Performance Evaluation of Error Correcting Techniques for OFDM Systems

Yasir Javed Qazi
Email: p060059@gmail.com

Safwan Muhammad
Email: safwan.mu11@gmail.com

Jawad Ahmed Malik
Email: reply.jawad@gmail.com
This thesis is submitted to the School of Electrical Engineering at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science Programme in Electrical Engineering with emphasis on Radio Communication. This thesis is equivalent to 30 credit hours of full time studies out of 120 credits.

**Supervisor:**

Dr. Muhammad Imran Iqbal  
School of Computing  
(Blekinge Institute of Technology, Sweden)

**Examiner:**

Dr. Sven Johansson  
Department of Electrical Engineering, School of Engineering  
(Blekinge Institute of Technology, Sweden)
Abstract

Orthogonal frequency-division multiplexing (OFDM) systems provide efficient spectral usage by allowing overlapping in the frequency domain. Additionally, they are highly immune to multipath delay spread. In these systems, modulation and demodulation can be done using Inverse Fast Fourier Transform (IFFT) and Fast Fourier Transform (FFT) operations, which are computationally efficient. OFDM allows suppression of inter-symbol interference (ISI), provides flexible bandwidth allocation and may increase the capacity in terms of number of users.

In this work, we have investigated the performance of different error correcting techniques for OFDM systems. These techniques are based on Convolutional codes, Linear Block codes and Reed-Solomon codes. Simulations are performed to evaluate the considered techniques for different channel conditions.

By comparing the three techniques, the results show that Reed-Solomon codes performs the best for all error rates due to its consistency in performance at both low and high code rates which we verified by results.
Acknowledgment

We would like to acknowledge the contributions made by our teachers for building our basic knowledge and skills in the field of electrical engineering. Then we would especially like to thank our supervisor Dr. Imran Iqbal for his support and continuous guidance without which it would not have been possible to complete this project. We would also like to thank our examiner Sven Johansson who guided us in earlier phase of our thesis. We would like to thank our parents and siblings for their love and care, and we would like to thank our friends for their support and help.
Contents

Abstract ................................................................................................................................................ ii
Acknowledgment ................................................................................................................................ iii
Contents .............................................................................................................................................. iv
Chapter 1. Introduction ................................................................................................................... 1
  1.1. Background ............................................................................................................................ 2
  1.2. Introduction to problem ........................................................................................................ 4
  1.3. Related work ......................................................................................................................... 4
  1.4. Work done in this thesis ........................................................................................................ 5
Chapter 2. OFDM Systems ............................................................................................................... 7
  2.1. History ................................................................................................................................... 8
  2.2. OFDM Signal .......................................................................................................................... 9
    2.2.1. Orthogonality ................................................................................................................. 9
  2.3. Efficient Multi Carrier Modulation Techniques ................................................................. 11
    2.3.1. Interpretation of FFT .................................................................................................... 11
    2.3.2. OFDM Modulation Using IFFT ...................................................................................... 12
  2.4. Designing an OFDM System ................................................................................................ 14
    2.4.1. Guard Time and Cyclic Extension ................................................................................. 14
    2.4.2. Number of Sub-Carriers ............................................................................................... 15
    2.4.3. Modulation and Coding Schemes ................................................................................ 15
    2.4.4. Building Blocks of an OFDM System ............................................................................ 15
  2.5. Advantages of OFDM ........................................................................................................... 16
    2.5.1. Multipath Delay Spread Tolerance .............................................................................. 16
2.5.2. Immunity to Frequency Selective Fading Channels ..................................................... 16
2.5.3. High Spectral Efficiency ............................................................................................. 17
2.5.4. Efficient Modulation and Demodulation .................................................................... 17
2.6. Applications of OFDM ................................................................................................. 17

Chapter 3. Channel Coding ................................................................................................... 19
3.1. Linear Block Codes ......................................................................................................... 19
3.1.1. Encoding ................................................................................................................... 20
3.1.2. Decoding .................................................................................................................. 23
3.1.3. Maximum likelihood detection .................................................................................. 23
3.1.4. The standard array and syndrome table decoding .................................................. 24
3.2. Convolutional Codes ...................................................................................................... 26
3.2.1. Encoding .................................................................................................................. 27
3.2.2. Viterbi Decoding ..................................................................................................... 29
3.3. Reed-Solomon Codes ..................................................................................................... 30
3.3.1. Encoding .................................................................................................................. 32
3.3.2. Decoding .................................................................................................................. 33
3.4. Common Uses ................................................................................................................ 37
3.5. Error detection and correction capabilities ..................................................................... 37

Chapter 4. Performance Evaluation of Error Correcting Techniques for OFDM Systems ....... 39
4.1. Research methodology ................................................................................................. 39
4.2. Implementation ............................................................................................................. 39
4.2.1. OFDM Transmitter ................................................................................................. 40
4.2.2. OFDM Receiver ....................................................................................................... 40
4.2.3. Linear Block Coding ............................................................................................... 41
4.2.4. Convolutional Coding ............................................................................................ 41
Chapter 1. Introduction

Wireless communication, as the name suggests is wireless way of transmitting information from one place to another, is replacing most of the wired transmission of today’s world. Research in the field of wireless communication is still a hot topic to discover new possibilities [32]. The goal of every research in this topic is to find more effective communication methods. Wireless communication helped the user to move freely without worrying about transfer of data. It dramatically changed the concept of information transfer in homes and in offices. Some of the key advantages gained by wireless communication are [33]:

1. **Efficiency Increase** - It improved communications that leads to faster transfer of information within businesses and between partners/customers.

2. **Always in reach** –There is no need to carry cables or adaptors in order to access some data in your office or home.

3. **Greater flexibility and mobility for users** –Workers in an office don’t need to sit on dedicated PCs. They can be wirelessly networked together.

4. **Reduced costs** –Compared to wired communication, wireless systems are usually cheaper to use, easy to install and maintain.

The basic building blocks of a typical wireless communication system are shown in Figure 1.1.

---

**Figure 1.1: Wireless Communication System**
In the figure the source block is the source of information that we need to transmit from one place to another. It can be a voltage signal, a voice signal or anything else. This information is first converted to digital data and then source encoded. Source encoding reduces the amount of the data present in the signal to reduce the bandwidth required to transmit the associated data. Then the data proceeds to channel encoder block, which is responsible for adding extra bits in the data to help correcting errors inflicted to the data due to fading and noise. Our thesis contains three techniques for channel encoding that will be explained in later chapters. The modulator modulates the message signal on the transmission frequency so that the signal is ready for transmission.

The signal, when transmitted, through wireless channel and faces multiple problems. One major problem faced by the signal is fading [40]. Fading can be caused by natural weather disturbances, such as rainfall, snow, fog, hail and extremely cold air over a warm earth. Fading can also be created by man made disturbances, such as irrigation, or from multiple paths [1]. All these factors introduce errors into the transmitted data.

When received at the receiver the signal is added with noise, since receiver antenna is designed to receive any signal present within a certain frequency range and noise is also present in that range. Now that data is passed to the demodulator whose job is convert the signal back from the carrier frequency to its normal form. After that the channel decoder helps to recover original signal from the degraded signal due to channel fading and noise. This is done by using the redundant bits that were added by the channel encoder [34]. This signal after recovery is passed to the source decoder, which converts the signal back to its original form.

With the advancements in wireless many multiplexing techniques were introduced like Frequency Division Multiplexing (FDM), Time Division Multiplexing (TDM) and Code Division Multiplexing (CDM) are some of those techniques [38]. Orthogonal Division Frequency Multiplexing (OFDM) is an advance form of FDM that has been studied in this thesis.

1.1. Background

Most of the transmission channels are frequency selective. This means that the frequency components from the input signal are affected differently by the channel. In other words, the channel’s transfer function $H(f)$ is not flat over the whole frequency band but behaves differently.
for different frequencies. This introduces Inter-symbol interference (ISI) in the received signal and equalizer (filters the received signal to cancel the isi) becomes necessary to deal with ISI [35]. ISI is a time-domain manifestation of the frequency selectivity.

![Figure 1.2: Single and multicarrier systems](image)

Figure 1.2 shows a single and multicarrier system. The difference is to use the whole band as a single carrier or divide into small bands and use multiple carriers for transmission.

In a single carrier system the whole available spectrum is used for transmission of a single message signal. The small time duration of message signal leads to ISI due to multipath signals. While in a multicarrier system the available spectrum is divided into many narrow bands and data is divided into parallel data streams each transmitted in a separate band. OFDM [36] is an example of multicarrier systems in which each carrier has a very narrow bandwidth, having lower symbol rate. This results in the signal having a high tolerance to multipath delay spread, as the delay spread must be very long to cause significant inter-symbol interference.

![Figure 1.2: Single and multi carrier systems](image)
1.2. Introduction to problem

OFDM is a technique for transmission of data stream over a number of sub-carriers. In OFDM, a high rate bit stream is divided into bit streams of lower rate and each of them are modulated over one of the orthogonal subcarriers. In a single carrier system a single fade can cause the entire link to fail while in a multi carrier system only a few bits will be disturbed and they can be corrected by applying error correction codes. [2]

OFDM overcomes the problem of inter-symbol interference by transmitting a number of narrow band subcarriers together with a guard interval. But this gives rise to another problem that all subcarriers will arrive at the receiver with different amplitudes. Some carriers may be detected without error but the errors will be distributed among the few subcarriers with small amplitude. Channel coding can be used across the subcarriers to correct the errors of weak subcarriers [3].

In OFDM systems error correction has a significant role since OFDM along with error correcting techniques help to deal with fading channels. Error correction helps in recovery of faded information by providing a relation between information and transmitted code such that errors occurring within the channel can be removed at the receiver [37]. A lot of such techniques for error correction are given till date. In this thesis we will consider some of these techniques, implement them for OFDM system and then a comparison between them will be made.

In particular, we will evaluate Linear Block codes, Convolutional codes and Reed-Solomon codes [30][31][39]. We chose these three techniques due to difference of categories to which they belong to and their implementation style (systematic and non-systematic).

1.3. Related work

We would like to mention some previous related works here. In [4] a comparison of turbo codes and convolutional codes for broadband fixed wireless access (FWA) systems is made. Another similar work is presented in [17]. It focuses on OFDM system’s shortcomings in terms of high peak-to-average power ratio (PAPR) that causes nonlinear distortion in the transmitted signal, degrading the performance of the system.
Another work in [18] focuses on OFDM tones that are severely degraded when spectral nulls are present in the channel so forward error correction (FEC) techniques with Code-Spread OFDM (CS-OFDM) are used to spread the data symbols across the frequency band before OFDM modulation.

Many related works are carried out on OFDM and error correcting techniques like [19], which was based on the Eureka-147 Digital Audio Broadcasting (DAB) system in which coded OFDM technology was used in making the receivers highly robust against the channel multipath effects to accomplish the demand of high quality sound services in mobile environment.

Another work [20] focuses on ATM cell oriented data transmission. This method enables accurate comparison of coding and interleaving techniques and optimization of OFDM system parameters.

Other related works include, [21] focuses on the effect of various concatenated forward error correction (FEC) codes on the performance of a wireless OFDM system. The study is done on a recorded audio signal under additive white Gaussian noise (AWGN) channel. The results show that the audio message was received effectively under noisy channel conditions.

The work done in our thesis and the work done in above related works follow the same context (the comparison of bit error rate (BER) vs. signal to noise ratio (SNR)) but the work that we cover is to compare three techniques in this single research. All the related work that we discussed they do not cover these three techniques in the single research.

1.4. Work done in this thesis

In our work we focused on implementation of an OFDM system with 64 carriers. The transmitted data is passed through Rayleigh fading channel and additive white Gaussian noise (AWGN) is added. The BER is calculated at the receiver for different SNR values to estimate the amount of error produced during the transmission of data. Later the system is enhanced with channel encoder and decoder to correct the errors caused by the channel fading and noise.

