NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

REED-MULLER CODES IN ERROR CORRECTION IN WIRELESS ADHOC NETWORKS

by

Serdar U. Tezeren

March 2004

Thesis Advisor:	Murali Tummala
Co-Advisor:	Roberto Cristi

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** March 2004	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: Reed-Muller Codes in Error Correction in Wireless Adhoc Networks			**5. FUNDING NUMBERS**
6. AUTHOR(S) Tezeren, Serdar Umit			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

The IEEE 802.11a standard uses a coded orthogonal frequency division multiplexing (COFDM) scheme in the 5-GHz band to support data rates up to 54 Mbps. The COFDM was chosen because of its robustness to multipath fading affects. In the standard, convolutional codes are used for error correction. This thesis examines the performance of the COFDM system with variable rate Reed-Muller (RM) error correction codes with a goal to reduce the peak-to-average power ratio (PAPR). Contrary to the expectations, RM codes did not provide expected improvement in PAPR reduction. Peak clipping and Hanning windowing techniques were investigated in order to reduce the PAPR. The results indicate that a tradeoff exists between the PAPR and the bit-error rate (BER) performance. Although peak clipping yielded considerable reduction in PAPR, it required high signal-to-noise ratios. On the other hand, Hanning windowing provided only a small reduction in PAPR with reasonable BER performance.

14. SUBJECT TERMS COFDM, Reed-Muller Error Correction Codes, Convolutional Codes, PAPR, QPSK, Multipath Fading, Peak Clipping, Hanning Windowing, IEEE 802.11a Standard, Outdoor Wireless Digital Communication Channel, Indoor Channel Characteristics, Delay, Doppler Effect.			**15. NUMBER OF PAGES** 153
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

REED-MULLER CODES IN ERROR CORRECTION IN WIRELESS ADHOC NETWORKS

Serdar U. Tezeren
Lieutenant Junior Grade, Turkish Navy
B.S., Turkish Naval Academy, 1998

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March 2004

Author:	Serdar U. Tezeren
Approved by:	Murali Tummala Thesis Advisor
	Roberto Cristi Co-Advisor
	John Powers Chairman, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The IEEE 802.11a standard uses a coded orthogonal frequency division multiplexing (COFDM) scheme in the 5-GHz band to support data rates up to 54 Mbps. The COFDM was chosen because of its robustness to multipath fading affects. In the standard, convolutional codes are used for error correction. This thesis examines the performance of the COFDM system with variable rate Reed-Muller (RM) error correction codes with a goal to reduce the peak-to-average power ratio (PAPR). Contrary to the expectations, RM codes did not provide expected improvement in PAPR reduction. Peak clipping and Hanning windowing techniques were investigated in order to reduce the PAPR. The results indicate that a tradeoff exists between the PAPR and the bit-error rate (BER) performance. Although peak clipping yielded considerable reduction in PAPR, it required high signal-to-noise ratios. On the other hand, Hanning windowing provided only a small reduction in PAPR with reasonable BER performance.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

First, I would like to thank my long-time friend, A. Yasin Erdogan for his friendship and cooperation during my education at the Naval Postgraduate School and the development of our theses.

Secondly, I thank my thesis advisors, Dr. Murali Tummala and Dr. Roberto Cristi for their suggestions and advise during the development of this thesis.

Special thanks to Mr. Benjamin Cooke from the University of Duke for his support of and patience with my never-ending questions, and also to my mother, Seval Tezeren, for the sacrifices she made in support of the completion of this thesis in her two-month vacation.

This thesis is dedicated to my father, Cetin Tezeren.

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

Since, wireless local area networks (WLAN) were increasingly becoming more and more important for broadband wireless communications in both military and commercial application, the main goal of this thesis was to investigate the performance of a coded orthogonal frequency multiplexing (COFDM) system utilizing Reed-Muller (RM) codes. In order to achieve this goal, three different types of RM codes and for comparison a rate one-half convolutional code are investigated through simulations. Also Hanning windowing and peak clipping techniques were investigated for reducing the peak-to-average ratio (PAPR) in COFDM systems.

The performance of the COFDM system was studied under indoor and oudoor channel environments using five different channels. The bit-error performance curves for each of the RM codes and convolutional code were obtained.

The results showed that the COFDM system is robust in indoor channel environments. However, for the outdoor channel environments, the system requires higher signal-to-noise ratios (SNR) to achieve the same bit-error rate (BER) performance as in the indoor case. Also, BER performance curves showed that the RM codes provide better performance in indoor channel environment at low SNR. However, convolutional codes provide better performance in outdoor channel environments.

Reed-Muller codes are straightforward to implement, and they provide a wide range of coding options. However, RM codes did not provide the expected improvement in PAPR.

The addition of Hanning windowing and peak clipping improve PAPR reduction. While improvement with Hanning windowing is limited, peak clipping provides remarkable reduction in PAPR but at the cost of increased bit error.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

Wireless local area networks (WLAN) are increasingly becoming more and more important for broadband wireless communications in both military and commercial applications. The IEEE 802.11a WLAN standard uses the orthogonal frequency division multiplexing (OFDM) to support data rates from 6 to 54 Mbps and operates in the 5-GHz band. OFDM is a multicarrier transmission method in which a high-rate data sequence is split into multiple low rate subblocks, and each subblock of data is modulated onto separate subcarriers. Specifically, IEEE 802.11a uses coded OFDM (COFDM) because of its robustness under multipath fading conditions.

A. OBJECTIVE

It is well known that multicarrier communication systems suffer from high peak-to-average power ratio (PAPR) because of the constructive interference of the signals. There are numerous methods to reduce the PAPR of the COFDM signal. One of the mitigation techniques is to use Reed-Muller (RM) codes.

The main goal of this thesis was to analyze the COFDM system utilizing RM codes. In order to achieve this goal, three different types of RM codes and for comparison a rate one-half convolutional code were investigated through simulation studies. Also, Hanning windowing and peak clipping techniques were investigated for reducing the PAPR in OFDM systems.

