next page), again indicated that Thomson was again a better performer than Permutation.

Comparison of Maximum Likelihood Decoder Against Rank Decoder

The rank decoder as outlined in section 4.1.4 on page 31 was quantitatively compared against maximum likelihood. The experimental results shown in figure 4.5 on the next page show that maximum likelihood provides a small improvement over rank decoding for Variant 1, but only by fractions of a decibel (when considering the point at which BER = 10^{-3}). Under Variant 2 (with sign flips), the BERs appeared to be much closer between rank and maximum likelihood, with rank having marginally better performance than maximum likelihood. This result may be due to the way the sign detector assigns bits, providing some noise robustness even when the specific permutation cannot be determined. This result is significant as rank decoding allows for significantly less computation to occur



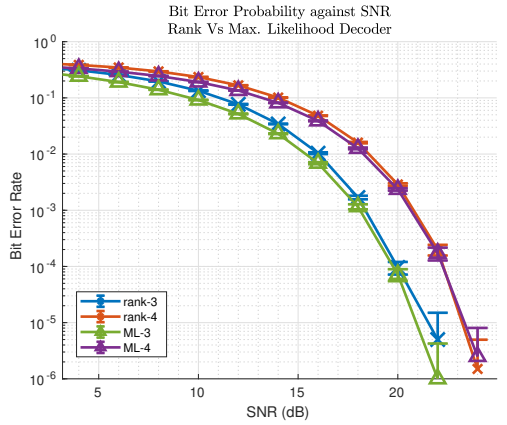
(a) Performance of 32-Thomson ($\mathcal{C} \subset \mathfrak{S}^2$ and $\mathcal{C} \subset \mathfrak{B}^4$) vs Permutation in $\mathbb{R}^3$, Variant 2



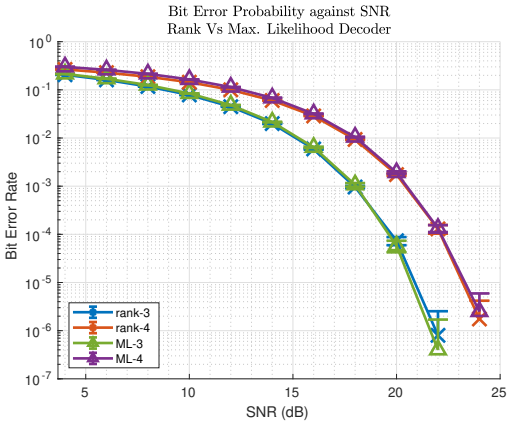
(b) Performance of 256-Thomson ($\mathcal{C} \subset \mathfrak{S}^3$ and $\mathcal{C} \subset \mathfrak{B}^4$) vs Permutation in $\mathbb{R}^4$, Variant 2. 16-QAM is compared as two 16-QAM channels have an identical capacity

Figure 4.4: Performance of Permutation Variant 2 compared with Thomson in 3 and 4 dimensions, this is used to gauge how well Permutation uses the available space

to find the assigned symbol (i.e. no high dimensional search is required), and its output is naturally prepared for conversion to Lehmer codes and therefore binary. The result shows that by only using comparatively few mathematical operations $\mathcal{O}(n \log(n))$ scaling can be exercised in practise.



(a) Permutation vs Rank decoding for the *optimised codebook*, Variant 1



(b) Permutation vs Rank decoding for the *optimised codebook*, Variant 2

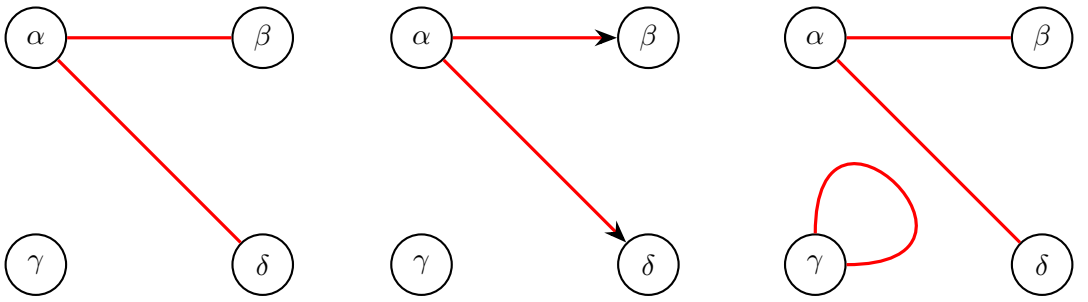
Figure 4.5: Comparing Permutation Modulation with the Maximum Likelihood decoder against the Rank decoder

4.2 Graph Modulation

4.2.1 A brief introduction to graphs

Graph modulation looks at modulation from a different angle, the goal is the find a scheme which allows for the combination of multiple channels to transmit an object structured as a *graph*. A graph in this context is a combinatoric structure amounting to a set of objects (vertices, $\mathcal{V}$) in which some pairs are related (connected by edges). Some examples of graphs are visible in figure 4.6.

The nomenclature for labelling graphs in this section is to let the nodes have letters of the Greek alphabet, such that the set $\mathcal{V} = \{\alpha, \beta, \gamma, \delta, \dots\}$ contains all the *labelled* nodes in the graph.