We used three different error correcting codes namely Linear Block codes, Convolutional codes and Reed-Solomon codes. The performance of these three codes is investigated under various channel conditions. To study in more detail each of them was tested for three different code rates. This helped to further investigate the effect of code rate on error correction performance. The
system parameters and channel characteristics were kept same to test the codes under same conditions.

This work is organized as follows.

We will start with the modulation of data. After modulation we will apply our particular channel coding scheme to enhance the systems error detection and correction abilities. After channel coding we will convert the serial stream of data into parallel streams of data depending on the desired size of FFT and the number of sub-carriers used. After converting the data into parallel streams we apply IFFT on each stream to convert the data from frequency domain to time domain. Then after that we will apply guard interval to give some protection between sub carriers. The performance of OFDM depends on the modulation scheme used, the channel coding scheme and the channel used. We are focussing on the channel coding schemes and their performance in OFDM systems with QPSK modulation and fading channel.

Chapter 2 gives a brief detail about OFDM Systems, Multi carrier modulation technique and its applications and advantages.

Chapter 3 describes the Channel Coding and different procedures involve in the process of coding.

Chapter 4 explains the research methodology and performance evaluation of error correcting techniques for OFDM systems.

Chapter 5 is about simulation results in different coding schemes, validity threats, conclusion and future work.
Chapter 2. \hspace{1cm} OFDM Systems

OFDM is a special type of multicarrier modulation in which a single data stream is transmitted over a number of lower rate subcarriers. In OFDM a high rate bit-stream is split into; say $N$ parallel bit-streams of lower rate and each of them is modulated using one of $N$ orthogonal sub-carriers. OFDM increases the immunity against frequency selective fading or narrowband interference. A single fade causes the entire link to fail in a single carrier system, while only few subcarriers will be disturbed in multicarrier system like OFDM. Error correction coding can then be used to correct those few errors.

In parallel data system such as OFDM, the frequency band is divided into $N$ non-overlapping frequency sub-channels. To eliminate inter-channel interference we avoid spectral overlap of channels, which results in inefficient use of available spectrum. The idea proposed to use parallel data and FDM with overlapping sub-channels comes from the mid 1960s. [22]

To realize the overlapping multicarrier technique, however, it is needed to reduce the crosstalk between subcarriers, which demands orthogonality between the different modulated carriers. Figure 2.1 shows a comparison of single carriers, non-overlapping multicarrier and overlapping orthogonal multi carrier modulation.

In a normal frequency-division multiplex system, by using conventional filters and demodulators, the signal can be received. Guard bands result in lowering of spectrum efficiency, which are introduced between different carriers in frequency domain.
Some of the main advantages of OFDM are its multi-path delay spread tolerance, robustness against frequency selective fading and efficient spectral usage by allowing overlapping of two or more signals in the frequency domain and its robustness against frequency selective fading. Additionally, for an OFDM system, the modulation and demodulation can be done using IFFT and FFT operations, which are computationally efficient.

OFDM will be covered in more detail in this chapter. Chapter 3 will cover the Channel coding techniques of Linear Block Coding, Convolutional Coding and Reed-Solomon Coding. Chapter 4 will cover the implementation methodology and chapter 5 will cover the simulation results.

## 2.1. History

The history of OFDM dates back to mid 1960’s. The concept of using parallel data transmission and frequency multiplexing was published in the mid-1960s [23][25]. After more than thirty years of research and development, OFDM has been widely implemented in high-speed digital communications. It has been recently recognized as an excellent method for high-speed bi-directional wireless data communication [24]. OFDM technique was used in numerous high frequency military systems such as KINEPLEX, ANDEFT and KATHRYN in the 1960s [25]. The
variable data modem in KATHRYN used up to 34 parallel low-rate phase-modulated channels with a spacing of 82 Hz [25].

Discrete Fourier transform (DFT) was applied to parallel data transmission by Weinstein and Ebert as part of the modulation and demodulation process in 1971 [25]. OFDM techniques were also used for multiplexed QAM using DFT and pilot tone, to stabilize carrier and clock frequency control and also the implementation trellis coding. Moreover, various speed modems were developed for telephone networks.

2.2. **OFDM Signal**

OFDM is similar to FDM technique except that in OFDM the ‘N’ sub-carriers are made orthogonal to each other over the symbol (frame) duration $T_s$. By orthogonality of the carriers, we mean that the carrier frequencies satisfy the following requirement:

$$f_k = f_0 + k/T_s \quad k = 1, 2, ..., N - 1$$

Equation (2.1) when converted to the time domain, shows that there will be integer number of cycles translated to the time domain, means that there must be integer number of cycles of each carrier over the duration $T_s$. 

2.2.1. **Orthogonality**

Signals are orthogonal if they are mutually independent of each other [26]. Orthogonality is a property that allows multiple information signals to be transmitted perfectly over a common channel and detected, without interference. Many common multiplexing schemes are inherently orthogonal. Time division multiple access (TDMA) allows transmission of multiple information signals over a single frequency channel by assigning unique time slots to each separate information signal. During each time slot only the signal from a single source is transmitted preventing any interference between the multiple information sources, because of this TDMA is orthogonal in nature. In the frequency domain most FDM systems are orthogonal as each of the separate transmission signals are well spaced out in frequency preventing interference [26]. Although these methods are orthogonal the term OFDM has been reserved for a special form of FDM [26]. The subcarriers in an OFDM signal are spaced as close as is theoretically possible while
maintain orthogonality between them. An OFDM system with four subcarriers is shown in Figure 2.2.

OFDM achieves orthogonality in the frequency domain by allocating different subcarriers to information signals. OFDM signals are made up from a sum of sinusoids, with each corresponding to a subcarrier.

Some of the sub-carriers in some of the OFDM symbols may carry pilot signals for measurement of the channel conditions [7][8]. The base band frequency of each subcarrier is chosen to be an integer multiple of the inverse of the symbol time, resulting in all subcarriers having an integer number of cycles per symbol. As a consequence the subcarriers are orthogonal to each other [26][27]. Sets of functions are orthogonal to each other if they match the conditions mentioned.

\[
\int_{0}^{T} S_i(t)S_j(t)d(t) = \begin{cases} \epsilon & i = j \\ 0 & i \neq j \end{cases} \quad (2.2)
\]

If any two different functions within the set are multiplied, and integrated over a symbol period, the result is zero, for orthogonal functions.
2.3. **Efficient Multi Carrier Modulation Techniques**

To implement a system such as OFDM we need ‘N’ modulators and ‘N’ demodulators at the transmitter and the receiver, respectively. This is a spectrally inefficient method to send data bits in parallel we use a bank of oscillators with centre frequencies spaced so that the channels do not overlap. Bits are feed to a serial to parallel converter whose parallel output modulates the individual carriers. In the receiver, filters tuned to each sub channel decode the bits transmitted in each band. The bits, which exit the filters in parallel, pass through a parallel to serial converter to reconstruct the transmitted bit stream.

Since we cannot construct a perfect brick wall filter to separate the sub channels, they are spaced wider than the Nyquist minimum fs. A spacing of \((1 + \alpha)fs\) (where \(\alpha\) represents the excess bandwidth) results in a bandwidth efficiency of \(fs/\Delta f = \alpha/(1 + \alpha)\). Allowing the sub channels to overlap increases the efficiency of the system.

Such systems require the use of numerous high-order filters and oscillators in both the transmitter and receiver. This degree of complexity increases both the cost and size of the system.

Fortunately, another far more efficient method exists. This method utilizes FFT. The FFT specifies the weights assigned to N orthonormal functions.

### 2.3.1. Interpretation of FFT

The FFT is an essential operation in OFDM transmission and reception. By definition, we need N orthogonal vectors to define an OFDM symbol. We denote the exponential used in Fourier transform as \(W_N^{kn}\), where

\[
W_N^{kn} = e^{j\frac{2\pi nk}{N}} \quad n = 0, 1, \ldots, N - 1
\]  

When using these complex exponentials, orthogonality is achieved by fulfilling the condition

\[
\sum W_N^{\alpha n} W_N^{\beta n} = \begin{cases} 1 & a = b \\ 0 & a \neq b \end{cases} \quad a, b \in \{0, 1, \ldots, N - 1\} \tag{2.4}
\]

Now, the FFT analysis equation is given as:
Similarly, the FFT synthesis equation is given as:

$$\text{IFFT}: \quad x[n] = \frac{1}{N} \sum_{n=0}^{N-1} X[k] e^{j \frac{2\pi nk}{N}} \quad (2.6)$$

### 2.3.2. **OFDM Modulation Using IFFT**

The OFDM transmitter operates on sets of $M$ bits at a time where groups of $nk$ bits modulate subcarrier $k$, where $k = 0, 1, \ldots, N-1$. In general, the number of bits assigned to each subcarrier vary among the subcarriers. We must choose each $nk$ such that $\sum_{k=0}^{N-1} n_k = M$.

If the transmitter uses 4-QAM on each channel, then $nk = 2$, $k = 0 \ldots N - 1$ and $M = 2N$ since each transmitted symbol carries two bits worth of information. Similarly, if 128 QAM is substituted for 4-QAM, then $nk = 7$, $k = 0 \ldots N - 1$ and $M = 7N$.

When using a two-dimensional signal format, the transmitted point in the signal constellation for subcarrier $k$ (which codes $nk$ bits) is written $d[k] = x[k] + j y[k]$, where $x[k]$ represents the in-phase component and $y[k]$ is the quadrature component. The subcarriers are spaced in frequency at the symbol rate to keep them orthogonal. Hence, an OFDM symbol duration of $T$ seconds results in a subcarrier spacing of $1/T$ Hz. Longer symbol durations allow to pack the subcarriers closer together in frequency.

We can describe the band pass continuous time transmitted waveform $D(t)$ during one symbol interval as:

$$D(t) = \sum_{k=0}^{N-1} (x[k] \cos(2\pi f_k t) + y[k] \sin(2\pi f_k t)) \quad (2.7)$$
Where $f_0$ is the base frequency, $\Delta t$ the serial symbol duration, $\Delta f = 1/N \Delta t$ the subcarrier spacing and $f_k = f_0 + k\Delta f$ the frequency of the $k^{th}$ subcarrier. Note that the individual subcarriers are separated by $\Delta f = 1/N \Delta t = 1/T$, the correct amount to keep them orthogonal.

Using Euler’s equation $e^{j\theta} = \cos(\theta) + j \sin(\theta)$, we may rewrite Equation (2.7) as follows:

$$D(t) = \text{Real} \left\{ \sum_{k=0}^{N-1} d[k] e^{j2\pi f_k t} \right\}$$

A single real function can entirely describe a bandpass signal. However, two such functions, or one complex one, are required for a baseband description. We can translate $D(t)$ into a baseband signal by sampling it at intervals of $\Delta t$ and maintaining the complex notation. Keeping in mind that $w_k = 2\pi f_k$ gives:

$$D[n] = \sum_{k=0}^{N-1} d[k] e^{j2\pi nk/N}$$

In fact, we can rewrite Equation 2.9 as:

$$D[n] = DFT\{d[n]\}$$

Modulating the carrier signal with $D[n]$ effectively windows the signal with a rectangular baseband shaping function $h(t)$ which is defined mathematically as:

$$h(t) = \begin{cases} 1/T & 0 \leq t < T \\ 0 & \text{otherwise} \end{cases}$$

This rectangular pulse shape imposes a sinc frequency response on each subchannel. This sinc function $(\sin(\omega T)/\omega T)$ has zeros at multiples of $1/T = \Delta f$, satisfying the Nyquist condition for ISI free transmission. Hence, in theory, the receiver can recover all of the transmitted symbols.