Although binary pulse shift keying (BPSK), quadrature pulse shift keying (QPSK), 16 quadrature amplitude modulation (16-QAM), and 64-QAM techniques are used in IEEE 802.11a WLAN systems, only the QPSK modulation technique was implemented in the simulations here. Using four different simulation steps, the performance of the COFDM system was studied under both indoor and outdoor channel environments. The bit-error rate (BER) performance curves for each of the RM codes and the convolutional code were obtained.

B. RELATED WORK

The RM coding algorithms in [1] provided a starting point for the implementation of encoding and decoding algorithms of RM codes in the simulations here. The Hanning windowing and the peak clipping algoritms used in this work were first introduced in [2].

The MATLAB simulation code used in this thesis was first developed by Roderick [3] for line-of-sight ship-to-ship, ship-to-shore and ship-to-relay type of connectivity. Tan [4] modified the MATLAB code in [3] to develop a simulation of the IEEE 802.11a standard's physical layer for indoor channel environments. Convolutional codes with hard decision decoding technique were used. The author used the MATLAB code in [4] as a starting point for developing the simulations in this thesis.

C. THESIS ORGANIZATION

Chapter II presents the basics of COFDM and its schematic structure, the definition of PAPR, and a general overview of the various functional blocks in an OFDM system. Interleaving and guard interval are explained as part of the COFDM system. Chapter III focuses on the coding scheme. The Hanning windowing and peak clipping techniques are described. The simulation methodology and test results are presented in Chapter IV. Finally, the thesis concludes with Chapter V.

Appendix A describes the MATLAB architecture. The transmitter, receiver and channel parts of the simulation model are explained, and MATLAB functions are described. Appendix B includes the MATLAB code.

II. CODED ORTHOGONAL FREQUENCY DIVISION MULTIPLEXING (COFDM) AND PEAK-TO-AVERAGE POWER RATIO (PAPR)

This chapter presents the basics of the coded orthogonal frequency division multiplexing (COFDM) technique and the advantages and disadvantages of COFDM communication systems. As it is one of the main disadvantages of the multicarrier communication systems, the peak-to-average power ratio (PAPR) in OFDM systems is also discussed.

A. CODED ORTHOGONAL FREQUENCY DIVISION MULTIPLEXING (COFDM)

In COFDM, the total bandwidth is divided into N nonoverlapping frequency subchannels. Each subchannel is modulated with a different symbol and then the N subchannels are frequency-multiplexed. While conventional frequency division multiplexing (FDM) successfully avoids spectral overlap of channels and eliminates interchannel interference (ICI), it makes the frequency spectrum usage inefficient. OFDM uses the spectrum more efficiently than the conventional frequency division multiplexing (FDM) by allowing frequency overlapping.

Figure 1 illustrates the difference between the nonoverlapping and overlapping multicarrier modulation techniques. It can be seen that almost half of the bandwidth is saved by overlapping the spectra.

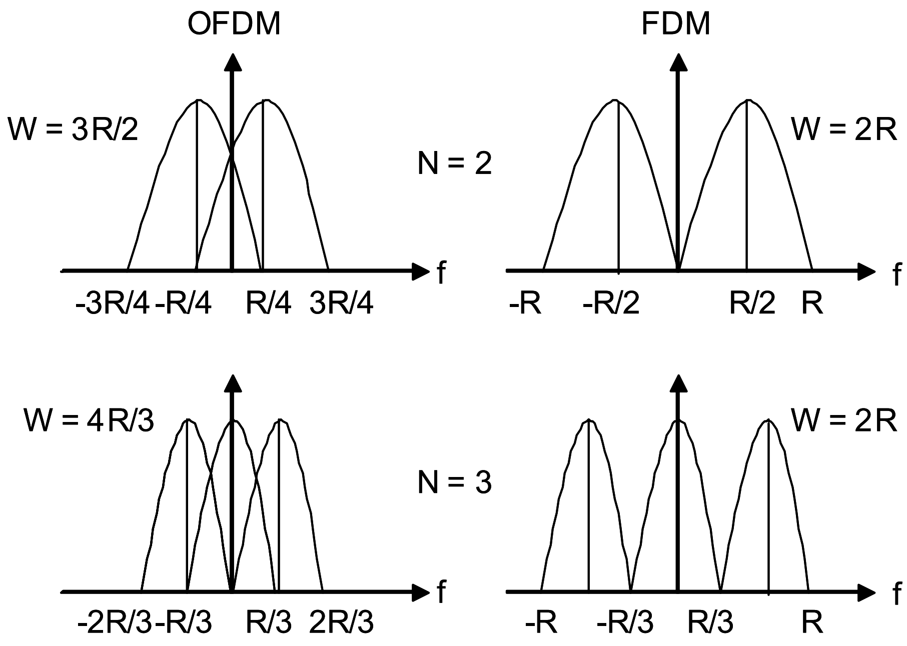


Figure 1. Frequency Efficiency of OFDM over FDM Illustrated for Two and Three Subchannels (After Ref. 5.)

OFDM also requires that the subchannel carrier frequencies are orthogonal. Two subcarriers are orthogonal if their inner product is zero. The COFDM subcarrier signals can be arranged such that the sidebands of the carriers overlap but the signals can still be received without ICI. In COFDM, cross talk among the subcarriers is prevented by making the subcarriers orthogonal. In other words, each subcarrier must be spaced at intervals of $1/T_s$ where T_s is the symbol duration for each subcarrier. A representative COFDM signal is shown in Figure 2. Individual COFDM subchannel response is shown in Figure 2(a) and the COFDM signal spectrum for $N = 4$ is shown in Figure 2(b). It can be seen that the spectra of the subcarriers are not separated but overlapped in frequency.

The discrete Fourier transform (DFT) is used to accomplish frequency multiplexing and subcarrier orthogonality. The transmitter uses an IDFT operation to modulate subblocks of data symbols. A DFT is used at the receiver where the signal is sampled at the center frequency of each subchannel, and the transmitted data are recovered without ICI. From Figure 2, it can be seen that, at the center frequency of a subchannel, the sidelobes due to all the other channels produce a null.