(a) An *undirected* graph (b) A *directed* graph (c) A graph with a self-loop

Figure 4.6: Examples of *labelled graphs* with four nodes, with differing properties

Adjacency matrices are another method of representing *graphs*. Using figure 4.6, some examples of adjacency matrices with A, B and C being subfigures a, b, and c are:

$$A = \begin{array}{c} \\ \alpha \\ \beta \\ \gamma \\ \delta \end{array} \begin{array}{c} \begin{array}{cccc} \alpha & \beta & \gamma & \delta \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \end{array}, \quad B = \begin{array}{c} \\ \alpha \\ \beta \\ \gamma \\ \delta \end{array} \begin{array}{c} \begin{array}{cccc} \alpha & \beta & \gamma & \delta \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \end{array}, \quad C = \begin{array}{c} \\ \alpha \\ \beta \\ \gamma \\ \delta \end{array} \begin{array}{c} \begin{array}{cccc} \alpha & \beta & \gamma & \delta \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \end{array} \tag{4.17}$$

On the adjacency matrix, a simple graph with no self-loops will have zeros on the diagonal and undirected graphs will be symmetric. With the goal in mind that a modulation scheme needs to be designed, the decision was made to work with *undirected, simple graphs* for this section as they are the simplest case with graphs.

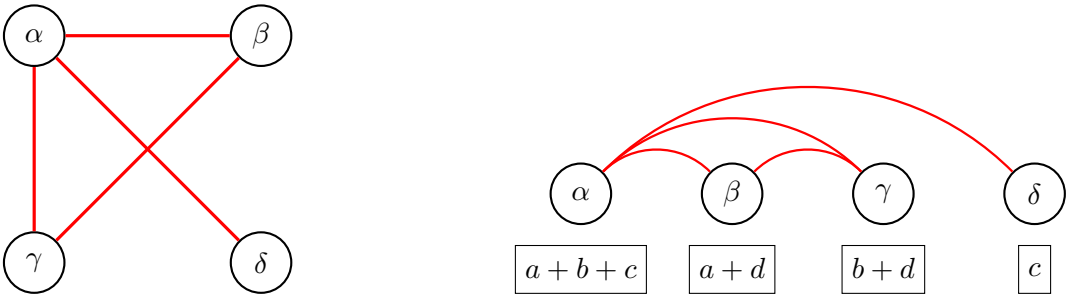
Number of different possible graphs

The number of different possible graphs given a set of vertices $\mathcal{V}$ where $|\mathcal{V}| = r$, and working under the *undirected* and *no cycles* constraints the number of possible distinct edges is $N_{\mathcal{V}} = \frac{r(r-1)}{2}$, giving $2^{N_{\mathcal{V}}}$ different possible graphs. This is an excellent motivator, as if a scheme can be developed

that considers each edge as a bit and each channel as a node, a $\mathcal{O}(n^2)$ growth in capacity would be possible. For example, under the case of a fifty node graph, $\frac{50(50-1)}{2} = 1225$ edges are present, in a communications channel where there are 50 nodes and each individual node can be adequately represented by a single channel, a capacity of $\frac{1225}{50} = 24.5$ bits per channel per symbol is achieved. Under comparison with M-QAM, this would require $M \approx 2.37 \times 10^7$, which is entirely infeasible.

4.2.2 Distinct Sets of Subset Sums

The first method considered to tackle the problem of encoding a graph of $\frac{n(n-1)}{2}$ bits on n channels was to use a property called *distinct sets of subset sums*. This strategy would be to try and 'relate' resources together through a shared element in each of their sub-symbols. Using figure 4.7 as a guide, each related channel pair would share the same element, for example with channel α and γ, they both share the element b in each of their sub-symbols.



(a) Unstructured example

(b) Distinct sets of subset sums example

Figure 4.7: Both figures show the same *undirected* graph but one is represented in a format to demonstrate how encoding with *distinct sets of subset sums* would appear

The key detail is for the sums on each channel to be distinct, such that the elements used (a, b, c, d in the example above) can be determined. To obtain some intuition on how the distinct sets of subset sums works for each channel's individual sub-symbol, each sub-symbol X_i must be some distinct sum of elements of a set $\mathcal{S}$

$$X_i = \sum_{g \in \mathcal{G}} g \text{ with } \mathcal{G} \subset \mathcal{S} \tag{4.18}$$

If the detail on $\mathcal{S}$ is ignored and distinct subset sums is assumed true and a 'sub-codebook' is created for each channel to use,

$$\mathcal{C}_{\text{sub}} = \left\{ \sum_{g \in \mathcal{G}} g \text{ with } \mathcal{G} \subset \mathcal{S} \right\} \tag{4.19}$$

there are a maximum of $2^{|\mathcal{S}|}$ different possible sums, and therefore a maximum of $2^{|\mathcal{S}|}$ elements present in $\mathcal{C}_{\text{sub}}$. This result is easily seen through each element in $\mathcal{S}$ either being added or not, which can be written as a binary array signifying if each element is added. What this result implies is the binary representation of numbers points to the only way to create the finite set $\mathcal{S}$ of which subsets

can be taken, and their elements summed to *distinctly* describe *all integers* up to a limit. For example the set $\mathcal{S} = \{2^0, 2^1, 2^2, \ldots, 2^{k-1}\}$ can distinctly generate any integer up to $2^k - 1$. For the case of encoding graphs as in the earlier example, $|\mathcal{S}| = n - 1$.

This result gives no benefit, as it merely describes the communications problem in a convoluted way. Namely, if an additive codebook is made, there still needs to be $2^{|\mathcal{S}|}$ one-hot vectors in the sub codebook for each channel. This realisation made it clear that although not a solution to graph modulation, this route can be explored through simplifying it by encoding the edges present with an existing scheme.