So, the IFFT and FFT operations are used to achieve OFDM modulation and demodulation.
2.4. **Designing of an OFDM System**

The design of an OFDM system requires a trade-off between various parameters like in all communication system designs. Usually, the input parameters to the design are the bit rate, available bandwidth and the maximum delay spread introduced by the channel. The design involves calculation of symbol duration, guard time, number of subcarriers, and the modulation and coding schemes among other factors.

2.4.1. **Guard Time and Cyclic Extension**

OFDM is immune to multi-path delay spread, which causes Inter-symbol Interference (ISI) in wireless networks. Making the symbol duration larger reduces the effect of delay spread. It is done by converting high rate data signal into lower rate data signal. ISI is eliminated by the introduction of guard time. This is done by making the guard time duration larger than that of the estimated delay spread in the channel. If the guard period is left empty, the orthogonality of the sub-carriers no longer holds, i.e., Inter Carrier Interference (ICI) comes into picture. In order to eliminate both the ISI as well as the ICI, part of the OFDM symbol called ‘cyclic prefix (CP)’ is cyclically extended into the guard period. The orthogonality of the sub-carriers is preserved within the FFT interval so that delayed versions of OFDM always have an integer number of samples. The cyclic prefix is a copy of the last part of the OFDM symbol which should be presented to the transmitted symbol (see Figure 2.3) and removed at the receiver before demodulation. The cyclic prefix should be as long as the significant part of the channel impulse response experienced by the transmitted signal. This doubles the benefit of cyclic prefix. First, it avoids ISI. Second, it also converts the linear convolution with channel impulse response into a cyclic convolution. As a cyclic convolution in the time domain translates to multiplication in the frequency domain, the subcarrier remains orthogonal and there is no ICI.

By setting the guard time larger than the delay spread and cyclically spreading the OFDM signal into guard period we can eliminate ISI and ICI. The guard time is made longer than the maximum delay spread introduced by the channel. But the guard time cannot be made very large since no information bits are transmitted during the guard time. The symbol duration must be fixed in such a way that the overhead associated with the guard time is minimal. This can be achieved by making the symbol duration much longer than the guard time. However large symbol duration
means more number of sub-carriers and thus causes implementation complexities and increased peak-to-average power problems. Thus a practical design choice for the symbol duration is around 5-6 times the guard time.

**Figure 2.3: OFDM Symbol Time Structure, Cyclic Prefix (Source: IEEE 802.16)**

### 2.4.2. Number of Sub-Carriers

Once the symbol duration is fixed, the spacing between the subcarriers can be obtained as the inverse of the symbol duration minus the guard time. The number of the subcarriers can then be calculated as the ratio of the available bandwidth to the carrier spacing.

### 2.4.3. Modulation and Coding Schemes

The decision of which modulation and coding technique to use depends on various issues. The decision significantly overlaps with the design of the number of subcarriers discussed above and usually a back-and-forth approach is followed. For example, if the number of bits that are to be assigned to each subcarrier is known, then the modulation and coding for each subcarrier can be designed based on this. On the other hand, if the modulation and coding schemes are specified, then the number of subcarriers can be determined.

### 2.4.4. Building blocks of an OFDM System

After knowing the important concepts of an OFDM system, we provide here the block diagram in Figure 2.4 of the entire system. At the transmitter, first, information bits are encoded at channel, which reduces error probability at the receiver. Then the bits are mapped to symbols. Usually, the bits are mapped into the symbols of either 16QAM or QPSK. The symbol sequence is converted to
parallel first then after that OFDM modulation (IFFT) takes place and then it is converted once again to serial format. These OFDM symbols are also provided by Guard time in between. The resulting sequence is then transmitted to RF channel and at the receiver reverse process takes place, which also includes channel estimation and equalization.

Figure 2.4: Block Diagram of an OFDM System

2.5. **Advantages of OFDM**

OFDM has several advantages that make it a possible alternative for CDMA and other future wireless technologies. Some of the main advantages are discussed below.

2.5.1. **Multipath Delay Spread Tolerance**

OFDM is immune to multi-path delay spread, which causes ISI in wireless networks. Making the symbol duration larger reduces the effect of delay spread. It is done by converting high rate data signal into lower rate data signal. ISI is eliminated by the introduction of guard time.

2.5.2. **Immunity to Frequency Selective Fading Channels**

For single carrier modulation techniques, complex equalization techniques are required if channel imposes frequency selective fading, while in OFDM the bandwidth is split in many orthogonal narrow flat fading subcarriers. Hence it can be assumed that the subcarriers experience flat fading only, though the channel gain/phase associated with the subcarriers may vary. In the receiver,
each subcarrier just needs to be weighted according to the channel gain/phase encountered by it. Even if some subcarriers are completely lost due to fading, the user data can be recovered by proper coding and interleaving at the transmitter.

### 2.5.3. High Spectral Efficiency

OFDM allows the subcarriers to overlap in the frequency domain. The sub-carriers are made orthogonal to each other at the same time. The total bandwidth required for \( \text{“N”} \) number of sub-carriers is:

\[
BW_{\text{total}} = \frac{N + 1}{T_s}
\]

(2.12)

For large values of \( N \), the total bandwidth required can be approximated as

\[
BW_{\text{total}} = \frac{N}{T_s}
\]

(2.13)

For serial transmission of the same data the required bandwidth is:

\[
BW_{\text{total}} = \frac{2N}{T_s}
\]

(2.14)

Compared to the single carrier serial transmission, we get a spectral gain of nearly 100% in OFDM.

### 2.5.4. Efficient Modulation and Demodulation

Modulation and demodulation of the sub-carriers is done using IFFT and FFT methods respectively, which are computationally efficient. The modulation and demodulation in digital domain avoids the need of high frequency stable oscillators.

### 2.6. Applications of OFDM

Due to recent advances of digital signal processing (DSP) and very large scale integrated circuit (VLSI) technologies, the initial obstacles of OFDM implementation such as massive complex computation, and high speed memory do not exist anymore. The use of FFT algorithms eliminates
arrays of sinusoidal generators and coherent demodulation required in parallel data systems and makes the implementation cost effective. OFDM has recently been applied widely in wireless communication systems due to its high data rate transmission capability with high bandwidth efficiency and its robustness to multi-path delay. It has been used in wireless LAN standards such as American IEEE802.11a and the European equivalent HIPERLAN/2 and in multimedia wireless services such as Japanese Multimedia Mobile Access Communications [28]. It is also being considered for the IEEE-ISTO BWIF. It is used in the European digital broadcast radio system and its use in wireless applications such as digital video broadcast and mobile communication system is currently being investigated. OFDM is also being used for wideband data communications over mobile radio FM channels [29], asymmetric digital subscriber line (ADSL; up to 6 Mbps), very high speed digital subscriber line (VDSL; 100Mbps), digital audio broadcasting (DAB), and high definition television (HDTV) terrestrial broadcasting.
Chapter 3. Channel Coding

Channel coding is basically from the class of signal transformations designed to improve the communication performance by enabling the transmitted signal to better resist the effects of various channel impairments such as noise fading and jamming. The goal of channel coding is to improve the bit error rate (BER) performance of power limited and/or band limited channels by adding redundancy to the transmitted data [5].

We have already explained that how OFDM avoids the problem of inter symbol interference by transmitting a number of narrowband subcarriers together with using a guard time. This give rise to another problem which is the fact that in multipath fading channels all subcarriers will arrive at the receiver with different amplitudes. In fact some subcarriers may be completely lost because of the deep fades.

Although most subcarriers may be detected without errors and the overall BER will be largely dominated by a few subcarriers with the smallest amplitudes. Forward-error correction coding is essential to avoid this domination by the weakest subcarriers. By using coding across the subcarriers, errors of the weak subcarriers can be corrected up to a certain limit that depends on the error control code and the channel.

Channel codes are a very important component of any modern digital communication system. Channel codes are used to improve the performance of a communications system when other means of improvement such as increasing transmitter power or using a more sophisticated demodulator are impractical. We will discuss linear block codes, convolutional codes and reed-solomon codes in following.

3.1. Linear Block Codes

A block code is defined as a code in which k symbols are input and n symbols are output and is denoted as a (n, k) code. If the input is k symbols, then there are \(2^k\) distinct messages. For each k input symbols output is n symbols known as a codeword where n is greater than k. Figure 3.1 shows input and output of a block code and the size of codeword formed.
A block code of length $n$ with $2^k$ codewords is called a linear $(n, k)$ code if and only if its $2^k$ codewords form a $k$-dimensional subspace of the vector space of all $n$-tuples over the Galois field $GF(2)$. Since there are $n$ output bits so there are $2^n$ combinations possible for codewords but all of them are not codewords rather $2^k$ are codewords. Rest of the combinations usually come forward when there is an erroneous transmission and codewords are corrupted by change in bits. There are some rules which codewords have to fulfil. One of them is that the sum of any two codewords is also a codeword and being a linear vector space, there is some basis, and all codewords can be obtained as linear combinations of the basis.

### 3.1.1. Encoding

We design a generator matrix for encoding of linear block coding. To start with we can designate \{${g_0, g_1, \ldots, g_{k-1}}$\} as the basis vectors. It means that we can represent the coding operation as matrix multiplication. We can make a generator matrix as

$$G = \begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_{k-1} \end{bmatrix}$$

(3.1)

$G$ is a $k \times n$ matrix. If $m = (m_0, m_1, \ldots, m_{k-1})$ is the input sequence of dimension $1 \times k$, the corresponding output is the codeword

$$mG = m_0g_0 + m_1g_1 + \cdots + m_{k-1}g_{k-1}$$

(3.2)
Note that the all-zero sequence must be a codeword and therefore the minimum distance of the code is the codeword of smallest weight. We have a vector space of dimension \( k \) embedded in a vector space of dimension \( n \), the set of all \( n \)-tuples. Associated with every linear block code generator \( G \) is a matrix \( H \) called the parity check matrix whose rows span the nullspace of \( G \). Then if \( c \) is a codeword, then

\[
cH^T = 0
\]

(3.3)

It means a codeword is orthogonal to each row of \( H \). We can also deduce that

\[
G H^T = 0
\]

(3.4)

This code has a dual code in which \( H \) is the generator matrix. If \( G \) is the generator for an \( (n, k) \) code then \( H \) is the generator for an \( (n, n-k) \) code.

An example of \( (7,4) \) code can be generated by generator matrix given below:

\[
G = \begin{bmatrix}
1 & 0 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 & 1
\end{bmatrix}
\]

The 16 codewords from this generator matrix are:

- \( 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \)
- \( 0 \ 0 \ 0 \ 1 \ 1 \ 1 \ 1 \)
- \( 0 \ 0 \ 1 \ 0 \ 1 \ 1 \ 0 \)
- \( 0 \ 0 \ 1 \ 1 \ 0 \ 0 \ 1 \)
- \( 0 \ 1 \ 0 \ 0 \ 1 \ 0 \ 1 \)
- \( 0 \ 1 \ 0 \ 1 \ 0 \ 1 \ 0 \)
- \( 0 \ 1 \ 1 \ 0 \ 0 \ 1 \ 1 \)
- \( 0 \ 1 \ 1 \ 1 \ 1 \ 0 \ 0 \)
- \( 1 \ 0 \ 0 \ 0 \ 1 \ 1 \)
- \( 1 \ 0 \ 0 \ 1 \ 0 \ 0 \)
- \( 1 \ 0 \ 1 \ 0 \ 1 \ 0 \)
- \( 1 \ 0 \ 1 \ 1 \ 0 \ 1 \)
- \( 1 \ 1 \ 0 \ 0 \ 1 \ 1 \)
- \( 1 \ 1 \ 0 \ 1 \ 0 \ 0 \)
- \( 1 \ 1 \ 1 \ 1 \ 1 \ 1 \)

The parity check matrix is,

\[
H = \begin{bmatrix}
0 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 1 \\
1 & 1 & 0 & 1 & 0 & 0 & 1
\end{bmatrix}
\]
If it is considered as a generator of a (7, 3) code, the codewords of this dual code are:

\[
\begin{align*}
0000000 & \hspace{1cm} 1101001 & \hspace{1cm} 1011010 & \hspace{1cm} 0110011 \\
0111100 & \hspace{1cm} 1010101 & \hspace{1cm} 1100110 & \hspace{1cm} 0001111
\end{align*}
\]

When we want to do the encoding, it is often convenient to have the original data explicitly evident in the codeword. Coding of this sort is called systematic encoding. For the codes we considered that it will always be possible to determine a generator matrix in such a way the encoding is systematic, simply perform row reductions and column reordering on \( G \) until an identity matrix is revealed. We can thus write \( G \) as

\[
G = [P|I_k]
\]

(3.5)

Where \( I_k \) is the \( k \times k \) identity matrix and \( P \) is \( k \times (n-k) \) matrix.