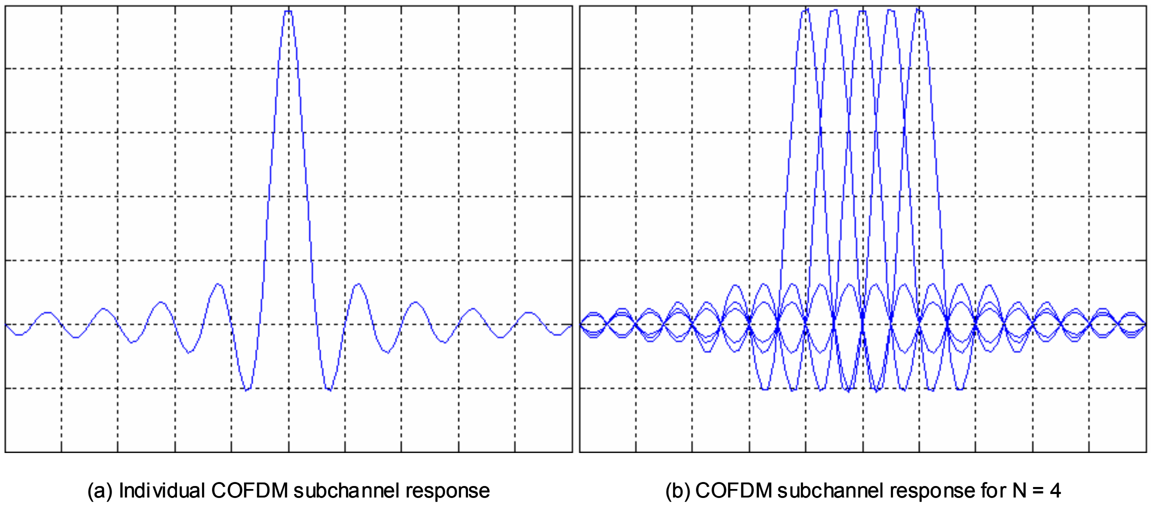


Figure 2. Spectra of an OFDM Subchannel and OFDM Signal (From Ref. 5.)

The fast Fourier transform (FFT) algorithm is used to implement both the IDFT at the transmitter and the DFT in the receiver, thus making it computationally efficient.

The COFDM divides a symbol sequence into *N* symbol groups so that each group contains a symbol sequence of rate *1/N* of the original sequence. The COFDM signal consists of *N* orthogonal subcarrier signals, where each of subcarrier is modulated by a different symbol sequence. Lengthening the symbol duration helps eliminate the use of an equalizer in the receiver.

COFDM reduces the effects of frequency selective fading or narrowband interference. If fading or interference takes place in a single carrier system, it can fail the whole system while only some of the subcarriers will be affected in a multicarrier system. Additionally, by using error correction coding, the COFDM system performance can be further improved.

A COFDM system uses the frequency spectrum very efficiently since it allows overlapping. In COFDM, the use of cyclic extension prevents intersymbol interference (ISI) and ICI problems. Dividing the channel into subchannels makes the COFDM system more robust to frequency selective fading effects while channel coding and interleaving make it more robust to errors due to the channel noise. These features of the COFDM are considered the key advantages. However, besides these advantages, the system has also drawbacks as it requires power amplifiers with a high PAPR in order to provide a large effective range and is prone to frequency offset relative to single carrier systems.

The efficiency in frequency spectrum usage and the robustness to multipath fading make COFDM desirable in a number of applications, such as digital audio broadcasting (DAB), high rate digital subscriber line (HDSL), very high rate digital subscriber line (VHDSL), asymmetric DSL (ADSL), HDTV terrestrial, IEEE 802.11 and HiperLAN/2, general switched telephone network (GSTN), cellular radio, and digital video broadcasting (DVB-T). [6]

B. COFDM SCHEMATIC DIAGRAM

The basic principle of COFDM, as mentioned in the previous section, is to divide a high-rate data stream into *N* lower rate streams and to transmit them at the same time over a number of subcarriers. Since the symbol duration is increased, the relative amount of dispersion in time caused by multipath delay spread is decreased. Intersymbol interference (ISI) is another problem, which can almost be eliminated by introducing a guard time in every COFDM symbol. In order to avoid the ICI, a COFDM symbol is cyclically extended by adding a guard time. A general block diagram of the transmitter and the receiver for the COFDM scheme is shown in Figure 3. The COFDM is used in 802.11a standard. In the following discussion as we describe the COFDM system, frequent references will be made to this standard.