4.2.3 Edge-Encoding Schemes

The next step in searching for a solution to graph modulation was to look at using the graph's structure to encode bits. As noted in section 4.2.1 on page 36, the number of edges in a graph grows as the square of the number of nodes, so a logical approach is to attempt to encode the edges as bits.

A scheme was devised which uses the properties noted in section 4.2.2 on the preceding page of 'relating' resources together, and while it has the potential to offer benefits, the scheme ends up reducing to repetition coding.

First (by a slight abuse of standard notation), the *Star Graph* H_i is defined, this is a complete bipartite graph with a single internal node and and edges connecting to all other nodes. An example for the case H_α is visible in figure 4.8.

For sake of example the $n = 4$ ($N_\mathcal{V} = 6$) case is shown, with the example data $d = \langle 1, 1, 1, 0, 1, 0 \rangle$, giving graph G as visible in figure 4.9.

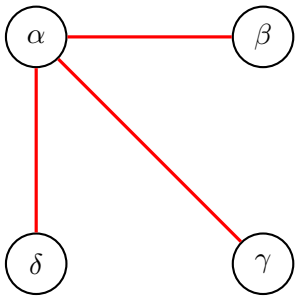


Figure 4.8: A star graph, H_α

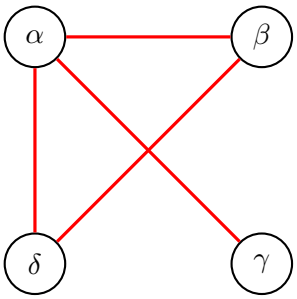


Figure 4.9: The example graph G which will be used to demonstrate this scheme

The devised scheme works as follows, $d \in \mathbb{F}_2^{N_\mathcal{V}}$ is transformed into a graph by *raster filling* the adjacency matrix's lower and upper triangles

$$G = \begin{array}{c} \\ \alpha \\ \beta \\ \gamma \\ \delta \end{array} \begin{array}{c} \begin{array}{cccc} \alpha & \beta & \gamma & \delta \end{array} \\ \begin{pmatrix} 0 & d_0 & d_1 & d_2 \\ d_0 & 0 & d_3 & d_4 \\ d_1 & d_3 & 0 & d_5 \\ d_2 & d_4 & d_5 & 0 \end{pmatrix} \end{array} = \begin{array}{c} \\ \alpha \\ \beta \\ \gamma \\ \delta \end{array} \begin{array}{c} \begin{array}{cccc} \alpha & \beta & \gamma & \delta \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix} \end{array} \tag{4.20}$$

This graph G is then intersected with H_i, giving partial graphs E_i,

$$E_i = G \cap H_i \quad \forall i \in \mathcal{V} \tag{4.21}$$

The action of creating these partial graphs shows for a single node how it is connected to the rest of the graph. From the point of having partial graphs available, *partial codes* are generated for each node. A diagrammatic view of how this is achieved is shown in figure 4.10, where by looking clockwise around each node and tracking edge presence or absence, the *partial code* can be generated. This can also be achieved by reading each *column* of the adjacency matrix and ignoring the diagonal, as shown in figure 4.11. Each of the n *partial codes* are then encoded and sent on n separate channels with a standard APSK method (for the sake of simplicity, 2^{n-1}-QAM was used as it has consistently displayed high performance throughout this report).

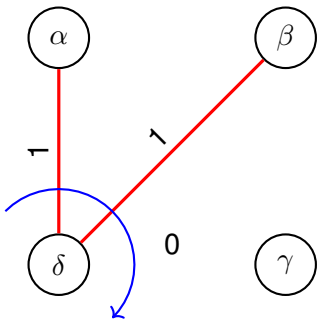


Figure 4.10: The Partial Graph, E_δ, giving the Partial Code $\langle 1, 1, 0 \rangle$

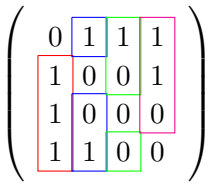


Figure 4.11: A method of obtaining *Partial Codes* involving only the adjacency matrix. Codes are read on the column from *top to bottom*, with the diagonal ignored. Note the magenta code $\langle 1, 1, 0 \rangle$ being the equivalent to figure 4.10

The demodulator obtains n separate complex values (one from each channel) which *can be* individually decoded to the original partial codes, but the decision is delayed until all the partial codes are obtained. Once all the codes are obtained a *maximum likelihood* decision is made using all the information available, as any single channel contains some information on all the other channels.

As G is symmetric and every element in G is transmitted, there are two copies of the binary data

being sent, so this scheme theoretically breaks down to standard repetition coding. This gives the scheme a capacity of $C = \frac{n(n-1)}{2}$ but only $\frac{n-1}{2}$ bits per channel. This scheme also requires 2^{n-1} one-hot vectors *on each channel* (this can be interpreted as having a 2^{n-1}-QAM sub codebook for each channel), making its performance with scale (letting n grow large) poor.

A potential benefit (above regular repetition coding) of this scheme is in the fact that *any two bits are on the same channel only once*. An interesting feature of this scheme is that it operates with the support of $\mathfrak{D}^{2|\mathcal{V}|}$ as *each channel* has its own maximum power constraint, suggesting it should have better point spacing than a scheme like permutation which is constrained to $\mathfrak{S}^{n-1}$.

Implementation Details and Results

The graph scheme used QAM constellation points which were Gray coded with the aim of ensuring the performance is as high as possible. The decoder used was a *maximum likelihood decoder* which has a codebook of all possible combinations of QAM symbols over all the 5 channels.

