University of The Witwatersrand

Masters Dissertation

Investigation of the use of Infinite impulse response filters to construct linear block codes

Author: Aneesh Chandran *Supervisor: Jaco Versfeld*

A dissertation submitted in fulfilment of the requirements
for the degree of Masters in Science

in the

Information Engineering
School of Electrical and Information Engineering

August 2016

The financial assistance of the Center for Telecommunications Access and Services (CeTAS) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the CeTAS.

Declaration of Authorship

I, Aneesh Chandran, declare that this dissertation titled, 'Investigation of the use of Infinite impulse response filters to construct linear block codes' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this dissertation has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this dissertation is entirely my own work.
- I have acknowledged all main sources of help.
- Where the dissertation is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

UNIVERSITY OF THE WITWATERSRAND

Abstract

Engineering and the Built Environment
School of Electrical and Information Engineering

Master of Science

Investigation of the use of Infinite impulse response filters to construct linear block codes

by Aneesh Chandran

The work presented extends and contributes to research in error-control coding and information theory. The work focuses on the construction of block codes using an IIR filter structure. Although previous works in this area uses FIR filter structures for error-detection, it was inherently used in conjunction with other error-control codes, there has not been an investigation into using IIR filter structures to create codewords, let alone to justify its validity. In the research presented, linear block codes are created using IIR filters, and the error-correcting capabilities are investigated. The construction of short codes that achieve the Griesmer bound are shown. The potential to construct long codes are discussed and how the construction is constrained due to high computational complexity is shown. The G-matrices for these codes are also obtained from a computer search, which is shown to not have a Quasi-Cyclic structure, and these codewords have been tested to show that they are not cyclic. Further analysis has shown that IIR filter structures implements truncated cyclic codes, which are shown to be implementable using an FIR filter. The research also shows that the codewords created from IIR filter structures are valid by decoding using an existing iterative soft-decision decoder. This represents a unique and valuable contribution to the field of error-control coding and information theory.

Acknowledgements

I would like to acknowledge my supervisor Dr. D.J.J Versveld and thank him for being a fantastic role model and a source of inspiration for myself and countless others. I would also like to thank him for being a superb mentor and for his encouragement as well as his words of wisdom.

Funding and support from the following organisations is gratefully acknowledged:

- The National Research Foundation (NRF) for supporting me to do this research at Wits.
- The CeTAS group for providing assistance and resources to do telecommunications based research at wits.
- CrunchYard for providing the machines on which the research simulations were conducted.

I would like to express my gratitude towards Dr. Rainier Dreyer for organizing the machines from CrunchYard. I would also like to thank Professor Ken Nixon for helping me setup the machines for simulation ready status, otherwise it would have been impossible to finish this work.

Thanks to my telecoms research group for providing support and constructive criticism towards my research.

Finally, it is important to acknowledge my parents and friends who gave me their undying support and encouragement which aided me in completing my research.

Contents

Declaration of Authorship i

Acknowledgements iii

Contents iv

List of Figures vi

List of Tables vii

Abbreviations viii

1 Introduction 1
- 1.1 Brief History of Error-Control Coding . . . 1
- 1.2 Problem and Research Statement . . . 2
- 1.3 Research Claims and Objectives . . . 3
- 1.4 Author's Contribution . . . 4
- 1.5 Conclusion . . . 5
- 1.6 Structure of Dissertation . . . 6

2 Literature Survey 7
- 2.1 Introductory Algebraic Theory . . . 7
- 2.2 Linear Block Codes . . . 9
  - 2.2.1 Cyclic Codes . . . 12
  - 2.2.2 Reed-Solomon Codes . . . 14
  - 2.2.3 Quasi-Cyclic Codes . . . 15
- 2.3 Filter Structures . . . 16
  - 2.3.1 FIR filter . . . 16
  - 2.3.2 IIR filter . . . 17
- 2.4 Convolutional Codes . . . 21
  - 2.4.1 Feed-forward Convolutional Codes . . . 22
  - 2.4.2 Feedback Convolutional Codes . . . 23
  - 2.4.3 Distance Bounds . . . 24
  - 2.4.4 Other forms of Non-binary Convolutional Codes . . . 24
- 2.5 Additive White Gaussian Noise Channel . . . 25
- 2.6 Binary Phase Shift Keying Modulation Scheme . . . 25

2.7 Decoding . . . 27
2.7.1 Parity-Check Transformation Algorithm . . . 27
2.8 Related Work . . . 30

3 Methodology **34**
3.1 Introduction . . . 34
3.2 Overview of Research Methodology . . . 34
3.3 Code Construction . . . 35
3.3.1 Generate Message book . . . 35
3.3.2 Filter Implementation . . . 36
3.3.3 Generate Coefficients . . . 37
3.3.4 Generate Code book . . . 37
3.4 Extracting the G-matrix . . . 38
3.5 Test for Cyclicity . . . 38
3.6 Distance Measurement . . . 38
3.7 Decoding . . . 39
3.7.1 Encoder . . . 40
3.7.2 Modulation and Noise models . . . 40
3.7.3 Decoder . . . 40
3.7.4 Monte Carlo Simulation . . . 41

4 Results **43**
4.1 Results of the construction . . . 43
4.1.1 (7,3) codes . . . 44
4.1.1.1 MDS . . . 44
4.1.1.2 Non-MDS . . . 45
4.1.2 (8,3) codes . . . 46
4.1.2.1 MDS . . . 46
4.1.2.2 Non-MDS . . . 46
4.1.3 (8,4) codes . . . 47
4.1.3.1 MDS . . . 47
4.1.3.2 Non-MDS (8,4) . . . 47
4.2 Analysis of the filter structure . . . 48
4.3 Achieved Distance Bounds . . . 50
4.4 Decoding . . . 51
4.4.1 Performance of PTA on codes . . . 52

5 Conclusion **54**
5.1 Construction of codes . . . 54
5.2 Decoding of Codes . . . 55
5.3 Analysis of codes . . . 55
5.4 Future Recommendations . . . 55

Bibliography **56**

List of Figures

1.1 Overview of how claims, sub-questions and evidences contribute to answering the research problem 5

2.1 Minimum distance d_{min} vs Block Length n for Singleton and Griesmer bounds for various k values 12
2.2 Encoding circuit for (n,k) cyclic code with a generator polynomial of $g(x) = 1 + g_1X^2 + \cdots + g_{n-k-1}X^{n-k-1} + X^{n-k}$ [24] 14
2.3 Direct Form I structure for an FIR filter 17
2.4 Direct Form I structure for an IIR filter 17
2.5 Structure of Direct Form II IIR filter structure 18
2.6 Rate 1/2 Feed-forward Convolutional Encoder 22
2.7 Rate 1/2 Feedback Convolutional Encoder 24
2.8 AWGN channel model 25

3.1 Overview of Research Methodology 35
3.2 Code Construction Processes 35
3.3 Sytem Layout 40

4.1 Performance of PTA on $(7, 3)$ Reed-Solomon, IIR MDS and IIR Non-MDS codes. 51
4.2 Performance of PTA on MDS codes for different rates 52
4.3 Performance of PTA on Non-MDS codes for different rates 52

List of Tables

2.1 Various Reperesentations of elements in GF(8) 9

3.1 Encoders used for various Block codes based on Distance Criteria 40

4.1 Distance Profile of codes with respect to the rate of the code and number of Memory Blocks 51

Abbreviations

ECC	**E**rror **C**orrecting **C**odes
BCH	**B**ose **C**haudhuri **H**ocquenghem
LDPC	**L**ow **D**ensity **P**arity **C**heck
MPI	**M**essage **P**assing **I**nterface
GF	**G**alois **F**ield
MDS	**M**aximum **D**istance **S**eperable
LSFR	**L**inear **S**hift **F**eedback **R**egister
RS	**R**eed **S**olomon
QS	**Q**uasi-**C**yclic
FIR	**F**inite **I**mpulse **R**esponse
IIR	**I**nfinite **I**mpulse **R**esponse
TCM	**T**rellis **C**oded **M**odulation
AWGN	**A**dditive **W**hite **G**aussian **N**oise
BPSK	**B**inary **P**hase **S**hift **K**eyeing
BER	**B**it **E**rror **R**ate
APP	**A** **P**riori **P**robability
HDD	**H**ard **D**ecision **D**ecoder
SSD	**S**oft **D**ecision **D**ecoder
KV	**K**oetter **V**ardy
GS	**G**uruswami **S**udan
PTA	**P**arity **T**ranformation **A**lgorithm
FIRPCC	**F**inite **I**mpulse **R**esponse **P**arity **C**heck **C**odes
SNR	**S**ignal to **N**oise **R**atio
SER	**S**ymbol **E**rror **R**atio

Chapter 1

Introduction

1.1 Brief History of Error-Control Coding

The significance of communication theory was shown by Shannon to the world in his famous paper [1], titled "A Mathematical Theory of Information", since then he has been widely regarded as the father of communication theory. Through probability theory, he showed that there exists a coding scheme with a low probability of error only if the information rate is less than the channel capacity. This was illustrated using an error-correcting code (ECC) invented by Hamming, since Hamming was the first person to introduce the basics of coding theory. In Hamming's paper [2], he describes a single ECC and introduces the basics, Hamming weight and Hamming distance which are now extensively used to describe modern ECCs. At the same time, another person who was working on error-control codes was Golay, which in his paper [3] generalised for coding theory what Shannon had concluded in his paper [1].

Research led to the discovery of Cyclic codes which had good algebraic properties. Some of the best cyclic error correcting codes such as Hamming, Golay, Reed-Solomon [4] and Bose-Chaudhuri-Hocquenghem (BCH) [5] were discovered. These are all subclasses of linear block codes. The beauty of such codes is that efficient decoders can be developed by exploiting their algebraic structure.

Hard-decision decoders exist for many of these codes, for cyclic codes, most of the decoders are developed based on syndrome decoding, which is derived from the algebraic structure of the codes. In 1969 Berlekamp and Massey came up with a decoding algorithm for both Reed-Solomon codes and Bose-Chaudhuri-Hocquenghem codes [6, 7]. With a recent emergence of Soft-decision decoding for linear block codes, it has shown to improve coding gain over hard decision decoding [8]. For Reed-Solomon codes, a soft-descion decoder was

created by combining soft information with a list decoder known as Koetter-Vardy and Guruswami-Sudan algorithm and has shown to improve the coding gain when compared to hard decision decoding [9, 10]. The drawback to this is that hard decision decoders are less complex and iterative in nature versus the complexity of soft-decision decoders.

Gallager founded Low-Density Parity Check (LDPC) codes in 1963 [11], but was disregarded due to high computational complexity which was not available then. This was the case until 1997 when Mackay and Neal [12] showed it again that by using the soft information to decode these codes, it gave a near Shannon limit performance. The encoding is based on density evolution and tanner graphs whilst the decoding is based on belief propagation, which is different from the approach of cyclic codes. This also reinforces that soft-decision is stronger than hard decision decoding.

Convolutional codes were introduced in 1955 by Elias [13] and was proposed as a good alternative to block codes. Convolutional codes are used almost exclusively as stream codes instead of block codes. Initially, Convolutional codes lacked a strong algebraic explanation and it was Fornay [14] who eventually derived a form of algebraic structure for analysis. Convolutional codes can be decoded using both a hard decision and soft-decision approaches, namely Threshold [15] and Viterbi [16] decoding. The Viterbi algorithm works on the trellis diagram of the convolutional code, which works out the maximum likelihood path on the Trellis diagram [16].

In the 1990's Turbo codes were discovered [17] and had a near Shannon limit error-correcting capability. Turbo codes are a combination of Recursive Systematic Convolutional encoders with an interleaver when combining the output streams from these encoders. Turbo codes are iteratively decoded with soft information and probability techniques [17]. Turbo codes entail two fundamental concepts, code that is designed using random-like properties which are not structural (due to the pseudo-randomness introduced by the interleaver) and decoding which heavily relies on soft-information based decoding.

This concludes the section where a brief history of notable error-control codes is introduced, which paints the context for this research. The following section will discuss the problem and research statement for this research.

1.2 Problem and Research Statement

The focus in Error-Control coding has always been the development of good codes which reached the Shannon limit. Error-Control codes are deemed "good" when it is capable of meeting a high minimum distance with large lengths and information symbols. These

requirements are hard to meet since an increase in parity will lead to a higher minimum distance and a lower code rate. Block codes are a form of error correcting codes and Reed-Solomon codes are some of the most widely used and studied linear block codes that are defined as "good" error-control codes. Another aspect that contributes to the preference of Reed-Solomon codes is their ease of construction and decoding. The drawback to Reed-Solomon codes is that they are designed in a particular field and constrained to a length and information symbols to achieve a good minimum distance. There have been approaches to extend the length of Reed-Solomon codes, but this is confined to a maximum of either a single or double symbols [8]. This is a very significant practical problem in the field of Error-Control coding. Reed-Solomon codes have been studied extensively in the past, which led to the use of using Linear Feedback Shift Registers in its implementation. The structure of a Linear Shift Feedback Register is analogous of that of a Finite Impulse Response filter. This similarity leads to the research problem which is stated as:

> "Investigate the use of Infinite Impulse Response filters to construct linear block codes that meet high minimum distance with flexible length and information symbols which are also easy to decode."

This research will attempt to investigate if it is possible to construct good block codes, which are not constrained by a length and number of information symbols by a field size, by utilising an IIR filter structure. The error correcting capabilities of the codes constructed from this scheme will be investigated. An existing decoder will be used to show the ease of which decoding is possible.

1.3 Research Claims and Objectives

The Research objectives are determined based on how they answer the claims hypothesised. The claims stated for this research are:

- valid codewords can be constructed through an IIR filter structure,
- the constructed codes are linear,
- these codewords are not cyclic codes or non-binary convolutional codes and has no quasi-cyclic properties,
- the Singleton bound isn't realistically achievable,
- these codes meet the Griesmer bound for block codes,

- these codes are decodable using an existing block code decoder.

The sub-questions that are formulated to warrant the claims made are:

- Can valid codewords be constructed using an IIR filter?
- Are these codes linear?
- Are these codes cyclic?
- Can they be defined as non-binary convolutional codes?
- Can Maximum distance separable codes be constructed?
- Do these codes reach the Griesmer bound?
- Are these codes decodable? If so can an existing decoder perform it?

This research attempts to answer these sub-questions and by answering them all evidences are obtained. These evidences are then used to warrant the claims made to support the research.

1.4 Author's Contribution

The author's contribution is split into two criteria namely, Knowledge contribution and Programming contribution. The Knowledge contribution discusses the valid contribution made to the field. The Programming contribution briefly mentions what setup is required for the experiments to be done. The programming contribution has no significance towards the research topic at hand. It is merely mentioned as a formality and thus concludes in this chapter.

Knowledge Contribution: Using IIR filters, it is possible to generate linear block codes that meet high minimum distance with flexible length and information symbols. These codes can be decoded using the Parity Transformation Algorithm, which is an existing decoder for Reed-Solomon codes. A further contribution was made when the algorithm was modified to work with non-MDS codes thus deeming it an iterative soft-decision decoder for any block codes.

Programming Contribution: The programming is done in Octave, where it is used to prototype and to run the simulations. In Octave, the communication package is used for generating the Galois field and defining the Filter functions. Octave is used because it is open source and is required to use Message Passing Interface (MPI) to split the computation across different machines. The packages used on Octave are [18–21]:

- MPI package
- Signals Package
- Control Package
- Communications Package

1.5 Conclusion

A brief history of error-control codes is introduced to cover the context, from which the problem and research statement are derived in this research. In conclusion, the dissertation attempts to answer the research problem, "Investigate the use of Infinite Impulse response filters to construct linear block codes that meet high minimum distance with flexible length and information symbols which are easy to decode." To achieve this, the research needs to warrant a set of claims. The general overview in which the claims and warrants are used to answer the research problem is illustrated in figure 1.1.