When \( G \) is systematic, it is easy to determine the parity check matrix \( H \). It has the following form:

\[
G = [I_{n-k}|P^T]
\]

(3.6)

The parity check matrix can be used to get some useful information about the code. Following theorem holds for linear block codes.

"Let a linear block code \( C \) has a parity check matrix \( H \). The minimum distance of \( C \) is equal to the smallest positive number of columns of \( H \) which are linearly dependent." 

On the receiver side let \( r \) be the vector, and syndrome is defined as:

\[
s = rH^T
\]

(3.8)

If the received vector \( r \) is a codeword the syndrome \( s \) is equal to zero and if it is not a codeword the syndrome is non-zero.

If received vector is an erroneous codeword corrupted with error vector \( e \) then

\[
r = c + e
\]

(3.9)
3.1.2. Decoding

To understand the process of decoding we start with the concept of maximum likelihood detection.

3.1.3. Maximum likelihood detection

Before talking about decoding, we should introduce a probabilistic criterion for decoding, and show that it is equivalent to finding the closest codeword. Given a received vector \( r \), the decision rule that minimizes the probability of error is to find that codeword \( c_i \) which maximizes \( P(c = c_i | r) \). This is called the maximum aposteriori decision rule. According to Bayes rule

\[
P(c|r) = \frac{P(c)P(r|c)}{P(r)}
\]

(3.11)

Where \( P(r) \) is the probability of observing the vector \( r \). Since \( P(r) \) is independent of \( c \), maximizing \( P(c|r) \) is equivalent to maximizing the term \( P(c)P(r|c) \). Assuming that each codeword is chosen with equal probability, then maximizing \( P(c)P(r|c) \) is equivalent to maximizing \( P(r|c) \).

A codeword selected on the basis of maximizing \( P(r|c) \) is said to be selected according to the maximum likelihood criterion. We define \( P(r|c) \) as

\[
P(r|c) = \prod_{i=1}^{n} P(r_i | c_i)
\]

(3.12)

Assuming a binary symmetric channel with crossover probability \( p \), we have

\[
P(r_i | c_i) = \begin{cases} 
1 - p & \text{if } c_i = r_i \\
p & \text{if } c_i \neq r_i 
\end{cases}
\]

(3.13)

Then

\[
P(r|c) = \prod_{i=1}^{n} P(r_i | c_i) = (1 - p)^{n-d(c_i \neq r_i)} p^{d(c_i \neq r_i)} = (1 - p)^n \left( \frac{p}{1-p} \right)^{d(c,r)}
\]

(3.14)
This example shows that if we want to maximize \( P(r|c) \) then select \( c \), which is closest to \( r \), since \( 0 \leq \frac{p}{1-p} \leq 1 \). Considering our assumption, now we can say that the maximum likelihood criterion is the minimum distance criterion, so we should choose the error vector of lowest weight.

3.1.4. The standard array and syndrome table decoding

Suppose a transmitter transmits a symbol \( c \) and receiver receives \( r = c + e \). We assume that error sequences with lower weight are more probable than error sequences with higher weight. We want to determine our decoded word \( c' \) such that the error sequence \( e' \) satisfying \( r = c' + e' \) has minimum weight.

One possible way of doing this is to create a standard array. We form the standard array by writing down a list of all possible codewords in a row with all-zero codeword being the first term. From the remaining \( n \)-tuples which have not already been used in the standard array, select one of smallest weight. Write this down as the coset leader under the all-zero codeword. On this row, add the coset leader to each codeword at the top of the column. Select another small weight from the remaining and repeat the same step until all the \( n \)-tuples are filled in the standard array.

The standard array satisfies the following properties:

- There are \( 2^k \) columns and \( 2^{n-k} \) rows, therefore, an \( (n, k) \) code is capable of correcting \( 2^{n-k} \) error patterns (the coset leaders).
- The difference of any two vectors in the same row is a codeword in \( C \), all vectors in the same row have the same syndrome.
- No two vectors in the same row are identical.
- Every vector appears exactly once in the standard array.

An example of a \( (7, 3) \) code is shown here, where

\[
G = \begin{bmatrix}
0 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 1 \\
1 & 1 & 0 & 1 & 0 & 0 & 1 
\end{bmatrix}
\]

The standard array is shown in Table 3.1 below:
Table 3.1: A standard array decoding table for (7, 3) Linear Block Code

Assume that the sequence \( r = 0011011 \) is received at the output of channel. Looking up this sequence in the standard array, we find that the corresponding coset leader is \( e = 0101000 \), and the sequence is decoded as \( c = 0110011 \).

Searching a received vector in a standard array is computationally expensive. We make a syndrome decoding table by calculation syndrome \( S = rH^T \), where \( r \) will be each coset leader.

The syndrome decoding is shown below in Table 3.2;

<table>
<thead>
<tr>
<th>Error Pattern</th>
<th>Syndrome</th>
<th>Error Pattern</th>
<th>Syndrome</th>
<th>Error Pattern</th>
<th>Syndrome</th>
<th>Error Pattern</th>
<th>Syndrome</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000000</td>
<td>0000</td>
<td>0001000</td>
<td>1000</td>
<td>0000011</td>
<td>0011</td>
<td>0010100</td>
<td>1010</td>
</tr>
<tr>
<td>0000001</td>
<td>0001</td>
<td>0010000</td>
<td>1101</td>
<td>0000110</td>
<td>0110</td>
<td>0010100</td>
<td>1001</td>
</tr>
<tr>
<td>0000010</td>
<td>0010</td>
<td>0100000</td>
<td>1011</td>
<td>0001100</td>
<td>1100</td>
<td>0010010</td>
<td>1111</td>
</tr>
<tr>
<td>0000100</td>
<td>0100</td>
<td>1000000</td>
<td>0111</td>
<td>0011000</td>
<td>0101</td>
<td>1110000</td>
<td>1110</td>
</tr>
</tbody>
</table>

Table 3.2: The Syndrome Decoding Table for (7, 3) Linear Block Code

Each pattern shown syndrome decoding table is a coset leader in standard array. We can summarize the decoding steps as follows:

- Compute the syndrome, \( S = rH^T \).
- Look up the error pattern \( e \) using \( s \) in the syndrome decoding table.
- Then calculate decoded codeword using \( c = r + e \).
- The first \( k \) bits of the decoded codeword are the message signal.
3.2. **Convolutional Codes**

Convolutional codes are different from the block codes since in convolutional coding the information sequences are not grouped into distinct blocks and encoded so a continuous sequence of information bits is mapped into a continuous sequence of encoder output bits. Convolutional coding can achieve a larger coding gain than can be achieved using a block coding with the same code rate [5]. Convolutional codes have their popularity due to good performance and flexibility to achieve different coding rates.

An important characteristic of convolutional codes is that the encoder has memory. That is the n-tuple output generated by the encoder is a function of not only the input k-tuple but also the previous N-1 input k-tuples. The integer $N$ is called the constraint length. The convolutional code is generated by passing the information sequence through a finite state shift register. In general, the shift register contains $N$ stages and $m$ linear algebraic function generators based on the generator polynomials. The input data $k$ bits is shifted into and along the shift register. The number of output bits for each $k$ input bits is $n$ bits. The code rate is given as $R = \frac{k}{n}$.

![Convolutional encoder with constraint length 7 and rate ½.](Source: IEEE 802.16 standard paper)

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 algorithms vary in complexity and performance. More complexity means better performance. The most important of these methods is the Viterbi
algorithm, which performs the maximum likelihood decoding. The Viterbi algorithm uses trellis diagrams to estimate the optimum path.

3.2.1. Encoding

Additive white Gaussian noise (AWGN) corrupts the signal that is transmitted. By using the convolutional channel coding significant improvement in SNR can be achieved. Shift registers are used to perform modulo two additions and combinational logic convolutionally encodes the data. (Chain of flip-flops is usually a shift register where n\textsuperscript{th} output of flip-flop is the input of the (n+1)\textsuperscript{th} flip-flop. The data is shifted one stage when the active edge of clock occurs). Cascaded exclusive-or (XOR) gates represent the combinational logic that implements as shown in table 3.3

<table>
<thead>
<tr>
<th>Input A</th>
<th>Input B</th>
<th>Output (A x-or B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 3.3: Convolutional encoder output table

When you cascade q two-input XOR gates, the output of first one is the input of the second input and the same goes for rest of the inputs, whose input is the output of the last output. Now that we have the two basic components of the convolutional encoder (flip-flops comprising the shift register and XOR gates comprising the associated modulo-two adders) defined, let's look at Figure 3.3 to see a picture of a convolutional encoder for a rate 1/2, k = 3 code. The generator polynomials can be defined as:

$x_1 = (1 1 1)_2 = (7)_8$

$x_2 = (1 0 1)_2 = (5)_8$
Table 3.4 gives the next state given the current state and the input, with the states given in binary.

![Convolution Encoder for code rate ½. K=3](image)

<table>
<thead>
<tr>
<th>Current State</th>
<th>Input = 0:</th>
<th>Input = 1:</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00</td>
<td>10</td>
</tr>
<tr>
<td>01</td>
<td>00</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>01</td>
<td>11</td>
</tr>
<tr>
<td>11</td>
<td>01</td>
<td>11</td>
</tr>
</tbody>
</table>

Table 3.4: State transition Table

k bits per second data bits are provided to the encoder. Channel symbols are output at a rate of \( n = 2k \) symbols per second. If an input clock edge occurs, the output of the left-hand flip-flop is clocked into the right-hand flip-flop and a new input bit becomes available. The first state output is obtained by two states using modulo-two adder, while in the second state it is obtained by the lower modulo-two adder. The encoder shown above encodes the \( k = 3 \), \( (7, 5) \) convolutional code. The octal numbers 7 and 5 represent the code generator polynomials, which when read in binary (111 and 101) correspond to the shift register connections to the upper and lower modulo-two adders.
adders, respectively. In our project we implement ½ rate convolutional encoder for different constraint lengths and used ½ rate k=7 in our simulation. The octal numbers 171 and 131 represents generator polynomials which when read in binary (1001111 and 1101101). It is also shown in Figure 3.4.

![Trellis diagram for convolutional encoder](image)

Figure 3.4: Convolutional encoder rate = ½. k=7
(Source: IEEE 802.16 Standard)

### 3.2.2. Viterbi Decoding

The single most important concept to aid in understanding the Viterbi algorithm is the trellis diagram [10]. Figure 3.5 shows the trellis diagram for our example rate = 1/2 k = 3 convolution encoder, for a 15-bit message.

![Trellis diagram for convolutional encoder](image)

The four parallel dots represent the four probable states of the encoder. For each time instant and the initial state of encoder there is one column in the message. 17 time instants are for 15-bit message and 2 memory-flushing bits, which represents the initial condition of the encoder.
When input bit is a zero the state transitions are shown by dotted lines, when it is a one these states are shown by solid lines. Since state 00 is the initial condition of the encoder, the arrows start and end out at the same state 00.

