

Design of a Forward Error Correction algorithm for a Satellite modem.

Mamphoko Nelly Sefara



Thesis presented in fulfillment of the requirements for the degree
of **Master of Science in Engineering Science**
at the University of Stellenbosch.

Supervisor: Prof. J.G. Lourens

November 2001

Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.

Abstract

One of the problems with any deep space communication system is that information may be altered or lost during transmission due to channel noise. It is known that any damage to the bit stream may lead to objectionable visual quality distortion of images at the decoder. The purpose of this thesis is to design an error correction and data compression algorithm for image protection, which will allow the communication bandwidth to be better utilized. The work focuses on Sunsat (Stellenbosch Satellite) images as test images. Investigations were done on the JPEG 2000 compression algorithm's robustness to random errors, putting more emphasis on how much of the image is degraded after compression. Implementation of both the error control coding and data compression strategy is then applied to a set of test images. The FEC algorithm combats some if not all of the simulated random errors introduced by the channel. The results illustrates that the error correction of random errors is achieved by a factor of 100 times (x100) on all test images and that the probability of error of 10^{-2} in the channel (10^{-4} for image data) shows that the errors causes little degradation on the image quality.

Opsomming

Een van die probleme met kommunikasie in die ruimte is dat informasie mag verlore gaan en/ of gekorrupteer word deur ruis gedurende versending deur die kanaal. Dit is bekend dat enige skade aan die bisstroom mag lei tot hinderlike vervorming van die beelde wat op aarde ontvang word. Die doel van hierdie tesis om foutkorreksie en datakompresie te ontwikkel wat die satelliet beelde sal beskerm gedurende versending en die kommunikasie kanaal se bandwydte beter sal benut. Die werk fokus op SUNSAT (Stellenbosch Universiteit Satelliet) se beelde as toetsbeelde. Ondersoeke is gedoen na die JPEG2000 kompresie algoritme se bestandheid teen toevalsfoute, met klem op hoeveel die beeld gedegradeer word deur die bisfoute wat voorkom. Beide die kompresie en die foutkorreksie is ge-implementeer en aangewend op die toetsbeelde. Die foutkorreksie bestry die gesimuleerde toevalsfoute, soos wat dit op die kanaal voorkom. Die resultate toon dat die foutkorreksie die toevalsfoute met 'n faktor 100 verminder, en dat 'n foutwaarskynlikheid van 10^{-2} op die kanaal (10^{-4} op die beelddata) weinig degradering in die beeldkwaliteit veroorsaak.

**With Love and affection to my lovely little girl Tahisha,
Matome, Khomotjo and My Mom Mosima.**

Acknowledgements

Firstly, I would like to thank the Lord for the strength and ability to complete the project, my study leader for his support during the project. Prof. J.G Lourens, without your advice I would have gotten lost on the way to completion.

I would also like to thank my Family, for their undivided support during all those trying times and for believing in me. You gave me the reason to finish the things that I would never have finished without your confidence in me. Lastly thanks to everyone who contributed to the progress of the project.

List of Abbreviations and Acronyms

ARQ	Automatic repeat request
BCC	Binary convolutional codes
BCH	Bose, Chaudhuri, Hocquenghem
BER	Bit error rate
BG	Background
DCT	Discrete cosine transform
DSTV	Digital satellite television
DWT	Discrete wavelet transform
ECC	Error control coding
FEC	Forward Error correction
JPEG	Joint photographic Expert group
JPEG-2k	JPEG 2000
MLD	Maximum likelihood decoding
MQ	multiple quantization
OBC 2	On board computer 2
PSNR	Peak signal to noise ratio
RMSE	Root mean square error
ROI	region of interest
RS	Reed Solomon coding
SNR	Signal to noise ratio

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	I
LIST OF ABBREVIATIONS AND ACRONYMS.....	II
TABLE OF CONTENTS.....	III
LIST OF TABLES	VI
LIST OF FIGURES	VII
1 INTRODUCTION.....	1
1.1 THESIS OVERVIEW	4
1.2 GOAL	6
2 ERROR CONTROL CODING	8
2.1 CHANNEL CODING	9
2.2 INTRODUCTION TO FORWARD ERROR CORRECTION	10
2.2.1 <i>General Description (What is error correction?).....</i>	<i>11</i>
2.2.2 <i>What mathematics is involved in the theory of error correction?</i>	<i>13</i>
2.3 BLOCK CODES.....	14
2.3.1 <i>Properties of Block codes.....</i>	<i>16</i>
2.4 CYCLIC CODES.....	17
2.4.1 <i>Cyclic codes in detail</i>	<i>18</i>
2.5 CONVOLUTIONAL CODING	22
2.5.1 <i>Description.....</i>	<i>22</i>
2.5.2 <i>Binary Convolutional Codes</i>	<i>24</i>
2.5.3 <i>Encoder Structure.....</i>	<i>25</i>
2.5.4 <i>Viterbi algorithm.....</i>	<i>27</i>
2.6 INTERLEAVING	29

2.6.1	<i>Block interleaving</i>	30
2.6.2	<i>Convolutional interleaving</i>	30
2.7	REED SOLOMON CODES	31
2.7.1	<i>Encoding</i>	33
2.7.2	<i>Decoding</i>	33
2.8	CONCATENATED CODES.....	36
2.9	CONCLUSION.....	37
3	IMAGE COMPRESSION	39
3.1	INTRODUCTION	39
3.2	JPEG.....	40
3.3	JPEG 2000.....	41
3.3.1	<i>Introduction</i>	41
3.3.2	<i>ROI</i>	45
3.3.3	<i>ROI mask creation</i>	46
3.4	ARCHITECTURE OF THE JPEG 2000	46
3.5	APPLICATION REQUIREMENTS FEATURES	49
3.6	COMPARISON METHODOLOGY	50
3.6.1	<i>Compression efficiency</i>	50
3.7	CONCLUSION.....	52
4	DESIGN AND DESIGN CONSIDERATIONS.....	54
4.1	JPEG 2000 ENCODING AND DECODING	55
4.1.1	<i>JPEG-2k Performance</i>	56
4.2	REED SOLOMON ENCODING	60
4.3	RS DECODING.....	61
4.4	CONVOLUTIONAL ENCODER.....	67
4.5	VITERBI DECODING ALGORITHM	68
4.6	CONCATENATED CODES.....	68
4.7	CONCLUSION.....	69
5	IMPLEMENTATION.....	70
5.1	JPEG COMPRESSION AND DECOMPRESSION	70
5.2	FEC	71

5.3	CONCLUSION.....	74
6	RESULTS.....	75
7	CONCLUSION AND RECOMMENDATIONS	84
7.1	CONCLUSION.....	84
7.2	RECOMMENDATIONS.....	86
APPENDIX A.....		87
A.1 BCH CODES		87
A.2.1	GALOIS FIELDS.....	88
A.2.2	THE BINARY BCH CODE	88
A.2 CONVOLUTIONAL CODING.....		90
APPENDIX B: THE SOURCE CODE.....		104
APPENDIX C: TEST IMAGES		105
BIBLIOGRAPHY		106

List of Tables

Table 4.1: Lena2 Image results for JPEG the process	58
Table 6.1: Lena2 JPEG-2k and FEC algorithm	77
Table 6.2: Hartswater Image.....	80
Table 6.3: Tibetan Image.....	82
Table 6.4: Tibetan 1 Image.....	84
Table 6.5: Syrie1 Image	86
Table 6.6: Syrie 1.1 Image	88

List of Figures

Figure 2.1: Systematic format of the Code word.....	14
Figure 2.2: The encoding circuit for cyclic codes	20
Figure 2.3: Symbols used in the encoder	21
Figure 2.4: The Feedforward Convolutional encoder	26
Figure 3.1: Block Diagrams of the JPEG 2000 (AA) Encoder (BB) Decoder	47
Figure 4.1: The graph of Mean square error versus Probability of error.....	59
Figure 4.2: The first step of the decoding algorithm	62
Figure 4.3: The third Step of the Decoding algorithm	64
Figure 4.4: The Third step of the decoding algorithm	66
Figure 4.5: The block diagram for the Concatenated code system and the JPEG codec.....	69
Figure 5.1: The thesis simulation diagram	73
Figure 6.1: The Mean square error versus the Probability of error for JPEG-2k, FEC algorithm.....	79
Figure 6.2: The Mean square error versus the Probability of error for All processes	81
Figure 6.3: The Mean square error versus the Probability of error for All processes	83
Figure 6.4: The Mean square error versus the Probability of error for All processes	85
Figure 6.5: The Mean square error versus the Probability of error for All processes	87
Figure 6.6: The Mean square error versus the Probability of error for All processes	89
Figure A. 1: Encoder state diagram.....	91
Figure A. 2: The symbols used in the encoder state diagram.....	92

1 Introduction

The purpose of the project was to implement and verify the data compression algorithm that will allow the communication bandwidth to be better utilized for the Sunsat (Stellenbosch University Satellite) satellite channel. The design for the algorithm requires the implementation of both the error control coding and the data compression algorithm.

Data transmitted from the ground station is prone to errors both random bit errors and block (nature) burst errors. It is extremely important to protect this data against any errors occurring undetected. For this reason, an error detection and correction scheme should be used.

The communication channel introduces noise and interference to corrupt the transmitted signal. At the receiver, the channel corrupted transmitted signal is mapped back to binary bits. The received binary information is an estimate of the transmitted binary information.

Bit errors may result due to the transmission and the number of bit errors depends on the amount of noise and interference in the communication channel.

Channel coding is often used in digital communication systems to protect the digital information from noise and interference and reduce the number of bit errors. Channel coding protects digital data from errors by selectively introducing redundancies in the transmitted data to improve link performance. It is mostly accomplished by selectively introducing redundant bits into the transmitted information stream. These additional bits will allow detection and correction of bit errors in the received data stream and provide more reliable information transmission.

The cost of using channel coding to protect the information is a reduction in data rate or an expansion in bandwidth. This reduces the bandwidth efficiency of the link in high Signal to noise ratio (SNR) conditions, but provides excellent Bit error rate (BER) performance at low SNR values. The presence of noise in the channel may cause some of the data to be corrupted. Noise sources affecting transmission range from atmospheric conditions and electrical equipment, while in storage devices, dust and surface defects cause the main problems.

All of these noise sources may be combated by error correcting codes. The main problem of error control coding is the construction of efficient codes that can correct many errors while at the same time maintaining a reasonable rate of transmission of information. In order to be practically useful, such codes must be equipped with efficient encoding and decoding algorithms. Solutions to these problems depend on algebraic coding theory, which involves the use of tools from modern algebra.

In 1948, Shannon demonstrated that by proper encoding of the information, errors induced by a noisy channel could be reduced to any desired level without sacrificing the rate of information transfer [17,pp.304]. Error control coding provides the means to protect data from errors. Data transferred from one place to the other has to be transferred reliably.

Unfortunately, in many cases the physical link can not guarantee that all bits will be transferred without errors. It is then the responsibility of the error control algorithm to detect those errors and in some cases correct them so upper layers will see an error free link. Two error control strategies have been popular in practice. They are the FEC (Forward Error Correction) strategy, which uses error correction alone, and the ARQ (Automatic Repeat Request) strategy which uses error detection, combined with retransmission of corrupted data.

The ARQ strategy is generally preferred for several reasons. The main reason is that the number of overhead bits needed to implement an error detection scheme is much less than the number of bits needed to correct the same error. The FEC strategy is mainly used in links where retransmission is impossible or impractical.

The FEC strategy is the strategy that will be discussed and is usually implemented in the physical layer and is transparent to upper layers of the protocol. When the FEC strategy is used, the transmitter sends redundant information along with the original bits and the receiver makes its best to find and correct errors. The number of redundant bits in FEC is much larger than in ARQ.

There are currently different image compression techniques available. The two main

categories that differentiate these technologies are lossless and lossy compression. Lossless compression is the ability to reduce image data for storage and transmission while retaining the quality of the original images. The decoded image quality is required to be identical to the image quality prior encoding and it is important to use lossless compression where one needs perfect reconstruction of the source.

Lossy compression is applicable in high image compression and is achieved by degrading the image quality meaning the decompressed image is not quite the same as what you started with. The decoded image quality is reduced compared to the quality of the original images prior encoding.

The main purpose of this work was to investigate the JPEG 2000 encoded image sensitivity at the receiver side after the image has been transmitted through the channel. In order to achieve this goal, more must be learned of coding techniques themselves and this will consider the Convolutional codes, Block codes, Reed Solomon coding with Concatenated codes and interleaving methods in some detail.

Having this knowledge of coding techniques, the rest of this work will be focused on the JPEG 2000 compression method and the sensitivity of this technique applied to satellite images. The document layout follows in section 1.1.

1.1 Thesis Overview

The thesis is structured as follows:

Chapter 1: Introduction

This chapter introduces the channel coding techniques and the data compression techniques, and the theory behind channel coding.

Chapter 2: Error control coding

The chapter discusses channel coding solutions, which defines the purposes of error detection and error correction techniques. The basic purpose of channel coding is to introduce redundancies in the data to improve performance. This introduction of redundant bits increases the raw data rate used hence increases the bandwidth requirements for a fixed source data rate. The theory of each coding technique will be discussed in brief and the description on how each is applicable to the final system will be illustrated in chapter 4.

Chapter 3: Image compression

The specific lossy compression techniques will be discussed in this chapter. These are JPEG and JPEG 2000 respectively. The theory of each method will be discussed and a brief description will be given on the history of the development of JPEG and JPEG 2000 image compression. By the end of chapter 3, the understanding of the lossy compression would be achieved and focus will be shifted to the JPEG-2k compression technique. The objective of this work is to measure the degradation of the image after compression.

Chapter 4: Design

Conventional forward-error-correction (FEC) codes, which reduce the required transmit power for a given BER at the expense of increased bandwidth or a reduced data rate use block or convolutional code designs. Block codes add parity bits to blocks of messages. Convolutional codes map a continuous sequence of information bits onto a continuous

sequence of encoded bits. The performance of the JPEG-2k algorithm is determined.

Chapter 5: Implementation

The designed coding techniques are implemented and the source code with functions discussed in the design chapter is implemented.

Chapter 6: Results

The simulation of the complete FEC, JPEG-2k algorithm is performed and the results found were discussed.

Chapter 7: Conclusion and Recommendations

This chapter concludes the project by summarizing the results and areas for future research is recommended.

Appendices

1.2 Goal

1. To determine the sensitivity of JPEG 2000 encoded image (process) to errors

2. To protect the encoded data and determine the nature of the channel.

The choice of channel coding depends on the type of errors on the channel. It is known that the information sent over the satellite channel, is susceptible to both random and burst errors then information should be encoded in such a way that both types of errors can be corrected. Channel coding protects data by adding redundancy.

3. To determine the requirements of the system?

In a satellite system we cannot afford to retransmit data each time when it is corrupted, because retransmission will be an expensive task to perform. In order to combat errors we have to design a FEC system, which is a one way system that is accomplished by employing error correcting codes that automatically correct errors detected at the receiver

2 Error Control Coding

Information Theory is an area of mathematics dealing with the transfer of information from one location to another, or the storage of information for later retrieval. Both of these operations can be regarded as passing information or data through a channel from a source to a receiver. There are two central aspects of the subject that are:

- Efficiency - efficient encoding of the information (source coding or data compression).
- Reliability - removal of errors caused by *noise* in the channel (error control coding).

Error control coding provides the means to protect data from errors. Data transferred from one place to the other has to be transferred reliably. Unfortunately, in many cases the physical link can not guarantee that all bits will be transferred without errors. It is then the responsibility of the error control algorithm to detect those errors and in some cases correct them so upper layers will see an error free link.

Two error control strategies have been popular in practice. They are the FEC (Forward Error Correction) strategy, which uses error correction alone, and the ARQ (Automatic Repeat Request) strategy, which uses error detection, combined with retransmission of corrupted data. The ARQ strategy is generally preferred for several reasons. The main reason is that the number of overhead bits needed to implement an error detection scheme is much less than the number of bits needed to correct the same error.

2.1 Channel Coding

Channel coding protects digital data from errors by selectively introducing redundancies in the transmitted data. It basically refers to the class of signal transformations designed to improve communications performance by enabling the transmitted signals to better withstand the effect of various channel impairments, such as noise, fading and jamming [1, pp.246]. The main goal of channel coding is to reduce the probability of bit error (P_B) or to reduce the required E_b/N_0 , at the cost of expending more bandwidth than would otherwise be necessary.

No electronic data transmission or storage is perfect. Each system makes errors at a certain rate. As data transfer rates and storage density increase, the raw error rate also increases.

The reasons for having channel coding in the communication system are:

- Very low signal-to-noise ratio for satellite communications due to limited transmit power in downlink
- Channel coding allows exchange of signal power and signal bandwidth without

performance loss (adaptation to transmission conditions)

- Compressed data (e.g. audio and video signals) is very sensitive to transmission errors and should be protected from them

Channel coding can be partitioned into two areas, waveform (or signal design) coding and structured sequence (or structured redundancy). Structured sequences deals with transforming data sequences into better sequences, having structured redundancy. The redundancy bits can then be used for the detection and correction of errors.

2.2 Introduction to Forward Error Correction

In the digital era the need for reliable storage or reliable high-speed transmission of data causes applications of error control coding to become omnipresent. The use of data-compression techniques, increasing the impact of digital errors on the data-recovery process, only contributes to the importance of error-free transmission and or storage.

Forward Error Control is well known for both error detection and error correction in data communications. FEC has the effect of increasing transmission overhead and hence reducing usable bandwidth for the payload data. Therefore, it must be used judiciously in video services that are very demanding in bandwidth but can tolerate a certain degree of loss.