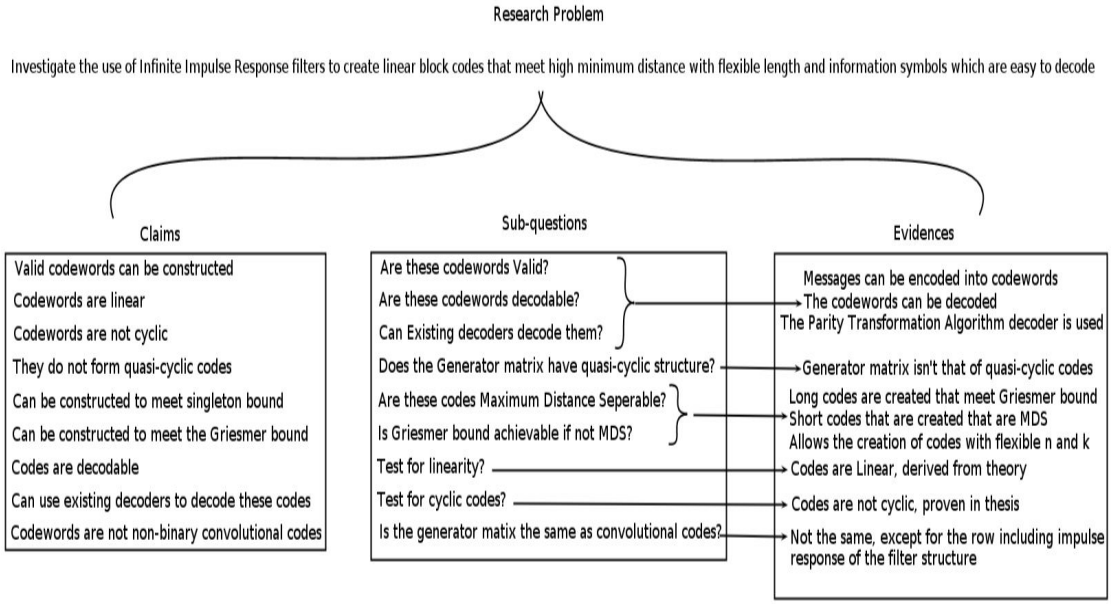

FIGURE 1.1: Overview of how claims, sub-questions and evidences contribute to answering the research problem

1.6 Structure of Dissertation

The dissertation is structured as follows: Chapter 2 contains the Literature Survey, which includes a discussion on basic algebra, block codes, convolutional codes, filter structures, decoding algorithm and related work. Chapter 3 contains the research methodology

and experimental setup adopted to tackle this research problem. Chapter 4 will lay out the results obtained as well as detailed analysis. Finally, Chapter 5 will conclude the research.

Chapter 2

Literature Survey

During the transmission of information over a channel, noise can corrupt some of the information transmitted. Error-Control coding is a practice for combating the receiving of errors while transmitting information. This approach allows the decoder to reconstruct the original information without any need of retransmission of information. Before introducing the different techniques of error-control coding some basic but necessary field theory is discussed. This chapter introduces the necessary theory required for block codes, convolutional codes, and related work.

In Section 2.1 the introductory algebraic theory is explored in depth with emphasis on basic concepts in the field of communication theory. Section 2.2 discusses block codes, in particular, significance is given to Reed-Solomon codes. A basic introduction to filters and different filter structures are provided in Section 2.3. Convolutional codes, their structure, and uses are reviewed in depth in Section 2.4. Section 2.5 explores the Additive White Gaussian Noise (AWGN) channel whilst Section 2.6 illustrates the Binary Phase Shift Keying (BPSK) modulation scheme and how soft information is extracted. Section 2.7 analyses the decoders used for the research. Finally, Section 2.8 scrutinizes work related to the research question and objective.

2.1 Introductory Algebraic Theory

In order to discuss the error-correcting codes some concepts regarding the algebra of finite fields, polynomial rings, and some basic error-control preliminaries need to be defined first.

Definition 2.1 A finite field $\mathbb{F}$ is defined as a set of finite elements in which two operations(addition and multiplication) exist. This is such that under the laws of addition

it creates an additive group and when multiplied without zero forms a multiplicative group [22]. Some of the properties the field must satisfy are:

1. The commutative and distributive laws must hold true i.e. $x, y, z \in \mathbb{F}$ then $xy + xz = x(y + z)$, $xz + zy = (x + y)z$, $x + y + z = z + y + a$ and $xyz = zyx$ must be true.
2. The associate property must hold true: where $x, y, z \in \mathbb{F}$ and the operations are $(x + y) + z = x + (y + z)$ and $(xy)z = x(yz)$.
3. The addition and multiplication operations must be closed i.e. $x, y \in \mathbb{F}$ such that $x + y \in \mathbb{F}$ and $xy \in \mathbb{F}$.
4. The field must have an additive and multiplicative identity such that given $x, 1, 0 \in \mathbb{F}$ then $x + 0 = x$ and $x \cdot 1 = x$.
5. Additive and Multiplicative inverses must exist for all elements in the field i.e. $x, 1, 0 \in \mathbb{F}$ has $x + (-x) = 0$ and $x \cdot x^{-1} = 1$. The exception to this is for element zero, where it does not have a multiplicative inverse.

Error-control coding is interested in a particular finite field known as the Galois Field.

Definition 2.2 A Galois Field (GF) is a Finite Field with finite number of elements which is derived from the powers of prime numbers and is denoted as $\mathbb{F}_{p^r}$.

A GF field is said to have an order of q where $q = p^r$, p is the prime number and r is the order of power of the prime number($r \in \mathbb{N}$ and $r \geq 1$). The order of the field also indicates the number of elements in the finite field. A GF field with an order q is also represented as $GF(q)$. There are specific Galois fields when $r = 1$ the field is called a prime field GF(p)(where $q = p^1$) [22].

Each element within the GF field is represented in terms of a polynomial with degree less than or equal to $r - 1$ and coefficients from $GF(p)$, the prime field. For a prime Galois field the addition and multiplication is done in terms of modular arithmetic i.e $a, b \in \mathbb{F}_p$ where $a + b = (a + b) mod(p)$ and $(a \cdot b) = (a \cdot b) mod(p)$ which insures the field is closed under addition and multiplication [22, 23].

Definition 2.3A Polynomial ring is the set of polynomials in $F_q[x]$ of degree less than some polynomial $f(x)$, where addition and multiplication of those set of polynomials will be modulo $f(x)$. This is very similar to a field but the difference is that it is in terms of polynomials with factors that lie in $GF(q)$. A ring of polynomials over $f(x)$ is denoted as $F_q[x]/f(x)$ and the number of elements contained is $q^{deg(f(x))}$. Suppose

if $a(x), b(x) \in F_q[x]$ and the $deg(a(x)) < deg(f(x)), deg(b(x)) < deg(f(x))$ then the polynomial $f(x)$ is deemed reducible over the ring $F_q[x]$ [22, 24].

Definition 2.4 An irreducible polynomial $f(x)$ in a field $\mathbb{F}$ is a polynomial which does not contain factors that are present in that field $\mathbb{F}$ [24].

Definition 2.5 A primitive polynomial $p(x)$ of some degree r is the minimal polynomial of some primitive element of $GF(p^r)$. It has the following properties [22, 24]:

1. A primitive polynomial always has a non-zero constant term.
2. An irreducible polynomial $f(x)$ over $GF(p)$ with degree r can be a primitive polynomial $p(x)$ if and only if it divides $x^n - 1$ where n is the smallest possible n in which it divides and $n = p^r - 1$.

A GF(8) field can be constructed using the irreducible polynomial $f(x) = X^3 + X + 1$ where the elements are expressed in terms of powers of α and the polynomials expressed in terms of indeterminate x [24].

TABLE 2.1: Various Reperesentations of elements in GF(8)

Powers of α	Polynomials in x	Binary Representation	Decimal Representation
0	0	000	0
$\alpha^0 = 1$	1	001	1
α^1	x	010	2
α^2	x^2	100	4
α^3	$x + 1$	011	3
α^4	$x^2 + x$	110	6
α^5	$x^2 + x + 1$	111	7
α^6	$x^2 + 1$	101	5

This concludes the introductory algebraic theory necessary to understand block codes and convolutional codes. The following section will introduce block codes, namely Reed-Solomon codes and will cover its construction and theory in detail.

2.2 Linear Block Codes

This section focuses on linear block codes which falls under a category of error-control coding. A block code is defined in terms of (n, k) where n is the length of the codeword, k is the number of information bits, and $n - k$ the number of parity bits. Intuitively it

can be stated that block codes are constrained to a particular length [22–24]. The rate of a block code is calculated as

$$R = \frac{k}{n}. \tag{2.1}$$

Definition 2.6 A Generator matrix with dimension $k \times n$ exists for linear block codes which is expressed as

$$\begin{bmatrix} g_{1,1} & g_{1,2} & \cdots & g_{1,n} \\ g_{2,1} & g_{2,2} & \cdots & g_{2,n} \\ \vdots & \vdots & \cdots & \vdots \\ g_{k,1} & g_{k,2} & \cdots & g_{k,n} \end{bmatrix}. \tag{2.2}$$

This is also known as the encoder and the codewords (c) are formed by multiplying the messages (m) with the encoder (G), $c = mG$.

Definition 2.7 A Parity Check matrix H also exists, which is commonly used as an alternate description to a linear block code. The parity check matrix $(n-k) \times n$ is defined as

$$\begin{bmatrix} h_{1,1} & h_{1,2} & \cdots & h_{1,n} \\ h_{2,1} & h_{2,2} & \cdots & h_{2,n} \\ \vdots & \vdots & \cdots & \vdots \\ h_{n-k,1} & h_{n-k,2} & \cdots & h_{n-k,n} \end{bmatrix}. \tag{2.3}$$

A codeword c is only a valid codeword if it satisfies the condition that $cH^t = 0$.

Definition 2.8 Hamming weight for a vector c is the maximum number of non-zero elements contained within the vector [22–24].

Definition 2.9 The Hamming distance between two vectors c and v is the number of positions in which both vectors are different [22–24].

Definition 2.10 The Minimum distance is the minimum Hamming distance between any two codewords of a codebook C, it is denoted as d_{min} [22–24].

One of the most important characteristics of an error correcting code is the minimum distance of the code, as this directly relates to the error correcting capability of the code. The minimum distance of a code is directly proportional to the minimum number of errors that the code can detect. In theory a code with a minimum distance d_{min} can detect upto $d_{min} - 1$ errors [24].

Definition 2.11 The Singleton Bound is the measurement of minimum distance obtainable by a maximum distance seperable(MDS) linear block code [22–24]. For a block code (n, k), and codeword $c \in C$ over $GF(q)$ with a d_{min}, the Singleton bound implies that

[25],

$$q^k \leq q^{n-d_{min}+1} \tag{2.4}$$

such that

$$k \leq n - d_{min} + 1 \tag{2.5}$$

which can be transposed and simplified with d_{min} as the subject of the formula

$$d_{min} \leq n - k + 1. \tag{2.6}$$

Going from equations 2.4 to 2.5 it can be noted that the field size is completely ignored. It is true that only MDS codes can achieve the Singleton bound, but it also does not indicate how large the field size q should be for those codes to exist, therefore it can be concluded that the Singleton bound does not take into account the field size and it is one of the major drawbacks [25]. From this the maximum number of errors a block code will correct is determined as

$$e = \frac{d_{min} - 1}{2}. \tag{2.7}$$

Definition 2.12 The Griesmer Bound is the measurement of minimum distance obtainable by a linear non-MDS codes. For example, a block code (n, k) with a Hamming distance of d the Griesmer bound is defined as

$$n \geq \sum_{i=0}^{k-1} \left\lceil \frac{d}{q^i} \right\rceil. \tag{2.8}$$

The Singleton bound is always the maximum achievable distance for a block code. But the draw back to the Singleton bound is that it does not take into account the field size. Therefore in higher fields and larger lengths n, for a block code, the Singleton bound is considered to be unrealistic to achieve in nature. Therefore the Griesmer bound is used as an alternative distance bound that is achievable for linear block codes. For the derivations and comparisons refer to reference [25].

The comparison between the Singleton and Griesmer bound is depicted in Figure 2.1 for a block code in $GF(4)$ where the information symbols are $k = 3$ and $k = 4$ for varying block lengths n. From Figure 2.1 it can be seen that for an increased k symbol the margin of difference between the bounds are still the same, thus it is independent of the relation of k. This indicates that if n is sufficiently larger than k then it holds that the Griesmer bound will always be realistically achievable.

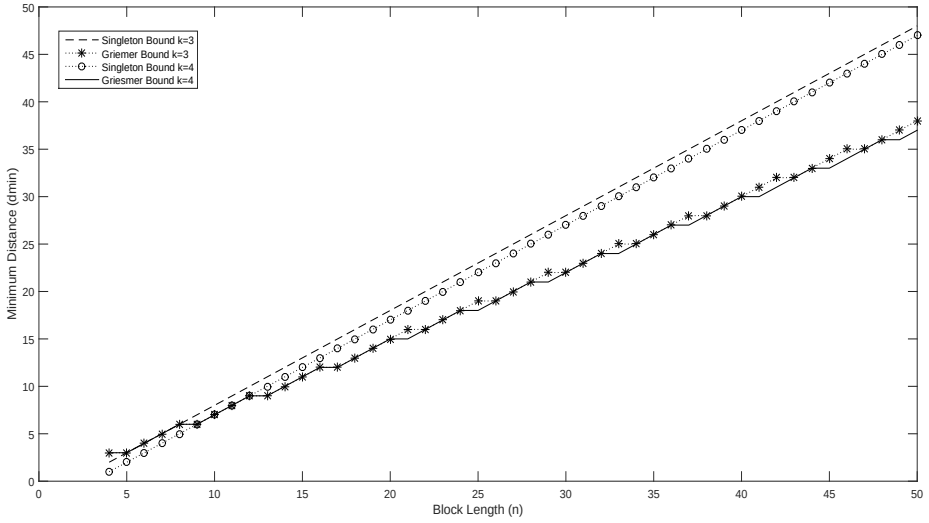

FIGURE 2.1: Minimum distance d_{min} vs Block Length n for Singleton and Griesmer bounds for various k values

2.2.1 Cyclic Codes

Cyclic codes forms part of a subclass of codes within linear block codes. They are studied due to their relative ease for implementation using Linear Shift Feedback Registers (LSFR) and because of strong algebraic properties which help in encoding and decoding [22–24]. This section will briefly introduce their algebraic definition. Suppose a codeword vector c exists containing n elements

$$(c_0, c_1, \cdots c_{n-1}), \tag{2.9}$$

if another vector c^1 contains

$$(c_{n-1}, c_0, \cdots c_{n-2}), \tag{2.10}$$

and if both c and c^1 are $\in \mathbb{C}$ then the codeword vector c^1 is considered a cyclic shift of c. The equivalent polynomial representation of the codeword c (2.9) is

$$c(x) = c_0 + c_1 x^1 + \cdots + c_{n-1} x^{n-1}. \tag{2.11}$$

This representation will always confine a codeword polynomial to a maximum degree less than or equal to $n-1$. To indicate the cyclic shift in (2.10) the polynomial representation is

$$xc(x) mod(x^n - 1) = c_{n-1} + c_0 x^1 + \cdots + c_{n-2} x^{n-1}. \tag{2.12}$$

From this it can be inferred that a cyclic shift is the multiplication of $c(x)$ and x in the polynomial ring $F_q[x]/x^n - 1$ [22–24].

Theorem 2.1 The generator polynomial for a cyclic code is represented as

$$g(x) = g_0 + g_1 x^1 + \cdots + g_l x^l \tag{2.13}$$

where g_0 will always be 1 assuming $GF(2)$.

Proof. By assuming that $g_0 = 0$, then

$$g(x) = g_1 x^1 + \cdots + g_l x^l = x(g_1 + g_2 x^1 + \cdots + g_l x^{l-1}). \tag{2.14}$$

By referring to equation (2.12), it is seen that (2.14) is a cyclic shift of some generator polynomial, $xg(x)$, and thus it can be concluded that $g_0 \neq 0$ [24]. □

Theorem 2.2 Provided that a generator polynomial in the form of (2.13) exists, there exists another polynomial $b(x)$ that has a degree of $n-1$ or less and is a multiple of (2.13).