3.3. Reed-Solomon Codes

These error correcting techniques are block-based and are used to correct errors in various systems including storage devices, wireless or mobile communications, satellite communications etc.

The mechanism by which Reed-Solomon codes correct errors is that the encoder adds redundant bits to the digital input data block. The decoder attempts to correct and restore the original data by removing the errors that are introduced in transmission, due to many reasons that might be the noise in the channel or the scratches on a CD. There are different families of Reed-Solomon codes and each has its own abilities to correct the number and type of errors.

These codes are linear and a subsection of BCH codes [12] and can be denoted as RS(n,k) with s-bit (where s are symbols represented as a bit) symbols.

An n-symbol codeword is made by the k data symbols of s bits each and the encoder adds parity bits. Errors in a codeword can be corrected by the decoder up to $t$ symbols, where $2t = n - k$.

Figure 3.6 shows a typical Reed-Solomon codeword:
Figure 3.6: A typical Reed-Solomon Codeword

Given a symbol size $s$, the possible codeword length $n$ for a Reed-Solomon code is $n = 2^s - 1$. For example, the maximum possible length of a code with 8-bit symbols ($s=8$) is 255 bytes.

R-S codes perform really well against burst noise. For example, consider an $(n, k) = (255, 247)$ R-S code, where each symbol is made up of $m = 8$ bits or 1 byte. Since $n - k = 8$ and this code can correct any four symbol errors in a block of 255. Imagine the presence of a noise burst, lasting for 25-bit durations and disturbing one block of data during transmission, as illustrated in Figure 3.7.

![Figure 3.7: Data block disturbed by 25-bit noise burst.](image)

In this example, notice that a burst of noise that lasts for a duration of 25 continuous bits must disturb exactly four symbols. The RS decoder for the $(255, 247)$ code corrects any four-symbol errors regardless of damage type. In other words, when a decoder corrects a byte, it replaces the incorrect byte with the correct one, regardless of number of bit errors in the symbol. Thus if a symbol is wrong, it might as well be wrong in all of its bit positions. This gives an RS code a tremendous burst-noise advantage over binary codes, even allowing for the interleaving of binary codes. In this example, if the 25-bit noise disturbance had occurred in a random fashion rather than as a contiguous burst, it should be clear that many more than four symbols would be affected (as many as 25 symbols might be corrupted). Of course, that would be beyond the capability of the RS $(255, 247)$ code.
3.3.1. Encoding

The encoding of RS Code starts with the understanding of the codeword it forms. Equation 3.15 defines Reed-Solomon (RS) codes in terms of the parameters \( n, k, t, \) and any positive integer \( m > 2; \)

\[
(n, k) = (2^m - 1, 2^m - 1 - 2t)
\] (3.15)

Where \( n - k = 2t \) is the number of parity symbols, and \( t \) is the symbol-error correcting capability of the code. The generating polynomial for an RS code takes the following form [12]:

\[
g(X) = g_0 + g_1X + g_2X^2 + \ldots + g_{2t-1}X^{2t-1} + X^{2t}
\] (3.16)

The degree of the generator polynomial is equal to the number of parity symbols. RS codes are a subset of the Bose, Chaudhuri, and Hocquenghem (BCH) codes. This relationship between the degree of the generator polynomial and the number of parity symbols holds, just as for BCH codes. Since the generator polynomial is of degree \( 2t \), there must be precisely \( 2t \) successive powers of \( \alpha \) that are roots of the polynomial. We designate the roots of \( g(\lambda) \) as \( \alpha, \alpha^2, \ldots, \alpha^{2t} \). It is not necessary to start with the root \( \alpha \), starting with any power of \( \alpha \) is possible.

Consider as an example the \((7, 3)\) double-symbol error correcting RS code. We describe the generator polynomial in terms of its \( 2t = n - k = 4 \) roots, as follows:

\[
g(X) = (X - \alpha)(X - \alpha^2)(X - \alpha^3)(X - \alpha^4)
\]

This can be simplified using the addition and multiplication defined for 4 finite field elements of \( GF(2^2) \):

\[
g(X) = X - \alpha^3X + \alpha^0X - \alpha^1X + \alpha^3
\]

Reed-Solomon codes are cyclic codes, encoding in systematic form is analogous to the binary encoding procedure [11]. It can be related to shifting a message polynomial \( m(\lambda) \) into the rightmost \( k \) stages of a codeword register and then appending a parity polynomial, \( p(\lambda) \), by placing it in the leftmost \( n - k \) stages. Therefore we multiply \( m(\lambda) \) by \( \lambda^{n-k} \), and by handling the message polynomial algebraically so that it is right-shifted \( n-k \) positions. Next step is dividing \( \lambda^{n-k}m(\lambda) \) by the generator polynomial \( g(\lambda) \), which is written in the following form:
\[ X^{n-k}m(X) = q(X)g(X) + p(X) \]  \hspace{1cm} (3.17)

where \( q(X) \) and \( p(X) \) are quotient and remainder polynomials respectively. Equation 3.17 can be expressed as:

\[ p(X) = X^{n-k}m(X) \text{ modulo } g(X) \]  \hspace{1cm} (3.18)

The resulting codeword polynomial, \( U(X) \) can be written as

\[ U(X) = p(X) + X^{n-k}m(X) \]  \hspace{1cm} (3.19)

A valid codeword is of the following form:

\[ U(X) = g(X)m(X) \]  \hspace{1cm} (3.20)

When the codeword is evaluated at any root of generator polynomial \( g(X) \), it should produce zero. For example for a RS(7, 3) code the four roots were \( \alpha, \alpha^2, \alpha^3, \alpha^4 \), the codeword \( U(X) \) evaluated at these roots must be zero.

\[ U(\alpha) = U(\alpha^2) = U(\alpha^3) = U(\alpha^4) = 0 \]

### 3.3.2. Decoding

Reed-Solomon algebraic decoding procedures can correct errors within its error correcting capability. A decoder can correct up to \( t = \frac{n-k}{2} \) errors.

When a codeword is decoded and the errors are not more than the error correcting capability limit, then the original transmitted code word will always be recovered. If this is not the case then there are two possibilities:

- The decoder will detect that it cannot recover the original code word and indicate this fact.
- The decoder will mis-decode and recover an incorrect code word without any indication.

The probability of any possibilities depends on the particular Reed-Solomon code and on the number and distribution of errors. If there is an erroneous transmission, received corrupted-
codeword polynomial \( r(X) \) is then represented by the sum of the transmitted-codeword polynomial and the error-pattern polynomial in Equation 3.21.

\[
r(X) = U(X) + e(X)
\]  

(3.21)

An important difference between the nonbinary decoding of \( r(X) \) and binary decoding is that in binary decoding the decoder only needs to find the error locations [11]. In binary decoding the error location value is just flipped because it is either 0 or 1. While dealing with nonbinary symbols like in this case we not only learn the error location but also determine the correct symbol values at those locations. We start with the syndrome decoding.

**Syndrome Decoding**

The syndrome is the result of a parity check performed on \( r \) to determine whether \( r \) is a valid member of the codeword set [12]. We get the value 0 by the syndrome \( S \), If \( r \) is a member of codeword set. Any nonzero value of \( S \) indicates the presence of errors. Similar to the binary case, the syndrome \( S \) is made up of \( n - k \) symbols, \( \{S_i\} \) \( i = 1, \ldots, n - k \).

In case of RS(7, 3) code, there are four symbols in every syndrome vector. Their values can be computed from the received polynomial \( r(X) \). Equation (3.20) is rewritten below;

\[
U(X) = g(X)m(X)
\]

The above equation states that the generator polynomial \( g(X) \) roots must also be the codeword polynomial roots \( U(X) \). We also know that \( r(X) = U(X) + e(X) \) then \( r(X) \) evaluated at each of the roots of \( g(X) \) will yield zero when received polynomial is a valid codeword. Any errors will result in a nonzero value. The computation of a syndrome symbol can be described as:

\[
S_i = r(X)|_{X=\alpha^i} = r(\alpha^i) \quad i = 1, \ldots, n - k
\]

(3.22)

Each syndrome symbol \( S_i \) will be equal to 0 if \( r(X) \) is a valid codeword.
An essential property of codes regarding the standard array is that of same syndrome in each rows [12]. This is also valid for RS codes. The syndrome value will be same if it is obtained by evaluating \( e(\lambda) \) or \( r(\lambda) \) at the roots of \( g(\lambda) \).

**Error Location**

Let there be \( \nu \) errors in a codeword at position \( X^{i_1}, X^{i_2}, \ldots, X^{i_\nu} \). Then the error polynomial \( e(X) \) can be written as follows:

\[
e(X) = e_{i_1} X^{i_1} + e_{i_2} X^{i_2} + \cdots + e_{i_\nu} X^{i_\nu}
\]

The error location here is referred by the index \( j \), and the first, second, \ldots, \( \nu \)th errors are denoted by the indices \( 1, 2, \ldots, \nu \). For correction of the degraded codeword, each error location \( X^{i_l} \), where \( l = 1, 2, \ldots, \nu \) and its error value \( e_{i_l} \) must be determined. We define an error locator number as \( \beta_l = \alpha^{i_l} \). Next, we obtain the \( n - k = 2t \) syndrome symbols by substituting \( \alpha_i \) into the received polynomial for \( i = 1, 2, \ldots, 2t \):

\[
S_1 = r(\alpha) = e_{i_1} \beta_1 + e_{i_2} \beta_2 + \cdots + e_{i_\nu} \beta_\nu \\
S_2 = r(\alpha^2) = e_{i_1} \beta_1^2 + e_{i_2} \beta_2^2 + \cdots + e_{i_\nu} \beta_\nu^2 \\
\vdots \\
S_{2t} = r(\alpha^{2t}) = e_{i_1} \beta_1^{2t} + e_{i_2} \beta_2^{2t} + \cdots + e_{i_\nu} \beta_\nu^{2t}
\]

Due to the nonlinearity of the equations these 2\( t \) equations cannot be solved in the usual way. For these kind of equations we use the technique that is a Reed-Solomon decoding algorithm.

An error is signified, after it has been received when a nonzero syndrome vector has been calculated. Next, it is necessary to learn the location of the error or errors. An error-locator polynomial, \( \sigma(X) \), can be defined as follows:

\[
\sigma(X) = (1 + \beta_1 X)(1 + \beta_2 X) \cdots (1 + \beta_\nu X) \\
\sigma(X) = 1 + \sigma_1 X + \sigma_2 X^2 + \cdots + \sigma_\nu X^\nu
\]

The roots of \( \sigma(X) = -\frac{1}{\beta_1}, -\frac{1}{\beta_2}, \ldots, -\frac{1}{\beta_\nu} \). The reciprocals of roots of \( \sigma(X) \) are the error-location numbers of the error pattern \( e(X) \). By using auto regressive modelling techniques [12], we form a
matrix from the syndromes, where the first $t$ syndromes are used to predict the next syndrome. That is,

$$
\begin{bmatrix}
S_1 & S_2 & S_3 & \ldots & S_{t-1} & S_t \\
S_2 & S_3 & S_4 & \ldots & S_t & S_{t+1} \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
S_{t-1} & S_t & S_{t+1} & \ldots & S_{2t-3}S_{2t-2} \\
S_t & S_{t+1}S_{t+2} & \ldots & S_{2t-2}S_{2t-1}
\end{bmatrix}
\begin{bmatrix}
\sigma_t \\
\sigma_{t-1} \\
\vdots \\
\sigma_2 \\
\sigma_1
\end{bmatrix}
= 
\begin{bmatrix}
-S_{t+1} \\
-S_{t+2} \\
\vdots \\
-S_{2t-1} \\
-S_{2t}
\end{bmatrix}
$$

By using the largest dimensioned matrix with nonzero determinant, we apply the autoregressive model of Equation 3.26. For the solution of the error-locator polynomial, $\sigma(X)$ and coefficients $\sigma_1$ and $\sigma_2$, we start as:

$$
A = 
\begin{bmatrix}
S_1 & S_2 & S_3 & \ldots & S_{t-1} & S_t \\
S_2 & S_3 & S_4 & \ldots & S_t & S_{t+1} \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
S_{t-1} & S_t & S_{t+1} & \ldots & S_{2t-3}S_{2t-2} \\
S_t & S_{t+1}S_{t+2} & \ldots & S_{2t-2}S_{2t-1}
\end{bmatrix}
$$

$$
x = 
\begin{bmatrix}
\sigma_t \\
\sigma_{t-1} \\
\vdots \\
\sigma_2 \\
\sigma_1
\end{bmatrix}
$$

$$
b = 
\begin{bmatrix}
-S_{t+1} \\
-S_{t+2} \\
\vdots \\
-S_{2t-1} \\
-S_{2t}
\end{bmatrix}
$$

We solve for $x$ as

$$
x = A^{-1}b$$
The roots of $\sigma(X)$ are the reciprocals of the error locations. Once these roots are located, the error locations will be known. To locate an error, we test each of the field elements with the $\sigma(X)$ polynomial and any element $X$ is a root if it results in $\sigma(X) = 0$.