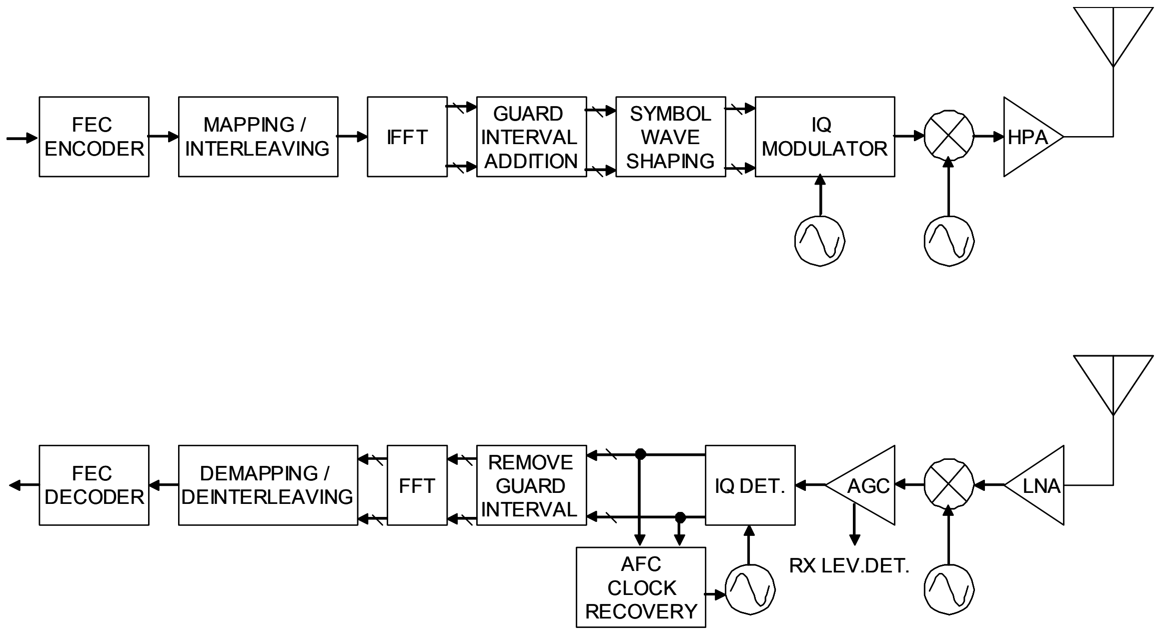


Figure 3. Block Diagram of COFDM Transmitter and Receiver (From Ref. 7.)

1. Channel Coding

In the IEEE 802.11a standard, data is encoded with a convolutional encoder with a coding rate $R = 1/2,\ 2/3$ or $3/4$. The convolutional encoder uses the industry-standard generator polynomials, $g_0 = 133_8$ and $g_1 = 171_8$ of rate $R = 1/2$ [6]. The code rate for the convolutional code can be changed by using a puncturing process. In this thesis, the data are also encoded with RM codes. The details of RM codes are explained in Chapter III.

2. Block Interleaver

If decoding errors occur in a codeword and these are passed to the next block, they may affect the performance of the entire system. The performance of the system can be improved if these errors are distributed over the other code words. This can be achieved by interleaver/deinterleaver.

A block interleaver consists of a two-dimensional array, into which the data are read along its rows. When the array is full, the data are read out by the columns, thus the order of the data is permuted. The original order can be received by the corresponding deinterleaver in which the data are read in by columns and read out by rows.

3. Symbol Mapping

The interleaved and rearranged data are mapped onto constellation points in accordance with the modulation type. Figure 4 shows QPSK constellation points.

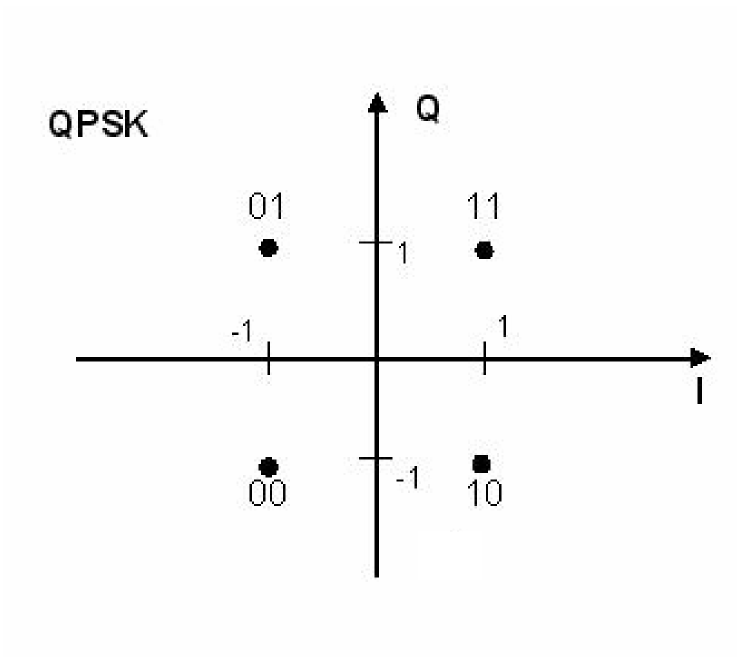


Figure 4. QPSK Constellation points (From Ref. 7.)

4. Discrete Fourier Transform (DFT)

There are 52 subcarriers per channel in a IEEE 802.11a standard WLAN system, where 48 of these subcarriers carry data and the remaining four subcarriers are used as pilot tones. After serial-to-parallel conversion, each COFDM symbol is modulated over 52 subcarriers by applying an inverse fast Fourier transform (IFFT).

5. Guard Interval and Cyclic Extension

A guard time is added to each COFDM symbol to eliminate the ISI and ICI, and it is removed before the FFT operation at the receiver. Since the other parameters are chosen according to the guard interval time, it is an important parameter for the COFDM system. As long as the guard time is larger than the expected delay spread, multipath components from one symbol do not cause interference with the other symbol.

A zero-padding extension could be used as guard interval; however, this can not eliminate ICI effects. Therefore, the COFDM symbol is cyclically extended into the guard interval. The cyclic prefix for the transmission signal consists of a specific part of last samples that are added to the beginning of the signal. Consequently, we have an extended signal. To ensure orthogonality between different subcarriers, it is required that all subcarriers differ by an integer number of cycles within the FFT integration time. Although the addition of guard interval eliminates the ICI and ISI effects, it causes inefficient use of the frequency spectrum

6. Modulator

The COFDM signal then is upconverted to the 5-GHz band and transmitted over the channel.

7. Channel

The presence of noise in the channel affects the ability to make correct decisions about the received symbols at the receiver part of the communication system, thereby limiting the data transmission rate.

For additive white Gaussian noise (AWGN), a received signal is typically modelled as an attenuated desired signal in AWGN with the noise having a power spectral density (PSD) given by

$$G_n(f) = \frac{N_0}{2} \text{ W/Hz} \tag{2-1}$$

A free space propagation model may be adequate to analyze the satellite communication systems since there is always a line-of-sight (LOS) component and almost no multipath fading effects. However, as a signal travels from the transmitter to the receiver in the channels of interest, the path it takes can vary from a simple LOS to one that is severely obstructed by buildings, hills, and trees, which lead to multipath propagation. The free space propagation model may not be adequate to analyze and describe the character istics of a channel. Because of the uneven terrain and other obstructions, a multipath model is desired. Consequently, the received signal undergoes fading as it may consist of multiple waves, which have randomly distributed amplitudes and phases.