Proof. Assume that there exists a codeword polynomial $v(x)$ over $GF(2)$ with a degree of $n-1$ or less. If that $v(x)$ is presumably a multiple of $g(x)$,

$$v(x) = (q_0 + q_1 x^1 + \cdots + q_{n-1-l} x^{n-1-l}) g(x) \tag{2.15}$$

Now it can be shown that (2.15) is an addition of various multiples of $g(x)$ and all of it is contained within the Codebook C. This shows that any polynomial with a degree $n-1$ or less is a subset in C. Through division of $v(x)$ by $g(x)$ the whole set can be expressed in terms of quotients and remainders as

$$v(x) = q(x)g(x) + r(x). \tag{2.16}$$

From (2.15) we can conclude that $v(x) = q(x)g(x)$ therefore $r(x) = 0$, thus proving that $v(x)$ is completely divisible by $g(x)$ due to being multiples [24]. □

Cyclic codes are encoded using the Generator matrix (G-matrix) described earlier in (2.2). That can also be achieved by implementing a circuit that is a linear $n-k$ shift register with feedback, also known as a Linear Shift Feedback Register (LSFR). This circuit is shown in Figure 2.2 and a three step process is involved to get the equivalent encoding of a G-matrix. So, what the encoding circuit achieves mathematically is that once a message polynomial $u(x)$ is multiplied by X^{n-k}, it is then divided by a generator polynomial $g(x)$ to obtain a remainder polynomial $b(x)$. This is combined to form the

codeword polynomial $b(x) + X^{n-k}u(x)$. The circuit achieves this in the following three step process:

1. Activate the gate and feed in the input sequences simultaneously into the registers. By shifting all the input sequences from the front is equivalent to multiplying $u(x)$ by X^{n-k}. As soon as all the messages have been fed the remaining $n-k$ digits form the remainder and are the parity-check digits.
2. Disconnect the feedback by switching off the Gate.
3. With the aid of a switch combine the parity-check digits $(b_0, b_1, \cdots, b_{n-k-1})$, with the message sequence to create the codeword sequence.

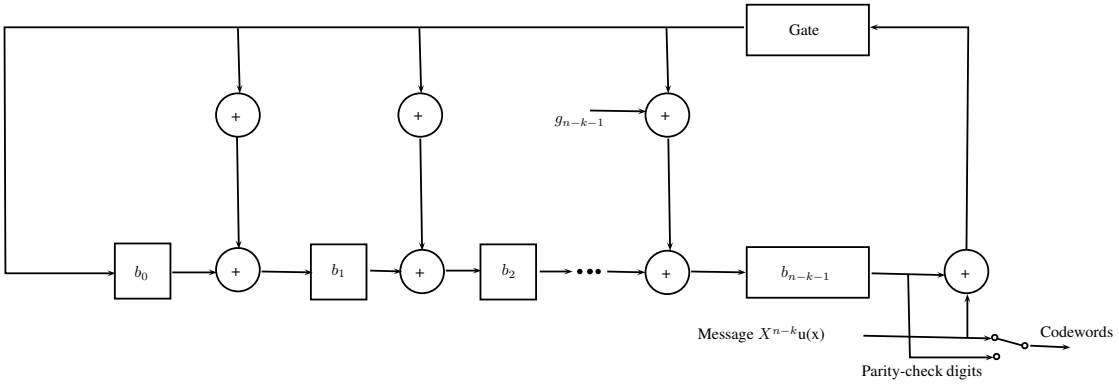

FIGURE 2.2: Encoding circuit for (n,k) cyclic code with a generator polynomial of $g(x) = 1 + g_1X^2 + \cdots + g_{n-k-1}X^{n-k-1} + X^{n-k}$ [24]

2.2.2 Reed-Solomon Codes

Reed-Solomon (RS) codes are a subset of cyclic codes and the most widely used linear block code. RS codes meet the Singleton bound under the following conditions

1. $n \leq q - 1$,
2. the parity bits must be $n - k$

From these conditions it is easy to derive that the most number of errors a code will correct is

$$e = \frac{n-k}{2}. \tag{2.17}$$

By combining (2.17) with equations (2.6) and (2.7), the Singleton bound can be expressed in terms of the maximum number of errors to correct as

$$d_{min} = 2e + 1. \tag{2.18}$$

It is more logical to design a code by determining the number of errors it requires to correct and thus leading to a logical approach for creating the appropriate RS code. The generator polynomial for a RS code is given as

$$g(x) = (x - \alpha)(x - \alpha^2)...(x - \alpha^{n-k}). \tag{2.19}$$

A generator matrix G with dimensions $k \times n$, which contains linearly independent codewords exist for RS codes. Following that the parity-check matrix also exists from the G matrix with the dimensions $(n - k) \times n$. Due to the existence of linearly independent rows in the G matrix it can be said that all codewords formed are a combination of those rows. The total number of codewords from such combinations are calculated as q^k. The systematic form of the generator matrix is shown as

$$G = \begin{bmatrix} I_k & P \end{bmatrix}, \tag{2.20}$$

where I_k is an $(k \times k)$ identity matrix and P is the $(k \times (n - k))$ parity matrix [22, 24]. This property holds true for any linear block code (n, k). Following this approach the parity check matrix H is constructed as

$$H = \begin{bmatrix} P' & I_{n-k} \end{bmatrix}, \tag{2.21}$$

where P' stands for the transposed $(n - k) \times k$ parity matrix and the $(n - k) \times (n - k)$ identity matrix I_{n-k} [22, 24]. Another approach based on the Vandermonde matrix also exists to generate the H matrix. Assuming β is a primitive element of $GF(q)$ for a RS code with (n, k, d_{min}) the H matrix is constructed as

$$H = \begin{bmatrix} 1 & \beta & \cdots & \beta^{n-1} \\ 1 & \beta^2 & \cdots & \beta^{2(n-1)} \\ \vdots & \vdots & \cdots & \vdots \\ 1 & \beta^{(d_{min}-1)} & \cdots & \beta^{(d_{min}-1)(n-1)} \end{bmatrix}. \tag{2.22}$$

2.2.3 Quasi-Cyclic Codes

This section introduces Quasi-Cyclic (QC) codes, a formal definition and an example is discussed in this section. The discussion is kept brief due to the research not being focused on QC codes, but an understanding is necessary to draw conclusions in later chapters.

Definition 2.11 Let a block code be defined as (sn, ak), where s, n, a and k are all positive real numbers with the condition that $ak < sn$. Assuming this codebook C is in $GF(q)$, the rules required to be a QC code are:

1. Every codeword has s sections of n bits.
2. Cyclic shift of those s sections of a codeword leads to codewords within C

An interesting insight is that if $s = 1$ and $a = 1$ we have a block code (n, k) which is a normal cyclic code. Therefore it can be concluded that cyclic codes are a subclass of QC codes. For example, a QC code in $GF(2)$, where $1 \geq a < s$, the generator matrix of that code consists of $a \times s$ sets of $n \times n$ circulant matrices. A circulant matrix is where each row of the matrix is a right cyclic shift of the previous row, the next column will be a downward shift of the previous column. The systematic G matrix of QC code (sn, ak) is given as

$$G = \begin{bmatrix} G_{0,0} & G_{0,1} & \dots & G_{0,s-a-1} & I & O & \dots & O \\ G_{1,0} & G_{1,1} & \dots & G_{1,s-a-1} & O & I & \dots & O \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ G_{a-1,0} & G_{a-1,1} & \dots & G_{a-1,s-a-1} & O & O & \dots & I \end{bmatrix}, \tag{2.23}$$

where $G_{w,m}$ is a circulant $n \times n$ matrix with $0 \geq w < a$ and $0 \geq m < s - a$. Where there is an I it represents an identity matrix and O is a zero matrix. Note that all of these matrices are circulant with dimensions $n \times n$ [26].

2.3 Filter Structures

Filter Structures are of great importance in Signal processing. The two most commonly implemented filters are namely, the Finite Impulse Response (FIR) and the Infinite Impulse Response (IIR) filters. The significance of introducing filter structures from signal processing is to highlight its use in coding theory. Convolutional codes are codes which have implementation of such filter structures and are introduced in the following section. The subsections below introduce the structural and mathematical definitions of these filters.

2.3.1 FIR filter

A Finite Impulse Response filter produces an impulse response which will terminate after a certain period of time T_p, this is achieved because it does not have feedback. From this it can be inferred that the transfer function is dependent only on the delay or memory

operators as the signal traverses through time.The structure of an FIR filter is illustrated in Figure 2.3 [27, 28]

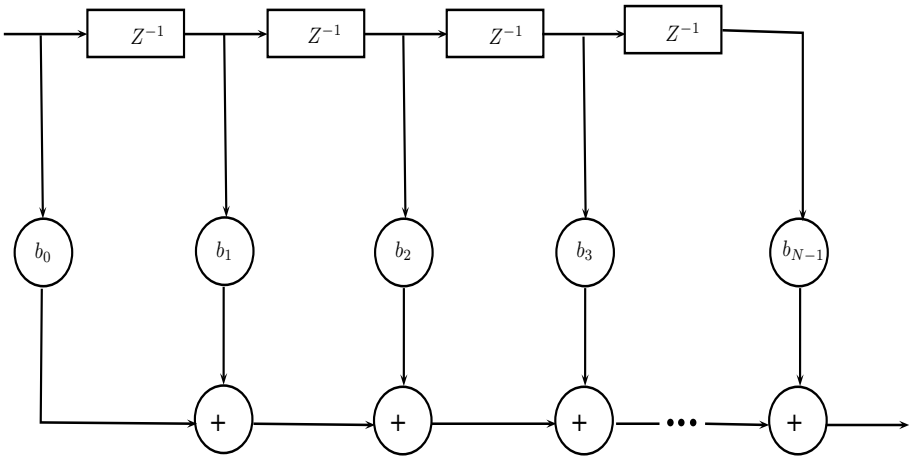

FIGURE 2.3: Direct Form I structure for an FIR filter

The transfer function is mathematically represented as

$$T(z) = b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_{N-1} z^{-(N-1)}, \tag{2.24}$$

where $N-1$ is the order of the filter and is represented in the z-domain which is where it is commonly analysed in.

2.3.2 IIR filter

Infinite Impulse Response (IIR) filters produces an impulse response which does not terminate after a period of time T_p, and it carries on infinitely. The structure of an IIR filter is provided in Figure 2.4 [27, 28]

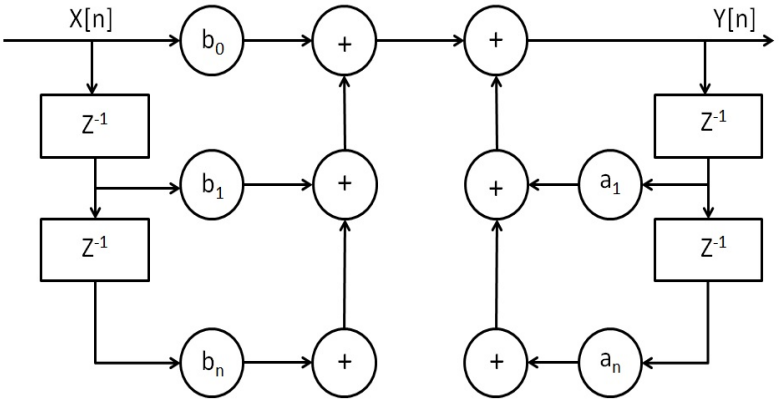

FIGURE 2.4: Direct Form I structure for an IIR filter

The traditional transfer function, where $a_0 = 1$ is given as

$$T(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_{N-1} z^{-(N-1)}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_{M-1} z^{-(M-1)}}, \tag{2.25}$$

where $N-1$ is the order of the feed-forward side and $M-1$ is the order of the feedback side. From this it can be seen that the output of the filter plays an important role in the form of feedback which influences the future of outputs. Equation 2.25 can also be expressed in terms of a rational function and power series as

$$t(z) = \frac{b(z)}{a(z)} = \frac{\sum_{i=0}^{N-1} b_i z^i}{\sum_{j=0}^{M-1} a_j z^j} = \frac{\sum_{i=0}^{N-1} b_i z^i}{1 + \sum_{j=1}^{M-1} a_j z^j}, \tag{2.26}$$

where t_j are coefficients of the power series in terms of z. The transfer function in equation 2.25 is derived from the following difference equation

$$y[n] = \frac{1}{a_0} \left(\sum_{i=0}^{N-1} b_i x[n-i] - \sum_{j=0}^{M-1} a_j y[n-j] \right), \tag{2.27}$$

where $y[n]$ is the discrete output signal, $x[n]$ the discrete input signal, $N-1$ the order of the feed-forward side, $M-1$ the order of the feedback side, b_i the coefficients of the feed-forward side, and a_j the coefficients of the feedback side.

Octave implements a direct form II transposed model for the filter function. The direct form transposed II depicted in Figure 2.5 looks different from Figure 2.4. The structure indicates that coefficients a_1 and a_2 are both negative, this is due to the difference equation shown in 2.27. This subtraction exists in the normal field, but in the Galois field subtraction and addition are the same process. The transfer function (2.25) for both the structures is the same and this is shown in [29, 30].

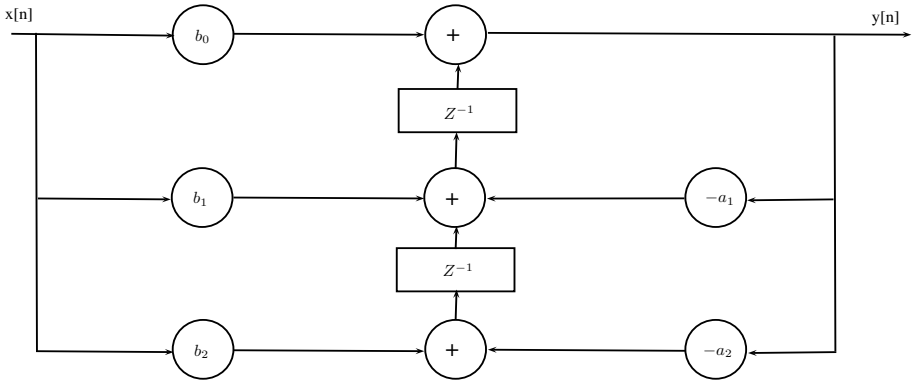

FIGURE 2.5: Structure of Direct Form II IIR filter structure

The filter structure in Figure 2.5 assumes $a_0 = 1$ and because 1 is a multiplicative identity, it does not need to be represented in a block in the diagram. To understand how

the direct form II filter structure is implemented, Algorithm 1 will illustrate it better. Algorithm 5 only shows the logic for the filter structure in Figure 2.5 and explains how the logic can be scaled for larger numbers of coefficients.

Input: 1. Row vector of filter coefficients for the numerator
Input: 2. Row vector of filter coefficients for the denominator
Input: 3. Row vector which contains the message
Output: Row vector which contains the codeword

4. Set the coefficients for $\begin{bmatrix} b_0 & b_1 & b_2 & \cdots & b_M \end{bmatrix}$, for the filter from row vector for numerator

5. Set the coefficients for $\begin{bmatrix} a_0 & a_1 & a_2 & \cdots & a_M \end{bmatrix}$, for the filter from row vector for numerator

6. InputAccumulator=0

7. OutputAccumulator=0

8. Summer1=0

9. Summer2=0

FOR (Index=0, Index < Length of the message, Index++)

The feedback side Logic

10. InputAccumulator = Message value at current Index
11. InputAccumulator = InputAccumulator - ($a_1\times$ Summer1)
12. InputAccumulator = InputAccumulator - ($a_2\times$ Summer2)

Repeat this logic to the number of summers and a_n

The feed forward side logic

13. OutputAccumulator = Message value at current index $\times b_0$
14. OutputAccumulator = OutputAccumulator + ($b_1\times$ Summer1)
15. OutputAccumulator = OutputAccumulator + ($b_2\times$ Summer2)

Repeat this logic to the number of summers and a_n

16. Summer2 = Summer1
17. Summer1 = InputAccumulator

18. Output value at current Index = OutputAccumulator

END

19. The codeword row vector will be the accumulation of the output values

Algorithm 1: Pseudo code for Direct form II IIR filter structure

2.4 Convolutional Codes

Convolutional codes and their necessary theory can be found in detail from reference [31]. The authors of these references have done well to introduce the theory of convolutional codes. This section contains a basic introduction to convolutional codes and their different types. The focus will be spent more on understanding the distance bounds for these codes. A detailed introduction to cyclic convolutional codes are also covered. The significance of this chapter is to draw insight from the understanding of convolutional codes to the filter structures.