Transmission errors can be classified into two categories: random bit errors and (burst) erasure errors. Random errors are caused by imperfections of physical channels, which

result in bit inversion, bit insertion, and bit deletion. Depending on the coding methods and the affected information content, the impact of random bit errors can range from negligible to objectionable.

The FEC strategy is mainly used in links where retransmission is impossible or impractical. When the FEC strategy is used, the transmitter sends redundant information along with the original bits and the receiver makes its best to find and correct errors. The number of redundant bits in FEC is much larger than in ARQ for good channels.

Forward error correction adds enough redundancy that is transmitted in the code, and errors can be corrected by the receiver without retransmission and provides more bandwidth efficient ways to improve the bit error rate. Later in this section, we briefly describe the major categories of FEC: block coding, cyclic codes, Reed Solomon codes, and convolutional codes.

2.2.1 General Description (What is error correction?)

In 1948, Claude Shannon showed that controlled redundancy in digital communications allows for the existence of communications at arbitrarily low bit error rates (BER) [5, pp.518].

In digital electronic systems, information is represented in binary format (1's and 0's). When binary information is passed from one point to another, there is always some chance that a mistake can be made, a 1 interpreted as a 0 or 0 taken to be a 1. Media can cause this defects, electronic noise, component failures, and poor connection

deterioration due to age and other factors. When a bit is mistakenly interpreted, a bit error has occurred. A group of bits in a computer has conventionally been referred to as a word. Each bit can be thought of as one or two "letters", 0 or 1.

Error control coding (ECC) uses redundancy to detect and correct errors. Error correcting system adds extra or "redundant" letters to computer words. The extra letters (bits) add a certain structure to each word. If that structure is altered by errors, the changes can be detected and corrected. The method of error control coding used depends on the requirements of the system (e.g. data, voice and video) and the nature of the channel.

The main obstacle in error control coding research is to find a way to add redundancy to the channel so that the receiver can fully utilize that redundancy to detect and correct the errors and to improve the coding gain. Codes have varying degrees of efficiency and only a limited few cases actually make use of all the redundancy in the channel.

Different codes also work better in certain conditions like high bit error rates (BER), high throughput, or bursty channels. Part of Shannon's work was discovering the theoretical limit on the lowest signal to noise ratio required to achieve throughput. He also proved that channels have a maximum capacity, C , that can not be surpassed regardless of how good the code is. However, he proved that codes exist such that by keeping the code structure without having to lower the effective information throughput [17, pp.304].

Since adding redundancy to the channel effectively reduces the bandwidth of the channel, there is a strong reason to come as close to the theoretical limit. Redundancy in digital

communications takes the form of parity bits on the information bits. The amount of redundancy is equal to the number of bits transmitted minus the original length of the message or:

$$n - k = \text{parity check bits}$$

Where k = (length of the message, the actual data that the user wants to be sent)

n = (length of the total transmission with redundancy added)

$R = n / k$ = (the code rate) {because $n \geq k$ and $k > 0$, $0 < R \leq 1$ }

Error control coding bases its mathematical foundation on linear algebra. Each code is distinguished by the method used to add redundancy and how much of this redundancy is added to the information going out of the transmitter. Here, the receiver corrects the error without requiring any further information from the sender. This requires a minimum amount of redundancy in the transmission. Not only must an error be detected, however also its location must be determined. For one-bit error correction on an n -bit frame, we must be able to identify one out of the n bit positions if there is an error, or state that there is no error.

2.2.2 What mathematics is involved in the theory of error correction?

Error correcting encoders operate on bits or bytes. From a mathematical point of view, each bit or byte is an element in a finite field, which is an abstract algebraic system with a finite number of elements and two operations and their inverses called division and subtraction.

The main problem of error control coding is the construction of efficient codes that can

correct many errors while at the same time maintaining a reasonable rate of transmission of information. In order to be practically useful, such codes must be equipped with efficient encoding and decoding algorithms. Solutions to these problems depend on algebraic coding theory that involves the use of modern algebra.

2.3 Block Codes

Block codes are FEC codes that enable limited amount of errors to be detected and corrected without retransmission. They can be used to improve the performance of a communication system when other means of improvement are impractical [2, pp.51].

In case of block codes, the source data are segmented into blocks of k data bits, also called information bits or message bits, each block codes represent one of 2^k distinct messages. The encoder transforms each k -bit data block into a larger block of n bits, called code bits or channel symbols. The $(n-k)$ bits, which the encoder adds to each data block, are called redundant bits, parity bits or check bits, and they carry no information.

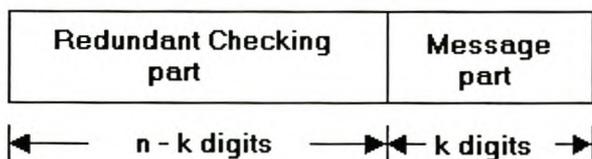


Figure 2.1: Systematic format of the Code word

A desirable property for block code is to possess the systematic structure of the codeword as shown in figure 2.1, where a code word is divided into two parts; the redundant checking part and the message part. As described above the Message part has unaltered information digits, redundant checking part consist of $n-k$ parity check digits which are linear sums of the Message part.

This is referred to as a (n, k) block code and the rate of the code is defined as $R_c = k / n$ and is equal to the rate of information divided by the raw channel rate (the ratio of data bits to total bits is called the code rate). The ratio of redundant bits to data bits $(n-k)/k$, within a block is called the redundancy of the code.

An encoder outputs a unique n -bit code for each of the 2^k possible input k -bit blocks. One would think that as the number of parity bits is increased that it should be possible to correct more and more errors. However, as more and more parity check bits are added, the required transmission bandwidth goes up as well (more bits are packet into the same time slot). In light of the resultant increase in bandwidth, more noise is introduced and the probability of error increases.

As code length is incremented it becomes increasingly more difficult and expensive to design good codes to deal with the increasing probability of error. The minimum distance d between two code words is the number of positions in which the two code words differ. The error correcting and error detecting capabilities of a code are related to the minimum distance between code vectors. Thus the goal is to choose parity bits to correct as many errors as possible (getting as close as possible to the $(d-1)/2$ error correction limit, while keeping the communications efficiency as high as possible.

For error correction with short codes it is common to compare the received vector with a table of possible code words and pick the closest one (least number of bits that differ). Toward the end of this Chapter we will cover Viterbi decoding, which for the case of convolutional codes provides a very efficient algorithm for error correction decoding.

The Block code family is most commonly used for error detection, by modulo-2 summing the individual data bits and appending a parity check bit. Many different types of codes exist and they have different properties (i.e. correcting single errors, burst errors, and providing relatively error-free synchronization). In this section we will concentrate on linear (parity bits are obtained by summing of data bits), systematic (parity bits appear at the end of the code word, behind the original uncoded data bits) block codes. The ability of a block code to correct errors is a function of the code distance.

2.3.1 Properties of Block codes.

Distance of a code - The distance of a code word is the number of elements in which two code words C_i and C_j differ:

$$d(C_i, C_j) = \sum_{w=1}^N C_{i,w} + C_{j,w}$$

2.1

where d is the distance of the code word and with modulo- q , and q is the number of

possible values of C_i and C_j . If the code used is binary, the distance is known as the Hamming distance. The minimum distance d_{\min} is the smallest distance for the given set and is given as:

$$d_{\min} = \min d(C_i, C_j)$$

2.2

Systematic - A systematic code is one in which the parity bits are appended to the end of the information bits.

2.4 Cyclic Codes

Cyclic codes are also block codes, on the other hand the code words are simple lateral shift of one another. This structure enables cyclic codes to correct larger blocks of errors, than what non-cyclic block codes are capable of correcting and specific rules for generating these codes may be set up.

They are attractive for two reasons: encoding and syndrome computation can be implemented easily by using shift registers with feedback connections and because they have considerable inherent algebraic structure. As a result of their structure and ease of implementation they are often used for both error correction and error detection.

2.4.1 Cyclic codes in detail

A cyclic code can be generated by using a generator polynomial $g(X)$ of degree $(n-k)$. The generator polynomial of an (n,k) cyclic code is a factor of x^n+1 and has the general form:

$$g(X) = g_0 + g_1X + \dots + g_{n-k}X^{n-k} \quad 2.3$$

A message polynomial $x(p)$ can also be defined as

$$u(X) = u_0 + u_1X + \dots + u_{k-1}X^{k-1} \quad 2.4$$

where (u_{k-1}, \dots, u_0) represents k information bits. It's not possible to have a non-zero parity bit sequence for all-zero data bits (recall that cyclic codes are also block codes), so the last element in the generator matrix must always be a 1 (in order to avoid this case). This requirement determines the properties of this family of codes.

In order to better understand cyclic codes, let's look at them in polynomial form, as a $(n-1)$ degree polynomial:

$$c(x) = c_1x^{n-1} + c_2x^{n-2} + \dots + c_{n-1}x + c_n \quad 2.5$$

Each power of x is a one bit shift in time. The higher order coefficient c_1 represents the first bit of the code word and so on. In order to obtain other code words one repeatedly multiplies by x $c(x) \bmod (x^n+1)$ (all generator polynomial of a cyclic code is always a divisor of x^n+1).

2.4.2 Encoding

Suppose the message to be encoded is $u = (u_0, u_1, \dots, u_{k-1})$. The corresponding message polynomial is:

$$u(X) = u_0 + u_1X + \dots + u_{k-1} X^{k-1} \quad 2.6$$

Multiplying $u(X)$ by X^{n-k} we obtain a polynomial of degree $n-1$ or less,

$$X^{n-k}u(X) = u_0X^{n-k} + u_1X^{n-k+1} + \dots + u_{k-1} X^{n-1}. \quad 2.7$$

Dividing $X^{n-k}u(X)$ by the generator polynomial $g(X)$, we have

$$X^{n-k}u(X) = a(X)g(X) + b(X) \quad 2.8$$

where $a(X)$ and $b(X)$ are the quotient and the remainder, respectively. Since the degree of $g(X)$ is $n-k$, the degree of $b(X)$ must be $n-k-1$ or less that is:

$$b(X) = b_0 + b_1X + \dots + b_{n-k-1} X^{n-k-1} \quad 2.9$$

To obtain the code polynomial the polynomials $b(X)$ and $X^{n-k}u(X)$ are combined and consists of k unaltered information digits $(u_0, u_1, \dots, u_{k-1})$ followed by $n-k$ parity check digits. The $n-k$ parity check digits are simply the coefficients of the remainder resulting from dividing the message polynomial $X^{n-k}u(X)$ by the generator polynomial $g(X)$.

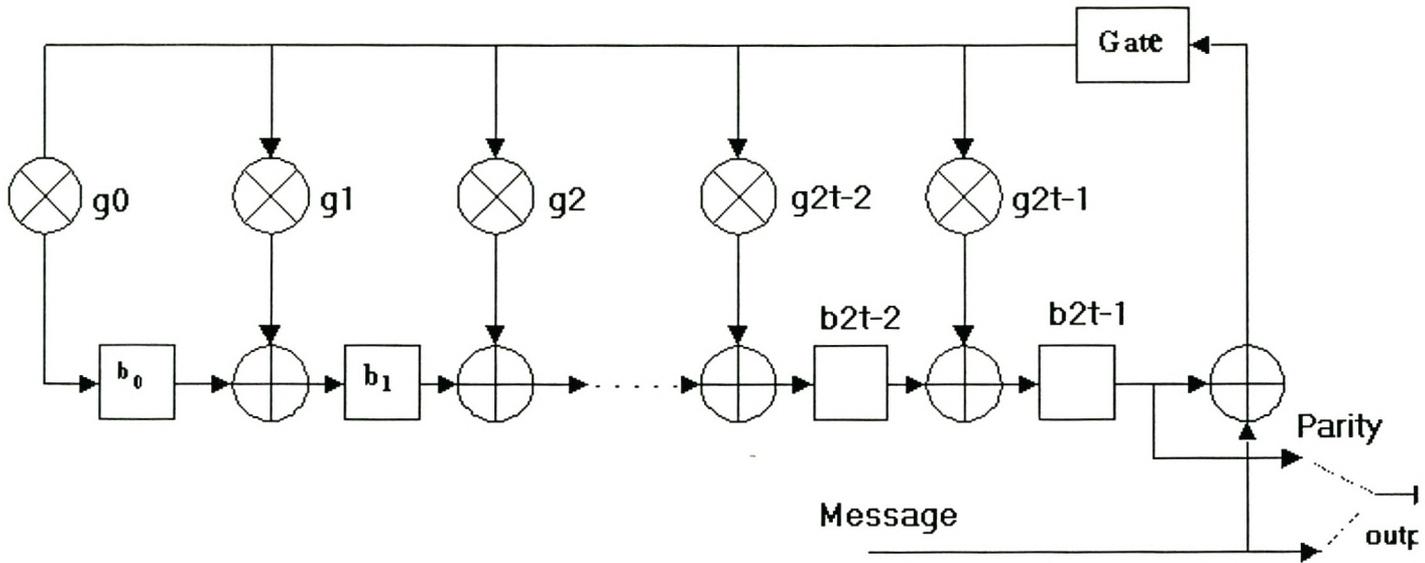


Figure 2.2: The encoding circuit for cyclic codes

In figure 2.2 above the numbered squares represent elements of the shift register.

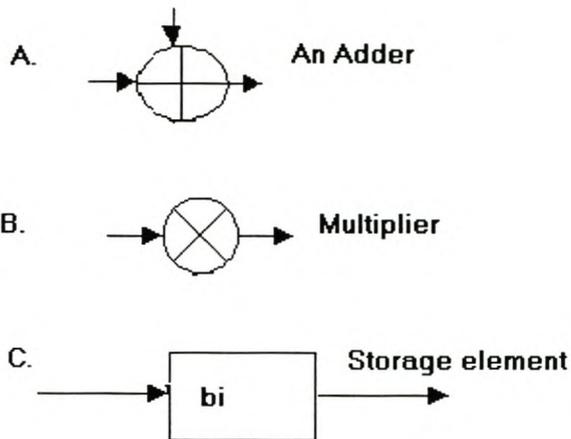


Figure 2.3: Symbols used in the encoder

Symbol A denotes an adder that adds up two elements from $GF(2^m)$. B denotes the multiplier that multiplies a field element from $GF(2^m)$ by a fixed element G_1 from the same field. The symbol C is the storage device that is capable of storing a field element b_i from $GF(2^m)$.

For this particular implementation the number of shift register bits is equal to the number of parity bits. As the k (Message) bits are shifted into the encoder, they are concurrently read out as " k data bits out", since they are the first k bits of the n -bit systematic code word noted as output. After the last data bit has been shifted out, the switch is moved from Message to Parity, and the $(n-k)$ parity bits are read out. The above example can be adapted for any other code, by changing the binary weights (coefficients of the $g(x)$ matrix).

The next section looks at the basic features of one of the categories of channel codes

which is Convolutional coding emphasizing more on the binary convolutional codes and the viterbi decoding technique.

2.5 CONVOLUTIONAL CODING

Convolutional codes are widely used to encode digital data before transmission through noisy or error-prone channels. During encoding, k input bits are mapped to n output bits to give a rate k/n coded bitstream. The encoder consists of a shift register of kL stages, where L is described as the constraint length of the code.

At the receiver, the bitstream can be decoded to recover the original data, correcting errors in the process. The optimum decoding method is maximum-likelihood decoding where the decoder attempts to find the closest "valid" sequence to the received bitstream.

The most popular algorithm for maximum-likelihood decoding is the Viterbi Algorithm. The possible received bit sequences form a "trellis" structure and the Viterbi Algorithm tracks likely paths through the trellis before choosing the most likely path. In this section we will study the basic parameters and properties of these codes and their decoding.

2.5.1 Description

The convolutional codes offer an alternative to the block codes for secure (error-free) data transmission over a noisy channel. Unlike in the case of block codes the redundancy bits added to the message will depend not only on the present information bits, but also

on the bits transmitted earlier. The algebraic presentation of the resulting collection of messages (=code) differs somewhat from that of the block codes. Indeed, a convolutional encoder could be viewed as finite state automation.

Convolutional codes have a structure that effectively extends over the entire transmitted bit streams, rather than being limited to code word block. The convolutional structure is especially well suited to space and satellite communication systems that require simple encoders and achieve high performance by sophisticated decoding methods [2, pp.533].

Convolutional encoding is a technique to add redundancy to the data systematically. The input bits are convolved in such a way that each bit influences the output more than once. Each input bit entering a shift register and the output of the encoder is derived by combining the bits in the shift register in a way determining by the encoder in use.

There are three generic methods for decoding convolutional codes:

1. The Viterbi algorithm based on maximum-likelihood decoding, which achieves optimum performance but requires extensive hardware for computation and storage.
2. Feedback decoding, which sacrifices performance in exchange for simplified hardware.
3. Sequential decoding, which approaches optimum performance to a degree that depends upon the decoder's complexity.

If the decoder uses the Viterbi algorithm based on maximum-likelihood decoding, it must examine an entire received sequence and find a valid path that has the smallest Hamming

distance. As there are 2 to the power of N possible paths for an arbitrary message sequence of N bits, an exhaustive comparison with all valid paths would be an absurd task in the usual case of $N \gg 1$. The Viterbi algorithm applies maximum-likelihood principles to limit the comparison to 2 to the power of kL surviving paths, independent of N , which brings maximum-likelihood decoding into the realm of feasibility.

2.5.2 Binary Convolutional Codes

Block codes such as Hamming codes and Reed-Solomon codes break a message stream up into fixed size blocks and add redundancy symbols to offer error correction capability. They are usually decoded via algebraic methods.