The scheme devised above was found to perform *extremely poorly* when compared to other identical rate per channel schemes. An experiment was run with $|\mathcal{V}| = 5$, which gives $\frac{5 \cdot 4}{2} = 10$ bits per symbol, or two bits per channel per symbol, the results are demonstrated in figure 4.12. Comparing with which SNR schemes have a BER $= 10^{-3}$, 4-QAM beat 16-QAM repetition coding by 3.9 dB, and beat Five Node Graph Modulation by 29.2 dB. A particular feature to highlight from this experiment was the slow roll-off of the Graph Modulation scheme, a tenfold decrease in BER occurring every 10 dB. The dependence between symbols may cause them to be poorly spaced, i.e. not maximising distance properties, which may be a feature causing the slow roll-off in BER.

The results found suggest that there are better methods than the proposed graph scheme for an identical number of bits per symbol per channel. Similarly, that distributing the bits over all channels may not be the most effective solution to increasing performance. The graph scheme could not be compared for other numbers of nodes due to the dramatic exponential increase in sub codebook size.

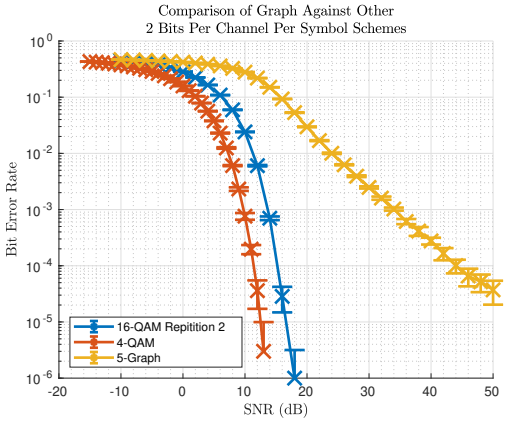
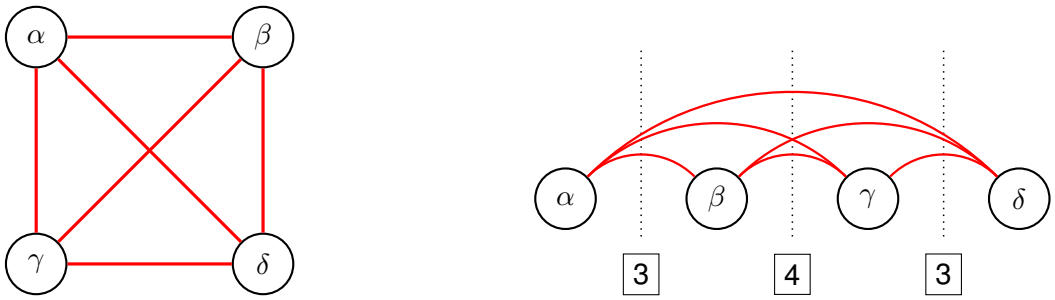


Figure 4.12: The proposed *Edge Encoding* scheme has identical performance and rate to Repetition Coding

4.2.4 A Potential Cut-Encoding Scheme

After the attempts with earlier schemes, and finding they are either impossible or simplify to known methods, a more heuristic approach was employed to try and find a method that offers $\mathcal{O}\left(n^{1+\epsilon}\right)$ capacity. A *potential method* may lie in looking between the nodes and enumerating their connections, rather than explicitly defining how they are connected to one and other. The framework for a scheme was developed that takes a graph, arranges the nodes inline, and looks in the gaps between the nodes to see how many crossings occur with the edges. A tangible example to demonstrate this is in figure 4.13, which shows how by considering the number of crossings of each dotted line gives a code (noted in the boxes below), for the example in figure 4.13b, the code is $X_{63} = \langle 3, 4, 3 \rangle$.



(a) Unstructured example (b) Cut-code example

Figure 4.13: Both figures show the same *fully connected, undirected* graph but one represented in a format that can be used for *cut-codes*

There does however exist *ambiguity* in this method. Cutting a graph in this manner is a *many to one* mapping, with a tangible example in figure 4.15 on the following page. Another way to demonstrate that ambiguity *must exist* is by viewing the maximum number of uniquely codable bits by the 'cut codes' and comparing it to the number of bits in the graph. This is demonstrated in figure 4.14, which shows for $|\mathcal{V}| > 5$ there are no unique mappings for a graph.

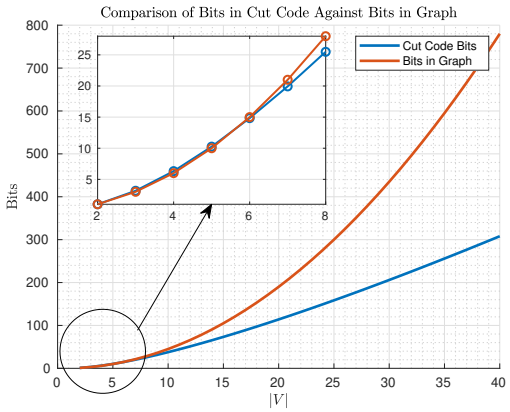
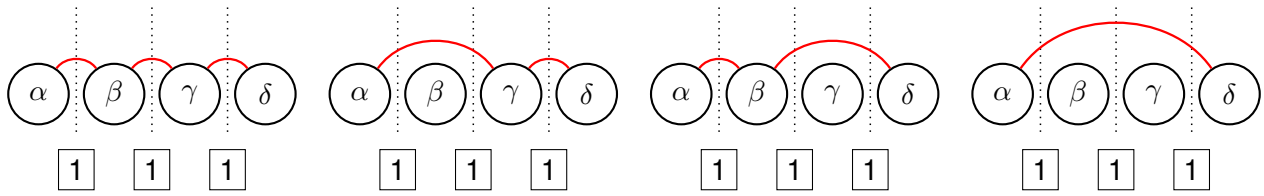