When we know the number of error locations we will know the length of error polynomial defined in Equation 3.23 and rewritten below:

$$e(X) = e_{j_1}X^{i_1} + e_{j_2}X^{i_2} + \ldots + e_{j_l}X^{i_l}$$

**Error Values**

We represent an error by $e_{jl}$, where the index $j$ and index $l$ refers to the error location and $l^{th}$ error. We can simply use it as $e_l$ instead of $e_{jl}$.

Now by using Equation (3.24), we have limited number of equations that will be used to find the error values.

After finding these error values put in error polynomial defined in Equation (3.26), use this error polynomial to find estimate of received codeword.

$$\hat{U}(X) = r(X) + \hat{e}(X)$$  \hspace{1cm} (3.27)

Where $\hat{U}(X)$ is estimated received polynomial and $\hat{e}(X)$ is estimated error polynomial.

### 3.4. Common Uses

Linear block codes are commonly used in digital communication systems and data storage systems [11]. Convolutional codes are used in digital video, mobile and satellite communication [31]. Reed-Solomon codes are used in CDs, DVDs and Blu-Ray Disk. Other applications are DSL, WiMAX, DVB, ATSC and in computer applications [30].

### 3.5. Error detection and correction capabilities

In linear block codes $d_{\text{min}}$ is an important parameter to know the error correction capability. Erroneous codewords are at least a distance $d_{\text{min}}$ away from the transmitted codeword. If any error pattern of weight $[(d_{\text{min}} - 1)/2]$ occurs, the Hamming distance between the received vector and the transmitted codeword is less than the distance between the received vector and
other codewords. A linear block code with minimum distance $d_{\text{min}}$ can correct all error patterns of weight less than or equal to $[(d_{\text{min}} - 1)/2]$, denoted by $t$ which is called the error-correction capability of a code with minimum distance $d_{\text{min}}$. It is the upper bound on the weights of all error patterns for which one can correct.

In Convolution codes it is difficult to tell the error correction capability due to its convolutional structure. A Reed-Solomon decoder can detect $d$ errors where $d = n - k + 1$ and can correct up to $\lfloor d/2 \rfloor$ errors.
Chapter 4. Performance Evaluation of Error Correcting Techniques for OFDM Systems

In this chapter we will discuss the work done in this thesis. We will start with the research methodology and then we will explain the implementation. Our main target was to test OFDM systems for channel coding and evaluate the performance based on the improvement in BER. In this chapter we have discussed the way this research was carried out and its implementation technique.

4.1. Research methodology

We adopted quantitative research methodology in this work. We used MATLAB as a tool to implement the OFDM system, error correcting techniques for the OFDM system and after that we made a performance comparison between these techniques.

The outcomes of this thesis will be a BER(Bit Error Rate) vs. SNR(Signal to Noise Ratio) comparison, which will tell the behaviour of all these three codes (Linear block codes, Convolutional codes and Reed-Solomon codes) under different SNR. The assessment will help us to analyze the performance of the three coding techniques in combination with OFDM. The finding of this work may help the OFDM system designers to choose the error correcting codes that match their requirements.

4.2. Implementation

The basic building blocks of an OFDM system are shown in Figure 2.4. The channel encoder and the channel decoder part will be discussed in section 4.2.4 and 4.2.5. The blocks from data modulation to guard interval are discussed in 4.2.1 and the blocks from guard removal to data demodulator are discussed in section 4.2.2. The remaining blocks and flow of the code is discussed in simulation setup in section 5.1.
4.2.1. **OFDM Transmitter**

The OFDM transmitter includes:

1) QPSK modulation
2) Pilot Insertion.
3) IFFT modulation.
4) Cyclic Prefix addition.

QPSK modulation uses 4 distinct phased signals and is assigned to each two bit data. It is done using two MATLAB functions ‘modem.pskmod()’ and ‘modulate function’ ‘modem.pskmod()’. ‘modem.pskmod()’ generates a PSK modulator object which is used by ‘modulate()’ function to modulate the data stream. We used an offset of \( \pi/4 \) and Gray coding is used to have minimum error.

Pilot Insertion is done for the measurement of channel condition. In our work it is done by adding zeros in the 12 unused channels out of the 64 channels of OFDM.

The use of Discrete Fourier Transform (DFT) in the parallel transmission of data using frequency division multiplexing was investigated in 1971 by Weinstein and Ebert [13]. We used IFFT for the multiplexing of data for OFDM.

Cyclic prefix addition is the prefixing of a symbol with repetition at the end. It acts as a guard interval to eliminate the inter symbol interference from the previous symbol [14].

The MATLAB implementation of OFDM transmitter is given in appendix A-4.

4.2.2. **OFDM Receiver**

The OFDM receiver includes:

1) Cyclic prefix removal.
2) FFT demodulation.
3) Pilot removal.
4) QPSK demodulation.
All modules in reverse order to the transmitter and do exactly the opposite job done by their counterpart in the transmitter. But before all these parts, noise is added to the signal as per SNR and there is a channel equalization part to counter the effects of channel fading that occurred in the channel. Cyclic prefix removal is done after channel equalization. FFT demodulation performs the job of de-multiplexing [13]. Pilot removal is done afterwards and QPSK demodulation is done at the end.

Two MATLAB functions ‘modem.pskdemod()’ and ‘demodulate()’ are used to perform the QPSK demodulation.

In our implementation, the transmitter and receiver part do not include the channel coding and decoding but that is done separately.

The MATLAB implementation of OFDM receiver is given in appendix A-5.

4.2.3. **Linear Block Coding**

We used MATLAB function ‘Encode()’ that includes linear block codes. We need to provide the message length \(k\), codeword length \(n\) and generator matrix for computing the codeword. The encoder and decoder for linear block coding are given in appendix A-6 and A-7.

4.2.4. **Convolutional Coding**

The MATLAB function ‘poly2trellis()’ is used for computing the trellis from the polynomial and ‘convenc()’ is the convolutional encoder. At the decoder side the function ‘poly2trellis()’ is used once again for the same purpose. After that Viterbi decoding is used for the convolutional decoding. ‘vitdec()’ function is used and hard decoding is applied on the coded data. The encoder and decoder function for convolutional code are given in appendix A-8 and A-9.

4.2.5. **Reed-Solomon Coding**

For Reed-Solomon encoding we made some changes to the binary data. We first converted the binary data to decimal symbols required to the Reed-Solomon encoder. ‘fec.rsenc()’ function is used for generating Reed-Solomon encoder and ‘encode()’ function is used to encode the data.
The data after encoding is once again changed to binary. The Reed-Solomon encoder and decoder functions are given in appendix A-10 and A-11.

At the decoder side the data is once again converted from binary data to decimal symbols. Function ‘fec.rsdec()’ is used for Reed-Solomon decoder and ‘decode()’ function is used to decode the data. After that the data is once again converted to binary.
Chapter 5. Results and Conclusion

This chapter discusses the results of simulations performed to investigate the performance of the considered error control codes for OFDM systems. On the basis of simulations we have tried to explain the results and discussed the reasons leading to such results.

5.1. Simulation setup

To keep the conditions same for all the simulations the input data, the fading channel and the noise is once generated and saved. The input data is kept same for testing of all coding schemes. The total number of carriers chosen in our OFDM system is 64 and used is 52. We have used Eb/No (Energy per bit to noise power ratio) in our testing. Eb/No is useful when comparing BER in digital modulation schemes without taking the bandwidth into account.

Rayleigh fading [15] channel is once generated and saved. The same fading is used for all the simulations. The fading is defined by the following MATLAB equation [15]:

\[
\text{rayleigh} = \sqrt{0.5 \ast (\text{randn}(N, 1) + i \ast \text{randn}(N, 1))}
\]  

(5.1)

Where ‘randn()’ generates values from the standard normal distribution function and ‘N’ defines the length of fading signal.

The random noise is generated using the ‘randn()’ function and is also generated once and saved. It is recalled in the receiver section and its power is adjusted according to the given Eb/No, after that it is added to the received signal. This way same AWGN is added to the data for all the simulations. Same fading and AWGN helps to get the same error patterns for all the coding techniques.

The final Eb/No changes due to the following reasons:

1. The number of bits after channel coding is more than number of bits before, so it causes decrease in Eb/No.
2. The number of modulated symbols is less than the total number of bits and this increases the value of \( E_b/N_0 \). In our case QPSK was used, which means two bits now make one modulated signal and total four types of modulated signals are there.

3. There are total 64 OFDM carriers in our simulation out of which 52 are used. The unused carriers cause decrease in the effective \( E_b/N_0 \).

4. After cyclic prefix addition there are 16 bits that are repeated in the start of each symbol, so now in each symbol the bits are increased from 64 to 80. This also decreases the value of \( E_b/N_0 \).

So the new adjusted \( E_b/N_0 \) can be given by.

\[
\frac{E_b}{N_0} \text{(adjusted)} = \frac{E_b}{N_0} + 10 \log_{10} \left( \text{code rate} \right) + 10 \log_{10} \left( \log_2(M) \right) \\
+ 10 \log_{10} \left( \frac{\text{Data bits}}{\text{Data bits + Pilot bits + Carrier bits}} \right)
\]

Where ‘code rate’ is code rate of the error control and it becomes 1 for uncoded OFDM, ‘M’ is the number of signals in modulation scheme (For QPSK, \( M=4 \)), Point 3 and 4 are combined together to get Equation 5.2. A similar form of adjusted \( E_b/N_0 \) is given in [16] and we have changed the equation according to our requirement.

The ‘biterr()’ function of MATLAB is used for the computation of BER and number of errors by comparing the binary data. We have used BER vs. \( E_b/N_0 \) plot to evaluate the coding schemes. The three main MATLAB scripts for the three channel coding schemes are given in appendix A-1, A-2 and A-3.

The bits input to the encoder are matched to the bits output of the decoder. BER is defined as the ratio of total number of unmatched bits to the total number of bits.

### 5.2. Results for Linear Block Codes

The simulation is done for code rates of 1/3, 1/2 and 2/3. The generator matrices used for the code were:

\[
G = \begin{bmatrix}
1 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 1 & 1
\end{bmatrix}
\]
The first generator matrix takes two bits as input and outputs six bits so the code rate becomes 1/3. Same is the case with other two generators to get desired code rate. These generators are selected as simplest possible generator matrices. The results of Linear block codes for all the three code rates are shown in Figure 5.1.

\[
G_1 = \begin{bmatrix}
1 & 0 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 1 & 1
\end{bmatrix}
\]

\[
G_2 = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0
\end{bmatrix}
\]

Figure 5.1: BER for Linear Block Codes with different code rates
In Figure 5.1 BER is on logarithmic scale while adjusted Eb/No is in dB. It shows a decrease in BER when the Eb/No is increased. Similarly while comparing the BER before and after channel coding we see an improvement. The BER increases with increase in code rate for a specific value of adjusted Eb/No. The code rate of 1/3 shows approximately 8 dB improvement, the code rate 1/2 shows approximately 6 dB improvement and the code rate 2/3 shows approximately 4 dB improvement as compared to uncoded OFDM at a BER of 10^-3.