The mean excess delay, rms delay spread, and excess delay spread are parameters used to characterize multipath channels. The mean excess delay ($\bar{\tau}$) and rms delay spread (σ_τ) of multipath channels are used to quantify the time dispersive properties of a multipath channel. The mean excess delay is the first moment of the power delay profile as given by

$$\bar{\tau} = \frac{\sum_k a_k^2 \tau_k}{\sum_k a_k^2} = \frac{\sum_k P(\tau_k)\tau_k}{\sum_k P(\tau_k)} \tag{2-2}$$

and the rms delay spread is the square root of the second central moment of the power delay profile defined as

$$\sigma_\tau = \sqrt{\overline{\tau^2} - (\bar{\tau})^2} \tag{2-3}$$

where

$$\overline{\tau^2} = \frac{\sum_k a_k^2 \tau_k^2}{\sum_k a_k^2} = \frac{\sum_k P(\tau_k)\tau_k^2}{\sum_k P(\tau_k)}, \tag{2-4}$$

where a_k is absolute power level. These parameters are measured relative to the first detectable signal arriving at $\tau_0 = 0$. Typical values of rms delay spread are on the order of microseconds in outdoor channels and on the order of nanoseconds in indoor channels. [8]

Due to the relative motion between the transmitter and the receiver, each multipath signal is subjected to *Doppler shift*. The Doppler shift is computed by

$$f_d = \frac{v}{c} f_c \tag{2-5}$$

where v is the velocity of the receiver, *c* is the speed of light ($c = 3x10^8$ m/s), and f_c is the carrier frequency. [8]

Different transmitted signals undergo different types of fading depending on signal and channel parameters. Four possible effects that the time dispersion and frequency dispersion in a channel cause are frequency selective fading, flat fading, slow fading and fast fading.

8. Receiver

On the receiver side of the COFDM system, the reverse operations are performed. At the front end, a low-noise amplifier (LNA) that reduces the effective noise temperature of the receiver and an automatic gain control (AGC) that estimates the power of the pilot tone and controls the power at the demodulator output are used. The guard interval is removed once the symbols are detected. The symbol constellations are recovered by passing the signal through FFT. The resulting data are deinterleaved and channel decoded.

C. PEAK TO AVERAGE POWER RATIO (PAPR)

COFDM signals are generated as the sums of a number of subchannel signals, which are continuous-time sinusoidal signals. The summation may cause constructive interference, thereby resulting in a high peak-to-average power ratio (PAPR).

The instantaneous power of the signal is

$$P_c(t) = |S_c(t)|^2 \tag{2-6}$$

where $S_c(t)$ is the transmitted COFDM signal.

The peak-to-average power ratio (PAPR) is given by

$$\text{PAPR} = \frac{\max\limits_{t \in [0,T]} \left|(S_c(t))\right|^2}{\varepsilon\{\left|(S_c(t))\right|^2\}} \tag{2-7}$$

where T is the symbol duration and $\varepsilon\{\}$ denotes expectation.

If the peak transmission power is physically limited or limited by regulatory rules, the system must be designed with reduced average transmitter power. This may reduce the effective transmission range of the system. Additionally, costs may rise as more equipment may be needed to cover the same transmission range. [9]

In brief, high PAPR is a disadvantage of COFDM communication systems. In this thesis, we investigate ways to reduce PAPR.

In this chapter, the COFDM system architecture is presented and the advantages and the drawbacks of the system discussed. The PAPR problem in COFDM systems is examined. The next chapter presents the coding scheme and the Reed-Muller error correction codes as a mitigation technique to reduce the required PAPR.

III. CODING SCHEME

This chapter begins with an overview of error correction coding in digital communication systems and the reason why channel coding is used. Also, we discuss how error correction codes improve the performance of digital communication systems and present a summary of commonly used error correction codes. The Reed-Muller error correction codes are discussed in detail. Also, peak clipping and Hanning windowing are explained.

A. ERROR CORRECTION SYSTEMS

Due to the effects of noise and multipath fading in the channel, the transmitted signal arrives at the receiver with some errors. The errors in the demodulated data are characterized in terms of a BER, which is directly proportional to the symbol rate and inversely proportional to transmitter power and bit-energy to noise power spectral density ratio (E_b/N_0). The bit error rate is an important performance parameter of digital communication systems.

Error control strategies basically have two main categories, automatic repeat request (ARQ) and forward error correction (FEC). In this chapter, forward error correction coding is discussed.

In forward error correction coding, a certain number of redundant bits are added to data bits in a particular pattern according to the type of the code. In other words, for every k data bits, n coded bits are transmitted, where $n>k$. In the receiver, the k data bits can be recovered by performing a decoding operation on the n received coded bits.

The transmission conditions in wireless communication channels are severe due to multipath fading and the variation of the signal-to-noise power ratio. Therefore, in order to design a communication system with an acceptable BER, error correction coding must be used to protect the data from transmission errors.

As long as the signal-to-noise ratio (SNR) is high and the channel is relatively flat, error correction coding may be unnecessary in OFDM systems. However, uncoded OFDM systems do not perform well in fading channels. [10]

The code rate $r = k/n$ is a ratio of the number of data bits to the total number of coded bits transmitted per code word.

As a result of redundancy, the number of bit errors can be expected to increase. However, the reliability of demodulated data is increased because redundancy is used to correct some of the errors. A better BER can be achieved at the output of the decoder by using an appropriate coding scheme. The BER improvement provided by the channel coding is generally expressed in terms of the required E_b/N_0 to achieve the same performance without coding. This difference in E_b/N_0 is called coding gain as given by

$$G_{\text{dB}} = \left(\frac{E_b}{N_0}\right)_{\text{uncoded}_{\text{dB}}} - \left(\frac{E_b}{N_0}\right)_{\text{coded}_{\text{dB}}} \qquad (3\text{-}1)$$

Using commercially available standard error correction systems, coding gains up to 6 to 9 dB are achievable. [11, 12]

Linear block codes, convolutional codes and turbo codes are the most widely used error correction codes. Although only convolutional codes and Reed-Muller codes are used in the COFDM simulation undertaken in this thesis, linear block codes and turbo codes are also mentioned.