Figure 4.14: The number of bits able to be uniquely coded by the cut-code compared to the number of bits in the graph, demonstrating for a sufficiently large graph ambiguity must exist



(a) Cut-code *3 edges* (b) Cut-code *2 edges* (c) Cut-code *2 edges* (d) Cut-code *1 edge*

Figure 4.15: *Ambiguity* exists in the cut-code structure, it is a many-one mapping, as demonstrated by subfigures a, b, c and d having the same *cut-code*

Ambiguity was attempted to be removed by considering the *graph vertex order* of the central nodes. The graph vertex order is known for the two edge nodes (as denoted by the first and last elements of the cut-code), so the central nodes' orders are transmitted[2] in series after the cut code.

An attractive feature this method is that the numbers generated as the *cut-code* grow quadratically with respect to the number of edges, whereas typical encoding methods require exponential growth. The maximum values a cut-code can have (given $n = |\mathcal{V}| - 1$ channels) is bounded by the multiplication table read by antidiagonals[3] , whereas the maximum value that will exist in a cut-code of n channels is bounded by the quarter-squares[4], $\lfloor \frac{n}{2} \rfloor \lceil \frac{n}{2} \rceil$.

A Potential Decoder

The decoder used relies on a *linear programming optimiser* to attempt to solve the system. For $|\mathcal{V}| < 5$ the system has an equal or greater number of constraints to the number of bits present, but for larger $|\mathcal{V}|$ this is not the case, hence requiring the optimiser.

First the 'net connections from left to right' for each node are calculated from the given cut-code. This describes the node's connections, if it's negative there are more edges connecting from the left than there are edges leaving to the right, the calculation to achieve this is shown by the pseudocode

```
% cutcode is an n-1 length cut-code
cutcode(N) = 0; % a zero is appended - no edges leave the last vertex to the right
for j = 1:N
  if j > 1;
    netconns(j) = (cutcode(j) - cutcode(j-1)); % the difference is calculated
  else
    netconns(1) = cutcode(1); % no edges join the first vertex from the left
  end
end
```

Now the 'net connections' array (of $|\mathcal{V}| - 1$ elements) is generated, the *graph vertex orders* (of $|\mathcal{V}| - 2$ elements) are concatenated on the end, and the information is ready to be put into a solver as the vector X_i'.

[2]Although just an implementation detail the integers for each the cut code and order channels can be sent by simply enumerating the QAM lattice.

[3]The multiplication table read by antidiagonals are the OEIS sequence A003991

[4]The quarter-squares are the OEIS sequence A002620

The solver has the structure

$$\begin{aligned} &\text{Maximise} && J_{1,N_\mathcal{V}} d' \\ &\text{Subject to} && Ad' \leq X'_i \\ &\text{and} && 0 \leq d' \leq 1 \end{aligned} \tag{4.22}$$

where $J_{i,j}$ is an i by j matrix of ones, A is the constraint matrix[5], and d' is the binary data out. An intuitive way to view A is is by looking at the structure of graph when put in a line. If k resembles which element of the array d' is being operated in, blue elements show edges that can be connected to node k on the right, whereas red elements show edges that can be connected to node k on the left. Green elements dictate the vertex order, which is the sum of leftward and rightward edges.

For the $|\mathcal{V}| = 6$ case (15 bits) the matrix A used in the linear programming optimiser is shown in figure 4.16. A significant drawback of this decoder is that A is not *totally unimodular*, meaning the problem is not guaranteed to solve to integers 0 and 1.

```
bits: 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15   Channel:
    [ 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0;  nc1/ord1
     -1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0;  nc2
      0,-1, 0, 0, 0,-1, 0, 0, 0, 1, 1, 1, 0, 0, 0;  nc3
      0, 0,-1, 0, 0, 0,-1, 0, 0,-1, 0, 0, 1, 1, 0;  nc4
      0, 0, 0,-1, 0, 0, 0,-1, 0, 0,-1, 0,-1, 0, 1;  nc5
      0, 0, 0, 0,-1, 0, 0, 0,-1, 0, 0,-1, 0,-1,-1;  nc6/ord6
      1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0;  ord2
      0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0;  ord3
      0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0;  ord4
      0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1]  ord5
```

Figure 4.16: Matrix used by the linear programming solver for $|\mathcal{V}| = 7$ cut codes. `nc` dictates a net connection channel, whereas `ord` dictates a vertex order channel.

Preliminary Results

A simulation was run to decode graphs in a noiseless channel. Since the address space (in bits) grows as $\mathcal{O}(n^2)$, 1×10^6 graphs were uniformly sampled *for each* $|\mathcal{V}|$ with the data d being distributed as in section 2.1.1 on page 7, and were processed in parallel. The results showed the 'cut-code' scheme is far from a complete solution. Adding the node orders only pushed the problem of ambiguity further down the road[6] and graphs are still not uniquely decodable under this method. Figure 4.17 on the following page shows that when considering orders or not, ambiguity is still rife in this scheme, and a better method of removing it is required for any real progress to be made.

The space of solutions for d' grows so rapidly that it cannot be constrained by this method, such that cut-codes are an invalid scheme under these circumstances.