Binary convolutional codes (BCC) take a different approach to error control coding. The message stream is not broken up into blocks with redundancy added to each block independently. Convolutional coding differs from block codes in that the encoder contains memory and the n encoder outputs at any given time unit depend not only on the k inputs but also on the m previous input blocks.

An (n, k, m) convolutional code can be implemented with a k input, n output linear sequential circuit with input memory m . Instead, redundancy is added continuously and is dependent on past bits of the message. This converts the entire message stream into one long code word. One further difference between block codes and convolutional codes is that BCC's are also decoded by a heuristic approach rather than algebraically.

2.5.3 Encoder Structure

In the encoder, message bits are fed into a shift register. The set of delay elements in the shift register can be thought of as a state machine that has 2^d possible states where d is the number of delay elements in the register. Code word bits are generated as functions of the current state and input. These functions are sums of fixed patterns of taps into the shift register.

What is referred to, as a code word here is not the same as in block coding. In block coding, each block is independent of all other blocks, this is not so for BCC's. Each code word is dependent on the current message word and the state of the register which stores information about the past values of the message bits. Therefore successive code words are not independent of one another.

The state naming convention used is that the first delay element that the message is fed into is the least significant bit of the shift register whose contents represent the state. The code words in BCC's are the collection of code bits that form the output of the encoder circuit for one period of the shift register. The rate of a BCC is computed the same way as with block codes.

Instead of representing the rate in the form (n, k, d_{min}) , the rate of a BCC is written k/n . The minimum distance of a block code is the free distance, which is defined as the number of state changes that give a minimum weight code word sequence.

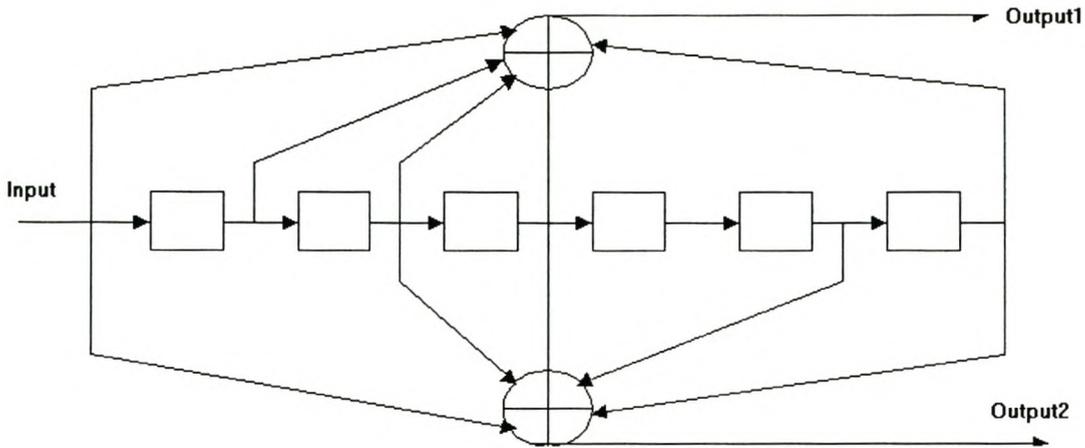


Figure 2.4: The Feedforward Convolutional encoder

The fundamental hardware unit for a convolutional encoder is a tapped delay line or a shift register with K stages [1, pp.318]. The message bits in the register are combined by modulo-2 addition to form the encoded bit. A particular message bit influences a span of K successive encoded bits as it is shifted through the register.

Convolutional codes operate on a sliding sequence of data bits in order to generate the coded stream of bits. Convolutional codes can also operate on symbol groups and binary convolutional coding.

In figure 2.4 the convolutional coder consists of a K -stage shift register, where K is also the constraint length of the coder. Input data bits are shifted into a shift register one bit at

a time. Alternating modulo-2 sums are shifted out v times as fast. (where v equals the number of different sums that are coded into the output stream). A coder putting out v bits/input bit is also called a rate $1/v$ coder. The shift register is initialized with all 0's. The convolutional coder may be visualized as a finite state machine, moving from one stage, represented by $K-1$ shift register contents (past input bits), to another:

2.5.4 Viterbi algorithm

The function of the decoder is to estimate the encoded input information using a rule or method that results in the minimum possible number of errors. There are a number of techniques for decoding convolutional codes. The most popular technique to find the metrics and at the same time minimizing the number of paths is the Viterbi decoding algorithm which performs maximum likelihood decoding (MLD) of convolutional codes. The viterbi algorithm performs MLD however it reduces the computational load by taking advantage of the special structure in the code trellis.

Its decode complexity is not a function of the number of symbols in the code word sequence as compared to the brute-force decoding. It is well known that a minimal trellis, which can be used for maximum likelihood soft decision decoding by the Viterbi algorithm, can represent every linear block code.

In practice, the maximum likelihood algorithm based on the Viterbi method can be applied only to codes with small redundancy or a small number of code words. The main idea is in sorting all possible error patterns in respect to their Euclidean distance from the

received sequence. Starting from the closest one, we test them, one by one, successively increasing the distance from the received sequence, until the first code word is found.

The error patterns are organized into trellises. The decoder searches the trellis until it finds a code word. This results in a soft maximum likelihood decoding algorithm with computational complexity reduced compared to the known maximum likelihood decoding methods. The proposed MLD algorithm is subsequently modified to further simplify computations. As a result a sub-optimum algorithm is derived with only a slight sacrifice in the error performance.

The algorithm involves calculating a measure of similarity, or distance, between the received signal, at time t ; and all the trellis paths entering each state at time t .

The viterbi algorithm removes from consideration those trellis paths that could not possibly be candidates for the maximum likelihood choice.

When two paths enter the same state, the one having the best metric is chosen; this path is called the surviving path. The selection of the surviving paths is performed for all the states. The decoder continues in this way to advance deeper into the trellis, making decisions by eliminating the least likely paths. The early rejection of the unlikely paths reduces the decoding complexity. The goal of selecting the optimum path can be expressed, equivalently as choosing the code word with the maximum likelihood metric, or as choosing the code word with the minimum distance metric.

Throughout this chapter an assumption that the channel is memory less was made, since codes were designed to combat random independent errors. In the next section section the assumption that channel has memory will be adopted and the errors will no longer be characterized as single randomly distributed bit errors.

2.6 Interleaving

Interleaving is used to obtain time diversity in a digital communications system without adding overhead. By spreading the source bits over time, it becomes possible to make use of error control, which protects the source data from corruption by the channel. Since error control codes are designed to protect against channel errors that may occur randomly or in a bursty manner, interleavers scramble the time order of source bits before they are channel coded.

Interleaving the coded message before transmission and deinterleaving after reception causes bursts of channel errors to be spread out in time and thus to be handled by the decoder as if they were random errors. The channel memory decreases with time separation, the idea behind interleaving is to separate the code word symbols in time.

Interleaving shuffles the code symbols over span of several block lengths for block codes or several constraint lengths for convolutional codes. The span is determined by the burst duration expected on the channel. The details of the bit redistribution pattern must be known to the receiver in order for the symbol stream to be deinterleaved before being decoded.

2.6.1 Block interleaving

A block interleaver accepts the coded symbols in blocks from the encoder, permutes the symbols, and then feeds the rearranged symbols to the receiver/ modulator. The usual permutation of the block is accomplished by filling the columns of an M -row by N -column ($M \times N$) array with the encoded sequence.

After the array is completely filled, the symbols are then fed to the modulator/receiver one row at a time and transmitted over the channel. At the receiver, the deinterleaver performs the inverse operation, it accepts the symbols from the demodulator, deinterleaves them, and feeds them to the decoder. Symbols are entered into the deinterleaver array by rows and removed by columns.

2.6.2 Convolutional interleaving

The code symbols are sequentially shifted into the bank N registers; each successive register provides J symbols more storage than the preceding one. The zeroth register provides no storage (the symbol is transmitted immediately). With each new code symbol the commutator switches to a new register, and the new code symbol is shifted in while the oldest code symbol in that register, the commutator returns to the zeroth register and starts again.

The de-interleaver performs the inverse operation, and the input and the output commutators for both the interleaving and de-interleaving must be synchronized. The most important advantage of a convolutional over a block interleaver is that with convolutional interleaving the end to end delay is $M(N-1)$ symbols, where $M=NJ$, and the memory required is $M(N-1)/2$ at both ends of the channel. Therefore there is a reduction of one-half in delay and memory over the block interleaving requirements [1, pp.360].

2.7 Reed Solomon Codes

Reed Solomon (RS) codes are a subset of BCH (Bose-Chaudhuri-Hocquenghem) cyclic codes which are designed to provide multiple-error correction. RS codes are a class of non-binary, multisymbol codes defined over finite fields. This code is actually one subclass of nonbinary BCH codes. The theory for BCH codes is summarized in Appendix A.2.

RS codes with code symbols from the Galois field $GF(2^m)$, have an m -bit symbol which means there are 2^m possible symbols (code words) where $m \geq 3$. An (n, k) Reed Solomon code is also a block code, in which k data symbols are inserted into the coder and n code word symbols appear on the output.

The total number of code symbols in the encoded block is $n = 2^m - 1$. The number of parity symbols that must be used to correct errors is $n - k = 2T$ with k being the number of data symbols to be encoded and T being the error correcting capability. The minimum

distance:

$$d_{\min} = n - k + 1$$

2. 10

and the error correcting capability:

$$e_{\text{cc}} = \frac{z(\beta_i^{-1})}{\prod_{i=1, i \neq l}^v (1 + \beta_i \beta_i^{-1})}$$

2. 11

The code is capable of correcting T or fewer errors. RS codes achieve the largest possible minimum distance (d_{\min}) of any linear code. The coding/decoding ideas of RS codes are almost same as that of BCH codes. The difference is that operations are more complicated since RS codes are nonbinary.

Binary codes derived from RS's codes are effective against burst errors, which are prevalent on many real channels. For this reason RS codes are widely used for error control with applications including deep space telecommunication systems, compact disks, digital broadcasting and frequency hopping spread spectrum systems. The most popular decoding method in current applications relies on algebraic hard decisions for each received code symbol.

The additional information provided by soft decision can enable between 2 and 3dB coding gain of Gaussian and in excess of 10dB on Rayleigh fading channels [1, pp.330]. Further improvements in the error performance can be obtained by making use of the reliability information of the symbols within each code word.

2.7.1 Encoding

Let α be a primitive element in $GF(2^m)$. The generator polynomial of a t -error correcting RS code of length $2^m - 1$ is:

$$\begin{aligned} g(X) &= (X + \alpha)(X + \alpha^2)\dots(x + \alpha^{2t}) \\ &= g_0 + g_1X + g_2X^2 + \dots + g_{2t-1}X^{2t-1} + X^{2t} \end{aligned} \quad \mathbf{2.12}$$

$g(X)$ has $\alpha, \alpha^2, \dots, \alpha^{2t}$ as its roots and has coefficients from $GF(2^m)$. The code generated by $g(X)$ is an $(n, n - 2t)$ cyclic code which consists of those polynomial of degree $n-1$ or less with coefficients from $GF(2^m)$ that are multiples of $g(X)$. The encoding is similar to the encoding for binary cyclic codes as in section 2.4.2. More information on encoding the RS codes is provided in [2, pp.171][2, pp.95].

In hardware implementation a division circuit as illustrated in figure 2.2 is used as the encoder. As soon as the message has entered the channel and the circuit, the parity-check digits are in the register.

2.7.2 Decoding

How can a receiver know if one or more errors have occurred during the transmission of an encoded block? The first step below which is the computation of syndromes gives an indication of whether the received vector is erroneous. To decode the received vector $r(X)$, the following four steps are required.

2.7.2.1 Syndrome Computations

The first step in decoding a T-error correction Reed Solomon code is to compute the $2T$ syndrome components S_1, S_2, \dots, S_{2T} . These syndrome components may be obtained by substituting the field elements $\alpha, \alpha^2, \dots, \alpha^{2T}$ into the received polynomial $r(X)$. If one of the $2T$ components or more are not equal to zero, then one or more symbol errors have occurred in the received block.

2.7.2.2 Find the Error-Location polynomial $\sigma(X)$

At this step we have knowledge as to whether or not there are errors in the received block. To have an indication of how many symbols are affected or the locations of errors, the error location polynomial $\sigma(X)$ is determined from the syndrome components obtained above using the Berlekamp's iterative algorithm. Proof of the Berlekamp algorithm is in [3 cited in 2, pp.155].

The first step of the iteration is to find a minimum degree polynomial $\sigma^{(1)}(X)$ whose coefficients satisfy the first Newton identities [2, pp.154]. The next step is to test whether the coefficients of $\sigma^{(1)}(X)$ also satisfy the second Newton's identity. If the coefficients of $\sigma^{(1)}(X)$ do satisfy the second Newton's identity then $\sigma^{(2)}(X) = \sigma^{(1)}(X)$. If the coefficients of $\sigma^{(1)}(X)$ do not satisfy the second Newton's identity a correction term is added to $\sigma^{(1)}(X)$ to form $\sigma^{(2)}(X)$ such that $\sigma^{(2)}(X)$ has minimum degree and its coefficients satisfy the first two Newton's identities.

The process continues, for every next n th step of iteration to find a minimum degree polynomial $\alpha^{(n)}(X)$ from $\alpha^{(n-1)}(X)$ such that the coefficients of $\alpha^{(n)}(X)$ satisfy the first n Newton's identities. A test is performed whether the coefficients of $\alpha^{(n-1)}(X)$ satisfy the n th Newton's identity. If they do, $\alpha^{(n)}(X) = \alpha^{(n-1)}(X)$ and if they do not a correction term is added to $\alpha^{(n-1)}(X)$ to form $\alpha^{(n)}(X)$. Iteration continues until $\alpha^{(2T)}(X)$ is obtained, then $\alpha^{(2T)}(X)$ is taken to be the error location polynomial $\sigma(X)$.

If the number of errors in the received polynomial $r(X)$ is T or less, then $\sigma(X)$ is the error location polynomial with correctable errors. If $\sigma(X)$ has degree greater than t , there are more than t errors in the received polynomial.

2.7.2.3 Finding the Error-Location Numbers

The inverses of the roots of the error-location polynomial $\sigma(X)$ yield the error location numbers. The roots are found by substituting $1, \alpha, \alpha^2, \dots, \alpha^{n-1}$ into $\sigma(X)$ and if the computation result after substituting a particular is zero then that value is the root for $\sigma(X)$.

2.7.2.4 Error values and Error Correction

Once $\sigma(X)$ is found, the error values can be determined. Let

$$Z(X) = 1 + (S_1 + \sigma_1)X + (S_2 + \sigma_1 S_1 + \sigma_2)X^2 + \dots \\ + (S_v + \sigma_1 S_{v-1} + \sigma_2 S_{v-2} \dots + \sigma_v)X^v$$

Then the error values at locations $\beta_i = \alpha^{il}$ (which is the error location number) is given by [2, pp.175]

$$e_{jl} = \frac{Z(\beta_l^{-1})}{\prod_{i=1, i \neq l}^v (1 + \beta_i \beta_l^{-1})}$$

2. 14

The error pattern is the error values obtained at each error location. Adding (modulo-2) the error pattern to the received vector completes the decoding of the code.

2.8 Concatenated Codes

A concatenated code is one that uses two levels of coding, an inner code and an outer code, to achieve the desired performance. The inner code is the one that interfaces with the channel and is usually a higher rate (lower redundancy) code, then reduces the probability of error to the specified level. Reason for using a concatenated code is to achieve a low error rate with an overall implementation complexity, which is less than that which would be required by a single coding operation.

An interleaver between the two coding steps is usually required to spread any error bursts that may appear at the output of the inner coding operation. Due to cyclic nature of the codes they can be shortened to fit any application.

2.9 Conclusion

In this chapter an overview of different error correcting codes was given. This chapter was made available to establish various capabilities of coding techniques, and to assist in making a decision before designing the FEC system depending on the application requirements.

In summary, this chapter discusses the basis of binary and nonbinary cyclic codes and an advantage of nonbinary codes can be illustrated by an example below. Consider a binary $(n,k) = (15,7)$ code. The entire n -tuple space amounts to $2^n = 2^{15} = 32768$ binary words, of which $2^k = 2^7 = 128$ (or $1/256$ of the n -tuple space) are code words. Next, consider a nonbinary $(n,k) = (15,7)$ code where each symbol is comprised of $m = 4$ bits.

The n -tuple space amounts to $2^{nm} = 2^{60} = 1,1529 \times 10^{18}$ binary words, of which $2^{km} = 2^{28} = 268435456$ (or $1/4294967296$ of the n -tuple space) are code words. With symbols, each made up of m bits, only a small fraction (i.e. 2^{km} of the large number 2^{nm}) of possible different words of n symbols become code words. This fraction decreases with increasing values of m . When a small fraction of the n -tuple is used for code words, a large d_{\min} can be created.

The Reed Solomon codes achieve the largest possible code minimum distance for any linear code with the same encoder input and output block lengths. They also have the capability to correct both random and burst errors.

The basic idea of convolutional coding is that when data is sent, the encoder adds coding bits to each data bit to be transmitted. The encoding rate is normally k/n ; i.e. each transmitted bit is encoded to n bits and k is the number of input bits. The transmitted bit is dependent on bits sent earlier.