Linear block codes generate *n* coded bits for *k* data bits where $n > k$. First, *k* data bits are transmitted and then $n - k$ redundant bits which are generated according to the generator matrices of the code are transmitted. The encoder consists of a *k*-stage shift register and $n - k$ modulo-2 adders. The shift register outputs are connected to modulo-2 adders according to the generator matrices of the code.

The main difference between block codes and convolutional codes is that a system utilizing block codes transmits the k data bits unaltered and then transmits the $n - k$ redundant bits. A system utilizing convolutional codes produces n coded bits from k data bits, and the code word does not contain unaltered k data bits.

As mentioned in Chapter II, in the IEEE 802.11a WLAN system, the data are coded using a convolutional encoder with coding rate $r = 1/2,\ 2/3$, or $3/4$. The convolutional encoder uses the half-rate industrial-standard generator polynomials, $g_0 = 133_8$ and $g_1 = 171_8$. Higher rates are obtained from the half-rate code by using a puncturing process, which omits some of the encoded bits in the transmitter and inserts a dummy zero in place of omitted bits in the convolutional decoder in the receiver. [7]

A concatenated code consists of two separate codes in series in which the first code, called outer code, directly takes the information bits and encodes them. The second code, called inner code, takes the bits coded by the outer code and encodes them. In order to prevent decoding errors from passing from one coder to the other, the errors are distributed by using the interleaving/deinterleaving operation. The interleaver is placed between the outer and inner encoders of a concatenated code in the transmitter and the deinterleaver is placed between the inner and outer decoders in the receiver. To make the coding system more efficient, the output of the outer decoder is reapplied to the inner decoder. This is the basis of the iterative decoding. It can be summarized as reapplying the decoded word not just to the inner code, but also to the outer and repeating as many times as necessary [13]. Turbo codes are based on parallel convolutional concatenated codes (PCCC). Additionally, pseudorandom interleavers instead of rectangular interleavers are used.

Since the goal of this thesis was to analyze the COFDM system with Reed-Muller (RM) error correction codes, the focus was on the encoding and decoding algorithms of RM codes.

B. REED-MULLER CODES

This section describes the encoding/decoding algorithm of the Reed-Muller (RM) coding and how it is used for the reduction of PAPR in COFDM systems. Since the encoding and decoding algorithms are complicated and different from the other schemes, some binary operations used with RM codes are first defined and then the encoding and decoding algorithms are presented.

RM codes were described by Muller in 1954, and Reed provided a better algebraic representation with a decoding algorithm in the same year. RM codes were used in space applications between 1969 and 1977. Although they seem to have lost their attraction in the space program because of the adoption of the newer codes, it does not necessarily mean that BCH (Bose-Chaudhuri-Hocquenghem) and Reed-Solomon codes are a better choice relative to RM codes for all applications. [11]

1. Definitions of Operations

Three possible operations performed in encoding and decoding are addition, multiplication and dot product. For the binary vectors $x = (x_1, x_2, ..., x_n)$ and $y = (y_1, y_2, ..., y_n)$, where each x_i or y_i is either 1 or 0, addition is defined as [1]

$$x{+}y = (x_1 + y_1, x_2 + y_2, ..., x_n + y_n) \quad (3\text{-}2)$$

and the complement $\overline{x}$ of vector x is the vector equal to $1 + x$. Multiplication is defined as

$$x * y = (x_1 * y_1, x_2 * y_2, ..., x_n * y_n)\,. \quad (3\text{-}3)$$

Finally, the dot product of x and y is defined as

$$x \,.\, y = (x_1 * y_1 + x_2 * y_2 + ... + x_n * y_n)\,. \quad (3\text{-}4)$$

2. Encoding Algorithm

The RM coding scheme is denoted in terms of $R(r, m)$. The parameters*, r* and *m*, are defined by

$$m = \log_2 n \quad (3\text{-}5)$$

where *n* is the number of the columns of the encoding matrix and

$$r = m - \log_2 d_{\min} \quad (3\text{-}6)$$

where $d_{\min}$ is the minimum Hamming distance. The number of the rows of the encoding matrix is given by [11]

$$k = 1 + \binom{m}{1} + \binom{m}{2} + \ldots\ldots + \binom{m}{r}. \tag{3-7}$$

The encoding matrix of the $R(r,m)$ code is defined as

$$G(r,m) = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_m \\ x_1 x_2 \\ x_1 x_3 \\ \vdots \\ x_{m-1} x_m \\ x_1 x_2 x_3 \\ x_1 x_2 x_4 \\ \vdots \\ x_{m-2} x_{m-1} x_m \\ x_1 x_2 x_3 x_4 \\ \vdots \\ x_{m-r+1} \ldots x_{m-1} x_m \end{bmatrix} \tag{3-8}$$

where x_i is the monomial vector and has a pattern of 2^{m-i} ones followed by 2^{m-i} zeros, repeated until n is completed. The first row of the encoding matrix x_0 is all ones or all zeros and the sequence of ones or zeros in the next rows are interchangeable. In this thesis, the first row is all ones and the sequence pattern is ones and zeros, respectively. For example, if *m* = 3, then the encoding matrix for R(1,3) is given by

$$G(1,3) = \begin{array}{c} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \end{array} \begin{array}{c} 1 \\ x_1 \\ x_2 \\ x_3 \end{array} \tag{3-9}$$

where the column on the right hand side is the corresponding monomial vector.