Convolutional codes unlike block codes introduced earlier have a different structure, (n, k, K) the parameter n is the length of the code, k is the information bits and K is the constraint length. Convolutional codes are defined by their rate shown below as

$$R = \frac{n_d}{n_e}, \tag{2.28}$$

where n_d is the number of input streams and n_e is the number of output streams. This equation is valid if and only if it satisfies the condition $n_e \geq n_d$. Before the different structures are introduced the basic layout of a convolutional code needs to be introduced. Suppose u^1 is the input stream and contains values from $GF(2)$, which is a binary field. Then from some initial time onwards you would have some input starting at t_0 and the input stream can be presented as

$$u^1 = \left\{ u^1_{t_0}, u^1_{t_1}, ..., u^1_{t_f}, ... \right\} \tag{2.29}$$

From (2.29) we know that the first input stream started at some time t_0 and then went onto some time t_f, in practice the input stream is infinitely long. Depending on the rate of the convolutional code you will have different output streams, this is illustrated below for the output stream of a $\frac{1}{2}$ rate encoder as

$$v^1 = \left\{ v^1_{t_0}, v^1_{t_1}, ..., v^1_{t_f}, ... \right\}, \tag{2.30}$$

$$v^2 = \left\{ v^2_{t_0}, v^2_{t_1}, ..., v^2_{t_f}, ... \right\}, \tag{2.31}$$

where these streams are combined as:

$$v = \left\{ v^1_{t_0} v^2_{t_0}, v^1_{t_1} v^2_{t_1}, ..., v^1_{t_f} v^2_{t_f}, ... \right\}. \tag{2.32}$$

To express the convolutional code in terms of a polynomial, it must firstly be expressed in terms of an indeterminate D. In a polynomial ring $\mathbb{F}_q[D]$ there are infinite q polynomials, where q are the coefficients of the polynomials. So a (n_d, n_e) convolutional code C is an n_e-dimensional subspace of $\mathbb{F}_q(D)^{n_d}$ [32]. The encoding aspect of convolutional encoders are represented as generator matrices as done in block codes. For analysis purposes the input sequence is converted to a polynomial form $x(D)$, the polynomial transfer function matrix $G(D)$ (with dimensions $n_d \times n_e$) and finally the codeword $y(D)$. This is mathematically illustrated as

$$y(D) = u(D)G(D). \tag{2.33}$$

The indeterminate D is also known as a delay operator or memory, this is adapted from a signal processing theory. The total number of delay blocks present directly relates to the maximum power of the equations formed. There are many generator matrix structures for convolutional encoders and can be extensively studied from references [24, 31]. Convolutional codes in practice are used by passing information bits as streams to the encoder to obtain the output codeword sequence. There are fundamentally two different ways to encode convolutional codes namely, feed-forward and feedback approach, and these approaches are introduced in the next two sections.

2.4.1 Feed-forward Convolutional Codes

This section discusses the feed-forward convolutional encoders. The architecture of the feed-forward encoder is provided in Figure 2.6. The structure for a rate $\frac{1}{2}$ Feed-forward convolutional encoder is shown below

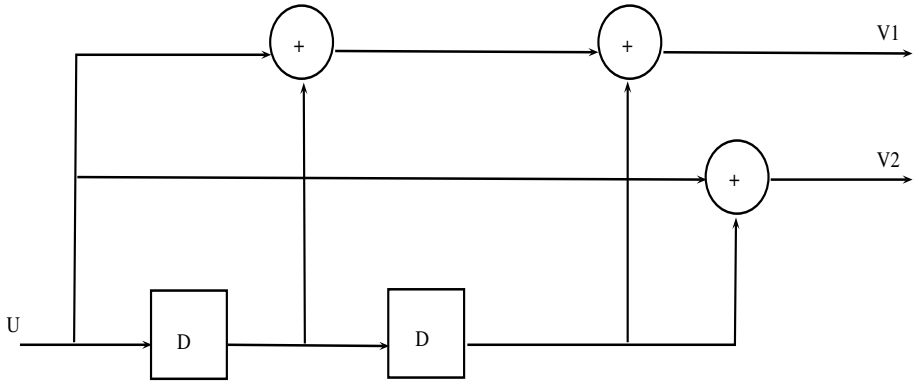

FIGURE 2.6: Rate 1/2 Feed-forward Convolutional Encoder

As shown above this is analogous to an FIR filter shown in Figure 2.4. The transfer function matrix for Figure 2.6 is

$$G(D) = \begin{bmatrix} 1 + D + D^2 & 1 + D^2 \end{bmatrix}. \tag{2.34}$$

Suppose the input is $u(D)$ then based on 2.34 the output for $V1$ is

$$V1(D) = u(D)(1 + D + D^2), \tag{2.35}$$

and for $V2$

$$V2(D) = u(D)(1 + D^2). \tag{2.36}$$

The G-matrix presented in equation 2.34 is obtained from the transform domain and the time domain version of the G-matrix is presented as

$$G = \begin{bmatrix} g_0^0 g_0^1 & g_1^0 g_1^1 & g_2^0 g_2^1 & \cdots & g_M^0 g_M^1 & & \\ & g_0^0 g_0^1 & g_1^0 g_1^1 & \cdots & g_{M-1}^0 g_{M-1}^1 & g_M^0 g_M^1 & \\ & & g_0^0 g_0^1 & \cdots & g_{M-2}^0 g_{M-2}^1 & g_{M-1}^0 g_{M-1}^1 & g_M^0 g_M^1 \\ & & \ddots & & & & \ddots \end{bmatrix} \tag{2.37}$$

where M is total number of memory blocks or delay operators and the blank areas are all zeros. This matrix is a semi-infinite matrix because it is dependent on the arbitrary length of the input stream u^1. If the input stream has a fixed length L then the number of rows present in the time domain G-matrix is also L.

2.4.2 Feedback Convolutional Codes

An example of a feedback convolutional encoder is provided in Figure 2.7. It is evident that the encoding structure varies from feed-forward when compared to Figure 2.6. The transfer function matrix for the this encoder is given as:

$$G(D) = \left[1 \quad \tfrac{1+D^2}{1+D+D^2}\right]. \tag{2.38}$$

They are widely known as recursive systematic convolutional codes, and mostly are encoded in a systematic format. Communication theory states that a feedback convolutional encoder can be used to implement feed-forward convolutional encoders. It is relevant to know that there are many different approaches for creating the encoder matrices and can be studied extensively from [24, 31]. The transfer function matrix is different from the one provided in (2.34). Assuming the input is assumed to be a rational function $u(D) = 1 + D + D^2$, then the outputs are obtained as

$$V^1(D) = u(D)(1) = 1 + D + D^2, \tag{2.39}$$

and

$$V^2(D) = u(D)(\frac{1 + D^2}{1 + D + D^2}) = 1 + D^2, \tag{2.40}$$

from which it has been shown to be the same as (2.35) and (2.36).

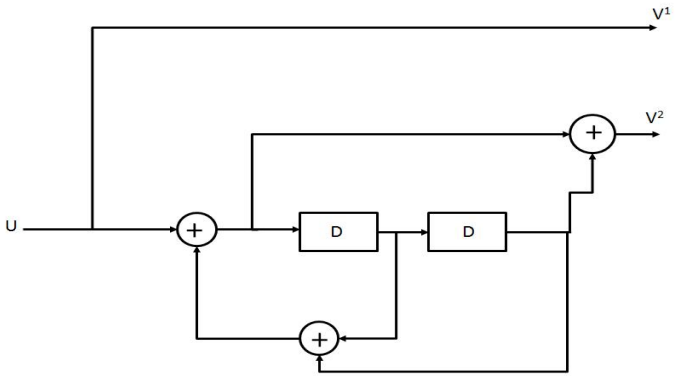

FIGURE 2.7: Rate 1/2 Feedback Convolutional Encoder

2.4.3 Distance Bounds

An important aspect of error-control coding is the ability to correct errors, and this is dependent on the d_{min} (minimum distance) of the codeword. Convolutional encoders from theory are regarded as linear block codes but are in practice used as stream codes. Various studies and explanations are proposed for distance bounds for convolutional codes [25] . The distance bounds can be separated into two categories, namely as a property of the encoder or of the codeword [31]. What is relevant for this research is the distance bounds associated with the codewords. The distance bounds that are most relevant are the Griesmer and Singleton bounds.

2.4.4 Other forms of Non-binary Convolutional Codes

The authors [33] discusses the different types of non-binary convolutional codes and their practical implementations. In order to decode Convolutional codes, its state diagram and trellis diagram is necessary as it leads to the Viterbi decoding algorithm [33]. From this extended a Trellis Coded Modulation (TCM) approach in which convolutional codes were combined with a modulation process [33]. There are two types, namely: Ring-TCM and Space-TCM codes. Further details will not be pursued since these codes serve no purpose to the research at hand.

2.5 Additive White Gaussian Noise Channel

This section covers the basic of the channel model of Additive White Gaussian Noise (AWGN), the channel adds White Gaussian noise to the transmitted signal.The mathematical description of this channel is

$$r(t) = T(t) + N(t) \tag{2.41}$$

where $r(t)$ is the received signal, $T(t)$ is the transmitted signal and $N(t)$ is the noise added to the signal. $N(t)$ is generally classified as white Gaussian noise with a spectral power density of $N_0/2$. White Gaussian Noise has a probability density function modelled as

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \tag{2.42}$$

where μ is a mean of zero and $\sigma^2 = \frac{N_0}{2}$ [34]. AWGN channel model is generally used to study deep space communication channels, and thus is deemed a very simple channel model of which is illustrated below [34]

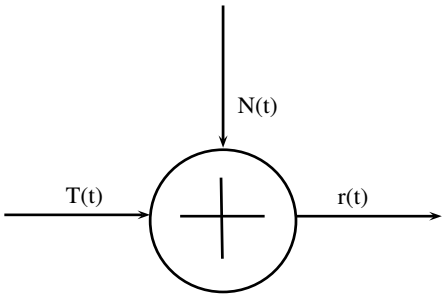

FIGURE 2.8: AWGN channel model

2.6 Binary Phase Shift Keying Modulation Scheme

The Binary Phase Shift Keying (BPSK) modulating scheme is briefly discussed in this section. The modulation scheme used for the transmission is BPSK and the bit error rate (BER), known as the probability of a bit error in demodulation (deemed as p in the equation) of the uncoded output from the AWGN channel is given as [34]

$$p = Q\left(\frac{2\epsilon_b}{N_0}\right) \tag{2.43}$$

whereby ϵ_b is the energy per bit. Assuming that a codeword $c \in C$ is transmitted over the AWGN channel with a two-sided spectral power density of $\frac{N_0}{2}$ using BPSK-modulation scheme with a signal energy of unity, a prior probability (APP) is calculated during

demodulation of the received sequence to obtain the soft-information from the channel [26]. A binary expansion of the codeword c is conducted and a new codeword sequence $c*$ is generated before transmission. Suppose the codeword c contains symbols which are of $GF(q)$, where $q = p^r$, then for every element $\alpha \in GF(q)$ the binary expansion is represented as an r-tuple $(\gamma_0, \gamma_1, \cdots, \gamma_{r-1})$ which is contained in $GF(2)$. For example a codeword vector of length N, $c = (c_0, c_1, c_2, \ldots, c_{N-1}) \in GF(q)$ will have the following binary expansion form

$$c* = ((c_{0,0}, c_{0,1}, \ldots, c_{0,r-1}), (c_{1,0}, c_{1,1}, \ldots, c_{1,r-1}), \ldots, (c_{N-1,0}, c_{N-1,1}, \ldots, c_{N-1,r-1})). \tag{2.44}$$

The BPSK modulation characterises the transmitted signal sequence as a bipolar sequence with +1 representing the binary bit 1 and -1 representing the binary bit 0. From this characterization the prior probabilities are calculated during the demodulation of received signal. The APP is also know as the likelihood ratio and is given as

$$l(x_i|y) \equiv \frac{Pr(x_i = 0|y)}{Pr(x_i = 1|y)}, \tag{2.45}$$

where x is the transmitted sequence ($x = [x_0, x_1, \ldots, x_{N-1}]$) and y is the received sequence ($y = [y_0, y_1, \ldots, y_{N-1}]$) both with a length N. This equations assumes that a specific bit from x equals 1 given y when there is no corruption due to noise. From this the formal equation for calculating the probability that the received $i-th$ bit being a 1, $y_{i,k}$, given the channel output $x_{i,k}$ is

$$p^1_{y_{i,k}} = \frac{1}{1 + e^{(\frac{4x_{i,k}}{N_0})}}; \tag{2.46}$$

where $0 \leq i < N$ and $0 \leq i < r$ ($q = p^r$). From this to calculate the probability of the received bit being 0 is simply

$$p^0_{y_{i,k}} = 1 - p^1_{y_{i,k}}. \tag{2.47}$$

After evaluating the probability of each received bit, it is now necessary to compute the probability of the symbol. So the probability of the $i-th$ symbol, α_i is

$$P_i^{\alpha_i} = \prod_{k=0}^{r-1} p^{\gamma_k}_{y_{i,k}}, \tag{2.48}$$

where it is the pi- product of the posterior probability of all the individual bits $(\gamma_0, \gamma_1, \cdots, \gamma_{r-1})$ that comprise the symbol $\alpha_i \in GF(q)$ [26].

2.7 Decoding

For Linear Block-codes such as Reed-Solomon codes there exists an efficient hard decision decoder (HDD) such as Berlekamp-Massey [6]. It is common knowledge that by performing HDD a penalty of 2-3dB coding gain is paid for in AWGN channel because HDD quantises real-valued information, thus ignores the information from the channel [8]. There are many soft decision decoders (SDD) that take in real-valued quantised information and creates a reliability matrix which in turn improves the coding when it comes to decoding. For example for RS codes one of the most popular SDD is known as the Koetter and Vardy (KV) decoder which is used in conjunction with a list decoder, Guruswami and Sudan (GS) which provides a coding gain with respect to HDD [9, 10]. In this dissertation, for the research conducted it is irrelevant to go into further detail about the GS and KV algorithm. Therefore the SD decoder used for this research is introduced in the next section.

2.7.1 Parity-Check Transformation Algorithm

The Parity-Check Transformation Algorithm (PTA) decoder is a soft-decision decoder designed to exploit the parity check equation of a linear block code. This algorithm tries to satisfy the syndrome checks based on the soft-information provided to it. This is a symbol level iterative SDD [35]. Ogundile *et al* has shown that for RS codes PTA outperforms KV-GS SDD, of which the latter was designed specifically for RS codes [35]. PTA can be broken down into two main stages namely the reliability matrix updating stage and the syndrome-check satisfying stage. For PTA to work it is imperative that the parity matrix, defined as the H matrix, and the reliability matrix (denoted as β) are both provided initially. The operations of PTA are as follows

1. Obtain a Parity Check Matrix (H) in systematic form.
2. Obtain the Reliability Matrix β, this varies according to the modulation scheme used. The β matrix has to be of a special format such that, for an (n,k) block code in $GF(q)$ the β matrix has to have the dimensions $q \times n$.
3. From the β matrix now find the highest reliability for each columns and populate a row vector of n highest reliability information hr.
4. Populate two vectors namely R and U, where R is a row vector which has the dimensions $1 \times k$ and U has the dimensions $1 \times (n-k)$. R is known as the reliable index matrix, where it contains the indices of k highest reliabilities from hr. U will

be the unreliable index matrix which contains the n-k lower reliability indices. It must be noted that U and R should have the indices sorted in an ascending order.

5. Create an H^i matrix which now contains the transformed systematic H matrix in such a way that the matrix has identity columns at the same indices within U and parity columns at the indices in R. This is done by following the method introduced in [36].

6. Compute the syndrome using H^i, from this operations n-k parity check equations are obtained. With equations either having equal to 0 or some symbol.

7. If all of the parity check equation is satisfied to be 0, then PTA stops. If some of the equations are unsatisfied then the reliability in the indices contained in U is modified by δ and the indices in R by $\frac{\delta}{2}$. This updates the reliability matrix β which is fed back to the decoder.