- Convolutional codes work on small blocks of data, thus the coding delay is small and Block codes use bigger data blocks, therefore their coding delay can be large.
- Convolutional encoding is simple to implement using shift registers.
- The Viterbi algorithm provides a computationally efficient means of decoding convolutional codes
- Implementation of convolutional codes requires less memory than that of block codes

3 Image Compression

3.1 Introduction

In this chapter the theory of image compression will be looked at and more emphasis will be on the two lossy compression techniques JPEG and JPEG-2k.

Image compression is defined as *the science and art of processing information so as to obtain a simple representation with at most tolerable loss of fidelity. Such data reduction may be required due to storage constraints, limited bandwidth or the desire to "sift" data for important attributes [7,pp.189]*. Image compression has the ability to reduce data volume and achieve reproduction of the original data without any perceived loss in data quality.

Each compression technique is a collection of mathematical and processing routines that manipulate the content of an image. In some cases the routines are similar, and in other cases the routines are quite different; in each case, the objective is identical: reduce the

amount of data required to represent the image. In order to explain this process, the primary steps involved in compression will be described.

Compression can be lossless or lossy. The theoretical possibility of image compression, no matter lossless or lossy, is primarily based on the redundancy inside the image. There are currently different methods for compressing image data, each with its own qualities and disadvantages. Currently popular compression techniques include the JPEG and wavelet compression techniques. Compression is lossless when it produces an exact copy of the original image with the related bounds on compression ratio.

Lossy image compression is performed when a specific level of degradation is allowed, and this in turn will influence the compression ratio. It is lossy when the data compressed could not be restored during the decompression process. Lossy compression can impact the visual quality of the image. The compression ratio or the degree of impact of lossy compression on an image can range from minimal to extreme depending on the amount of information discarded.

3.2 JPEG

This is the very well known ISO/ITU-T standard created in the late 1980s. There are several modes defined for JPEG, including baseline, lossless, progressive and hierarchical. The baseline mode is the most popular one and supports lossy coding only. JPEG is based on the discrete cosine transform. In the baseline mode, the image is divided in 8x8 blocks and each of these is transformed with the DCT.

The transformed blocks are quantized with a uniform scalar quantizer, zigzag scanned and entropy coded with Huffman coding. The quantization step size for each of the 64 DCT coefficients is specified in a quantization table, which remains the same for all blocks. The DC coefficients of all blocks are coded separately, using a predictive scheme. Hereafter we refer to this mode simply as JPEG. The lossless mode is based on a completely different algorithm, which uses a predictive scheme. The prediction is based on the nearest three causal neighbours and seven different predictors are defined (the same one is used for all samples).

The prediction error is entropy coded with Huffman coding. It is sometimes referred to as L-JPEG. The progressive and hierarchical modes of JPEG are both lossy and differ only in the way the DCT coefficients are coded or computed, respectively, when compared to the baseline mode. They allow a reconstruction of a lower quality or lower resolution version of the image, respectively, by partial decoding of the compressed bitstream.

Progressive mode encodes the quantized coefficients by a mixture of spectral selection and successive approximation, while hierarchical mode uses a pyramidal approach to computing the DCT coefficients in a multi-resolution way.

3.3 JPEG 2000

3.3.1 Introduction

With the continual expansion of multimedia applications, the needs and requirements of

the technologies used in multimedia grow and evolve. Many compression technologies and associated file formats have been investigated and developed within the last decade, on the other hand the most successful technique has without doubt been the JPEG standard. The JPEG committee decided to anticipate the needs and requirements for the second generation imagery applications by defining the need for the new standard JPEG2000.

JPEG2000 is not only intended to provide rate-distortion and subjective image quality performance superior to existing standards [8 cited in 4, pp. 2] but will also provide functionality that the existing compression techniques can either not address efficiently or not address at all.

JPEG2000 uses a wavelet-based coding system, based mainly on Embedded Block-based Coding with Optimized Truncation [14, pp.344-348][15, pp.1158-1170]. It is based on the discrete wavelet transform (DWT), scalar quantization, context modeling, arithmetic coding and post-compression rate allocation.

This means that the first step in the algorithm is to decompose the input image into a set of sub-bands via a wavelet transform. Each sub-band is divided into rectangular blocks (called code-blocks in JPEG 2000), typically 64x64, and entropy coded using context modeling and bit-plane arithmetic coding. The coded data is organized in so called layers, which are quality levels, using the post-compression rate allocation and output to the code-stream in packets. The generated code-stream is parseable and can be resolution, layer (i.e. SNR), position or component progressive, or any combination thereof.

Each sub-band will then contain different frequency components of the information in the original image with the appropriate subsampling. The most common decomposition (i.e. dyadic) is performed by recursively applying a low-pass filter and a high-pass filter to the image in both spatial directions, although other decompositions are also possible. The wavelet transform used can be either a floating- or a fixed-point wavelet, which implies lossy coding due to limited precision, or a reversible integer wavelet, which enables lossless coding.

The wavelet coefficients are explicitly quantized if needed and the quantizer indices, or the integer wavelet coefficients, are losslessly entropy encoded using arithmetic coding of its bitplanes. The fact that the subbands are encoded bitplane by bitplane makes it possible to select regions of the image that will precede the rest of the image in the codestream. Each subband is independently encoded, starting at the most significant bitplane in the subband down to the less significant one.

One of the requirements in JPEG2000 is the support of Region of Interest (ROI) coding, where ROI of the image can be coded with better quality than the background (BG). The ROI coding mode supported in JPEG2000 is based on scaling the wavelet coefficients and the general ideas are presented in [10, pp.856-860][11 cited in 6, pp.5]. The principle of a scaling based method is to scale up (shift up) coefficients so that the bits associated with the ROI are placed in higher bit-planes.

Then, during the embedded coding process, those bits are placed in the bit-stream before the non-ROI parts of the image (depending on the scaling value, some bits of ROI coefficients might be encoded together with non-ROI coefficients). Thus, the ROI will

be decoded, or refined, before the rest of the image.

Regardless of the scaling, a full decoding of the bit-stream results in a reconstruction of the whole image with the highest fidelity available. If the bit-stream is truncated, or the encoding process is terminated before the whole image is fully encoded, the ROI will have a higher fidelity than the rest of the image. In the JPEG2000 standard the scaling-based method is implemented as follows:

1. Calculate the wavelet transform
2. If a ROI is chosen, derive a mask (ROI mask) indicating the set of coefficients that are required for up to lossless ROI reconstruction (see section 3.3.2).
3. Quantize the wavelet coefficients
4. Downscale the coefficients outside the ROI mask by a specific scaling value.
5. Entropy encode the resulting coefficients progressively with the most significant bitplanes first.

The scaling value assigned to the ROI and the coordinates of the ROI are added to the bit-stream. If the scaling value were chosen arbitrarily, the shape of the ROI would also be needed in the bitstream. The decoder would also perform the ROI mask generation and scale up the downsampled coefficients.

The sub-band samples are scaled so that the bitplanes encoded first contains only the ROI information and the following bitplanes only contain background information. The only thing the decoder needs to receive is the factor by which the samples were scaled. The decoder can then invert the scaling based only on the amplitude of the samples. Other supported functionalities are error-resilience, random access, multicomponent images,

palletized color, compressed domain lossless flipping and simple rotation, to mention a few.

3.3.2 ROI

In many applications where digital images are involved (medical, remote sensing, etc.), there exists within the image, one or more regions of greater interest than the rest. By allowing the background (BG) to be coded with lower fidelity than the regions of interest (ROI's), significant gains can be achieved in terms of compression and hence in storage space and transmission times [10, pp.856-860][12, pp.87-91].

ROI means that a part of the image (the ROI) is coded with better quality when compared to the background. ROI coding can be done in two ways: either the image is stored with an ROI or the user can pick the ROI during transmission [11 cited in 6, pp.5] and only receive bits for that part of the image.

Region of interest coding happens in two steps: the wavelet coefficients that affect the ROIs must be identified, and then coded with a higher quality by some special means. For the first step, an ROI mask is generated, which is explained below. The approach to use in the second step depends on the application. One possibility is to separately encode ROI and BG coefficients in each subband, using a coarser quantization for the BG coefficients [8 cited in 4, pp.2].

Another approach [8 cited in 4, pp.3][12, pp.87-91] is to shift the quantization indexes, or integer wavelet coefficients, in the ROIs upwards in bitplanes so that they are coded first.

In the above methods, the ROIs must be defined before the encoding process and cannot be changed afterwards (unless a complete decoding and re-encoding is performed). A third method allows for the definition of new ROIs during the encoding process, or even during a progressive transmission (by using a transcoder), as explained further in [6, pp.389-391].

3.3.3 ROI mask creation

When an image is coded with an ROI, it should be possible to reconstruct the entire ROI with better fidelity (at a higher bitrate). It is therefore necessary to identify the wavelet coefficients needed for the reconstruction of the ROI's, so that they can be coded at a higher quality. For this purpose an ROI mask is created. This mask indicates which wavelet coefficients belong to an ROI and to which one if multiple ROI's are defined.

In order to reconstruct a pixel inside ROI at a higher quality, all the wavelet coefficients that contribute through the inverse wavelet transform are needed at a higher quality, and thus become part of the ROI mask. ROI masks are losslessly encoded, then the ROI can be losslessly reconstructed. The way to handle the coefficients of the ROI's depends on the application, but most common solutions are in [6, pp.4].

3.4 Architecture of the JPEG 2000

The block diagram of the JPEG 2000 encoder is demonstrated below in figure 3.1. The discrete wavelet transform is first applied on the source image data. The transform

coefficients are then quantized and entropy coded, before forming the output codestream (bitstream).

The decoder is the reverse of the encoder (Fig. 3.1BB). The codestream is first entropy decoded, dequantized and inverse discrete transformed, thus resulting in the reconstructed image data the decoder and sent from the source.

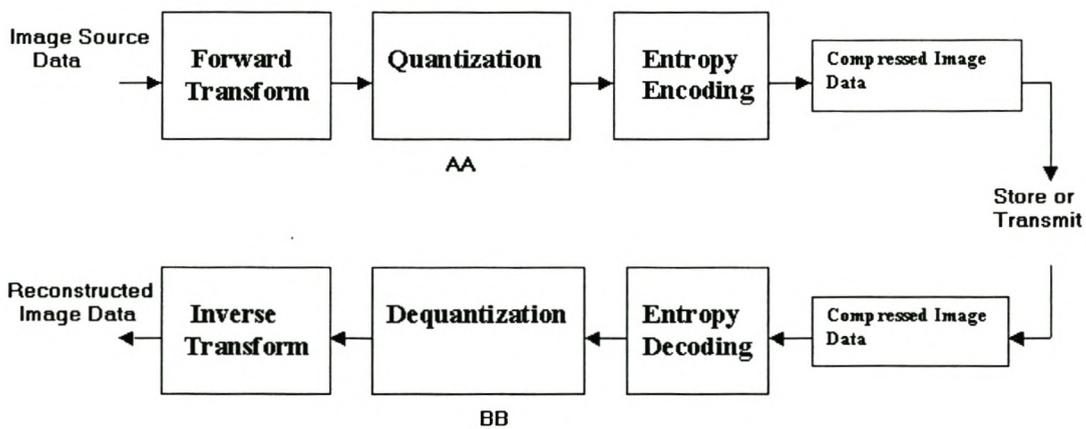


Figure 3.1: Block Diagrams of the JPEG 2000 (AA) Encoder (BB) Decoder

The standard works on image tiles and the term ‘tiling’ refers to the partition of the original (source) image into rectangular non-overlapping blocks (tiles), which are compressed independently, as though they were entirely distinct images. Prior to computation of the forward discrete wavelet transform (DWT) on each image tile, all samples of the image tile component are DC level shifted by subtracting the same quantity (i.e. the component depth).

DC level shifting is performed on samples of components that are unsigned only. If color transformation is used, it is performed prior to computation of the forward component. Otherwise it is performed prior to the wavelet transform. At the decoder side, inverse DC level shifting is performed on reconstructed samples of components that are unsigned only. If used, it is performed after the computation of the inverse component transform. Arithmetic coding is used in the last part of the encoding process. The multiple quantization MQ coder is adopted by the encoder, this coder is basically similar to the QM adopted in the original JPEG standard [8 cited in 4, pp.2].

The description details of each block in the encoding procedure is as follows:

- ◆ The source image is decomposed into wavelet components.
- ◆ The image and its components are decomposed into rectangular tiles. The tile-component is the basic unit of the original or reconstructed image
- ◆ The wavelet transform is applied on each tile. The tile is decomposed in different resolution levels.
- ◆ The decomposition levels are made up of subbands of coefficients that describe the frequency characteristics of local areas (rather than across the entire tile-component) of the tile-component.

- ◆ The subbands of coefficients are quantized and collected into rectangular arrays of “code-blocks”.
- ◆ The bit-planes of the coefficients in a “code-block” are entropy coded.
- ◆ The optional file format describes the meaning of the image and its components in the context of the application.
- ◆ The encoding can be done in such a way that certain ROI’s can be coded in a higher quality than the background.
- ◆ Markers are added in the bitstream to allow error resilience.
- ◆ The codestream has a main header at the beginning that describes the original image and the various decomposition and coding styles that are used to locate, extract, decode and reconstruct the image with the desired resolution, fidelity, region of interest and other characteristics.

3.5 *Application Requirements Features*

The JPEG2000 standard provides a set of features that are of importance to many high end and emerging applications by taking advantage of new technologies. It addresses areas where current standards fail to produce the best quality or performance and provides capabilities to markets that currently do not use compression.

The markets and applications better served by the JPEG2000 standard are Internet, color facsimile, printing, scanning (consumer and pre-press), digital photography, remote sensing, mobile, medical imagery, digital libraries / archives and E-commerce.

Each application area imposes some requirements that the standard should fulfil. Some of the most important features that this standard should possess are the described in [16, pp.1103-1127].

In the next sub paragraph comparison methodology will be described which emphasizes on the compression efficiency.

3.6 Comparison Methodology

Although one of the major, and often the only, concern in coding techniques has been that of compression efficiency, it is not the only factor that determines the choice of a particular algorithm for an application. Most applications require features in a coding algorithm other than simple compression efficiency. This is often referred to as functionalities.

Examples of such functionalities are ability to distribute quality in a non-uniform fashion across the image (e.g., ROI), or resiliency to residual transmission errors that occur in mobile channels. The next section reports on compression efficiency since it is still one of the top priorities in many imaging products.

3.6.1 Compression efficiency

Compression efficiency is measured for lossless and lossy compression. For lossless coding it is simply measured by the achieved compression ratio for each one of the test

images. For lossy coding the root mean square error (RMSE) is used, as well as the corresponding peak signal to noise ratio (PSNR), defined as

$$PSNR = -20 \log_{10} \frac{RMSE}{2^b - 1}$$

3.1

where b is the bit depth of the original image.

Although RMSE and PSNR are known to not always faithfully represent visual quality, it is the only established, well-known, objective measure that works reasonably well across a wide range of compression ratios. For images encoded with a Region of Interest (ROI) the RMSE, as well as the corresponding PSNR, are calculated both for the ROI and for the entire image.

At higher error rates the reconstructed image quality in JPEG 2000 is almost constant across all bitrates. This is due to the fact that in JPEG 2000 each sub-band block is coded by bitplanes. When the error rate is high enough almost all blocks are affected in the most significant bitplanes, which are transmitted first. When a particular bitplane is affected in a block, lower bitplanes can not be decoded and are therefore useless.

In JPEG the higher the encoding bitrate the lower the decoded quality. This can be explained by the fact that when an 8 by 8 block is affected by a transmission error the entire block is basically lost. The higher the encoding bitrate, the more the bits it takes to code a block, and therefore the probability of a block being hit by an error and lost is

higher, for the same bit error rate. The density of error protection decreases with an increase in bitrate.

3.7 Conclusion

JPEG2000 is the new standard for still image compression that is going to be in use mainstream by the beginning of next year. It provides a new framework and an integrated toolbox to better address increasing needs for compression. It also provides a wide range of functionalities for still image applications, like Internet, color facsimile, printing, scanning, digital photography, remote sensing, mobile applications, medical imagery, digital library and E-commerce.

Lossy coding, embedded lossy to lossless, progressive by resolution and quality, high compression efficiency, error resilience and lossy color transformations are some of its characteristics. Comparative results have shown that JPEG2000 is indeed superior to existing still image compression standards [16, pp.1103-1127]. Work is still needed in optimizing its implementation performance.

JPEG-2k is intended for low bit rate applications, exhibiting rate-distortion and subjective image quality performance superior to existing standards. In conclusion, JPEG-2k provides high compression efficiency, superior image quality performance and error resilience, which are some of the features of this compression algorithm. Therefore the JPEG-2k algorithm was chosen as the compression strategy to be implemented.

4 Design and Design Considerations

In order to develop software to simulate a system, it is imperative to understand the structure and operation of the system. Due to the unexpected disappearance of Sunsat, the nature and behaviour of the channel is not known. But most satellite channels are susceptible to both random and burst errors and the information should be encoded in such a way that both types of errors can be corrected.

FEC use the redundancy to detect and correct errors caused by noise on the channel. During transmission it is very likely that the transmitted data is altered by noise from the channel. Therefore on the receiver side it is very likely too that the received data are not the same as the transmitted. The RS codes are capable of correcting both random and burst errors. The RS codes are also attractive for two reasons: encoding and syndrome computation can be implemented easily by using shift registers with feedback connections and because they have considerable inherent algebraic structure.

If the channel behaviour was known then it was going to assist us in deciding which

coding techniques to design and implement. Due to the reasons above, the case study channel was chosen as the DSTV (Digital Satellite Television). The channel uses the *de facto* standards for the satellite communications channel specifications (7,1/2) and (204,188) RS for error correction and detection.