The R(2,3) code can be generated by adding the rows x_1x_2, x_1x_3 and x_1x_3 to the generator matrix in (3-9) as follows:

$$G(2,3)=\begin{array}{c}\left[\begin{array}{cccccccc}1&1&1&1&1&1&1&1\\1&1&1&1&0&0&0&0\\1&1&0&0&1&1&0&0\\1&0&1&0&1&0&1&0\\1&1&0&0&0&0&0&0\\1&0&1&0&0&0&0&0\\1&0&0&0&1&0&0&0\end{array}\right]\end{array}\begin{array}{c}1\\x_1\\x_2\\x_3\\x_1x_2\\x_1x_3\\x_2x_3\end{array} \tag{3-10}$$

where the column in the right hand side is the corresponding monomial vector. The generator matrix for R(3,3) is obtained by adding the $x_1x_2x_3$ term to the above and is given by

$$G(3,3)=\left[\begin{array}{cccccccc}1&1&1&1&1&1&1&1\\1&1&1&1&0&0&0&0\\1&1&0&0&1&1&0&0\\1&0&1&0&1&0&1&0\\1&1&0&0&0&0&0&0\\1&0&1&0&0&0&0&0\\1&0&0&0&1&0&0&0\\1&0&0&0&0&0&0&0\end{array}\right]\begin{array}{c}1\\x_1\\x_2\\x_3\\x_1x_2\\x_1x_3\\x_2x_3\\x_1x_2x_3\end{array}. \tag{3-11}$$

It must be noted that the $R(r,m)$ encoding matrix can be achieved by adding $\binom{m}{r}$ rows to the $R(r-1,m)$ encoding matrix. Since the encoding matrix has k rows, the message is sent to the encoding matrix in blocks of length k. If the message is defined as

$$\mathrm{m} = (m_1, m_2, \ldots\ldots, m_k), \tag{3-12}$$

then the encoded message M_c is

$$M_c = \sum_{i=1}^{k} m_i R_i \tag{3-13}$$

where R_i is the corresponding row of the encoding matrix.

3. Decoding Algorithm

The decoding algorithm is somewhat complicated and consists of a few steps. It is based on majority logic or threshold decoding. In general, majority logic techniques are fast and easy to implement.

An $R(r,m)$ code can correct up to e errors if the distance between two code words is greater than $2e$, where the distance is described as the number of different values at the corresponding places between two code words.

Characteristic vectors are the vectors whose dot product is one with the corresponding row and zero with all the other rows of the encoding matrix. The decoding algorithm begins with finding 2^{m-r} characteristic vectors for each row of the encoding matrix. After finding the characteristic vectors, the next step is the dot product of these characteristic vectors with the received encoded message M_e. This operation gives 2^{m-r} results and, by performing the majority logic, the coefficient of the row is calculated. In other words, if the number of the zeros is larger than the number of ones in the dot product operations, the coefficient is zero and one otherwise. These steps thus far are performed for every row of the encoding matrix except the first one. The coefficients are then multiplied with the corresponding row and the results are added. At this point, M_y is obtained, which is a row vector having 2^m columns. The addition of M_y and M_e provides the location of the error(s) and knowledge about the first row of the encoding matrix, that is, whether it is all ones or all zeros. If the number of the ones in this row is larger than the number of zeros, the first row of the matrix is all ones, otherwise it is all zeros. The 1's and 0's so obtained form the recovered message. [1]

It must be noted that the error-correcting capability of the RM codes can be increased by increasing the minimum Hamming distance.

4. Reduction of PAPR of COFDM Using RM Coding

While discussing the drawbacks of the COFDM system, it was mentioned that high PAPR is the main disadvantage of the system.

Some research has been performed in order to reduce the PAPR in the COFDM systems. One of the solutions to reduce PAPR requires the use of block codes. This approach is based on the selection of appropriate code words that reduce the PAPR, and then the use of these code words. However, performing a long search to find these appropriate codes and the necessity for large lookup tables make this approach inefficient [14]. Another approach is based on introducing a specific phase, which is independent and known to both the transmitter and the receiver for each coordinate of all the code words. This approach obtained 4.5 dB in PAPR reduction by using the computed phase shifts [10]. In this work, the use of RM codes in order to reduce PAPR is explored.

Davis and Jedwab in [15] showed that large sets of binary length of 2^m Golay complementary pairs can be obtained from the certain second-order cosets of the first order RM code. These cosets can be grouped according to their PAPR values for the codes of lengths 2^m, where $m = 2,3,4,5$. The first group of cosets has a PAPR of at most 2. The second one has a PAPR of at most 4 and so forth. An effective combination of the code rate, PAPR and minimum distance can be obtained by selecting second-order cosets from the list. This method can be efficiently used for the COFDM system for a small number of carriers. For a larger number of carriers, it becomes inefficient since the evaluation of PAPR values of a large set of cosets requires a large amount of computation. [15]

In summary, using RM codes with the COFDM system is expected to not only provide efficient encoding and decoding algorithms and high error correction capability with high code rates, but also to reduce PAPR, especially for a small number of carriers.

C. PEAK CLIPPING

Peak clipping is a simple approach to reduce the PAPR. Since the peaks in the COFDM signal occur with a low probability, peak clipping can be considered an effective technique for the reduction of PAPR. However, it must be noted that it is a nonlinear process and may worsen the BER performance due to the inband signal distortion. Additionally, it increases the out-of-band radiation and, therefore, reduces the frequency spectrum efficiency. The clipping ratio, CR, is defined by

$$\text{CR} = \frac{A}{\sigma} \tag{3-14}$$

where A is the clip level and σ is the rms power of the COFDM signal. The COFDM signal is clipped whenever it exceeds the clip level.