8. Process is carried on until all the syndrome checks are satisfied or the stopping condition is reached.

Input: 1. Parity Check Matrix H $(n-k) \times n$
Input: 2. Reliability matrix β $(q \times n)$
Output: Decoded Codeword C^1

3. Populate hr row matrix

4. The highest k reliability indexes in hr now populate R vector. The rest of the indexes populate the U vector.

5. Adapt the H matrix using the indexes in U. Populate a sub matrix H_{sm} and insert columns from the systematic H matrix which have the same indexes in U,
$H_{sm} = H(:, U)$. Then compute $H^i = H_{sm}^{-1} \times H$.
r= Received Vector which is the symbols chosen from the β matrix with the highest reliability.
Exit = false
$delta$ = some value

WHILE (Exit==false)
6. Compute the syndrome by:
For length of H^i multiply the elements in each row to the elements from r vector. Also keep track of the indexes of the non-zero elements of each row of H^i.

7. If all the syndrome values ==0
then Exit = true.

Else

IF syndrome value = 0 the indexes which participated are modified in the β matrix by the delta value as:

If the index is in U
Add by $delta$

Else
Add by $\frac{delta}{2}$

IF syndrome value $\neq$ 0 the indexes which participated are modified in the β matrix by the delta value as:

If the index is in U
Subtract by $delta$

Else
Subtract by $\frac{delta}{2}$

8. **REPEAT**
C^1= Received Vector which is the symbols chosen from the β matrix with the highest reliability.

Algorithm 2: PTA Algorithm

2.8 Related Work

There are many works done in terms of creating MDS codes. According to [37, 38] a Pseudocyclic code is defined as a codeword vector c

$$c = \{c_0, c_1, \cdots c_{n-1}\}, \tag{2.49}$$

with a pseudocyclic shift such as

$$c = \{ac_{n-2}, c_0, \cdots c_{n-2}\}, \tag{2.50}$$

where a is some element within $GF(q)$. By following this definition it can be seen that cyclic codes are a subset of pseudocyclic codes. MDS pseudocyclic codes are studied and classified in [37, 38]. In [37] the authors present proofs which indicate that there exists pseudocyclic block codes with length $q-1$ and $q+1$. This proof also validates that these codes are extensions of BCH codes. In which they indicate that although there exists pseudocyclic MDS codes at various values of n and k, these codes are not cyclic codes which are subsets of pseudocyclic codes. In [38] the authors also prove that such pseudocyclic codes exists which are extensions of RS codes. The ones notable are extended and doubly extended RS codes. In [39] an extended RS code is defined as a block code (n, k) over a $GF(q)$ where $n = q$ and $d_{min} = n - k + 1 = q - k + 1$. The extended RS code is obtained by adding an extra parity digit, assuming β to be the primitive element the G matrix is expressed as

$$G = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 & 1 \\ 1 & \beta & \beta^2 & \cdots & \beta^{(q-2)} & 0 \\ 1 & \beta^2 & \beta^4 & \cdots & \beta^{(q-2)2} & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots & 0 \\ 1 & \beta^{(k-1)} & \beta^{2(k-1)} & \ldots & \beta^{(q-1)(k-1)} & 0 \end{bmatrix}. \tag{2.51}$$

The work in this research paper states that there exists an extended RS code $(q, k, d_{min} = q - k + 1)$ if and only if q is prime which then has an equivalent cyclic code defined as $(x-1)^{q-k}$ [39]. The work also states that if q is not prime then there will be no MDS codes of length q and dimensions $2 \leq k \leq q-2$ [39]. Further theorems and proofs are provided in [39]. The process of adding an extra column is known as code extending and the vice versa is known as code puncturing. Code puncturing is when from a generator matrix the last column is deleted to form punctured codes. By puncturing or extending the only changes made are to the length n and redundancy $(n-k)$, and not to the dimensions k of the code.

Heegard *et al* in [40] introduced a new class of algebraic block codes termed FIR Parity Check Codes (FIRPCC) using FIR filters which can be expressed in terms of syndromes. The implementation is described as an FIR filter structure, it is implemented through the aid of an IIR filter. The process implemented by Heegard *et al* are

- Concatenate the codewords,
- Implement an FIR or Sliding window,
- The syndrome that is obtained through the filter output is passed through a syndrome detector circuit.

This approach was primarily created for synchronization and error detection. In order to understand this fully some basics need to be presented. From (2.3) we obtain the parity-check matrix H, and from block code theory we know that a codeword is a valid codeword when it satisfies the condition $cH^t = 0$. The syndrome of the linear block code can be explained as $cH^t = S$. Note that this will always be zero if the codeword is valid. This leads to the premise that if there is a non-zero syndrome then the codeword is not valid and thus it can be claimed such that the codeword has an error in it.

Concatenating the codewords: Once a codeword polynomial is generated (prior to transmission by some other error control code) $c(x)$, with p terms (which has the structure shown in (2.11)) it is then concatenated with some input sequence

$$w(x) = \sum_{i=0}^{\infty} c_i(x)x^{ip}, \tag{2.52}$$

where $c_i(x)$ is the codeword sequence from the codeword polynomial. This is basically a string of codewords where multiples of p shows the starting point of one codeword.

Implementing the FIR filter: In order to show this process a basic example of a periodic sequence and power series needs to be examined. Let $r(x)$ be some binary polynomial such that

$$r(x) = 1 + r_1x^1 + r_2x^2 + \cdots + r_{p-1}x^{p-1}. \tag{2.53}$$

If this is a binary polynomial it is accurate to even describe it as a polynomial with coefficients in $GF(2)$ as shown in (2.11). This polynomial can be expressed as a power series such that

$$\frac{1}{r(x)} = r^{-1}(x) = r^*(x) = \sum_{i=0}^{\infty} R_ix^i, \tag{2.54}$$

where R_i are coefficients in the binary domain(i.e wither 1 or 0). This power series is bounded by the condition $r(x)r^*(x) = 1$. The power series $r^*(x)$ can be represented as a

periodic sequence as

$$r^*(x) = r^{*P}(x)\sum_{j=0}^{\infty} x^{jP} = \frac{r^{*P}(x)}{1-x^P}, \tag{2.55}$$

where P is the fundamental period of $r(x)$ and $r^{*P}(x) = \sum_{i=0}^{P-1} R_i x^i$ is the truncated version of the power series in (2.54). The question that needs to be asked is "what relevance does this have ?" The answer is that if $r(x)$ is a primitive polynomial in $GF(2)$ then the power series is periodic of $P \leq 2^{p-1} - 1$. From this it is valid to claim that $r(x)$ represented as a power series in (2.54) is a pseudorandom sequence.The difference equation for an IIR filter is provided in (2.27), this equation can be rewritten as a power series as

$$y[n] = \sum_{j=0}^{\infty} d_j x[n-j]. \tag{2.56}$$

For further mathematical proof the reader is advised to read through reference [40]. At this point the relationship between power series and filter difference equation is shown. The filter implemented has the following structure

$$h^{d+1}(x) = \frac{m(x) - x^{d+1}n(x)}{r(x)}, \tag{2.57}$$

where $r(x)$ is the recursive polynomial, $m(x)$ and $n(x)$ are both some polynomial which are chosen based on some criteria. Note all these polynomials are chosen in $GF(2)$ by Heegard *et al* in their paper [40]. The difference equation for 2.57 is written as

$$y[n] = \left(\sum_{i=0}^{deg(m(x))} M_i x[n-i] - \sum_{j=0}^{deg(n(x))} N_j y[(n-(d+1))-j] \sum_{l=1}^{p-1} R_l y[n-l] \right), \tag{2.58}$$

where the first and second summations are the power series representation of the polynomials $m(x)$ and $x^{d+1}n(x)$ which has terms till the maximum degree of powers the respective polynomials contain, and finally the last summation is (2.54) which is the equivalent truncated power series till the maximum power of the recursive polynomial $r(x)$. What (2.57) and (2.58) achieve is it extracts the polynomial coefficients of degree d from the infinite series of $r(x)$ thus forming a subsequence of finite terms. This approach makes $h^{d+1}(x)$ appear as an FIR filter although it uses the structure of an IIR filter. To implement this Heegard *et al* [40] chose an $r(x)$ which was a primitive polynomial and then solved for $h^{d+1}(x)$ where appropriate $m(x)$ and $n(x)$ were chosen for the length of $d+1$ bits.

The Syndrome Calculator: The concatenated sequence is then passed through the filter implemented. Heegard *et al* [40] were trying to answer the problem of synchronization and error-detection. They achieved synchronization by passing $w(x)$ from (2.52) through the filter. By applying this technique they also exploit the properties of the

syndrome detection. It is known that a syndrome can be calculated as $S = cH^t$, this is known as a coset of the linear block code. In practice it is possible to encode on this coset of a linear block code. This can be achieved as $c = mG + \sigma$, where $\sigma H^t = S$. From this theory the syndrome sequence obtained is

$$S(x) = h^{d+1}(x)w(x) = \sum_{i=0}^{\infty}(y_i(x) + x^d s(x)x^{ip}), \tag{2.59}$$

where $y_i(x)$ is some output from the filter and $s(x)$ is the syndrome of the codeword $c(x)$. The Syndrome sequence is then passed on to the syndrome detector circuit. This circuit basically contains $p - d$ parity bits which then will have coefficients from $s(x)$ shifted into it to establish a desired sync pattern. For further enquiries the reader is advised to read Heegard *et al* [40].

The theory covered in this section will be used to test the block codes constructed, in this research, in order to prove its error correcting characteristics. The PTA algorithm will be used to decode the block codes constructed from the filter structure. In conclusion the use of filter structures in error-control coding to detect errors is discussed here. This will be the necessary foundation to show how IIR filter structures are used to construct linear block codes.

Chapter 3

Methodology

3.1 Introduction

The methodology adopted for this research must answer the claims made to support the research problem which is to

> "Investigate the use of Infinite Impulse Response filters to construct linear block codes that meet high minimum distance with flexible length and information which are also easy to decode."

The methodology must be able to prove the existence of block codes with high minimum distance which are constructed from an IIR filter structure. The methodology must also scrutinize the structure and error-correcting capabilities of the code.

This chapter is structured as follows: Section 3.2 will provide the overview of the research methodology implemented. Section 3.3 will go in depth on the specifics of irreducible polynomials that was used to implement the filter functions. Section 3.4 will elaborate on how the message books were generated and the following section 3.5 illustrates the different types of codebooks generated. The evaluation of the distance bounds for the codeword are covered in section 3.6 and finally section 3.7 covers the investigation of the periodic nature of the codewords generated.

3.2 Overview of Research Methodology

The methodology focuses on four sections namely, construction of the codeword, extraction of the G-matrix, testing for cyclicity, evaluation of the distance bounds and the decoding

of the codewords. The overview of the research methodology is provided in figure 3.1. This will attempt to answer the sub questions formed to warrant the claims made to support the research problem statement.

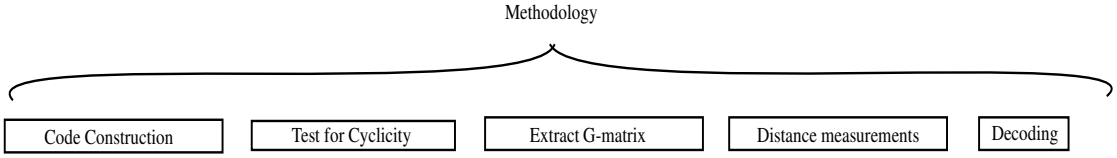

FIGURE 3.1: Overview of Research Methodology

3.3 Code Construction

The codes are constructed through the use of an IIR filter structure. The filter structure acts as an encoder, which receives a message of a particular block length and then produces an output codeword. The codewords are then combined to form a codebook. The IIR filter structure is constructed from appropriate coefficients which belong to the Galois field chosen. The processes followed to construct a codebook is shown in Figure 3.2, which is examined in detail in the following subsections.

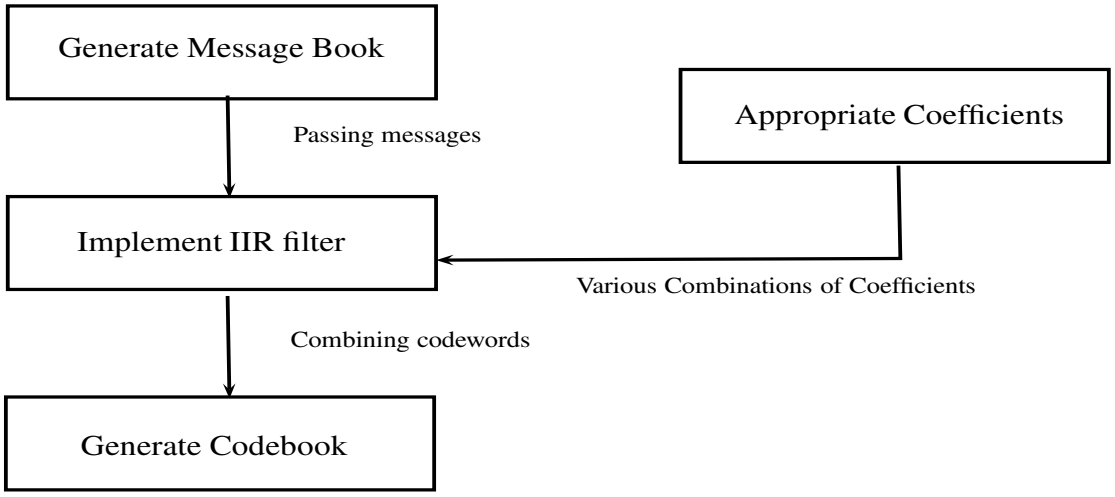

FIGURE 3.2: Code Construction Processes

3.3.1 Generate Message book

The messages will have a length of n, out of which k consecutive symbols will be the information and the remainder will be parity, thus rendering it in the systematic format. The message books vary depending on the number of information symbols that are required and can be represented mathematically as q^k, where q is the Galois field size

used and k the number of information symbols. Algorithm 3 illustrates the pseudo code on how the message book is generated.

Input: 1. Desired Length of block code n
Input: 2. Desired Rate of block code k
Input: 3. Desired Galois Field Size $q = p^r$
Output: Message book M_b

4. Preallocate matrix with dimensions $(q^k) \times n$

5. Create a sub matrix with dimensions $(q^k) \times k$

6. Generate permutations for all values ranging from $0 - (q-1)$, the number of permutations depends on the size of k

7. Fill the sub matrix $(q^k) \times k$ with all the permutations

8. Populate sub matrix into the initial matrix $(q^k) \times n$

9. The big matrix is the message book, where k subsequent symbols will contain information from the permutations and the rest $n - k$ symbols will be zero representing parity bits

Algorithm 3: Pseudo code for creating the Message book

Two sets of fields are used for this research, $q = 4$ and $q = 8$. The higher field size is used to prove MDS codes exist and is easier to computationally work with on shorter lengths. A smaller field size is used when working with a larger length in order to meet the Griesmer bound and to ease the computational operations. The information symbol starts at a minimum of 3 and is increased till 5, these ranges are chosen due to the computational complexity, which increases by a factor of q. The field chosen for this is $q = 4, GF(4)$ and the number of messages range from $64(k = 3)$ to $1024(k = 5)$. For $GF(8)$, k will remain at 3 since it is a shorter code with combinations of 512. The reason for choosing these values and how it affects the computational complexity is explained in the sections below.

3.3.2 Filter Implementation

The filter is implemented using the built in function of Octave. The filter transfer function requires two sets of inputs which can be expressed as

$$T_r = \frac{\begin{bmatrix} b_0 & b_1 & b_2 & \cdots & b_M \end{bmatrix}}{\begin{bmatrix} a_0 & a_1 & a_2 & \cdots & a_M \end{bmatrix}}, \tag{3.1}$$

where a_j and $b_j \in GF(q)$, $j \in \{0, 1, 2, \cdots, M\}$ and M is the number of memory blocks. The constraints for constructing the filters are

- The numerator must contain at least one non-zero coefficient, i.e. $b_j, j \in \{0, 1, 2, \cdots, M\}$ where it must have at least one b_j that is ≥ 1.
- The denominator must always have an a_0 such that $a_0 \in GF(q)$ and $a_0 \geq 1$.