The Convolutional encoder and the Viterbi decoding algorithm are chosen due to reasons mentioned in section 2.9. The type of convolutional encoder chosen is of $K=7$ and code rate $1/2$ as stated above. Appendix A.3 contains a summarized description of convolutional coding. The RS code (204,188) and code rate of $1/2$ for convolutional coding with the constraint length of $K=7$ were chosen.

Part of the project was to determine how vulnerable the JPEG 2000 compression encoded images will be to errors and to be able to know, with the protection strategy that is available, how much of the data can be protected.

4.1 JPEG 2000 Encoding and Decoding

An Implementation of the JPEG 2000 Standard in Java(TM) Version 3.2.2 was used because the standard was still in the formulation-conclusion process. JJ2000 release implements all JPEG 2000 part I features at the decoder side and almost all features at the encoder side, although JJ2000's decoder implements all JPEG 2000 part-I features, one functionality was still missing in the encoder: Packed Packet Headers. The part II technology have been removed and the source code has been reorganized according to the description given in the Final Committee Draft version 1.0[4, pp.2];

The Java language has been chosen to provide an implementation of the decoder of the JPEG2000 Verification Model. The most important advantage for using Java is that of platform independence making the decoder usable on a wide variety of architectures including handheld devices, embedded devices, computers, etc. "It is a dynamic language where new functionality can be incorporated on the fly". Java is also Web enabled, making it easy to provide JPEG2000 on the Web, where it is likely to play a major role [4, pp.2]. The main disadvantage for Java is its execution speed.

Supported image file formats are PGM, PGX, and PPM, Colour images can be used either by using a PPM file or using Red, Green and Blue components of PGM or PGX files.

4.1.1 JPEG-2k Performance

In the simulation, images are transmitted over a noisy channel at various probabilities of error and the compression bitrate per image is constant. Different options to simulate transmission errors shows that the nature of different types of image distortions has varying degrees of effect to the image. The options are described below:

Option1: The input source image is JPEG-encoded and the locations where errors should be introduced are randomly created. For each location created, the new value is randomly created which replaces the original value. This is indicated in the Comparejpg class file.

Option2: The input source image is JPEG-encoded and the locations where errors should be introduced are randomly created. For each location created, the existing value at the particular location is converted to binary format and one bit is flipped and the bits are converted back to the original format. Then a new value has been created at that location and this is indicated in the `Comparejpg1` class file.

Option3: The input source image is JPEG-encoded and the locations where errors should be introduced are randomly created. For each location created, the new value is randomly created which is added to the original value resulting in a new value. This is indicated in the `Comparejpg2` class file.

The sensitivity of the image to errors is determined by measuring the mean square error between the corrupted JPEG-2k decoded file and the JPEG-2k decoded file without errors.

For every bitrate the input file is JPEG-2k encoded, errors are randomly introduced and then decoded. Simultaneously the input file is JPEG-2k encoded and decoded without introducing errors. The two output files are compared and MSE is computed for each iteration. Then the steps are repeated 20 times and the average MSE is calculated per bitrate.

For every image the results that follow show the performance of JPEG-2k and give an indication of how much an image is degraded. In this section interest lies only in the

performance of JPEG-2k but later in chapter 6, the FEC simulations results will be shown to illustrate how channel coding and image compression are connected.

The graph shown below is the plot of the probability of error versus the mean square error and table 4.1 illustrates the results from 3 options that were used to randomly introduce errors in the JPEG-2k encoded file before transmission.

Table 4.1: Lena2 Image results for JPEG the process

Probability of Error	Mean Square Error		
	Option 1	Option 2	Option 3
0	0	0	0
10^{-6}	0	0	0
25×10^{-6}	1,170	0,545	1,103
5×10^{-5}	6,872	5,171	3,637
10^{-4}	15,539	17,920	9,684
10^{-3}	188,068	347,309	204,953
10^{-2}	1534,165	2043,890	889,981
10^{-1}	2645,692	2199,534	1091,525

The transmission channel with random errors was simulated and evaluated the average reconstructed image quality. The reconstructed image quality under transmission errors for JPEG-2k at bit error rates higher than 10^{-4} is very low. The results in figure 4.1 illustrates that at bit error rates higher than 10^{-4} , the errors introduced are detectable and are not all recoverable which gives high values of mean square error implying that the quality is degrading as the bit error rate increases.

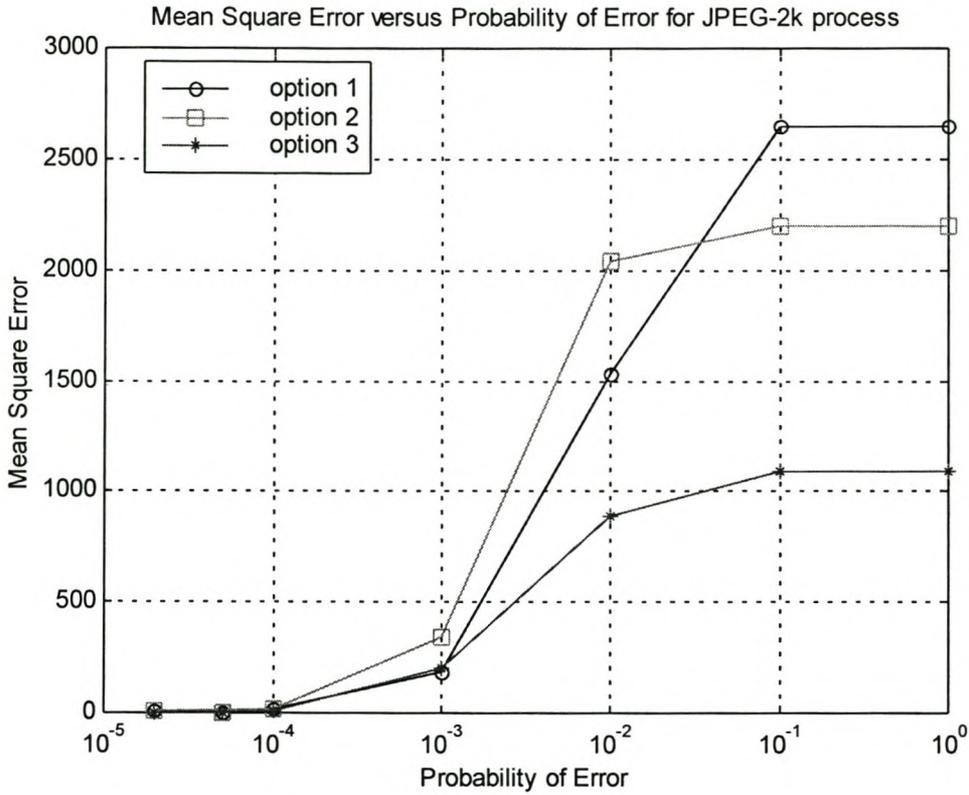


Figure 4.1: The graph of Mean square error versus Probability of error

The different options used to simulate errors shows that the nature of different types of image distortion has varying degrees of effect to the image as in figure 4.1. For all the other test images their graphs for the probability of error for JPEG-2k process versus mean square error are in the chapter 6 for Results.

4.2 Reed Solomon Encoding

The (255,239) RS code with 8-bit symbol is chosen. The generator polynomial of the code in octal form is 267543. The Reed Solomon's encoding is implemented by shift registers with feedback connections similar to the ones in Figure 2.2. The parity bits are added to blocks of message bits to make code words. A total of $(n - k)$ which is 16 bytes, redundant bits are added to the k information bits for the purpose of detecting and correcting errors.

Having the (255,239) cyclic code, the set of code vectors for which the 51 leading high order information digits are identical to zero are considered. If the 51 zero information digits are deleted from each of this code vectors, a set of vectors of length $255 - 51$ are obtained. These shortened vectors form a (255-51, 239-51) linear code. This code is a shortened cyclic and has at least the same error correction capability as the code (255,239) which is $T = 8$ per 255 bytes.

A shortened (204,188) Reed-Solomon encoding is performed on each received 188 symbols, with $T = 8$. This means that 8 erroneous symbols can be corrected. This process adds 16 parity symbols to each 188 symbols to give a block of 204 symbols. The shortened Reed-Solomon code is implemented by appending 51 bytes, all set to zero, before the information bytes at the input of a (255,239) encoder, after the coding procedure these bytes are discarded.

4.3 RS Decoding

To design the decoder four steps that are defined in section 2.7.2 have to be followed: The first step in the decoding algorithm is the computation of the syndrome components, which are obtained by substituting α^i into the received polynomial $2T$ times.

The value i keeps track of the number of iterations. If at a specific iteration i , the syndrome is non-zero then count is incremented, otherwise count is zero. After $(n-k+1)$ iterations if count is greater than zero then it implies that the received polynomial is corrupted. If count is zero, it means all the syndromes are zeros then the received vector is accepted as the transmitted vector which is not corrupted.

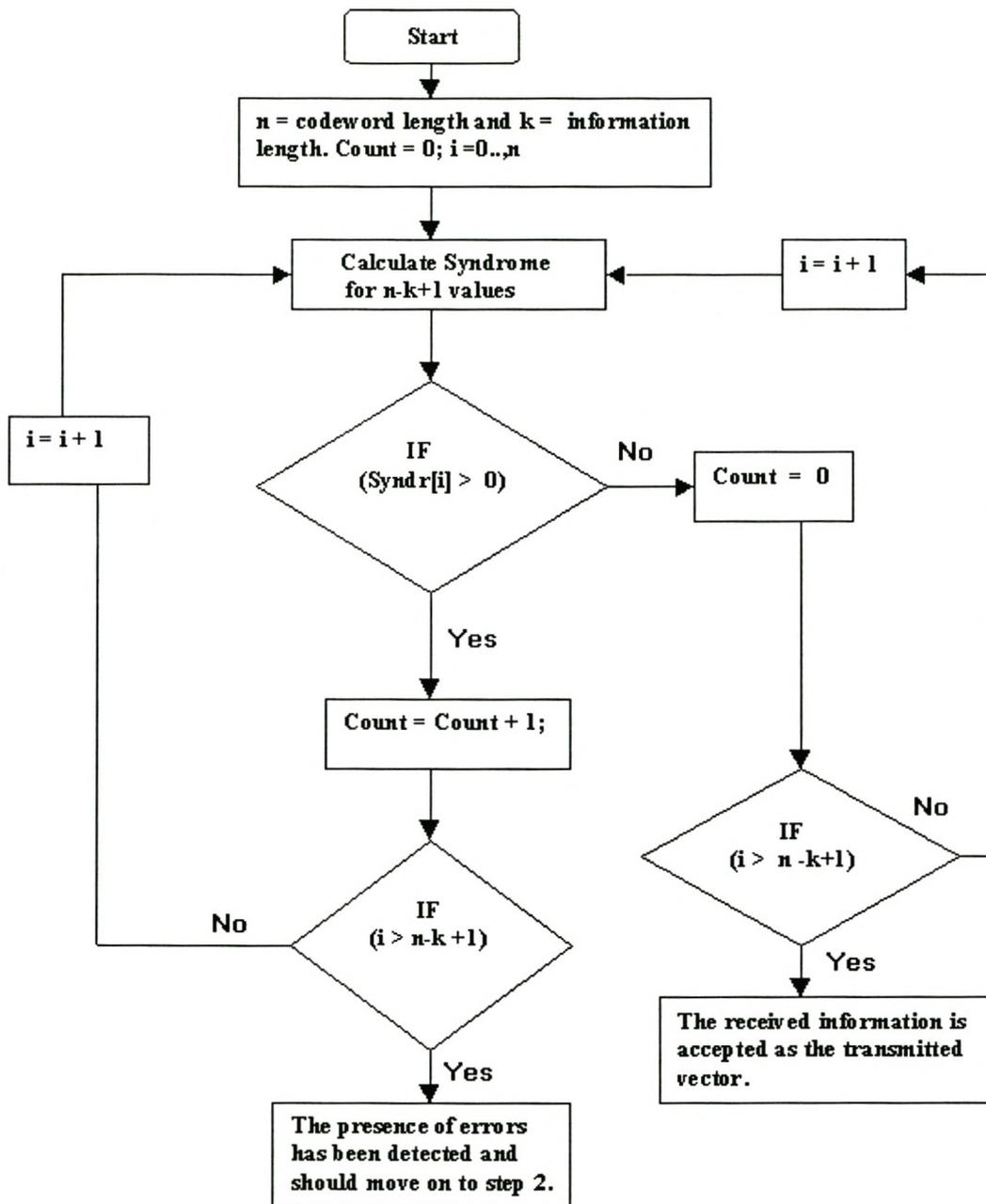


Figure 4.2: The first step of the decoding algorithm

The second step is for determining the error location polynomial as described in the flow diagram in figure 4.3 below. The second step uses the results of the Syndrome computations obtained from step 1.

The Berlekamp's algorithm initial conditions are from $\mu = -1$ to $\mu = 0$ whereas in this application due to the fact that if reading bytes in Java -1 represents a null, so for this implementation $\mu = -1$ is replaced by $\mu = 0$. The initial conditions for this case are $\mu = 0$ to $\mu = 1$.

At each iteration the algorithm must retain both an error location polynomial and a possible correction term. For each new symbol the algorithm tests to determine whether the current error location polynomial will predict this symbol correctly. If so, the error location polynomial is left unchanged and the correction term also is not modified. If the current error location polynomial fails, then adding the correction term modifies it.

The symbols used in the flow diagram are defined below as:

- Sig is the error location polynomial per iteration.
- d is the discrepancy determined per iteration.
- μ is the number of iteration in step 2 and
- Correction is the correction term used to add (modulo-2 addition) to the previous minimum degree polynomial.

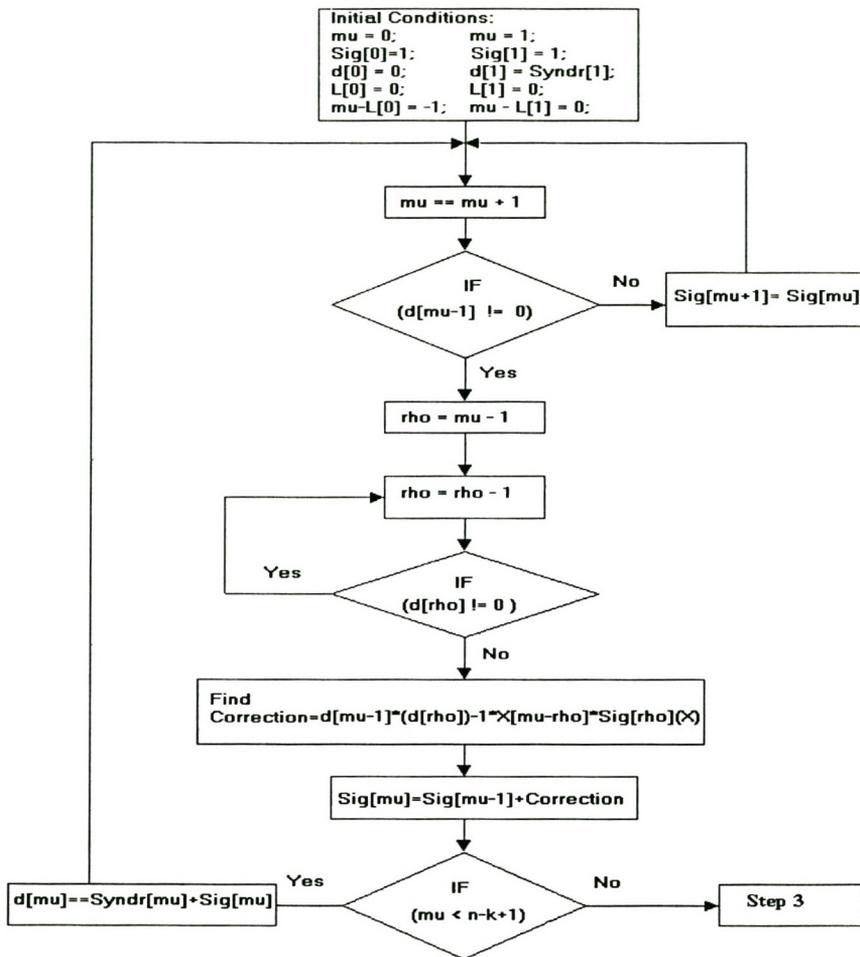


Figure 4.3: The third Step of the Decoding algorithm

From the second step we will be having the error location polynomial Sig or $\sigma(X)$. If the degree of $\sigma(X)$ is greater than the error correcting capability then there are more than T errors and no correction will take place. As illustrated in the flow diagram in figure 4.4 the elements of $\text{GF}(2^m)$ α^i , where $i = 1, \dots, 2t$ will be substituted in Sig to compute the roots. If substituting α^i in Sig gives the result of zero then α^{n-i} is an error location number otherwise it is not an error location number.

The error values are determined by substituting the error location numbers, which are the inverses of the roots, in equations 2.13 and 2.14. The result obtained after the substitution gives the error pattern vector. If the number of errors is equals to or less than T , then errors are corrected by adding the error pattern to the received polynomial.