[5] An example of the constraint matrix is given in figure 4.16

[6] When considering the point that the number of bits in the graph overtakes the number of uniquely decodable bits in the symbol as in figure 4.15 on the previous page, this point occurred at $|\mathcal{V}| = 9$

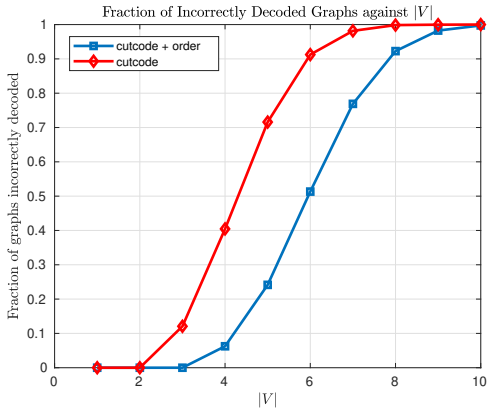


Figure 4.17: A plot to show the proportion of graphs that are incorrectly decoded, given the presence of orders and without

4.2.5 Future Work

More work is required on this scheme to find a solution, and one *may* lie in some advanced number or group theory that was outside the scope of this project. The fundamental problem with this scheme is that the $\mathcal{O}(n^2)$ growth is faster than a lot of mathematical methods, including permutation.

A next step may be in *ambiguity resolution methods* for the *cut-code* scheme, as the structure of the graph (and cut-code) only permit certain combinations of edges. Due to the number of bits per symbol in this scheme growing rapidly, and the data source being maximum entropy, there may be benefits in having ambiguity resolution for *cut-codes* if *on average* the ambiguity takes comparatively few bits to resolve.

A future method may rely on simplicial complexes, which are group of objects that graphs are a subset of. While graphs have nodes and edges, simplical complexes have nodes, edges, triangles, and high dimensional counterparts. The number of simplicial complexes given r vertices is known as a Dedekind number ($M'(r)$). Dedekind numbers are only known for[7] $0 \leq r \leq 8$, but upper and lower bounds are known to be [19]:

$$\binom{r}{r/2} \leq \log_2\left(M'(r)\right) \leq \binom{r}{r/2}\left(1+\mathcal{O}\left(\frac{\log(r)}{r}\right)\right) \tag{4.23}$$

This is an alarmingly fast rate of growth, as the *logarithm of the number of complexes grows factorially fast*. If this could be formed into a modulation scheme, the number of transmitted bits would grow factorially fast.

[7] As of 10th April 2019

4.3 Summary on Heuristic and Ordering based Schemes

The work performed on ordering based schemes revealed the benefit of $\mathcal{O}(n\log(n))$ type capacity scaling is available to be used in communications systems.

Permutation modulation's codebooks were optimised to use the space available from the scheme in an optimal manner, and were showed to perform strongly when compared to the expected random code performance. Despite the high performance, Thomson modulation had a faster and earlier rolloff to lower BERs, suggesting a more efficient usage of the space by the Thomson scheme.

The decoder of extreme interest from the general permutation scheme is the rank modulation, which does not require the storage of any large codebooks, or any high dimensional searches to be performed. Comparatively few mathematical operations need to be applied for this scheme to gain the $\mathcal{O}(n\log(n))$ capacity scaling offered by Permutation modulation, and the application of rank modulation's nonlinear sort did not result in any significant amount (if any) performance degradation. These details rather excitingly give rank modulation the ability to be applied in practise, meaning that n can be driven to a large value to benefit from the performance increases provided by the scheme.

After exploring permutation modulation, work was then focused on trying to find a scheme which has a capacity that scales as $\mathcal{O}(n^{1+\epsilon})$ while using the structure of a *graph*. No useful result was obtained from the search, suggesting that more work needs to be performed to adequately use the structure of a graph to gain the mentioned performance increase.

Chapter 5

Conclusions

5.1 A Summary on Capacity Growth Rates

Throughout this project the rate at which scheme capacity grows was explored. Figure 5.1 shows the concluding results of how examples of all the explored schemes capacities grow given the equations derived throughout this report.

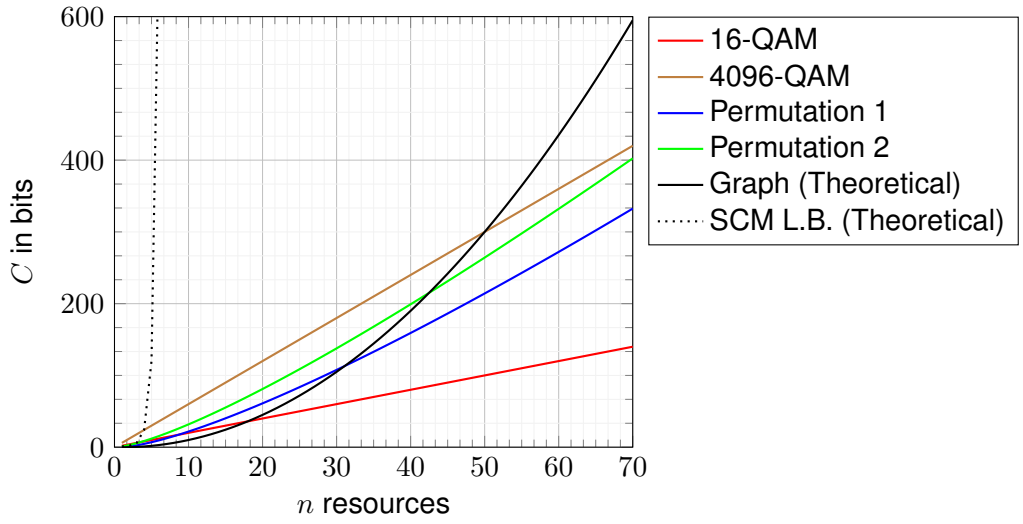


Figure 5.1: Scheme Capacity (per symbol) vs Number of Resources, note that one 'resource' is a single *dimension*, so for permutation a single channel yields $n = 2$ dimensions, and conventional schemes (QAM) have $n = 2$ being a single channel, with Graph Modulation and Simplicial Complex Modulation lower bound added as a future limit