5.3. **Results of Convolutional Codes**

For convolutional coding we also used code rates of 1/3, 1/2 and 2/3. The three convolutional encoders are shown below by generator polynomials:

For code rate 1/3 the constraint length $K$ is 3, one input bit and output bits are defined as:

- Output 1 = (1 1 1)_2 = (7)_8
- Output 2 = (0 1 1)_2 = (3)_8
- Output 3 = (1 0 1)_2 = (5)_8

MATLAB parameters can be defined as constraint length = [3], generator = [7 3 5]  

For code rate 1/2 the constraint length $K$ is 3, one input bit and output bits are defined as:

- Output 1 = (1 1 1)_2 = (7)_8
- Output 2 = (1 0 1)_2 = (5)_8

MATLAB parameters can be defined as constraint length = [3], generator = [7 5]  

For code rate 2/3 the constraint length $K$ is 3, two input bits and output bits are defined as:

- Output 1 = (1 1 1)_2 and (0 0 0)_2 = (7)_8 and (0)_8
- Output 2 = (1 0 1)_2 and (0 0 1)_2 = (5)_8 and (1)_8
- Output 3 = (0 1 1)_2 and (1 1 0)_2 = (3)_8 and (6)_8

MATLAB parameters can be defined as constraint length = [3, 3], generator = [7 5 3, 0 1 6].
The results of the convolutional block coding with OFDM are shown in Figures 5.2.

In figure 5.2 the code rate of 1/3 shows approximately 10 dB improvement, the code rate 1/2 shows approximately 7 dB improvement and the code rate 2/3 shows approximately 4.5 dB improvement as compared to uncoded OFDM at a BER of $10^{-3}$.

### 5.4. Results of Reed-Solomon Codes

The code rates of 1/3, 1/2 and 2/3 were also used for testing Reed-Solomon encoder of symbol length 4 bits. The codes used were $(15,5)$ for code rate 1/3, $(15,7)$ for code rate 1/2(approximately) and $(15,10)$ for code rate 2/3.

The results of Reed-Solomon coding with OFDM are shown in Figure 5.3.
In figure 5.3 the code rate of 1/3 shows approximately 8.5 dB improvement, the code rate 1/2 shows approximately 6.5 dB improvement and the code rate 2/3 shows approximately 4 dB improvement as compared to uncoded OFDM at a BER of 10^{-3}.

5.5. **Comparison of the three coding schemes**

After evaluating each technique for different code rates we will now compare all techniques for same code rates. Figure 5.4 shows the comparative plot for code rate 1/3.
Figure 5.4, which is for code rate 1/3 shows that Convolutional code has the best improvement in Eb/No of around 10 dB as compared to uncoded OFDM for a BER of $10^{-3}$. The improvement of RS code is almost the same as Convolutional code while Linear Block code showed less improvement. The codes start with the best BER for Convolutional at 0 dB among the three and all the codes tends to improve its performance as the Eb/No increases. Overall we see that Convolutional codes show best performance.

In Figure 5.5 we have a code rate of 1/2 and here we see that performance of linear block again shows the same low performance compared to the other two techniques. The RS code and convolutional code start performing better at a higher SNR as compared to that their performance for code rate of 1/3. The RS code performs the best among all with a SNR improvement of approximately 6.5 dB at BER=$10^{-3}$. The performance of convolutional is also better than linear block codes.
Figure 5.5: Comparison of uncoded OFDM and three coding schemes for code rate ½

Figure 5.6: Comparison of uncoded OFDM and three coding schemes for code rate 2/3
In Figure 5.6 the code rate was 2/3. The performance of linear block code is almost the same compared to other two techniques. The performance of convolutional code is also good and close but RS shows consistency in its performance and performed well than others at high code rates.

5.6. **Validity Threats**

There are mainly two types of validity threats that may have influence on our research. These two are

1. Internal Validity
2. External Validity

5.6.1 **Internal Validity**

Internal validity threats related to statistical operations in data analysis are avoided through equal sampling for all the techniques. So, sampling doesn’t have any effect on the results of the research. Different code rates are only responsible for the difference in results of all the three techniques as some techniques may perform better at different code rates.

We have used same tools and configurations for all the three techniques that are compared. Therefore there is no possibility of internal threats by instrument change.

5.6.2 **External Validity**

In this research we compared BER by varying signal Eb/No for three different coding techniques at three different code rates. If someone changes the generator matrix then it will change the results. We cannot generalize these results by having different environment other than described in section 5.1.

5.7. **Conclusions and Future Work**

Convolutional codes are very good in performance at lower code rates but due to complex decoding structure they are difficult to implement. Another problem that may arise in them is error propagation while decoding. Convolutional codes have problem of very complex decoding so if we increase the length of input data, the trellis used will be complex and the decoding will
become even more complicated. Linear block codes are very simple to implement. They also show good performance and are also good for lower code rates and highly suitable for application with less complexity requirement and not so high performance requirement. Reed-Solomon codes in comparison to linear block codes are difficult to implement but their performance is much better and consistent than linear block codes since they can handle long bursts of errors. These codes showed good performance on all three tested code rates. The systems where very high performance is required Reed-Solomon codes are recommended and systems that don’t allow complicated structure linear block codes are the preferable solution.

Future work of this thesis can be comparison of other error correcting techniques. Another future work can be finding the trade-off between performances and complexity when the number of bits to the encoder is increased.
References


[37] A. Bovik, ”The Essential guide to Image processing”, LondonWC1X 8RR, UK


Appendix

A-1: Main Code for Linear Block Codes

clc, close all, clear all

% Parameters
nbitpersym = 52; % number of bits per OFDM symbol (same as the number of
subcarriers for BPSK)
% nsym = 13500; % number of symbols
len_fft = 64; % fft size
sub_car = 52; % number of data subcarriers
EbNo = 0:16;

% Channel Coding setting
%For code rate = 1/3
n=6;,k=2;
genmat = [eye(2) [1 0 1; 0 1 1]]; % %For code rate = 1/2
% n=6;,k=3;
% genmat = [1 0 0 1 0 1;0 1 1 0 0 1 1 1 ];
%For code rate = 2/3
% n=9;,k=6;
% genmat = [eye(6),[1 1 0; 0 1 1; 1 0 1; 1 11; 0 0 1; 1 0 0]];

% Data generation
loadInputData.mat;

% Channel Coding
[t_data1] = LBlock_Encoder( t_data2,n,k,genmat );

% OFDM Transmitter
nsym = length(t_data1)/104;
[ser_data] = OFDM_transmitter(t_data1,nbitpersym,nsym); % ser_data is
transmitted data

% Passing through channel, applying fading.
loadfading.mat; % using fixed(once generated and saved) fading data for all
simulation
fading = fading(1:size(ser_data)); % Fading signal length adjusted to match
passing signal
chan_data = fading.*ser_data; % Applying Channel Fading

% Receiver end
no_of_error=[];
LB_BER=[];

for ii=1:length(EbNo)
ii
% OFDM Receiver
[demod_Datal] = OFDM_reciever( chan_data,EbNo(ii),fading,nbitpersym,nsym );
% Channel Decoding
[decoded_data] = LBlock_Decoder( demod_Data1,n,k,genmat );

% Error Calculation
[no_of_error(ii),LB_BER(ii)] = biterr(decoded_data,t_data2) ;
% error rate calculation ;

end

% plotting the result
close all
load uncoded_OFDM.mat
EbNo_LB=EbNo+10*log10(k/n)+10*log10(2)+10*log10(52/80);
figure, semilogy(EbNo_LB,LB_BER,'--or',EbNo_uncoded,OFDM_BER,'--.k');
gridon, xlabel('Eb/No'), ylabel('BER');
title('Bit error probability curve using Linear Block Coding and OFDM');
legend('BER LB Coding', 'BER Uncoded OFDM')

% For code rate =1/3
EbNo_LB1b3=EbNo_LB;
LB_BER1b3=LB_BER;
save('LBdata_1b3.mat','LB_BER1b3','EbNo_LB1b3');

% For code rate =1/2
% EbNo_LB1b2=EbNo_LB;
% LB_BER1b2=LB_BER;
% save('LBdata_1b2.mat','LB_BER1b2','EbNo_LB1b2');

% For code rate =2/3
% EbNo_LB2b3=EbNo_LB;
% LB_BER2b3=LB_BER;
% save('LBdata_2b3.mat','LB_BER2b3','EbNo_LB2b3');

A-2: Main code for convolutional codes with OFDM

clc, close all, clear all

% Parameters
nbitpersym = 52;  % number of bits per OFDM symbol (same as the number of
                  % subcarriers for BPSK)
nsym        = 13500; % number of symbols
len_fft     = 64;   % fft size
sub_car     = 52;   % number of data subcarriers
EbNo        = 0:15;

% Channel Coding
% For Code rate = 1/3
% k=1;n=3;t=1;
% const_length=[3];
% generator=[7 3 5];

% For Code rate = 1/2
% k=1;n=2;t=1;
% const_length=[3];
% generator=[7 5];
%
% For Code rate = 2/3
k=2;n=3;t=2;
const_length=[3 3];
generator=[7 5 3;0 1 6];

% Data generation
loadInputData.mat;

% Channel Coding
[ t_data1 ] = Conv_encoder( t_data2,const_length,generator);

% OFDM Transmitter
nsym = length(t_data1)/104;
[ ser_data ] = OFDM_transmitter(t_data1,nbitpersym,nsym); % ser_data is transmitted data

% Passing through channel
loadfading.mat; % using fixed(once generated and saved)fading data for all simulation
fading = fading(1:size(ser_data)); % Fading signal length adjusted to match passing signal
chan_data = fading.*ser_data; % Applying Channel Fading

% Receiver end
no_of_error=[];
CONV_BER=[];
for ii=1:length(EbNo)

% OFDM Receiver
[ demod_Data1 ] = OFDM_reciever( chan_data,EbNo(ii),fading,nbitpersym,nsym );

% Channel Decoding
tblen = 48;
[ demod_Data ] = Conv_decoder( demod_Data1,const_length,generator,tblen );

% Error Calculation
[no_of_error(ii),CONV_BER(ii)]=biterr(demod_Data(tblen*t+1:end),t_data2(1:end-tblen*t)); % error rate calculation

end

% plotting the result
closeall
loaduncoded_OFDM.mat
EbNo_CONV=EbNo+10*log10(k/n)+10*log10(2)+10*log10(52/80);
figure, semilogy(EbNo_CONV,CONV_BER,'--or',EbNo_uncoded,OFDM_BER,'--.k');
gridon, xlabel('Eb/No'), ylabel('BER');
title('Bit error probability curve using Convolutional Coding and OFDM');
legend('BER Convolutional Coding', 'BER Uncoded OFDM')
% %For code rate =1/3
% EbNo_CONV1b3=EbNo_CONV;
% CONV_BER1b3=CONV_BER;
% save('CONVdata_1b3.mat','CONV_BER1b3','EbNo_CONV1b3');

%For code rate =1/2
% EbNo_CONV1b2=EbNo_CONV;
% CONV_BER1b2=CONV_BER;
% save('CONVdata_1b2.mat','CONV_BER1b2','EbNo_CONV1b2');

% %For code rate =2/3
EbNo_CONV2b3=EbNo_CONV;
CONV_BER2b3=CONV_BER;
save('CONVdata_2b3.mat','CONV_BER2b3','EbNo_CONV2b3');