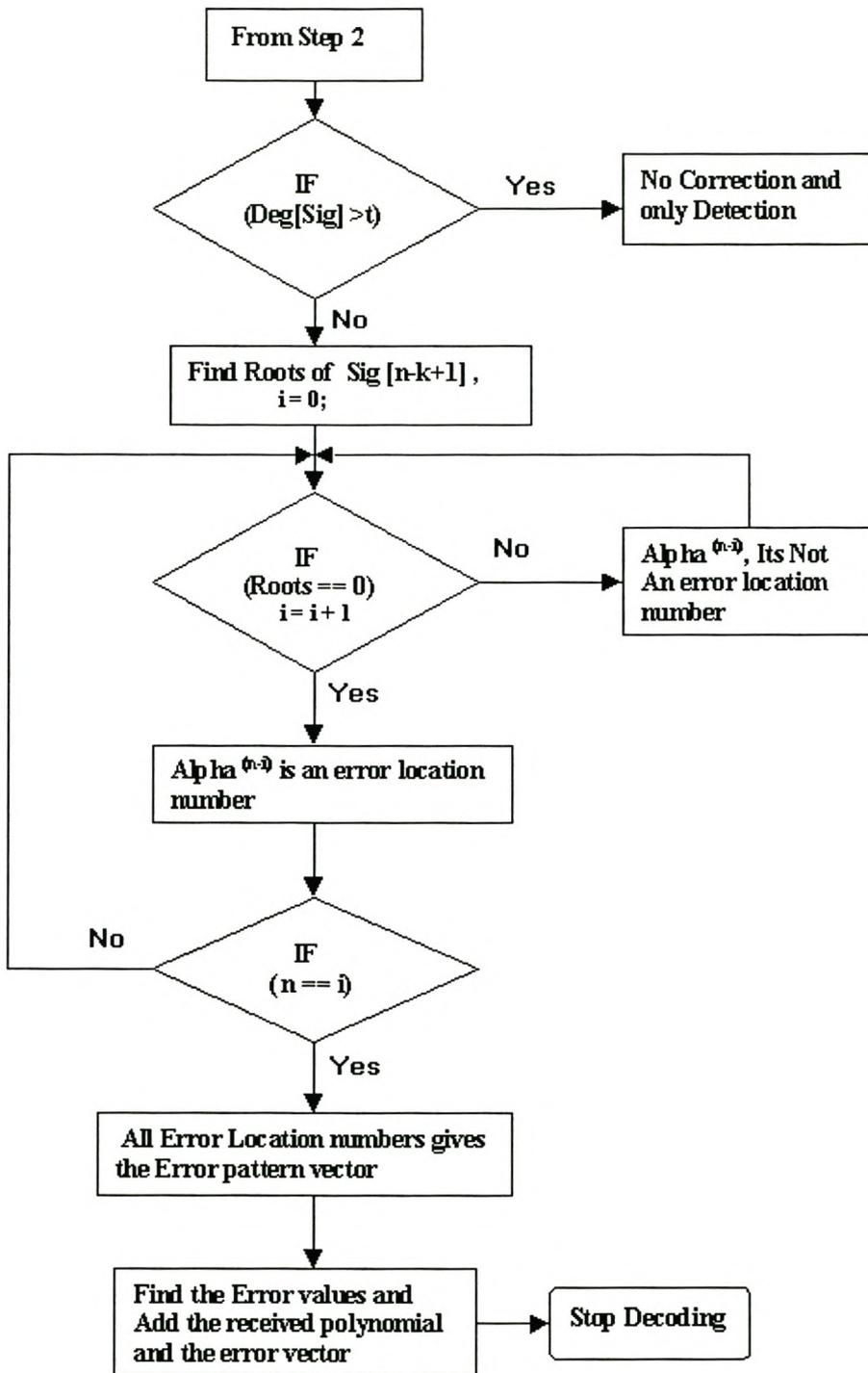


Figure 4.4: The Third step of the decoding algorithm

The decoding procedure listed above completes the RS decoding process and it will be performed on the ground station due to the algorithm's complexity to avoid too much power consumption on the satellite.

4.4 Convolutional Encoder

As illustrated in figure 2.4, input enters from the left end and two outputs are generated for every input bit. Each summing node represents modulo-2 addition and each box represents a memory register that holds the input value from previous times. For this application, there are six memory registers and the output at a given time depends on seven input values, including the current one. Thus the constraint length of the code K is 7. Since the code has one input value and two outputs, the code rate is $1/2$.

One output is produced by **xoring** (or modulo-2 add) the bits of shift register 1,2,3,4,7 and the other is **xor** output of 1,3,4,6,7. Initially we can assume that the flip-flops contain zero and subsequently they take on values depending on the input bits. The bit values represented by the flip-flops are called the "state" of the system. The output bits are transmitted through a communication channel and are decoded by employing the Viterbi decoder at the receiving end.

4.5 Viterbi decoding algorithm

The coded bit stream produced by the convolutional coder from the previous section is transmitted over the noisy channel. The noise can cause some of the bits to be decoded incorrectly causing one or more bit errors. What type of decoder is best suited for reconstructing the original bit sequence with minimum probability of error?

The Viterbi algorithm involves calculating a measure of similarity between the received signals, at time t_i , and all the trellis paths entering each state at time t_i . When two paths enter the same state, the one having the best metric is chosen this path. The decoder continues in this way to advance deeper into the trellis, making decisions by eliminating the least likely paths. This algorithm is effective for short constraint length of $K \leq 7$.

4.6 Concatenated Codes

The codes analyzed in this contribution for the channel is demonstrated in the block diagram in figure 4.5. It is the block diagram of the Concatenated Reed-Solomon, Convolutional Coding Scheme with the JPEG 2000 encoder and decoder incorporated in the system. It illustrates the functionality of both the data compression algorithm and the error control coding algorithm.

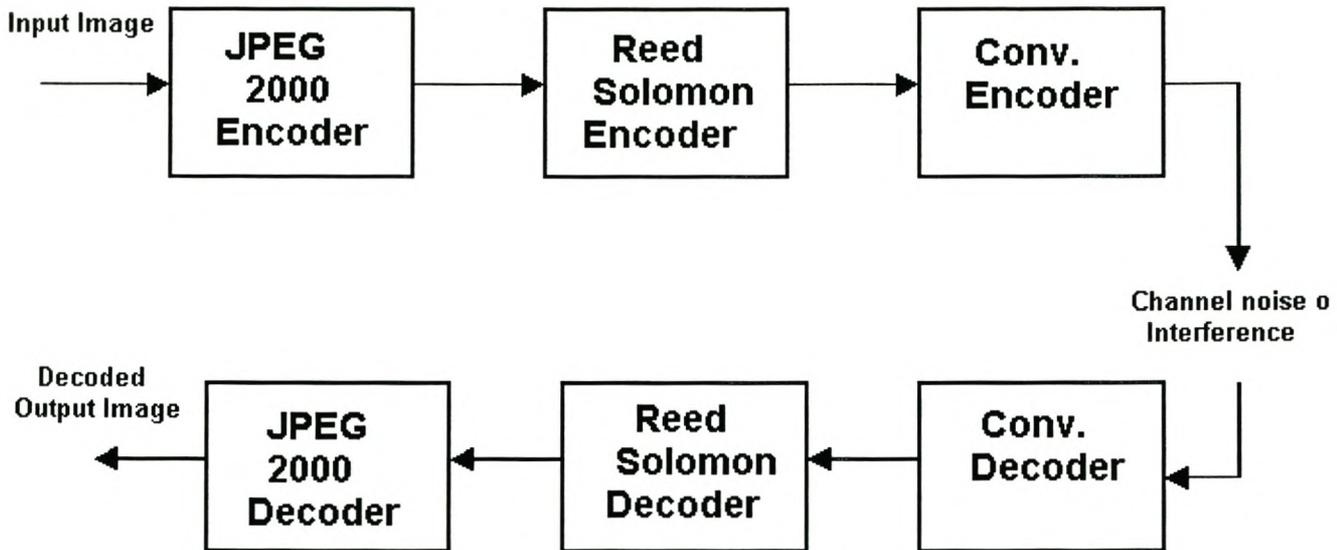


Figure 4.5: The block diagram for the Concatenated code system and the JPEG codec

4.7 Conclusion

Design criteria is as follows:

The compression algorithm to be used is the JPEG 2000. The designed FEC algorithm use RS (204,188) with 8 symbol error correcting capability per block, the convolutional encoder of (1/2) rate and viterbi algorithm with constraint length of $K=7$.

The graph in figure 4.1 shows the specification for the JPEG-2k algorithm, which gives an indication of the degradation of the image and shows the extent at which the JPEG-2k decoder can recover the data. For all the images the same test as above were performed with the compression rate constant at all the bit error rates for a particular image. The implementation of the designed system is outlined in the next chapter.

5 Implementation

As is well known Java is a relatively new language, which is well suited for, but not restricted to, Internet environments, and one that presents many advantages. Java is an object-oriented language, which means focus is on building classes to represent the data in the application, rather than thinking of the solution to a problem as a set of procedures that must be followed in a set order. It is robust, secure, encourages generation of modular and maintainable programs. For reasons stated above Java was chosen as the language to implement the error control coding as the data compression had already been implemented using the same language.

This chapter will be focusing on the development of the FEC software that was designed in the previous chapter. The JPEG-2k software is already available.

5.1 JPEG Compression and Decompression

The package JJ2000.j2k from the public domain is used in the encoder and decoder. To run the JPEG 2000 encoder the call is forwarded to the CmdLnEncoder class, which runs

JJ2000's encoder from the command line interface. It parses command-line arguments to fill a ParameterList object, which will be provided to an Encoder object.

A command line encoder object, is instantiated with the 'argv' command line arguments. The args is the arguments given to the encoder and the list of <args> used in the encoder algorithm is:

- '-i ' option denotes the input file
- the name of the file to be encoded
- '-o ' denotes the output file
- name of the file the encoded file should be written to
- '-Mct ' specifies color transform.

Decompression of the image can be achieved by applying the encoder's steps in reverse order. The JPEG 2000 algorithm was tested using a freely available compression program [4], which is included in Appendix B.

5.2 FEC

The figure below shows the implementation of the FEC encoding algorithm. To encode the image file the first step is to JPEG-2k encode it, then use the returned file in the RS encoder class. The following steps outlines the RS-encoder class:

- A block of 188 bytes is read and 51 zeros are appended to give a total of 239 bytes.
- The parity is calculated for the block and 51 zeros are discarded.
- Append the 16 bytes to the initial 188 bytes.
- The encoded block is written to a file.

These steps above are repeated until the end of file is reached. Then the file is convolutional encoded and there is an option of introducing random errors to a convolutional encoded file.

The decoding algorithm is achieved by having the convolutional decoder, RS decoder, JPEG-2k decoder respectively in the block named decode in figure 5.1. The steps are the inverse of the encoding steps and are in the RSDecodeShrt-class. All the class files for the encoding and decoding algorithms are in Appendix B.

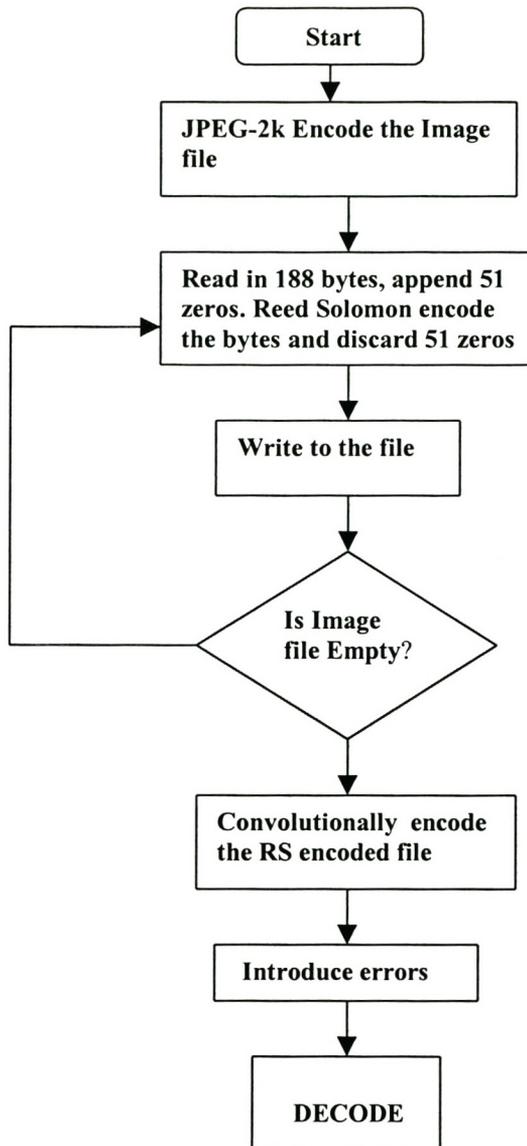


Figure 5.1: The thesis simulation diagram

5.3 Conclusion

The software developed for the encoder should be at the satellite, while the decoding should be at the ground (earth) station where the complexity of the system lies. In order for the system to be tested, the channel errors / noise are simulated. The sections JPEG-2k encoder-decoder, RS encoder-decoder and Convolutional encoder-decoder are tested individually before assembling the whole system. In the next chapter the simulations are performed to test the algorithms implemented.

6 Results

For the purpose of the thesis a set of 6 images were selected as test data. The images are all either satellite images or photographs. The image content varies in resolution, scale and quality.

The calculation/measurement of the quality of image compression differs from other types of quality measurements in that the original image is available for comparison and that therefore the error is exactly quantifiable to give what is expected to be a good objective judgement of the image quality [19, pp.49].

Mean square error is the sum of the squares of the difference between the desired response and the actual system output (the error).

$$MSE = \frac{\sum_{i=1}^N \sum_{j=1}^M (X_i - X_j)^2}{N \times M}$$

6.1

where x_i is the desired response and x_j is the actual system output. The MSE metric, measured image quality correlates well with the perceived image quality [20, pp.6].

The SNR is a measure of the amount of degradation introduced by the compression algorithm, which is not different from MSE.

Below is a list of the 6 test images used in the thesis and the first 5 are taken from the Sunsat micro satellite with a 15m resolution 3 band colour image.

1. Syrian agriculture area = Syrie 1
2. Syrian residential area = Syrie1.1
3. Tibetan country side = Tibetan
4. Bottom half section of Tibetan country side = Tibetan 1
5. Hartswater agricultural area, in Northern Cape South Africa = Hartswater
6. Lena2 image

The images used are included in Appendix C. The sensitivity of the data compression algorithm to simulated channel errors is determined in chapter 4 for each test image. The results from chapter 4 gives an indication of how much the Lena2 image was degraded under transmission errors and results for other images the results are indicated in the next section. How the channel coding scheme can help minimize the effects of random bit errors, the answer to this is shown next.

Testing the JPEG-2k and FEC algorithm:

For every bitrate, the input file is JPEG-2k encoded, RS encoded, Convolutionally encode and errors are introduced randomly at specific locations which are randomly chosen. The corrupted file is then decoded. Simultaneously the input file is JPEG-2k encoded then decoded without introducing errors. The two output files are then compared, using equation 6.1 above MSE is computed.

The steps above are repeated 20 times and for each iteration the MSE is computed. The average MSE is calculated per bitrate. The tests were performed for five test images.

The JPEG-2k is robust to errors, which allows the complete or acceptable partial decoding in the presence of errors in the codestream such as random errors. Since the error correction algorithm should have the ability to combat errors introduced during the transmission over a noisy channel or to protect the message, the bit error rate after the JPEG-2k, FEC algorithm should increase because of the ability to correct some of the errors introduced if not all. The table below shows the results for JPEG-2k and FEC algorithm.

Lena2

For the results shown in Table 6.1 the image size is 193 kB.

Table 6.1: Lena2 JPEG-2k and FEC algorithm

Probability of Error	Mean square Error
0	0
10^{-6}	0,287
10^{-5}	2,702
5×10^{-5}	9,232
2×10^{-5}	35,616
10^{-4}	74,699
10^{-3}	115,870
10^{-2}	586.683
10^{-1}	3025,857

Mean square error is calculated using equation 6.1 above. The rate of mean square error increases as the probability of error increases. The results of Table 4.1 in chapter 4

indicates that as the probability of error increases the mean square error value also increases which means as the MSE value increases the image quality decreases and ultimately the image will be destroyed at high probabilities of error.

The FEC was concatenated with a JPEG-2k algorithm and the simulations were performed. In figure 6.1, the graph shows that between the probability of error of 10^{-6} and 10^{-3} the MSE value is less than 600 and this shows that the addition of the error control coding algorithm has improved the results. This implies that the introduction of the FEC algorithm managed to protect some of the data and improved the bit error rate by 10 as illustrated in figure 6.1.