The scaling of scheme capacity is not the entire goal when communications is in mind, as the bit error rate for a given SNR must be adequately low to ensure the scheme is viable. The BER to SNR curves for schemes with identical capacity per channel was interrogated, and room for improvement was found to exist.

The way the power limit is defined for the communications problem makes comparing between schemes challenging, due to different spaces inside being used. Nonetheless schemes with similar

space usage were used to compare against, with the two dimensional case of QAM acting as a benchmark against the majority of them.

Results showed the bit error properties of the devised 'in ball' Thomson scheme were worse than that of QAM despite better distance properties being achieved. The Thomson scheme also proved useful as a meaningful comparison against Permutation schemes, highlighting the poor use of space permuting a vector gives. The flexibility of the Thomson scheme allowed for arbitrary numbers of codes to be defined for the 'on sphere' and 'in ball' cases, such that Permutation could be compared against for any rate. The Thomson scheme's method of spacing points provided a large performance improvement over what Permutation offered, but lacked the structure which would allow algorithms to efficiently decode a received code (as with rank decoding).

Overall the problem of communications is a trade-off and practical engineering decisions need to be made when choosing the best scheme for a system. Rank modulation's efficient decoding method coupled with the super-linear scaling highlight it as a very interesting scheme that needs more work focused on it. The same *optimal generating list* made for permutation was used to compare against rank, however (especially under Variant 2 systems) the true optimal generating list for rank may differ from the one used. Exploring the intricacies of rank modulation generating lists was beyond the scope of this project and is an interesting feature to look at in the future.

5.2 Project Conclusion

Conventional schemes like QAM were probed to find they have a linear growth rate, where the number of the elements in the constellation dictates the 'constant' multiplying the number of channels used.

The Thomson scheme was developed to space points optimally on the $(n-1)$-sphere, and experimentation revealed an improvement in minimum distance properties when compared to QAM and permutation. Since the problem of a general solution that places points inside a n-dimensional space is an open problem, it is not known if the Thomson modulation has an optimal solution for inside the ball. The capacity growth rate of this scheme, much like QAM, scales linearly but potential benefits are offered in high dimensional space usage. The efficiency of point-packing in high dimensional space was not explored due to the complexities of high dimensional spaces being outside the scope of this report, and therefore the capacity growth rate was unable to be analysed formally. An analysis of the high dimensional volume was run which revealed that the n-ball may not be the best possible support for a modulation scheme to use. The suggestion that a scheme could use a space inside the *Cartesian product of 2-balls* is especially interesting to follow up as it offers far more volume than the n-ball, due to the power limit being defined differently (per channel rather than per symbol).

Auto-encoders were used to try and find a higher level structure to reduce bit error rate, but the computational complexity of training a model scaled poorly and research was halted due to far many open questions regarding the scheme.

A combinatoric scheme, Permutation Modulation was then deeply investigated to see if there lie any benefits in its use. The scheme's super-linear growth rate of $\mathcal{O}(n\log(n))$ proved to be a valuable asset, and the presence of efficient encoding and decoding methods allow for n to easily be driven large without the storage of large and difficult to search codebooks. These practical benefits could allow for permutation modulation to have a presence in modern-day communications, but more experiments are needed for higher n to fully compare against identical rate schemes for QAM.

The proposal of using a graph as a method of achieving $\mathcal{O}(n^2)$ capacity was explored, but the analysis of the graph structure was found to be very applied and a bit beyond the scope of the project. The use of a graph makes sense as they are well understood objects in combinatorics, the same area of mathematics that allowed for the presence of the only other super-linear scheme explored. A solution to graph modulation has *not been found*, but many methods have been explored to try and exploit the graph structure, and only the two most promising schemes were included in this report. The devised 'edge encoding scheme' currently has extremely poor performance but has room to be improved due to its large amount of usable space and potentially robust encoding properties. The explored 'cut encoding scheme' was found to not be valid due to the many to one mapping of the codes produced, and the problem of ambiguity being a limiting factor to the scheme. A reliable method of resolving the ambiguity of the 'cut encoding' scheme is required if the method has any hope of working correctly.

Overall the properties of using a graph suggest that $\mathcal{O}(n^{1+\epsilon})$ scaling may be possible. More work needs to be performed on graph modulation to find a result that can scale sufficiently fast and without ambiguity.

References

[1] *Global Mobile Trends 2017*. 2017. URL: https://www.gsmaintelligence.com/research/?file=3df1b7d57b1e63a0cbc3d585feb82dc2&download.

[2] *Award of 2.3 and 3.4 GHz spectrum bands- Publication under regulation 111 of the Wireless Telegraphy (Licence Award) Regulations 2018 of results of auction*. 2018. URL: https://www.ofcom.org.uk/__data/assets/pdf_file/0018/112932/Regulation-111-Final-outcome-of-award.pdf.