A-3: Main Code for RS codes and OFDM

clc, close all, clear all

% Parameters
nbitpersym  = 52;   % number of bits per OFDM symbol (same as the number of
subcarriers for BPSK)
len_fft     = 64;   % fft size
sub_car     = 52;   % number of data subcarriers
EbNo        = 0:16;

% Channel Coding setting
% For Code rate = 1/3
% n=15;
% k=5;
% For Code rate = 1/2
% n=15;
% k=7;
% For Code rate = 2/3
n=15;
k=10;

% Data generation
loadInputData.mat;
% These two steps are to match the exact input length required by
si=floor(length(t_data2)/(log2(n+1)*104*k))*(log2(n+1)*104*k);
t_data2=t_data2(1:si);

% Channel Coding
[t_data1 ] = RS_Encoder( t_data2,n,k);

% OFDM Transmitter
nsym = length(t_data1)/104
[ ser_data ] = OFDM_transmitter(t_data1,nbitpersym,nsym); % ser_data is transmitted data

% Passing through channel, applying fading.
loadfading.mat; % using fixed(once generated and saved)fading data for all simulation
fading = fading(1:size(ser_data)); % Fading signal length adjusted to match passing signal
chan_data = fading.*ser_data; % Applying Channel Fading

% Receiver end
no_of_error=[];
RS_BER=[];

for ii=1:length(EbNo)
ii

% OFDM Receiver
[ demod_Data ] = OFDM_reciever( chan_data,EbNo(ii),fading,nbitpersym,nsym );

% Channel Decoding
[ decoded_output ] = RS_Decoder( demod_Data,n,k);

% Error Calculation
[no_of_error(ii),RS_BER(ii)]=biterr(decoded_output,t_data2) ;

end

closeall
loaduncoded_OFDM.mat
EbNo_RS=EbNo+10*log10(k/n)+10*log10(2)+10*log10(52/80);
figure, semilogy(EbNo_RS,RS_BER,'--or',EbNo_uncoded,OFDM_BER,'--.k');
gridon, xlabel('Eb/No'), ylabel('BER');
title('Bit error probability curve using RS Block Coding and OFDM');
legend('BER RS Coding', 'BER Uncoded OFDM')

% % For code rate =1/3
% EbNo_RS1b3=EbNo_RS;
% RS_BER1b3=RS_BER;
% save('RSdata_1b3.mat','RS_BER1b3','EbNo_RS1b3');
%
% For code rate =1/2
% EbNo_RS1b2=EbNo_RS;
% RS_BER1b2=RS_BER;
% save('RSdata_1b2.mat','RS_BER1b2','EbNo_RS1b2');
%
% For code rate =2/3
EbNo_RS2b3=EbNo_RS;
RS_BER2b3=RS_BER;
save('RSdata_2b3.mat','RS_BER2b3','EbNo_RS2b3');

A-4: OFDM Transmitter Function
function [ ser_data ] = OFDM_transmitter(data,nbitpersym,nsym)
% Transmitter Function for OFDM.
% INPUTS: The function takes 'data' as input bits, 'nsym' as number of symbols and
% 'nbitpersym' as number of bits per symbol.
% OUTPUTS: The output of function is 'ser_data' which is serial data ready for transmission and to be passed from channel.

data_bits=bi2de([reshape(data,2, length(data)/2)])'; % Reshape data for QPSK modulation
% QPSK modulating data
M= modem.pskmod('M', 4, 'PHASEOFFSET', pi/4, 'SYMBOLORDER', 'GRAY'); % QPSK Modulator with offset of pi/4 radian and gray code order
mod_data = modulate(M,data_bits); % Modulating data
% serial to parallel conversion
par_data = reshape(mod_data,nbitpersym,nsym).';
% pilot insertion
pilot_ins_data=[zeros(nsym,6) par_data(:,[1:nbitpersym/2]) zeros(nsym,1) par_data(:,[nbitpersym/2+1:nbitpersym]) zeros(nsym,5)];
% IFFT and normalizing
IFFT_data = (64/sqrt(52))*ifft(fftshift(pilot_ins_data.'));
% addition cyclic prefix
cyclic_add_data = [IFFT_data(:,[49:64]) IFFT_data.'];
% parallel to serial conversion
ser_data = reshape(cyclic_add_data,80*nsym,1);
end

A-5: OFDM Receiver Function

function [ output ] = OFDM_reciever( chan_data,EbNo,fading,nbitpersym,nsym )
% Reciever Function for OFDM.
% INPUTS: The function takes 'chan_data' which is data from channel, 'EbNo' is
% SNR, 'Ray_est' is estimation of channel fading,'nsym' as number of symbols and
% 'nbitpersym' as number of bits per symbol.
% OUTPUTS: The output of function is 'output' which is final data after
% reciever's processing and demodulation.

sigp_dB = 10*log10(norm(chan_data,2)^2/numel(chan_data)); % Signal Power in dB
noisep_dB = sigp_dB-EbNo; % Noise Power in dB
noisep = 10^(noisep_dB/10); % Noise Poise in linear scale
loadnoise1.mat; % using fixed(once generated and saved)noise for all simulation
% Noise*standard deviation to achieve SNR, length is adjusted according to
% signal length
noise = sqrt(noisep)*noise(1:length(chan_data),1);
noise = sqrt(noisep)*(noise(1:length(chan_data),1)+noise(1:length(chan_data),1)*li);
chan_awgn= chan_data+noise; % Noise addition in signal

% chan_awgn = awgn(chan_data,EbNo,'measured'); % awgn addition equivalent
% to above code of Noise but Noise pattern is different for every simulation
Ray_est = fading+0.01*(randn(size(fading))+randn(size(fading))*1i); % Fading Estimation
chan_ray = chan_awgn./Ray_est; % Rayleigh Channel Estimation and cancellation

% serial to parallel conversion
ser_to_para = reshape(chan_ray,80,nsym).';

%cyclic prefix removal
cyclic_pre_rem = ser_to_para(:,[17:80]);

% freq domain transform
FFT_recdata =(sqrt(52)/64)*fftshift(fft(cyclic_pre_rem.'));'

% Pilot Removal
rem_pilot = FFT_recdata (:,6+[1:nbitpersym/2] 7+[nbitpersym/2+1:nbitpersym]);
% pilot removal

% Serial Conversion
ser_data_1 = reshape(rem_pilot.',nbitpersym*nsym,1);

% QPSK Demodulation
z= modem.pskdemod('M', 4, 'PHASEOFFSET', pi/4, 'SYMBOLORDER', 'GRAY');
demod_Data = demodulate(z,ser_data_1); % demodulating the data
out2=[de2bi(demod_Data)]';
output =out2(:);

disp('A-6: Linear Block Encoder Function')

function [ coded_output ] = LBlock_Encoder( data,n,k,genmat )
% Function for Linear Block encoding of data.
% Inputs: The function takes 'data' as the data to be encoded, 'k' is
% length of msg bits into the encoder, 'n' is the coded msg bits out of the
% encoder while 'genmat' defines the generator matrix required for coding.
% Outputs: The coded output from the Linear Block encoder

coded_output = encode(data,n,k,'linear',genmat);
end

A-7: Linear Block Decoder Function
function [decoded_data] = LBlock_Decoder(code_data,n,k,genmat)
% Function for linear block decoding of data.
% Inputs: The function takes 'code_data' as the coded data to be decoded, 'k' is
% length of decoded msg bits out of the decoder, 'n' is the coded msg bits in to the
% decoder while 'genmat' defines the generator matrix required for decoding.
% Outputs: The decoded output from the linear block decoder

[decoded_data,err] = decode(code_data,n,k,'linear',genmat);
end

A-8: Convolutional Encoder Function

function [coded_data] = Conv_encoder(data,length,generator)
% Function of convolution encoding of data.
% Inputs: The function takes 'data' as the data to be encoded, length is
% the constraint length of the convolutional encoder, generator is the
% polynomial defining the convolutional encoder.
% Outputs: The coded output from convolutional encoder

trel = poly2trellis(length,generator); % Trellis designing from polynomials
coded_data = convenc(data,trel); % Encode the message using the trellis.
end

A-9: Convolutional decoder Function

function [decoded_data] = Conv_decoder(code_data,length,generator,tblen)
% Function of convolution encoding of data.
% Inputs: The function takes 'data' as the data to be encoded, length is
% the constraint length of the convolutional encoder, generator is the
% polynomial defining the convolutional encoder, tblen is the trackback
% length for viterbi decoder.
% Outputs: The coded output from convolutional encoder

trel = poly2trellis(length,generator);
qcode = quantiz(code_data,[0.001,.1,.3,.5,.7,.9,.999]);
decoded_data = vitdec(qcode,trel,tblen,'cont','soft',3);
end

A-10: Reed Solomon Encoder Function

function [coded_output] = RS_Encoder(data,n,k)
% Inputs: The function takes 'data' as the data to be encoded, length is
% the constraint length of the convolutional encoder, generator is the
% polynomial defining the convolutional encoder.
% Outputs: The coded output from convolutional encoder

l = log2(n+1);
r_data = bi2de([reshape(data,l,length(data)/l)]');
enc = fec.rsenc(n,k);
code = encode(enc,r_data);
coded_output = [reshape([de2bi(code)]',length(code)*l,1)];
A-11: Reed-Solomon decoder Function

```matlab
function [ decoded_output ] = RS_Decoder( code_data,n,k )
    r = log2(n+1);
    r_code_data = bi2de(reshape(code_data,l,length(code_data)/l))';
    decoder = fec.rsdec(n,k);
    [decoded,cnumerr,ccode] = decode(decoder,r_code_data);
    decoded_output = reshape([de2bi(decoded)'],length(decoded)*l,1);
end
```

A-12: Uncoded OFDM code

```matlab
clc, clear all

% Parameters
nbitpersym = 52;   % number of bits per OFDM symbol (same as the number of
subcarriers for BPSK)
nsym = 12600; % number of symbols
len_fft = 64;   % fft size
sub_car = 52;   % number of data subcarriers
EbNo = 0:14;
k = 1/3; % code rate of Encoder

% Data generation

% load data.mat;
% t_data1=randdata(1:(nbitpersym*2*nsym),1);
load InputData.mat;
nsym = length(t_data2)/104;

% OFDM Transmitter
[ ser_data ] = OFDM_transmitter(t_data2,nbitpersym,nsym); % ser_data is
transmitted data

% Passing through channel, applying fading.
sizel=size(ser_data);
loadfading.mat;
fading = fading(1:size(ser_data));
% fading = sqrt(0.5*(randn(sizel) + 1i*randn(sizel))); % Fading channel
% fading=1;
chan_data = fading.*ser_data; % Applying Channel Fading
% CHAN = RAYLEIGHCHAN;
% chan_data = filter(CHAN,ser_data); % Applying Channel Fading

% Receiver
no_of_error=[];
OFDM BER=[];
```
for ii=1:length(EbNo)
    % OFDM Receiver
    [ demod_Data1 ] = OFDM_reciever( chan_data,EbNo(ii),fading,nbitpersym,nsym );

    % Error Calculation
    [no_of_error(ii),OFDM_BER(ii)]=biterr(demod_Data1,t_data2) ; % error rate calculation
end

% plotting the result
figure,
EbNo_uncoded=EbNo+ 10*log10(2)+10*log10(52/64);
semilogy(EbNo_uncoded,OFDM_BER,'--.r');
gridon, xlabel('Eb/No'), ylabel('BER');
title('Bit error probability curve using uncoded OFDM');
save('uncoded_OFDM.mat','OFDM_BER','EbNo_uncoded');

A-12: Uncoded OFDM code

noise = randn(100000,1);
save('noise1.mat','noise');
t_data2 = randint(18720,1);
save('InputData.mat','t_data2');
fading = sqrt(0.5*(randn(10000,1) + 1i*randn(10000,1)));
save('fading.mat','fading');