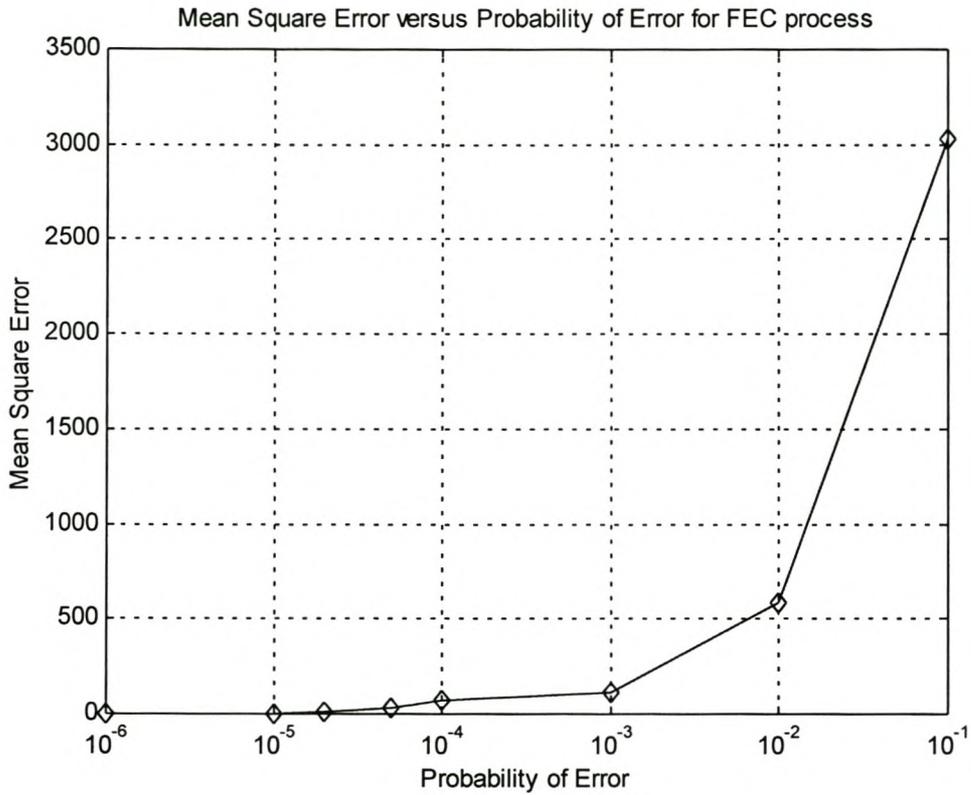


Figure 6.1: The Mean square error versus the Probability of error for JPEG-2k, FEC algorithm

The tables below indicate the results of JPEG-2k process and JPEG-2k and FEC process for other images. Each image has its own compression bitrate and the sizes for the images are as follows: 400kB, 893kB, 351kB, 522kB and 565kB and the compressed image sizes are: 37kB, 73kB, 34kB, 126kB and 104kB respectively.

Tables below shows the results from the JPEG-2K process which are indicated by three options and the results from JPEG-2K-FEC process which are in column named FEC in each of the tables.

Table 6.2: Hartswater Image

Probability of Error	Mean square Error			
	Option1	Option2	Option3	FEC
0	0	0	0	0
10^{-6}	0	0	0	0,055
10^{-5}	0	0	0	1,093
2×10^{-5}	0,177	0,367	0,116	6,550
5×10^{-5}	2,861	1,545	0,595	11,493
10^{-4}	6,330	25,825	19,62	54,911
10^{-3}	471,196	326,839	747,457	168,369
10^{-2}	793,655	1113,677	966,093	532,496
10^{-1}	1518,080	1928,735	2000,308	2038,262

The graphs shown in this section indicate the results from the two tests performed on each image. For every image the three graphs indicate a measure of how much the image is vulnerable to errors under the JPEG-2k compression algorithm and the fourth one is a measure of how much the FEC can protect the compressed image file from transmission random errors. The following section shows the results of the FEC, JPEG-2k algorithm tested for different images at various bitrates.

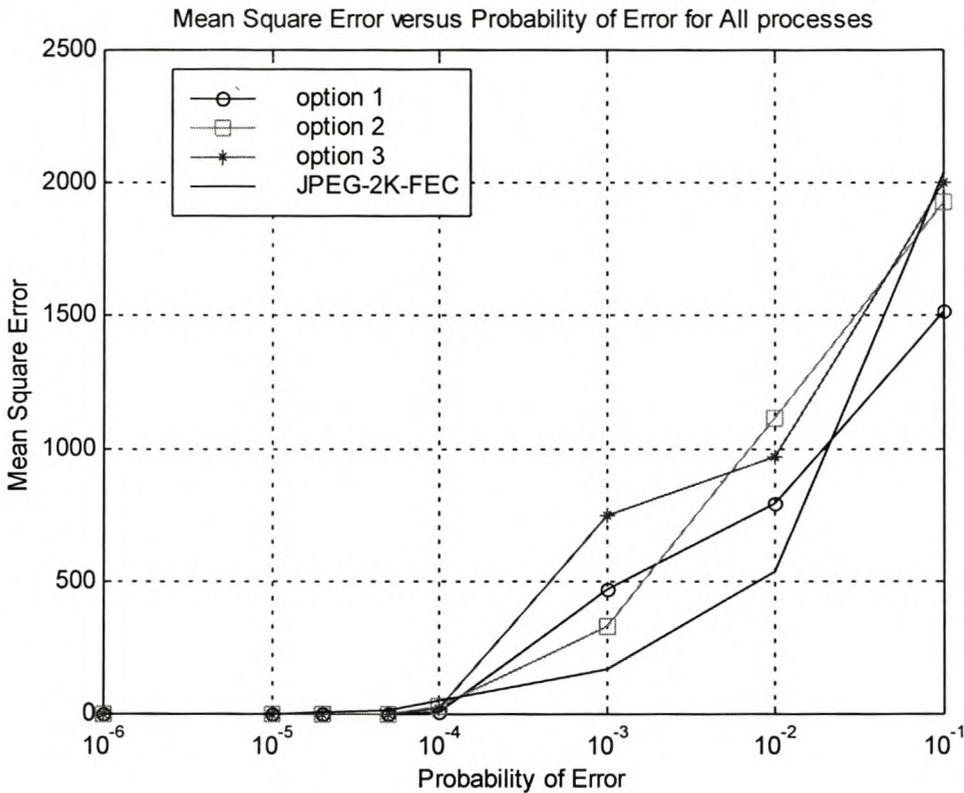


Figure 6.2: The Mean square error versus the Probability of error for All processes

For this image if the probability of error is 10^{-4} , then MSE = is less than 200 for JPEG-2k process at the same probability of error the MSE for FEC, JPEG-2k algorithm is also less than 200. But for probability of error of 10^{-3} in various options MSE is in the range 200 - 800 whereas in FEC MSE at the same probability of error is less than 200. Furthermore at 10^{-2} the FEC, JPEG-2k algorithm shows that the image quality is still better as the MSE value is still less than a 600. The results from the two graphs illustrate that there is an improvement in bit error rate when the FEC algorithm is included.

Table 6.3: Tibetan Image

Probability of Error	Mean square Error			
	Option1	Option2	Option3	FEC
0	0	0	0	0
10^{-6}	0	0	0	0,043
10^{-5}	0,175	0,647	0,145	6,295
2×10^{-5}	0,38	1,719	1,288	10,522
5×10^{-5}	1,953	4,075	3,984	26,532
10^{-4}	11,206	26,438	89,739	46,817
10^{-3}	492,680	458,935	555,416	240,456
10^{-2}	1214,655	894,185	715,554	430,745
10^{-1}	1442,678	1531,373	1949,410	1904,048

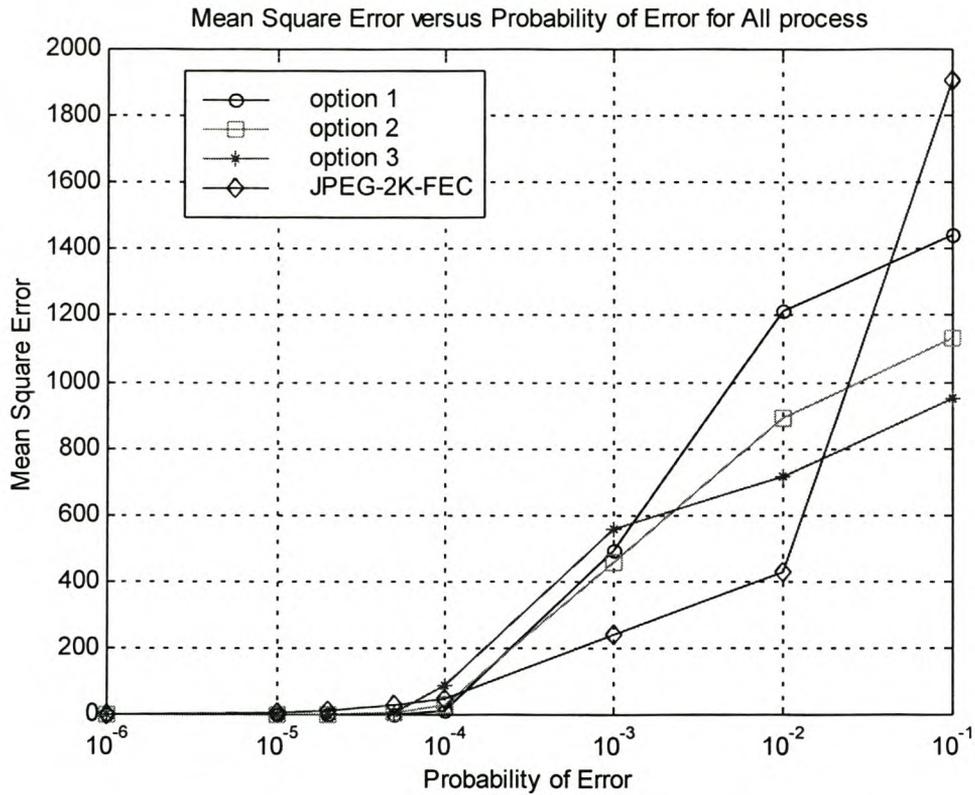


Figure 6.3: The Mean square error versus the Probability of error for All processes

For this image if the probability of error is 10^{-4} , then MSE = is less than 200 for JPEG-2k process at the same probability of error the MSE for FEC, JPEG-2k algorithm is also less than 200. But for probability of error of 10^{-3} in various options MSE is in the range 500 - 1000 whereas in FEC MSE at the same probability of error is less than 500. Furthermore at 10^{-2} the FEC, JPEG-2k algorithm shows that the image quality is still better as the MSE value is still less than a 500.

The results of figure 6.3 shows that there is an improvement in bit error rate for this image after the introduction of the FEC algorithm.

Table 6.4: Tibetan 1 Image

Probability of Error	Mean square Error			
	Option1	Option2	Option3	FEC
0	0	0	0	0
10^{-6}	0	0	0	0,224
10^{-5}	0	0	0	1,859
2×10^{-5}	0,794	0,134	0,254	9,836
5×10^{-5}	2,302	7,030	3,095	17,604
10^{-4}	27,602	21,994	18,632	30,549
2×10^{-4}	233,956	201,992	203,026	99,100
5×10^{-4}	306,333	677,957	623,202	186,139
10^{-3}	658,375	1107,909	847,384	345,780
10^{-2}	1039,802	1791,132	1768,901	453,466
10^{-1}	1757,12	2593,189	2570,202	2374,611

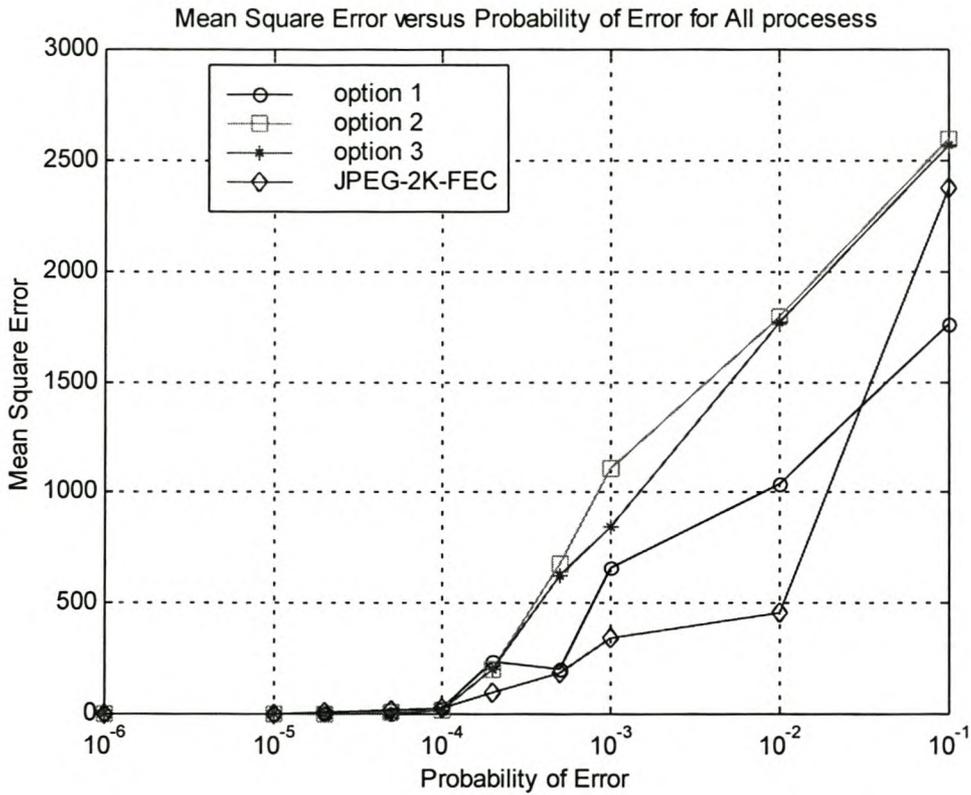


Figure 6.4: The Mean square error versus the Probability of error for All processes

For this image if the probability of error is 10^{-4} , then MSE = less than 500 for JPEG-2k process at the same probability of error the MSE for FEC, JPEG-2k algorithm is also less than 200. But for probability of error of 10^{-3} in various options MSE is less than 600 whereas in FEC MSE at the same probability of error equals 200. Furthermore at 10^{-2} the FEC, JPEG-2k algorithm shows that the image quality is still better as the MSE value is still less than a 500. Figure 6.4 shows that there is an improvement in bit error rate for this image after the introduction of the FEC algorithm.

Table 6.5: Syri1 Image

Probability of Error	Mean square Error			
	Option1	Option2	Option3	FEC
0	0	0	0	0
10^{-6}	0	0	0	0,287
10^{-5}	0,513	2,474	1,008	7,836
10^{-4}	80,758	76,722	280,155	57,646
10^{-3}	476,927	585,328	600,308	215,700
10^{-2}	992,928	1304,858	1083,152	547,166
10^{-1}	2316,160	3856,803	1202,393	2516,255

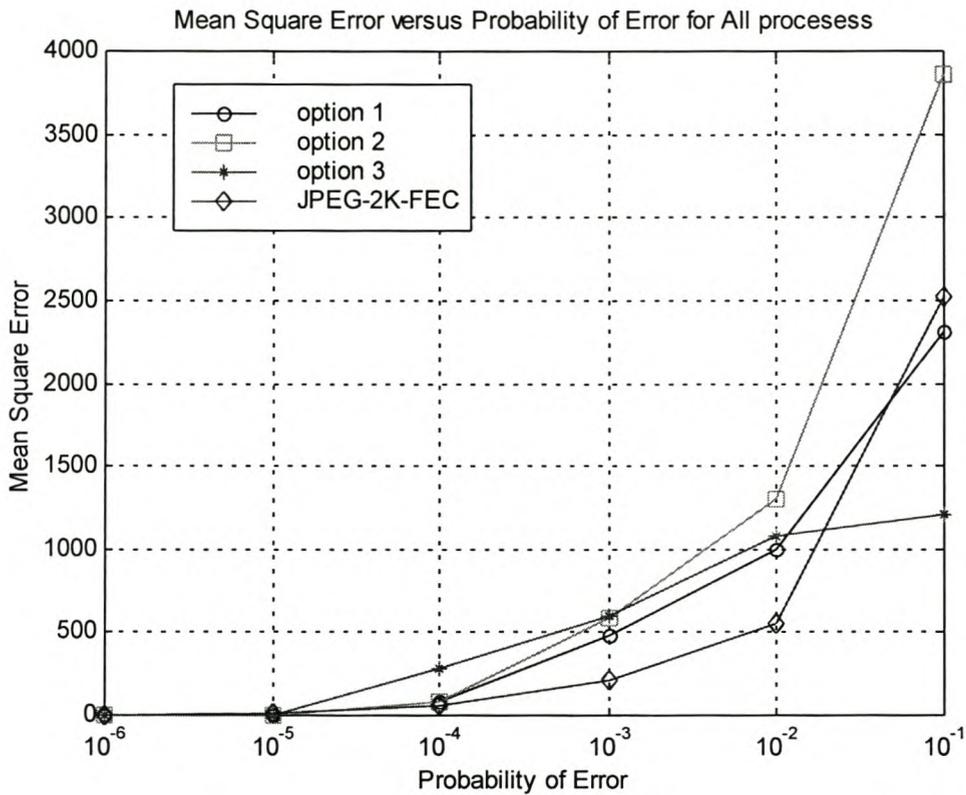


Figure 6.5: The Mean square error versus the Probability of error for All processes

For this image if the probability of error is 10^{-4} , then MSE = less than 200 for JPEG-2k process at the same probability of error the MSE for FEC, JPEG-2k algorithm is also less than 200. But for probability of error of 10^{-3} in various options MSE is greater than 600 whereas in FEC MSE at the same probability of error is less than 200. Furthermore at 10^{-2} the FEC, JPEG-2k algorithm shows that the image quality is still better as the MSE value is still less than 800. The graphs show that there is an improvement in bit error rate for this image after the introduction of the FEC algorithm.