From the constraints the number of combinations for T_r is worked out as the product of the number of memory blocks in the numerator (is worked out as q^{M+1}) and the number of memory blocks for the denominator (is $q^M \times (q-1)$). In $GF(4)$ with $M = 3$ the number of filter transfer functions are $4^4 \times (4^3 \times (3)) = 49344$ and for M=5 it is $4^6 \times (4^5 \times (3)) = 12582912$. For $GF(8)$ with $M = 3$ and $M = 4$ the number of combinations are 229376 and 14680064 respectively.

3.3.3 Generate Coefficients

The coefficients are constrained in relation to the maximum number of memory blocks that are used for the filters. From Chapter 2, Figure 2.4 and from (2.25), it can be seen that the number of memory blocks are 2 and the number of coefficients are 3. The coefficients can be represented as a row vector

$$L = \begin{bmatrix} l_0 & l_1 & l_2 & \cdots & l_M \end{bmatrix}, \tag{3.2}$$

where $l_i \in GF(q)$, $i \in \{0, 1, 2, \cdots, M\}$ and M is the number of memory blocks and $M+1$ the number of coefficients. To work in $GF(4)$, taking into consideration computational complexity and the constraints, the number of coefficients ranges from 4 to 6, and for $GF(8)$ are 3 and 4. For a memory block of 4 the number of combinations(number of vectors) are $4^4 = 256$ and for 6 is $4^6 = 4096$(in $GF(8)$ for 3 is $8^3 = 512$ and 4 is $8^4 = 4096$).

3.3.4 Generate Code book

For each filter transfer function that is implemented, all the messages contained within the message book is passed through the filter and the codewords that come out are combined to form a code book. The codebook will have the same dimensions and length as the message book.

3.4 Extracting the G-matrix

The G-matrix is extracted from the codebook through a computer search. The G-matrix is constructed by extracting codewords that are in the systematic format. The codewords are selected based on how they form an identity matrix in the structure. The approach taken to determine the appropriate codewords to form the G-matrix is covered in Algorithm. The G-matrix is extracted to test for linear independence and compared to the structure of quasi-cyclic codes.

Input: 1. $(q^k \times n)$ Codebook containing the codewords
Output: $k \times n$ Systematic Generator Matrix

While GenratorMatrixFound != True

2. Extract the codeword at the CodewordIndex

FOR (i =0; i < length of codeword; i++)
3. Search for the systematic first 3 symbols

4. If found, populate the Generator Matrix.

5. If found, increment the GeneratorMatrixIndex

END
6. If GeneratorMatrixIndex = k, then set GeneratorMatrixFound = True
END
7. Output the Generator Matrix.

Algorithm 4: Pseudo code for extracting a systematic G-matrix from the Codebook

3.5 Test for Cyclicity

The codebook is used to test for cyclicity. For cyclic codes, any cyclic shift of a codeword is present in the codebook. The pseudo code is provided in Algorithm on how the test for cyclic shift is done.

3.6 Distance Measurement

The distance measurement is conducted for both Griesmer and Singleton bounds. This is done by iteratively working out the Hamming distance of all the non-zero codewords contained within the codebook and then choosing the lowest hamming distance found. This becomes the distance profile for the codebook, which has a transfer function associated with it. The association is such that the codebook contains codewords created

Input: 1. $(q^k \times n)$ Matrix containing the codewords
Output: Column vector containing number of times the cyclic shift was true

While CodewordIndex != End of Codeword matrix

2. Extract the codeword at the CodewordIndex

FOR (i =0; i < length of codeword; i++)
3. Shifted codeword = Circular Shift the first element of the codeword by i^{th} place

4. Find Shifted Codeword in Codebook
5. If found increase the counter.

END
6. Populate a column vector with the number of times the circular shifts of the codeword was found
7. Output the column vector.

Algorithm 5: Pseudo code for Direct form II IIR filter structure

by the IIR filter with a particular transfer function. A distance profile is created for each transfer function which then is used to create codes that achieve the Griesmer and Singleton bounds.

3.7 Decoding

In order to truly show the validity of the codewords constructed from the filter structures, the codes need to be decoded by an existing decoder. A comparison is also drawn from comparing MDS codes created from the filter structure to another MDS code, namely Reed-Solomon codes. The decoding methodology focuses on implementing a simulated system that ranges from the encoding of messages to the decoding of codewords. The simulated system consists of the following: the Input messages are encoded using the filter structure, then transmitted through a modelled BPSK modulation scheme, with an AWGN model simulating a real valued noise model, alongside a demodulator which finally passes the codewords to the PTA decoder to decode the codewords. A brief layout of the system is provided in Figure 3.3 The Monte Carlo test is conducted on this system to generate the decoding performance of the decoder on the codewords. The test will give a distribution for the comparison of the Symbol Error Rate (SER) vs Signal to Noise Ratio (SNR).

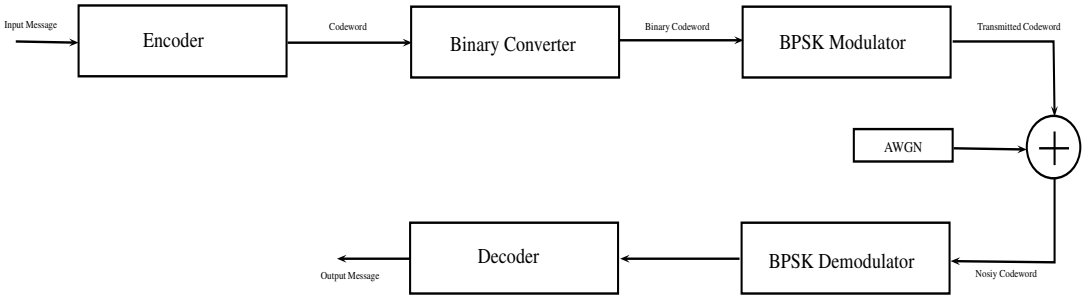

FIGURE 3.3: Sytem Layout

3.7.1 Encoder

The encoder is of two types, Reed-Solomon or the Filter Implementation, which is chosen to compare the two in terms of error-correcting capabilities. The encoder will encode messages in $GF(8)$ and illustrates the comparison of MDS and Non-MDS block codes as given in Table 3.1. The messages will be passed in a systematic format and thus it will be encoded in a systematic format. The codeword that comes from the encoder will be converted to binary as shown in (2.44).

TABLE 3.1: Encoders used for various Block codes based on Distance Criteria

Block Code	Distance Criteria	Block Code Type
(7,3)	MDS	Reed-Solomon
(7,3)	MDS	Filter Type
(7,3)	Non-MDS	Filter Type
(8,4)	MDS	Filter Type
(8,4)	Non-MDS	Filter Type

3.7.2 Modulation and Noise models

A modelled BPSK modulation scheme is used alongside a demodulator, which is known as a BPSK detector. The real valued noise is modelled based on an AWGN channel and this simulates the transmission of information on an AWGN noise infused channel (refer to chapter 2 for further detail on BPSK and AWGN noise).

3.7.3 Decoder

The decoder implemented uses the PTA algorithm. The PTA decoder is an iterative symbol level soft-decision decoder. The algorithm works primarily on adapting parity check matrix to decode the codeword. The algorithm for PTA provided in chapter 2 only

works on MDS codes , namely Reed-Solomon. The drawback for the PTA algorithm is that it fails to decode Non-MDS codes. The reason behind this failure is that any sub-matrix of a non-MDS code is rank deficient, thus invertible.

It is a well known theory that for (n, k) linear block codes there exists $n - k$ linearly independent rows in the H-matrix (refer to Chapter 2). For MDS codes any sub matrix of the H-matrix will also be linearly independent. Suppose an $s \times s$ sub matrix is extracted from the H-matrix, it will have s independent rows and is also deemed as full rank. Step 5 of adapting the H-matrix for PTA depends on that property [36]. This is not true for Non-MDS codes, although the H-matrix is full rank and has $n - k$ linearly independent rows, the sub matrix will not always be full rank. The original algorithm found the adapted H-matrix by multiplying the systematic H-matrix with the inverse of the sub matrix. In the case that it is not full rank the matrix cannot be inverted. By applying the Moore-Penrose algorithm it is possible to find a pseudo inverse of the sub matrix, but this fails if the sub matrix is rank deficient as well as symmetric, where a symmetric matrix A is defined as $A = A^T$, where A^T is the transpose of the A matrix. There are methods to invert such matrices, such as Cholesky decomposition, and from [41] an approach was taken to find the inverse of a sub matrix that is also symmetric and rank deficient. This holds true in the field of Natural numbers and Complex numbers and failed to provide a solution for Galois Field arithmetic. Both approaches are not applicable because after finding the pseudo inverse of the sub-matrix it needs to be multiplied back into the H-matrix, and this should result in a sparse adapted H-matrix but the pseudo inverse makes it more dense. Therefore the implemented modified method evaluates the adapted H-matrix as follows:

1. Create a matrix H_t which will have the columns of the systematic H-matrix organised in the indices contained U and R.
2. Perform Gaussian elimination on H_t and obtain a row reduced form.
3. H^i is now the sorted columns of the row reduced form of Ht

This modification makes the adapted H-matrix more sparse and because it relies on Gaussian elimination it will successfully execute on rank deficient matrices. This modification makes PTA applicable to both MDS and Non-MDS codes.

3.7.4 Monte Carlo Simulation

This experiment is conducted based on the Monte Carlo simulation. This is done 10000 times for SNR values ranging from 0 to 12dB. For each SNR value it sums up the total

of Symbol Error Rate (SER) and a semi-log plot is done to show that it forms a waterfall model.

Chapter 4

Results

The main objective of this research is to investigate the construction of block codes using the IIR filter structure. Error-control codes are classified as good ones based on a combination of factors such as ease of construction, achievable minimum distance and ease of decodability. The block codes generated from these filter structures are analysed with respect to their construction, distance profiles and decodability using an existent decoder. From the results several conclusions are drawn that validate the research problem.

Chapter 4 is structured in the following way: Section 4.1 discusses the results obtained from the construction of the codes; Section 4.2 presents the analysis of the filter structures; Section 4.3 illustrates the distance bounds achieved and finally Section 4.4 depicts the results obtained from decoding the codewords constructed through the IIR filter structures.

4.1 Results of the construction

The codes are constructed using the transfer function described in 2.25 for the filters. As shown in Chapter 3 equation (3.1), the codes are constructed with the coefficients found through the computer searches. A polynomial representation of these coefficients are used to explain the results. The construction of these codes are done in $GF(8)$ and done for various lengths and distance bounds. The following subsections are split in terms of (7,3), (8,3) and (8,4) block codes. Each section will address codes that are MDS and Non-MDS. Under these sections are the G-matrices for these codes that is obtained from the computer searches.

4.1.1 (7,3) codes

The block codes are constructed using the IIR filter structure, with the parameters $n = 7$ and $k = 3$. The coefficients chosen for the filter structure were found using computer searches. This section will illustrate how block codes can be created from the filter structure that are both MDS and Non-MDS codes.

4.1.1.1 MDS

Three random codes that achieve the Singleton bound were chosen to illustrate the systematic G-matrix formed. These three examples are IIR filters implemented with the maximum number of coefficients being 4 and achieved minimum distance of 5. Example 1 will have the first coefficients of the numerator and the denominator the same. Example 2 will have the first coefficient in the denominator as 1 and the final coefficients the same. Example 3 will have two distinct first and last coefficients in the numerator and denominator.

Example 1: The decimal and power representation (in terms of the primitive element α, refer to table 2.1) of the elements in the transfer function are presented below

$$T_r = \frac{\begin{bmatrix} 7 & 7 & 7 & 1 \end{bmatrix}}{\begin{bmatrix} 7 & 5 & 4 & 4 \end{bmatrix}} \Rightarrow T_r = \frac{\begin{bmatrix} \alpha^5 & \alpha^5 & \alpha^5 & 1 \end{bmatrix}}{\begin{bmatrix} \alpha^5 & \alpha^6 & \alpha^2 & \alpha^2 \end{bmatrix}}.$$

The algebraic representation of the transfer function is given as

$$T_r(x) = \frac{7 + 7x + 7x^2 + x^3}{7 + 5x + 4x^2 + 4x^3} \Rightarrow T_r(x) = \frac{\alpha^5 + \alpha^5 x + \alpha^5 x^2 + x^3}{\alpha^5 + \alpha^6 x + \alpha^2 x^2 + \alpha^2 x^3}.$$

The systematic G-matrix formed from this structure is

$$G = \begin{bmatrix} 1 & 0 & 0 & 5 & 2 & 1 & 6 \\ 0 & 1 & 0 & 4 & 2 & 6 & 5 \\ 0 & 0 & 1 & 3 & 1 & 1 & 5 \end{bmatrix} \Rightarrow G = \begin{bmatrix} 1 & 0 & 0 & \alpha^6 & \alpha & 1 & \alpha^4 \\ 0 & 1 & 0 & \alpha^2 & \alpha & \alpha^4 & \alpha^6 \\ 0 & 0 & 1 & \alpha^3 & 1 & 1 & \alpha^6 \end{bmatrix}.$$

Example 2: The second structure has a transfer function of

$$T_r = \frac{\begin{bmatrix} 3 & 4 & 5 & 4 \end{bmatrix}}{\begin{bmatrix} 1 & 1 & 4 & 4 \end{bmatrix}} \Rightarrow T_r = \frac{\begin{bmatrix} \alpha^3 & \alpha^2 & \alpha^6 & \alpha^2 \end{bmatrix}}{\begin{bmatrix} 1 & 1 & \alpha^2 & \alpha^2 \end{bmatrix}},$$

and has a polynomial representation of

$$T_r(x) = \frac{3 + 4x + 5x^2 + 4x^3}{1 + 1x + 4x^2 + 4x^3} \Rightarrow T_r(x) = \frac{\alpha^3 + \alpha^2 x + \alpha^6 x^2 + \alpha^2 x^3}{1 + x + \alpha^2 x^2 + \alpha^2 x^3}.$$

The systematic G matrix is obtained as

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 7 & 4 & 1 \\ 0 & 1 & 0 & 5 & 3 & 3 & 6 \\ 0 & 0 & 1 & 4 & 3 & 4 & 5 \end{bmatrix} \Rightarrow G = \begin{bmatrix} 1 & 0 & 0 & 1 & \alpha^5 & \alpha^2 & 1 \\ 0 & 1 & 0 & \alpha^6 & \alpha^3 & \alpha^3 & \alpha^4 \\ 0 & 0 & 1 & \alpha^2 & \alpha^3 & \alpha^2 & \alpha^6 \end{bmatrix}$$

.

Example 3: The transfer function represented in terms of coefficients are:

$$T_r = \frac{\begin{bmatrix} 3 & 1 & 4 & 1 \end{bmatrix}}{\begin{bmatrix} 5 & 5 & 6 & 6 \end{bmatrix}} \Rightarrow T_r = \frac{\begin{bmatrix} \alpha^3 & 1 & \alpha^2 & 1 \end{bmatrix}}{\begin{bmatrix} \alpha^6 & \alpha^6 & \alpha^4 & \alpha^4 \end{bmatrix}},$$

followed by the algebraic representation

$$T_r(x) = \frac{3 + 4x + 5x^2 + 4x^3}{1 + 1x + 4x^2 + 4x^3} \Rightarrow T_r(x) = \frac{\alpha^3 + \alpha^2 x + \alpha^6 x^2 + \alpha^2 x^3}{1 + x + \alpha^2 x^2 + \alpha^2 x^3}.$$

A systematic G matrix is obtained as

$$G = \begin{bmatrix} 1 & 0 & 0 & 5 & 2 & 1 & 6 \\ 0 & 1 & 0 & 4 & 2 & 6 & 5 \\ 0 & 0 & 1 & 3 & 1 & 1 & 5 \end{bmatrix} \Rightarrow G = \begin{bmatrix} 1 & 0 & 0 & 1 & \alpha^5 & \alpha^2 & 1 \\ 0 & 1 & 0 & \alpha^6 & \alpha^3 & \alpha^3 & \alpha^4 \\ 0 & 0 & 1 & \alpha^2 & \alpha^3 & \alpha^2 & \alpha^6 \end{bmatrix}.$$

Examples 1-3 shows that to obtain block codes with the MDS property the coefficients play no significant role, they are independent of the value of coefficients. From here on out for different lengths of codes only one example will be illustrated.