[3] D. Slepian. "Permutation modulation". In: *Proceedings of the IEEE* 53.3 (1965), pp. 228–236. ISSN: 0018-9219. DOI: 10.1109/PROC.1965.3680.

[4] Naoki Ishikawa, Shinya Sugiura, and Lajos Hanzo. "50 Years of Permutation, Spatial and Index Modulation: From Classic RF to Visible Light Communications and Data Storage". In: *arXiv e-prints*, arXiv:1803.03939 (2018), arXiv:1803.03939. arXiv: 1803.03939 [eess.SP].

[5] A. Jiang et al. "Rank Modulation for Flash Memories". In: *IEEE Transactions on Information Theory* 55.6 (2009), pp. 2659–2673. ISSN: 0018-9448. DOI: 10.1109/TIT.2009.2018336.

[6] Ertugrul Basar. "Index Modulation Techniques for 5G Wireless Networks". In: *CoRR* abs/1604.08315 (2016). arXiv: 1604.08315. URL: http://arxiv.org/abs/1604.08315.

[7] Shigenobu Sasaki, Jinkang Zhu, and G Marubayashi. "Performance of the Parallel Combinatory Spread Spectrum Multiple Access Communication System with the Error Control Technique". In: Jan. 1992, pp. 159 –162. DOI: 10.1109/ISSSTA.1992.665671.

[8] C. E. Shannon. "A Mathematical Theory of Communication". In: *The Bell System Technical Journal* 27.3 (1948), pp. 379–423. ISSN: 0005-8580. DOI: 10.1002/j.1538-7305.1948.tb01338.x.

[9] J.G. Proakis and M. Salehi. *Digital Communications (4th Edition)*. ISBN: 9781283387460.

[10] Andrea Goldsmith. *Wireless Communications*. Cambridge University Press, 2005. DOI: 10.1017/CBO9780511841224.

[11] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. New York, NY, USA: Wiley-Interscience, 2006. ISBN: 0471241954.

[12] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. "Estimating mutual information". In: *Physical review E* 69.6 (2004), p. 066138.

[13] J.J. Thomson. *The Corpuscular Theory of Matter*. Scribner's Sons, 1907.

[14] A.B.J. Kuijlaars, E.B. Saff, and X. Sun. "On separation of minimal Riesz energy points on spheres in Euclidean spaces". In: *Journal of Computational and Applied Mathematics* 199.1 (2007). Special Functions in Harmonic Analysis and Applications, pp. 172 –180. ISSN: 0377-0427. DOI: https://doi.org/10.1016/j.cam.2005.04.074. URL: http://www.sciencedirect.com/science/article/pii/S037704270500748X.

[15] T. O'Shea and J. Hoydis. "An Introduction to Deep Learning for the Physical Layer". In: *IEEE Transactions on Cognitive Communications and Networking* 3.4 (2017), pp. 563–575. ISSN: 2332-7731. DOI: 10.1109/TCCN.2017.2758370.

[16] Alex Graves et al. "Hybrid computing using a neural network with dynamic external memory". In: *Nature* 538.7626 (Oct. 2016), pp. 471–476. ISSN: 00280836. URL: http://dx.doi.org/10.1038/nature20101.

[17] D.H. Lehmer. "Teaching combinatorial tricks to a computer". In: 10 (1960), pp. 179–193.

[18] J. A. Reeds M. H. Wright Lagarias J. C. and P. E. Wright. "Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions." In: *SIAM Journal of Optimization* (1998).

[19] D. Kleitman and G. Markowsky. "On Dedekind's Problem: The Number of Isotone Boolean Functions. II". In: *Transactions of the American Mathematical Society* 213 (1975), pp. 373–390. ISSN: 00029947. URL: http://www.jstor.org/stable/1998052.

Department of Engineering Science

Supplementary Questions for 4th Year Project Students

UNIVERSITY OF OXFORD

Factor	Answer	Things to Consider	Record details here
Has the checklist covered all the problems that may arise from working with the VDU?	☑ Yes ☐ No	Keyboard + Ext. mouse used. Stand used.	
Are you free from experiencing any fatigue, stress, discomfort or other symptoms which you attribute to working with the VDU or work environment?	☑ Yes ☐ No	Any aches, pains or sensory loss (tingling or pins and needles) in your neck, back shoulders or upper limbs. Do you experience restricted joint movement, impaired finger movements, grip or other disability, temporary or permanently	
Do you take adequate breaks when working at the VDU?	☑ Yes ☐ No	Periods of two minutes looking away from the screen taken every 20 minutes and longer periods every 2 hours Natural breaks for taking a drink and moving around the office answering the phone etc.	
How many hours per day do you spend working with this computer?	☐ 1-2 ☐ 3-4 ☐ 5-7 ☑ 8 or more		
How many days per week do you spend working with this computer?	☐ 1-2 ☐ 3-5 ☑ 6-7		
Please describe your typical computer usage pattern	Extended periods of programming with breaks taken regularly. Some simulations run which require me to not use computer for ~3 hours common. Dark themes used to reduce Eye Strain.		

Student Declaration and Academic Approval

Student Declaration:

I have completed the DSE Workstation Checklist and the Supplementary Questions for my computer-related risk assessment for 4YP Project Number indicated below:

4YP Project Number: 11685

4YP Student's Name *(please print)* WILLIAM MATTHEWS

4YP Student's Signature: WMM

Academic Approval

I confirm my approval of this 4YP DSE Risk Assessment.

Academic Supervisor's Name: *(please print)* JUSTIN P. COON

Academic Supervisor's Signature: Justin P Coon