Table 6.6: Syrie 1.1 Image

Probability of Error	Mean square Error			
	Option1	Option2	Option3	FEC
0	0	0	0	0
10^{-6}	0	0	0	0,668
10^{-5}	0	0	0	3,081
2×10^{-5}	0,177	0,367	0,116	9,702
5×10^{-5}	2,861	1,545	0,595	19,356
10^{-4}	6,330	25,825	19,62	45,086
10^{-3}	471,196	326,839	747,457	113,992
10^{-2}	793,655	1113,677	966,093	603,739
10^{-1}	1518,080	1928,735	2000,308	1205,772

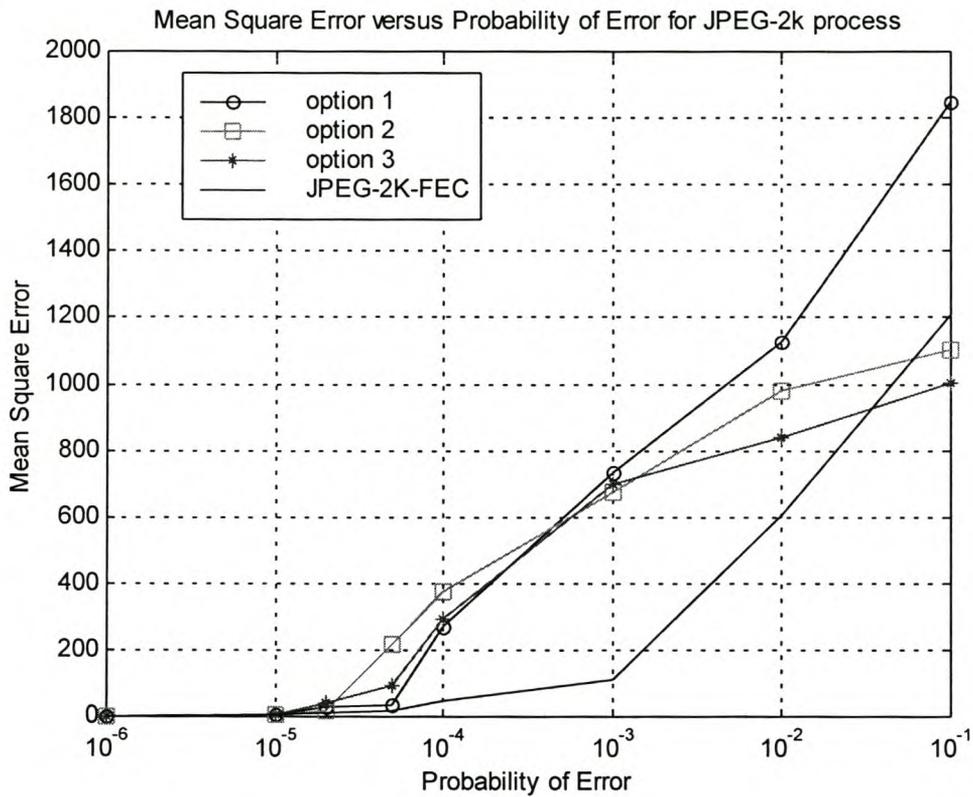


Figure 6.6: The Mean square error versus the Probability of error for All processes

For this image if the probability of error is 10^{-4} , then MSE = less than 200 for JPEG-2k process, at the same probability of error the MSE for FEC, JPEG-2k algorithm is also less than 200. But for probability of error of 10^{-3} in various options MSE is greater than 500 whereas in FEC MSE at the same probability of error is less than 500. Furthermore at 10^{-2} the FEC, JPEG-2k algorithm shows that the image quality is still better as the MSE value is still less than 600. The graphs show that there is an improvement in bit error rate for this image after the introduction of the FEC algorithm

The calculated quality is purely a function of the JPEG-2k, FEC, with errors and JPEG-2k compressed image without errors. This results clearly shows quality is highly dependent on which sequence is hit by an error. In summarizing the results, it is shown that JPEG-2k is robust to bit errors and channel coding helps to minimize the effects of bit error to a certain level.

The compression rates for all the test images are different and if they were the same the idea of averaging the quality for all images could be an option. But now it's a bit tricky to do.

The possible question from this section would be why only 20 iterations were chosen, instead of any other number or probably the number same as the image file size? The number of iterations same as the image size will give the overall average of the errors introduced, meaning the MSE will be representative of all the possibilities. In this case the MSE was selective over the image and due to time constraints further tests could not be performed. The conclusions and recommendations for the project follows in the next chapter.

7 Conclusion and Recommendations

The chapter concludes the document by summarizing the work done in relation to the thesis's main objectives. Further recommendations for future research will be proposed. This will begin by looking at the performance of the data compression technique discussed in section 4.1.

7.1 Conclusion

The JPEG 2k algorithm employed for this work was obtained from source code that is available in the JJ2000 public domain. The source code includes additional image enhancements such as ROI, error resilience and progressive transmission. In chapter 4, and 6 the performance of JPEG-2k was measured and the results showed that at bit error rates higher than 10^{-4} the MSE value increases to higher values meaning image quality decreases and the image is degraded more.

Overall the reconstructed image quality under transmission error is high at bit error rates less than 10^{-4} and lower at bit error rates greater than 10^{-4} . The goal for determining the sensitivity of JPEG-2k was achieved.

From theory it is known that the channel decoder duty is to reconstruct data based on the received symbols. In chapter 6 the results of simulated FEC, JPEG-2k algorithm are shown, and prove that the error correction section of the algorithm manages to recover some of the corrupted data symbols. In conclusion the goal of protecting the encoded data was achieved and the bit error rate improved from 10^{-4} to about 10^{-2} .

The amount of input data used for these results was not enough but the results obtained are good enough to make a conclusion that the FEC algorithm helped minimize the effects of bit errors to a certain degree. It would also be good practice to have more sets of test data using different types of images or sources of images and a wider range of distortion types to put together some form of a base for the algorithm to can measure its performance.

The results in chapter 4 and chapter 6 concludes the project by showing that error correction of random errors was achieved by a factor of 10 or more depending on the image type. Finally, the project showed that Channel coding improves performance by adding redundant bits in the transmitted bit stream that are used by the receiver to correct errors introduced by the channel, thus reducing the average bit error rate. If all the JPEG 2000 features and the FEC algorithm section for interleaving were implemented the performance might improve. The source code is ready to be implemented on the modem or On Board Computer 2 (OBC 2).

7.2 Recommendations

Having studied the error control coding techniques and using the knowledge and experience gained through the direct involvement with the development of the FEC system it is possible to make recommendations for further study.

Parallel and Serial concatenated codes called Turbo codes that achieve data communication at signal to noise ratios close to the Shannon limit [21, pp.1-8]. The codes should represent a major paradigm shift in the approach for coding systems for deep space communications by providing better performance, reduce the complexity of decoding and simplifying system integration.

More recent advances in coding technology, such as Turbo codes, exhibit superior error-correction performance, although they are generally very complex and impose large delays on end-to-end transmission, drawbacks that make them unsuitable for many wireless applications. This coding technology should be investigated as a possible substitution/replacement of RS coding due to reasons stated. It can be implemented with the complete JPEG-2k.algorithm.

Appendix A.

A.1 BCH CODES

The linear Code

A block code of length n and 2^k code words is called a linear (n,k) code if and only if the 2^k code words form a k dimensional subspace of the vector space of all the n tuples over the $GF(2)$.

The Linear Systematic Block Code

A linear block code possesses the property which is the systematic structure of the code words, where a code word is divided into two parts, the message and the redundant checking part. The message part consists of k unaltered information digits and the redundant checking part consists of $n-k$ parity check digits, which are linear sums of the information digits. A linear block code with this structure is a systematic block code.

The Cyclic Codes

An (n,k) linear code C is called a cyclic code if every cyclic shift of a code vector in C is also a code vector in C .

The Shortened Cyclic Code

Given an (n, k) cyclic code C . Consider the set of code vectors for which the l leading high order information digits are identical to zero. There are 2^{k-l} such code vectors and they form a linear subcode of C . If the l zero information digits are deleted from each of

these code vectors, we obtain a set of 2^{k-1} vectors of length $n - 1$. These 2^{k-1} shortened vectors form an $(n-1, k-1)$ linear code. This code is called a shortened cyclic code and is not cyclic. A shortened cyclic code has at least the same error correction capability as the code from which it originates.

A.2.1 Galois fields

Galois fields $GF(2^m)$ for $m \geq 3$. Each element in $GF(2^m)$ is expressed as a power of some primitive element α and as a linear sum of $\alpha^0, \alpha^1, \dots, \alpha^{m-1}$. The minimal polynomial of α is used to generate the field. For every m , the generating polynomial of the field is given. For the Galois field table of $GF(8)$ see [2,pp.564].

A.2.2 The Binary BCH Code

BCH codes are an important class of multiple error correcting codes which have cyclic property. A primitive t -error correcting BCH codes over $GF(q)$ of block length $n = q^m - 1$ has $\alpha^{m_0}, \alpha^{m_0+1}, \dots, \alpha^{m_0+2t-1}$ as roots of the generator polynomial $g(x)$ (for any m_0), where α is a primitive element of $GF(q^m)$.

For any specified m_0 and d_0 , the code generated by $g(x)$ is a BCH code if and only if $g(x)$ is the polynomial of the lowest degree over $GF(q)$ for which $\alpha^{m_0}, \alpha^{m_0+1}, \dots, \alpha^{m_0+d_0+2}$ are roots. Once the code generator polynomial, $g(x)$ is defined the code may be encoded using a feedback shift register. The generator polynomial of a cyclic code is:

$$g(x) = \text{LCM}[f_1(x), f_2(x), \dots, f_r(x)]$$

where $f_1(x), f_2(x), \dots, f_r(x)$ are minimal polynomials of the zeros of $g(x)$.

An important subclass of the BCH codes are the Reed Solomon codes with $m=m_0=1$. For these nonbinary codes defined over $GF(q)$ the block length is $n = q-1$, and the code generator is given by:

$$g(x) = (x-\alpha)(x-\alpha^2)\dots(x-\alpha^{2t})$$

$g(x)$ has degree $2t$ so that only $2t$ parity symbols are required for t -error correction.

Typically $q = 2^m$ is selected, this means the code provides correction of 2^m -ary symbols and hence burst errors.

The generator polynomial of Reed-Solomon code with code word length N , message length K . A valid N should be 2^M-1 , where M is an integer no less than 3. K should be an odd positive number. The error-correction capability is $(N - K) / 2$. The generator polynomial is a polynomial in $GF(2^M)$ and each element is represented by the power form.

For any positive integers m ($m \geq 3$) and t ($t \leq 2^{m-1}$), there exist a binary BCH code with the following parameters:

Block length: $n = 2^m - 1$

Number of parity check digits: $n - k \leq mt$

Minimum distance: $d_{\min} \geq 2t + 1$

This forms a t -error correcting BCH code. The encoding and decoding of binary and nonbinary BCH codes is the same.

A.2 Convolutional Coding

The encoding calculations of these codes are easily implemented using shift registers with feedback connections. BCH codes are a very efficient subclass of cyclic block codes which includes Hamming codes and Reed Solomon codes. There are considerations to determine which error control codes are suitable for a specific application.

Channel conditions determine the probability of random errors and burst errors. Random errors increase as the signal to noise ratio on the channel deteriorates. Channel fading leads to burst errors. Maximum acceptable error rate determines how well the transmitted information has to be protected and by execution the performance required from the error correction.

Implementation of a Convolutional encoder

Convolutional encoders are implemented using shift register and exclusive or gates. Figure (B.1) shows a simple convolutional encoder.

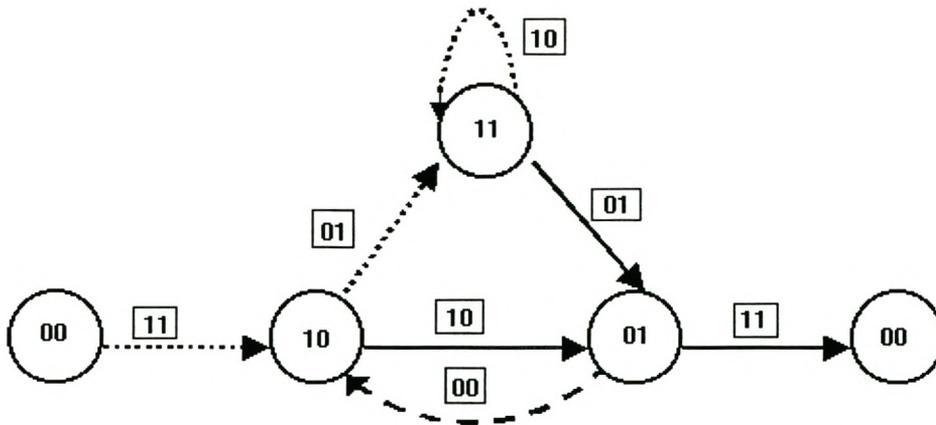


Figure A. 1: Encoder state diagram

One way to represent simple encoders is with a state diagram. The states shown by circles in figure A.5 represent the possible contents of the stages of the register, and the paths between the states represent the output branch words resulting from state transitions. There are only two transitions emanating from each state, corresponding to the two possible input bits.

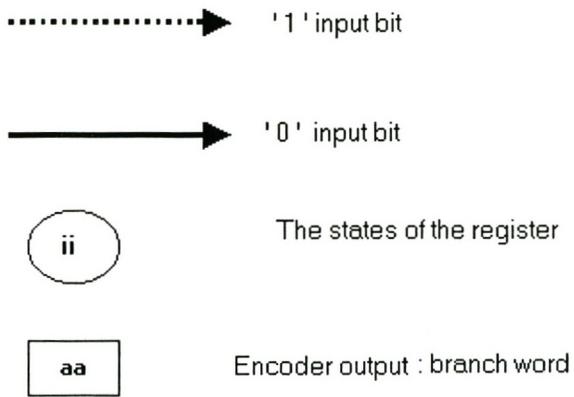


Figure A. 2: The symbols used in the encoder state diagram

Appendix B: The source code.

The accompanying floppy disk contains the source code written in Java. The order of the classes is as follows:

- RS encoder - RSEncode
- Rsdecoder - RSDecodeShrt
- Convolutional encoder and decoder - ConvShrt
- JPEG-2k encoder and decoder- folder JPEG 2000

and lastly the two main classes that has all the encoder and the decoder classes for the FEC-JPEG-2k in which all the subclasses are called from.

- TestEncoding.
- TestDecoding

The file for simulating the JPEG-2k performance depending of which option you choose are:

- CompareJpg – Option 1
- CompareJpg1- Option 2
- CompareJpg2 – Option 3

Functions used by the encoder and decoder to emulate the GF operations

Add, Adder, Invsr, Multi and Pwer.

Appendix C: Test images

The following images are used as test images and they are attached in the following order:

1. Syrian agriculture area = Syrie 1
2. Syrian residential area = Syrie1.1
3. Tibetan country side = Tibetan
4. Bottom half section of Tibetan country side = Tibetan 1
5. Hartswater agricultural area, in Northern Cape South Africa = Hartswater
6. Lena2 image

Bibliography

1. Bernard Sklar, Digital Communications Applications and Fundamentals, 1998 Prentice Hall, Englewood New Jersey, 07632.
2. Shu Lin and Daniell J. Costello. Jr., Error Control Coding Fundamentals and Applications, Prentice-Hall, Inc. Englewood Cliffs, New Jersey, 1983.
3. E.R. Berlekamp, Algebraic Coding Theory, McGraw-Hill, New York, 1968.
4. Charrier M., Santa Cruz D., Larsson M., JPEG 2000 The Next Millenium Compression Standard for still images, June 30,1999.
5. Taub, Schilling, "Principles of Communication Systems", Second Edition, McGraw-Hill, New York, 1987.
6. Santa Cruz D., Larsson M., Ebrahimi T., Askelof J., Christopoulos C., Region of Interest Coding in JPEG 2000 for interactive client/server applications, Proc. IEEE Third Workshop on Multimedia signal Processing (MMSP), pp. 389-394, September 13-15, 1999.
7. Davisson L.D., An introduction to Statistical Signal processing, McGraw-Hill, New York, November1998, pp. 189.
8. Ebrahimi T., Requirements and profile documents version 4.0, ISO/IEC JTC1/SC29/WG1 N1105, November 6,1998.
9. Nister D. and Christopoulos C., Lossless Region of Interest with a naturally progressive still image coding algorithm, ISO/IEC JTC1/SC29/WG1 N613, November 10-14, Sydney, Australia, 1997.
10. Nister D. and Christopoulos C., "Lossless Region of Interest with a naturally progressive still image coding algorithm", Proc. IEEE International Conference

- on Image Processing (ICIP 98), pp. 856-860, 4-7 October 1998, Chicago, Illinois.
11. Santa Cruz D., Larsson M., Ebrahimi T., Christopoulos C., Report on core Experiment ROI 1 refinement, ISO/IEC JTC1/SC29/WG1 N990, Los Angeles, USA, 02-06 November 1998.
 12. E. Atsumi and N. Farvardin, "Lossy/lossless region-of-interest image coding based on set partitioning in hierarchical trees", Proc. IEEE International Conference on Image Processing (ICIP-98), pp. 87-91, October 4-7, 1998 Chicago, Illinois, USA.
 13. D. Speck, "Binary arithmetic coder", ISO/IEC JTC1/SC29/WG1 N299.
 14. D. Taubman, "High Performance Scalable Image Compression With EBCOT", Proceedings IEEE International Conference Image Processing, Vol. III, pp. 344-348, Kobe, Japan, October 1999.
 15. D. Taubman, "High Performance Scalable Image Compression With EBCOT", IEEE Trans. Image Processing, Vol.9, No.7, pp. 1158-1170, July 2000.
 16. Christopoulos C., Athanassios S., and Ebrahimi T., "The JPEG Still Image Coding System", IEEE Transactions on Consumer Electronics, Vol. 46, No. 4, pp.1103-1127, November 2000.
 17. Roden M.S., Analog and Digital Communication System, Prentice-Hall, inc., Englewood Cliffs, New Jersey, 07632, 1979.
 18. <http://jj2000.epfl.ch>, JavaTM JPEG 2000 development,
 19. Nolte E, Image Compression quality measurement: A Comparison of the performance of JPEG and Fractal compression on satellite images, Stellenbosch, March 2000.
 20. Dolinar S, Divsalar D, Pollara F, Turbo codes and Space communications, California Institute of Technology, May 1998, pp.1-8.