4.1.1.2 Non-MDS

This section will provide an example of a Non-MDS $(7, 3)$ block code with a minimum distance of 4. The transfer function represented in terms of coefficients are:

$$T_r = \frac{\begin{bmatrix} 1 & 7 & 7 & 4 \end{bmatrix}}{\begin{bmatrix} 7 & 5 & 0 & 4 \end{bmatrix}} \Rightarrow T_r = \frac{\begin{bmatrix} 1 & \alpha^5 & \alpha^5 & \alpha^2 \end{bmatrix}}{\begin{bmatrix} \alpha^5 & \alpha^6 & 0 & \alpha^2 \end{bmatrix}},$$

followed by the algebraic representation

$$T_r(x) = \frac{1 + 7x + 7x^2 + 4x^3}{7 + 5x + 4x^3} \Rightarrow T_r(x) = \frac{1 + \alpha^5 x + \alpha^5 x^2 + \alpha^2 x^3}{\alpha^5 + \alpha^6 x + \alpha^2 x^3}.$$

A systematic G matrix is obtained as

$$G = \begin{bmatrix} 1 & 0 & 0 & 3 & 3 & 2 & 5 \\ 0 & 1 & 0 & 1 & 6 & 5 & 7 \\ 0 & 0 & 1 & 5 & 6 & 5 & 2 \end{bmatrix} \Rightarrow G = \begin{bmatrix} 1 & 0 & 0 & \alpha^3 & \alpha^3 & \alpha & \alpha^6 \\ 0 & 1 & 0 & 1 & \alpha^4 & \alpha^6 & \alpha^5 \\ 0 & 0 & 1 & \alpha^6 & \alpha^4 & \alpha^6 & \alpha \end{bmatrix}.$$

4.1.2 (8,3) codes

This section will show that by extending the length of the block code to $n = 8$, it is still possible to construct codes using the IIR filter structure that are MDS and Non-MDS.

4.1.2.1 MDS

An MDS code with a minimum distance of 6 is presented here. The transfer function is provided

$$T_r = \frac{\begin{bmatrix} 5 & 7 & 6 & 7 \end{bmatrix}}{\begin{bmatrix} 7 & 7 & 1 & 5 \end{bmatrix}} \Rightarrow T_r = \frac{\begin{bmatrix} \alpha^6 & \alpha^5 & \alpha^4 & \alpha^5 \end{bmatrix}}{\begin{bmatrix} \alpha^5 & \alpha^5 & 1 & \alpha^6 \end{bmatrix}},$$

followed by it's algebraic representation

$$T_r(x) = \frac{5 + 7x + 6x^2 + 7x^3}{7 + 7x + x^2 + 5x^3} \Rightarrow T_r(x) = \frac{\alpha^6 + \alpha^5 x + \alpha^4 x^2 + \alpha^5 x^3}{\alpha^5 + \alpha^5 x + x^2 + \alpha^6 x^3}.$$

The G matrix is given as

$$G = \begin{bmatrix} 1 & 0 & 0 & 3 & 6 & 4 & 7 & 6 \\ 0 & 1 & 0 & 1 & 7 & 1 & 2 & 3 \\ 0 & 0 & 1 & 4 & 7 & 6 & 4 & 4 \end{bmatrix} \Rightarrow G = \begin{bmatrix} 1 & 0 & 0 & \alpha^3 & \alpha^4 & \alpha^2 & \alpha^5 & \alpha^4 \\ 0 & 1 & 0 & 1 & \alpha^5 & 1 & \alpha & \alpha \\ 0 & 0 & 1 & \alpha^2 & \alpha^5 & \alpha^4 & \alpha^2 & \alpha^2 \end{bmatrix}.$$

4.1.2.2 Non-MDS

An (8,3) code with a minimum distance of 5 is presented here, and the transfer function is

$$T_r = \frac{\begin{bmatrix} 4 & 6 & 2 & 0 \end{bmatrix}}{\begin{bmatrix} 4 & 2 & 6 & 1 \end{bmatrix}} \Rightarrow T_r = \frac{\begin{bmatrix} \alpha^2 & \alpha^4 & \alpha & 0 \end{bmatrix}}{\begin{bmatrix} \alpha^2 & \alpha & \alpha^4 & 1 \end{bmatrix}},$$

followed by it's algebraic representation

$$T_r(x) = \frac{4 + 6x + 2x^2}{4 + 2x + 6x^2 + x^3} \Rightarrow T_r(x) = \frac{\alpha^2 + \alpha^4 x + \alpha x^2}{\alpha^2 + \alpha x + \alpha^4 x^2 + x^3}.$$

The G matrix is given as

$$G = \begin{bmatrix} 1 & 0 & 0 & 5 & 6 & 1 & 2 & 4 \\ 0 & 1 & 0 & 0 & 5 & 6 & 1 & 2 \\ 0 & 0 & 1 & 2 & 4 & 6 & 1 & 3 \end{bmatrix} \Rightarrow G = \begin{bmatrix} 1 & 0 & 0 & \alpha^6 & \alpha^4 & 1 & \alpha & \alpha^2 \\ 0 & 1 & 0 & 0 & \alpha^6 & \alpha^4 & 1 & \alpha \\ 0 & 0 & 1 & \alpha & \alpha^2 & \alpha^4 & 1 & \alpha^3 \end{bmatrix}.$$

4.1.3 (8,4) codes

In this section both the length and rate are increased to $n = 8$ and $k = 4$, when compared to $(7, 3)$. The results show that it is still possible to construct block codes using the IIR filter structure to produce codes that are MDS and Non-MDS.

4.1.3.1 MDS

With the number of coefficients being 4 there was no (8,4) MDS code found from the search. By increasing the coefficients to 5 an (8,4) MDS code was found. This code is represented as an FIR filter. This (8,4) code has a minimum distance of 5. The transfer function represented in terms of coefficients are:

$$T_r = \begin{bmatrix} 5 & 1 & 4 & 5 & 6 \end{bmatrix} \Rightarrow T_r = \begin{bmatrix} \alpha^6 & 1 & \alpha^2 & \alpha^6 & \alpha^4 \end{bmatrix}$$

followed by the algebraic representation

$$T_r(x) = 5 + x + 4x^2 + 5x^3 + 6x^4 \Rightarrow T_r(x) = \alpha^6 + x + \alpha^2 x^2 + \alpha^6 x^3 + \alpha^4 x^4.$$

The corresponding systematic G matrix is

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 3 & 4 & 1 & 5 \\ 0 & 1 & 0 & 0 & 2 & 7 & 2 & 3 \\ 0 & 0 & 1 & 0 & 7 & 7 & 5 & 5 \\ 0 & 0 & 0 & 1 & 2 & 3 & 1 & 7 \end{bmatrix} \Rightarrow G = \begin{bmatrix} 1 & 0 & 0 & 0 & \alpha^3 & \alpha^2 & 1 & \alpha^6 \\ 0 & 1 & 0 & 0 & \alpha & \alpha^5 & \alpha & \alpha^3 \\ 0 & 0 & 1 & 0 & \alpha^5 & \alpha^5 & \alpha^6 & \alpha^6 \\ 0 & 0 & 0 & 1 & \alpha & \alpha^3 & 1 & \alpha^5 \end{bmatrix}.$$

4.1.3.2 Non-MDS (8,4)

The transfer function represented in terms of coefficients are:

$$T_r = \frac{\begin{bmatrix} 3 & 4 & 5 & 4 \end{bmatrix}}{\begin{bmatrix} 4 & 1 & 1 & 1 \end{bmatrix}} \Rightarrow T_r = \frac{\begin{bmatrix} \alpha^3 & \alpha^2 & \alpha^6 & \alpha^2 \end{bmatrix}}{\begin{bmatrix} \alpha^2 & 1 & 1 & 1 \end{bmatrix}},$$

, followed by the algebraic representation

$$T_r(x) = \frac{3 + 4x + 5x^2 + 4x^3}{4 + x + x^2 + x^3} \Rightarrow T_r(x) = \frac{\alpha^3 + \alpha^2 x + \alpha^6 x^2 + \alpha^2 x^3}{\alpha^2 + x + x^2 + x^3}.$$

he algebraic representation. A systematic G matrix is obtained as

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 5 & 5 & 7 \\ 0 & 1 & 0 & 0 & 3 & 7 & 6 & 5 \\ 0 & 0 & 1 & 0 & 5 & 2 & 2 & 6 \\ 0 & 0 & 0 & 1 & 2 & 1 & 0 & 2 \end{bmatrix} \Rightarrow G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & \alpha^6 & \alpha^6 & \alpha^5 \\ 0 & 1 & 0 & 0 & \alpha^3 & \alpha^5 & \alpha^4 & \alpha^6 \\ 0 & 0 & 1 & 0 & \alpha^6 & \alpha & \alpha & \alpha^4 \\ 0 & 0 & 0 & 1 & \alpha & 1 & 0 & \alpha \end{bmatrix}.$$

In conclusion the results present the evidence to the construction of block codes that are both MDS and Non-MDS for block codes that have the parameters: $(7, 3)$, $(8, 3)$ and $(8, 4)$. The results also show the existence of G-matrices for all the codes constructed. These G-matrices are formed through computer searches and are of a systematic structure. Further analysis of the G-matrices shows that the rows are all independent of each other thus proving that these codes are linear block codes. If the G-matrices are scrutinized further it is also evident that none of the matrices has a Quasi-Cyclic G-matrix structure. This concludes that the codes are not of Quasi-Cyclic structure.

4.2 Analysis of the filter structure

In signal processing it is common theory that a truncated IIR filter to a finite length, can be represented as an FIR filter. This holds true in Galois fields and the coefficients of the FIR filter can be represented with the impulse response from the IIR filter. The IIR filter alongside with its respective FIR filter equivalent in terms of values and powers of α for various codes are shown below:

(7,3) MDS code

$$T_r = \frac{\begin{bmatrix} 7 & 7 & 7 & 1 \end{bmatrix}}{\begin{bmatrix} 7 & 5 & 4 & 4 \end{bmatrix}} \Rightarrow \begin{bmatrix} 1 & 3 & 1 & 1 & 5 & 1 & 7 \end{bmatrix}$$

$$T_r = \frac{\begin{bmatrix} \alpha^5 & \alpha^5 & \alpha^5 & 1 \end{bmatrix}}{\begin{bmatrix} \alpha^5 & \alpha^6 & \alpha^2 & \alpha^2 \end{bmatrix}} \Rightarrow \begin{bmatrix} 1 & \alpha^3 & 1 & 1 & \alpha^6 & 1 & \alpha^5 \end{bmatrix}$$

$$T_r(x) = 1 + 3x + x^2 + x^3 + 5x^4 + x^5 + 7x^6$$
$$T_r(x) = 1 + \alpha^3 x + x^2 + x^3 + \alpha^6 x^4 + x^5 + \alpha^5 x^6$$

(7,3) Non-MDS code

$$T_r = \frac{\begin{bmatrix}1 & 7 & 7 & 4\end{bmatrix}}{\begin{bmatrix}7 & 5 & 0 & 4\end{bmatrix}} \Rightarrow \begin{bmatrix}4 & 2 & 5 & 2 & 3 & 5 & 6\end{bmatrix}$$

$$T_r = \frac{\begin{bmatrix}1 & \alpha^5 & \alpha^5 & \alpha^2\end{bmatrix}}{\begin{bmatrix}\alpha^5 & \alpha^6 & 0 & \alpha^2\end{bmatrix}} \Rightarrow \begin{bmatrix}\alpha^2 & \alpha & \alpha^6 & \alpha & \alpha^3 & \alpha^6 & \alpha^4\end{bmatrix}$$

$$T_r(x) = 4 + 2x + 5x^2 + 2x^3 + 3x^4 + 5x^5 + 6x^6$$
$$T_r(x) = \alpha^2 + \alpha x + \alpha^6 x^2 + \alpha x^3 + \alpha^3 x^4 + \alpha^6 x^5 + \alpha^4 x^6$$

(8,3) MDS code

$$T_r = \frac{\begin{bmatrix}5 & 7 & 6 & 7\end{bmatrix}}{\begin{bmatrix}7 & 7 & 1 & 5\end{bmatrix}} \Rightarrow \begin{bmatrix}2 & 3 & 5 & 7 & 3 & 3 & 1\end{bmatrix}$$

$$T_r = \frac{\begin{bmatrix}\alpha^6 & \alpha^5 & \alpha^4 & \alpha^5\end{bmatrix}}{\begin{bmatrix}\alpha^5 & \alpha^5 & 1 & \alpha^6\end{bmatrix}} \Rightarrow \begin{bmatrix}\alpha & \alpha^3 & \alpha^6 & \alpha^5 & \alpha^3 & \alpha^3 & 1\end{bmatrix}$$

$$T_r(x) = 2 + 3x + 5x^2 + 7x^3 + 3x^4 + 3x^5 + 1x^6$$
$$T_r(x) = \alpha + \alpha^3 x + \alpha^6 x^2 + \alpha^5 x^3 + \alpha^3 x^4 + \alpha^3 x^5 + x^6$$

(8,3) Non-MDS code

$$T_r = \frac{\begin{bmatrix}4 & 6 & 2 & 0\end{bmatrix}}{\begin{bmatrix}4 & 2 & 6 & 1\end{bmatrix}} \Rightarrow \begin{bmatrix}1 & 1 & 4 & 1 & 4 & 7 & 7 & 6\end{bmatrix}$$

$$T_r = \frac{\begin{bmatrix}\alpha^2 & \alpha^4 & \alpha & 0\end{bmatrix}}{\begin{bmatrix}\alpha^2 & \alpha & \alpha^4 & 1\end{bmatrix}} \Rightarrow \begin{bmatrix}1 & 1 & \alpha^2 & 1 & \alpha^2 & \alpha^5 & \alpha^5 & \alpha^4\end{bmatrix}$$

$$T_r(x) = 1 + x + 4x^2 + x^3 + 4x^4 + 7x^5 + 7x^6 + 6x^7$$
$$T_r(x) = 1 + x + \alpha^2x^2 + x^3 + \alpha^2x^4 + \alpha^5x^5 + \alpha^5x^6 + \alpha^4x^7$$

(8,4) Non-MDS code

$$T_r = \frac{\begin{bmatrix} 3 & 4 & 5 & 4 \end{bmatrix}}{\begin{bmatrix} 4 & 1 & 1 & 1 \end{bmatrix}} \Rightarrow \begin{bmatrix} 2 & 4 & 2 & 0 & 4 & 4 \end{bmatrix}$$

$$T_r = \frac{\begin{bmatrix} \alpha^3 & \alpha^2 & \alpha^6 & \alpha^2 \end{bmatrix}}{\begin{bmatrix} \alpha^2 & 1 & 1 & 1 \end{bmatrix}} \Rightarrow \begin{bmatrix} \alpha & \alpha^2 & \alpha & 0 & \alpha^2 & \alpha^2 \end{bmatrix}$$

$$T_r(x) = 2 + 4x + 2x^2 + 4x^4 + 4x^5$$
$$T_r(x) = \alpha + \alpha^2x + \alpha x^2 + \alpha^2x^4 + \alpha^2x^5$$

Since the MDS $(8, 4)$ code is generated through an FIR filter there is no need to represent that structure here. From these FIR structures it can been seen that the number of coefficients can be less than the length of the codewords generated. Examples of such structures are the MDS (8,4) code and MDS (8,3) code.

Block codes with some n and k that achieve Singleton bound can be created using the IIR filter structure. The very same IIR structure can be replaced with an FIR filter structure to generate the same codewords. The coefficients for the FIR filter structure can only be used for that particular n and k to achieve that minimum distance. From the results presented it is seen that for some cases the equivalent FIR requires more memory blocks to be implemented to mimic the IIR filter.

4.3 Achieved Distance Bounds

The two bounds tested for these codes are the Singleton and Griesmer bounds. The results from the influence of number of coefficients and increasing minimum distance is shown in the table 4.1. The results presented are done in $GF(4)$ due to the computational complexity. The codes in $GF(4)$ have shown to achieve the Griesmer bound and it is shown that by increasing the number of coefficients in the filter transfer function can also increase the minimum distance of the codes. From equation 2.8 $n \geq 21$ is required to obtain a $d_{min} = 16$. Therefore the table contains code rates of $\frac{3}{21}$, $\frac{4}{21}$ and $\frac{5}{21}$. It can be seen that by increasing the memory blocks the achievable distances of the codes

improve and tend towards the Griesmer bound. The significance of the memory blocks are affected more towards a higher rate of code.

TABLE 4.1: Distance Profile of codes with respect to the rate of the code and number of Memory Blocks

Memory Blocks	d_{min} for $k = 3$	d_{min} for $k = 4$	d_{min} for $k = 5$
4	16	4	4
5	16	13	5
6	16	14	13

4.4 Decoding

The results shows the performance of the PTA algorithm on the codewords constructed from the IIR filter. A comparison is shown between RS codes and the codes generated from the filter structure to show that the algorithm is a fair comparison in terms of decoding. In figure 4.1 the comparison between RS and IIR codes both MDS and Non-MDS is presented. This confirms the accuracy of the PTA algorithm as well as a comparison between the codes created from the filter structures and RS codes.

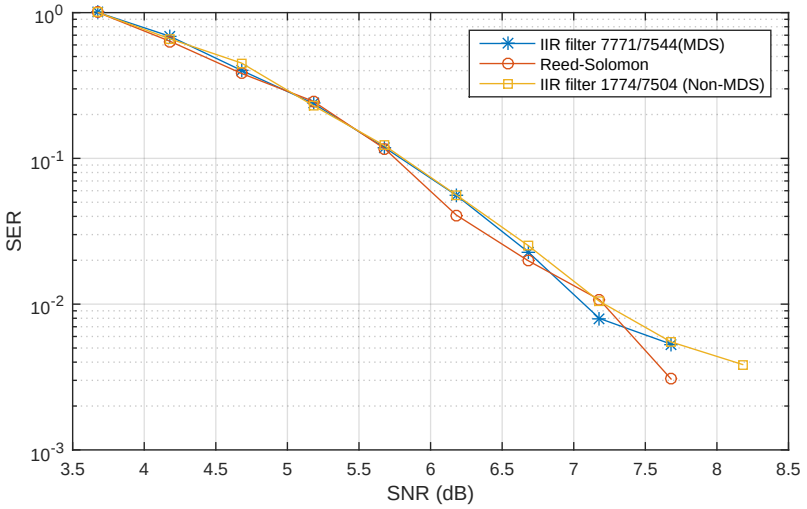

FIGURE 4.1: Performance of PTA on $(7, 3)$ Reed-Solomon, IIR MDS and IIR Non-MDS codes.

In Figure 4.1 it shows the comparison of an RS code with $d_{min} = 5$, an MDS code generated from the IIR filter with coefficients forming the transfer function of $\frac{7771}{7544}$ with a $d_{min} = 5$ and a non-MDS code with a $d_{min} = 4$, which is also constructed from the IIR filter with the transfer function of $\frac{1774}{7504}$.

4.4.1 Performance of PTA on codes

In this section the results from PTA on comparing different code rates are presented. Figure 4.2 depicts the comparison for MDS codes for different rates and figure 4.3 depicts the comparison for Non-MDS codes for different rates.

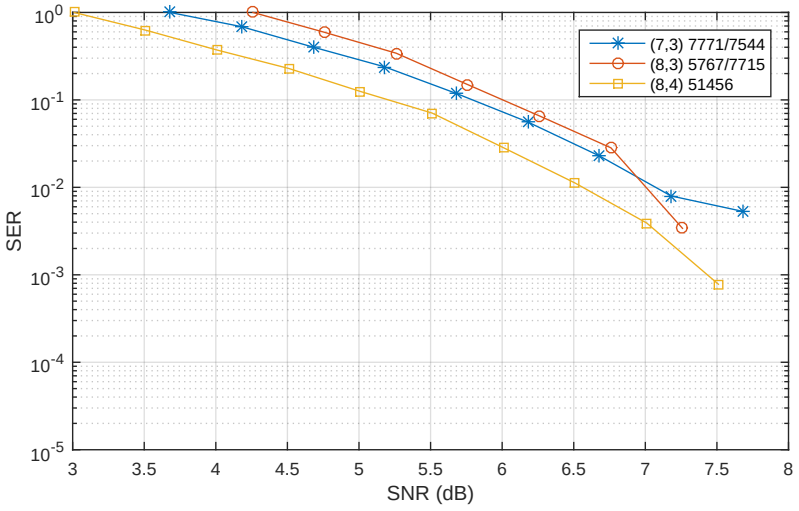

FIGURE 4.2: Performance of PTA on MDS codes for different rates

Evidently it can be stated that for MDS codes PTA performs slightly better with a higher rate code, as the $(8, 4)$ code has a coding gain of 0.8 dBs to code rate of $(8, 3)$ code. It shows that codes with a higher minimum distance performs better and this adheres to the theory of ECC.

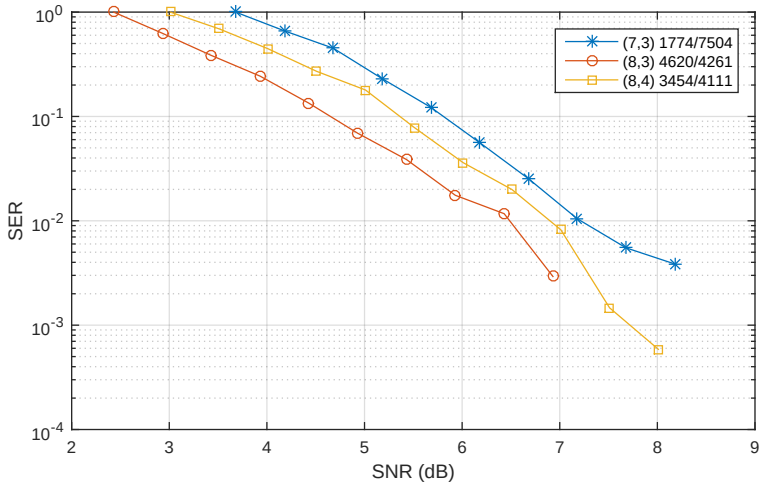

FIGURE 4.3: Performance of PTA on Non-MDS codes for different rates

For Non-MDS codes it can be seen that PTA performs better for a low rate code. A coding gain of 1 dB is achieved between $(8, 3)$ and $(7, 3)$ codes.

Chapter 5

Conclusion

The conclusions drawn from the research are summarized in this chapter. The main objective and conclusion is derived by answering the secondary objectives. The results are structured as:

1. Construction of codes
2. Decoding of codes
3. Analysis of codes

These are summarised from Sections 5.1 to 5.3 and Section 5.4 discusses future recommendations.

5.1 Construction of codes

Block codes were constructed using filter structures namely, IIR and FIR filters. From computer searches the generator matrices were obtained. The codes constructed are linear codes as is evident from the structure of the generator matrix where the rows were all linearly independent of each other. The generator matrices obtained from the codes are compared to quasi-cyclic codes and were concluded that they failed to have a quasi-cyclic structure. The generator matrices are different from the matrices constructed from Convolutional codes, although the generator matrices still contained the impulse response of the filter. From these structures Maximum Distance Separable codes were found. Non-MDS codes were also found and these codes were confirmed to obtain the Griesmer bound both for shot and long lengths. Results show that by adding more memory blocks it is possible to get closer to the Griesmer bound for large lengths.

5.2 Decoding of Codes

To confirm that these codes are decodable by block code decoder, a generic soft-decision decoder is used to decode the codes constructed. The PTA decoder is used to decode the codewords. The algorithms produced results for MDS codes but failed to decode for Non-MDS codes. A slightly modified PTA algorithm is used to decode the codes and is successful. PTA with the modification produced good results and successfully decodes both MDS and Non-MDS codes.

5.3 Analysis of codes

A further analysis on the structure of the codes generated from IIR filter indicates that such codes can also be represented by FIR filters with more memory blocks. The coefficients for the FIR filter can be found from the impulse response of the IIR filter. Although both the IIR and equivalent FIR filters produce the same code, and in some instances the memory block required by the equivalent FIR model might be less than or the same as the IIR filter, it is not scalable. The same IIR filter can produce MDS and Non-MDS codes, depending on the application, for various lengths without the need to change the coefficients in the memory blocks, and thus deemed scalable. From this it can be concluded that using filter structures MDS block codes can be encoded and decoded.

5.4 Future Recommendations

This research was an investigation if it was possible to classify block codes using filter structures, the conclusion drawn from this research is that such codes exist and can be decoded. The drawback is that there is no generalised solution in creating such codes nor is there an optimised computer search algorithm to find such codes from the filter structures. Therefore the recommendations for future work would be to provide a general formula for the construction of these codes.

The decoder used is a generic iterative soft-decision decoder, the possible future recommendation is that maybe devise a decoder that can decode this using the Markov model or employ a partial state observable decoder. The filters are finite state machines in which information is preserved therefore it is possible to apply a Markov model to deterministically determine the next state given an input sequence.

Bibliography

[1] C.E. Shannon. A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27(3):379–423, July 1948.

[2] R.W. Hamming. Error Detecting and Error Correcting Codes. *The Bell System Technical Journal*, 29(2):147–160, April 1950.

[3] M.J.E. Golay. Notes on Digital Coding. *Proc. IRE*, 37(2):657, June 1949.

[4] I.S. Reed and G. Solomon. Polynomial Codes Over Certain Finite Fields. *J. Soc. Ind. Appl. Math*, 8(2):300–304, June 1960.

[5] R. C. Bose and D. K. Ray-Chaudhuri. On a class of error correcting binary group codes. *Information and Control*, 3(1):68–79, March 1960. doi: 10.1016/s0019-9958(60)90287-4.

[6] Berlekamp E.R. *Algebraic Coding Theory*. McGraw-Hill, Inc., New York, 1 edition, 1968.

[7] J.L. Massey. Shift-register synthesis and BCH decoding. *IEEE Transactions on Information Theory*, 15(1):122–127, Jan 1969. ISSN 0018-9448. doi: 10.1109/TIT.1969.1054260.

[8] S. Wicker and V. Bhargava. *An Introduction to Reed-Solomon Codes*. Wiley-IEEE Press, 1994. ISBN 9780470546345. doi: 10.1109/9780470546345.ch1.

[9] V. Guruswami and M. Sudan. Improved decoding of Reed-Solomon and Algebraic-geometry codes. *IEEE Transactions on Information Theory*, 45(6):1757–1767, Sep 1999. ISSN 0018-9448. doi: 10.1109/18.782097.

[10] R. Koetter and A. Vardy. Algebraic soft-decision decoding of Reed-Solomon codes. *IEEE Transactions on Information Theory*, 49(11):2809–2825, 2003. doi: 10.1109/TIT.2003.819332.

[11] R.G. Gallager. Low-density parity-check codes. *IRE Transactions on Information Theory*, 8(1):21–28, January 1962. ISSN 0096-1000. doi: 10.1109/TIT.1962.1057683.

[12] D.J.C. MacKay and R.M. Neal. Near Shannon limit performance of Low Density Parity Check Codes. *Electronics Letters*, 33(6):457–458, Mar 1997. ISSN 0013-5194. doi: 10.1049/el:19970362.

[13] Elias P. Coding for Noisy Channels. *IRE Conv. Rec*, pages 4: 37–47, 1957.

[14] Jr. Forney, G.D. Convolutional codes i: Algebraic structure. *IEEE Transactions on Information Theory*, 16(6):720–738, Nov 1970. ISSN 0018-9448. doi: 10.1109/TIT.1970.1054541.

[15] Massey J.L. *Threshold Decoding*. MIT Press, Cambridge, 1961.

[16] A.J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, April 1967. ISSN 0018-9448. doi: 10.1109/TIT.1967.1054010.

[17] C. Berrou, A. Glavieux, and P. Thitimajshima. Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1. In *Communications, 1993. ICC '93 Geneva. Technical Program, Conference Record, IEEE International Conference on Communications*, volume 2, pages 1064–1070 vol.2, May 1993. doi: 10.1109/ICC.1993.397441.

[18] SOURCEFORGE.NET. MPI Octave-Forge Extra packages for GNU Octave, . URL `http://octave.sourceforge.net/mpi/overview.html`.

[19] SOURCEFORGE.NET. Signal Octave-Forge Extra packages for GNU Octave, . URL `http://octave.sourceforge.net/signal/overview.html`.

[20] SOURCEFORGE.NET. Control Octave-Forge Extra packages for GNU Octave, . URL `http://octave.sourceforge.net/control/overview.html`.

[21] SOURCEFORGE.NET. Communications Octave-Forge Extra packages for GNU Octave, . URL `http://octave.sourceforge.net/communications/overview.html`.

[22] A. A. Bruen and M. A. Forcinito. *Cryptography, Information Theory and Error-Correction. A Handbook for the 21st Century*. John Wiley & Sons, Inc., 2005.

[23] G. Cancellieri. *Polynomial Theory of Error Correcting Codes*. Springer Publishing Company, Incorporated, 2014.

[24] S Lin and D. J. Costello. *Error Control Coding, Second Edition*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2004.

[25] Schmale. W. Gluesing-Luerssen. H. Distance bounds for Convolutional codes and some optimal codes. *Information Theory, Optimization and Control, Cornell University Library*, pages 1–22, May 2003.

[26] W. E. Ryan and S Lin. *Channel Codes Classical and Modern*. Cambridge University Press, New York, USA, 2009.

[27] C L Phillips, J Parr, and E Riskin. *Signals, Systems, and Transforms*. Prentice Hall Press, Upper Saddle River, NJ, USA, 2007.

[28] B.P. Lathi. *Signal Processing and Linear Systems*. The Oxford series in electrical and computer engineering. Oxford University Press, 2009.

[29] S. J. Mason. Feedback theory-some properties of signal flow graphs. *Proceedings of the IRE*, 41(9):1144–1156, Sept 1953. ISSN 0096-8390. doi: 10.1109/JRPROC.1953.274449.

[30] S. J. Mason. Feedback theory-further properties of signal flow graphs. *Proceedings of the IRE*, 44(7):920–926, July 1956. ISSN 0096-8390. doi: 10.1109/JRPROC.1956.275147.

[31] R Johannesson and K. S. Zigangirov. *Fundamentals of Convolutional Coding*. Wiley-IEEE Press, 1999.

[32] W.C. Huffman and V. Pless. *Fundamentals of Error-Correcting Codes*. Cambridge University Press, 2010.

[33] R.A. Carrasco and Johnston M. *Non-Binary Error Control Coding for Wireless Communication And Data Storgae*. John Wiley & Sons, Ltd., West Sussex, United Kingdom, 2008.

[34] John G. Proakis. *Digital Communications*. McGraw-Hill, Avenue of the Americas, New York, 1995.

[35] O.O. Ogundile, Y.O. Genga, and D.J.J. Versfeld. Symbol level iterative soft decision decoder for Reed-Solomon codes based on parity-check equations. *Electronics Letters*, 51(17):1332–1333, 2015. ISSN 0013-5194. doi: 10.1049/el.2015.1932.

[36] D.J.J. Versfeld, J.N. Ridley, H.C. Ferreira, and A.S.J. Helberg. On Systematic Generator Matrices for Reed Solomon Codes. *Information Theory, IEEE Transactions on*, 56(6):2549–2550, June 2010. ISSN 0018-9448. doi: 10.1109/TIT.2010.2046104.

[37] A. Krishna and D.V. Sarwate. Pseudocyclic maximum-distance-separable codes. *Information Theory, IEEE Transactions on*, 36(4):880–884, Jul 1990. ISSN 0018-9448. doi: 10.1109/18.53751.

[38] J.P. Pedersen and C. Dahl. Classification of pseudo-cyclic MDS codes. *Information Theory, IEEE Transactions on*, 37(2):365–370, Mar 1991. ISSN 0018-9448. doi: 10.1109/18.75254.

[39] R.M. Roth and G. Seroussi. On cyclic MDS codes of length q over GF(q). *Information Theory, IEEE Transactions on*, 32(2):284–285, Mar 1986. ISSN 0018-9448. doi: 10.1109/TIT.1986.1057151.

[40] C. Heegard and A.J. King. FIR Parity Check Codes. *Communications, IEEE Transactions on*, 48(7):1108–1113, Jul 2000. ISSN 0090-6778. doi: 10.1109/26.855518.

[41] P Courrieu. Fast Computation of Moore-Penrose Inverse Matrices. *CoRR*, abs/0804.4809, 2008. URL `http://arxiv.org/abs/0804.4809`.