D. HANNING WINDOWING

Hanning windowing is another approach to reduce the PAPR of COFDM signals. The COFDM signal is multiplied by the window function when the signal exceeds the clipping level or falls below the bottom level. While the clipping operation directly chops off the peaks, windowing results in a smooth signal. The peak window is given by

$$w_c[n] = \begin{cases} 1 - k_c(0.5 - 0.5\cos(2\pi n / M)) & 0 \le n \le M \\ 0 & \text{otherwise} \end{cases} \tag{3-15}$$

where k_c is the attenuation factor of the window and *M* is the width of the window. The bottom window is given by

$$w_c[n] = \begin{cases} 1 + k_a(0.5 - 0.5\cos(2\pi n / M)) & 0 \le n \le M \\ 0 & \text{otherwise} \end{cases} \tag{3-16}$$

where k_a is the amplification factor of the window. Figure 9 shows a 9-point peak and a 9-point bottom Hanning windows. [2]

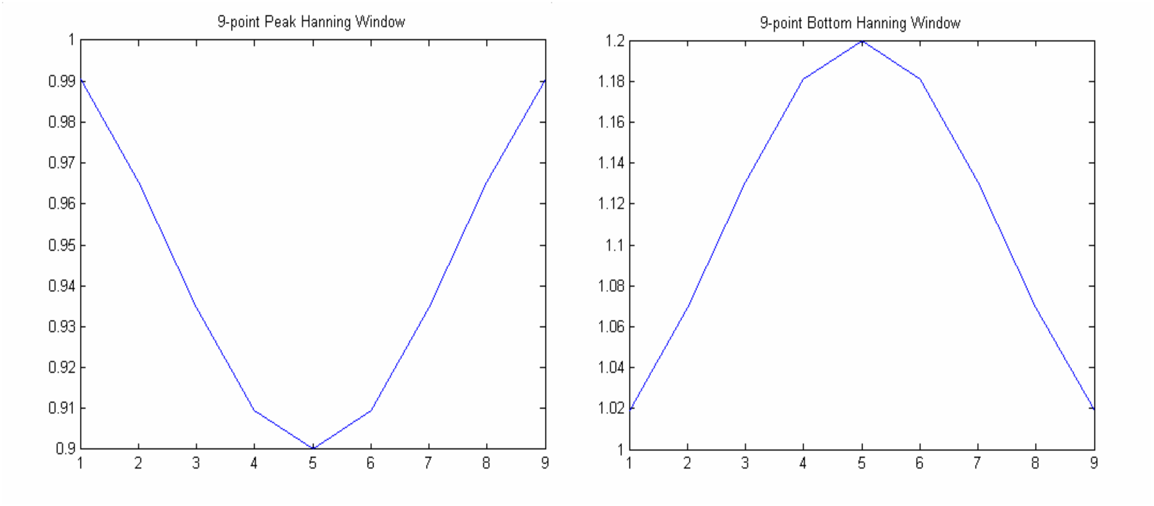


Figure 5. 9-Point Hanning Windows

When the windows are multiplied with the COFDM signal, the resulting spectrum is the spectrum of the windowed signal. Therefore, the window width is an important parameter that affects the BER performanceof the system.

This chapter discussed the error correcting coding schemes. The RM coding scheme was presented in detail. The RM coding is known to reduce the PAPR, which is a problem of the COFDM systems. The encoding and decoding algorithms of RM codes were presented. Peak clipping and Hanning window techniques were explained. The next chapter presents the COFDM simulation architecture and reports the results of simulation.

IV. SIMULATION RESULTS

This chapter describes the simulation model used in this thesis and presents the simulation results. The MATLAB simulation code in [1] was modified and expanded to implement the simulation model. Simulations were performed to study the system performance with RM codes under a variety of channel conditions. The system was also tested for PAPR reduction. In addition to the RM codes, peak clipping and Hanning windowing were implemented in order to reduce the PAPR.

A. SIMULATION MODEL

Since the main goal of this thesis was to simulate the COFDM system by utilizing RM codes, all signal processing including forward error correction coding (FEC) and different types of channel characteristics are performed at the base band. The block diagram of the entire system is shown in Figure 6. A detailed description of the different MATLAB functions corresponding each of the blocks in Figure 6 is provided in Appendix A.

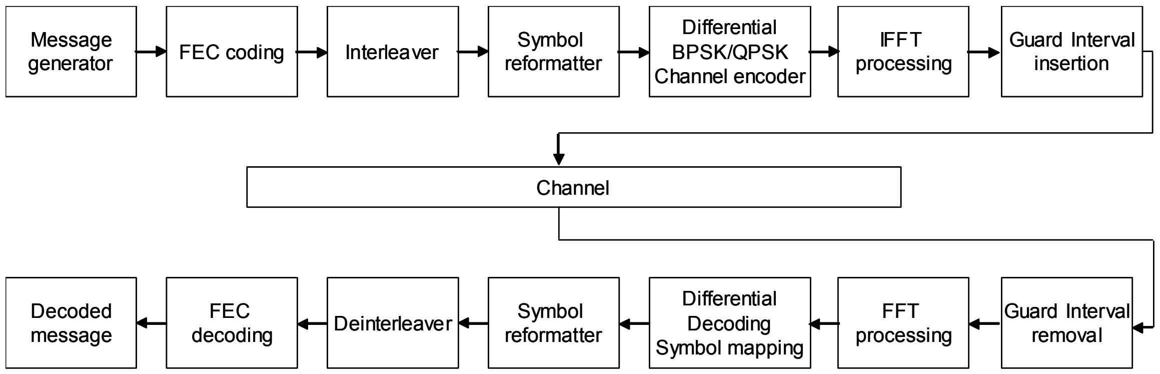


Figure 6. COFDM System Block Diagram.

B. SIMULATION PARAMETERS

During the simulations, in order to compare the results, the same random message was used. The seed value in *msg.m* was changed only for comparing the PAPR values of the system with different random messages. Although the code makes it possible to select a different number of symbols and interleaver pairs, all simulation runs were performed with 12,000 symbols and a (120, 100) interleaver pair. While testing the convolutional codes, the IEEE 802.11a standard half-rate convolutional codes with $k = 50$ and $n = 100$ were performed.

After construction of the subblocks and modification of the code in [1], five different channels were formed. Channel 0 is a noise-free channel with no AWGN and multipath effects. Channels 1, 2 and 3 incorporate AWGN, multipath, and AWGN and multipath, respectively. Channel 4 incorporates the outdoor channel characteristics of AWGN, severe multipath, and mobility.

In Channel 1, the standard deviation σ of white Gaussian noise is varied from 0 to 0.06 for different coding options.

The multipath fading parameters used in Channels 2 and 3 are tabulated in Table 1. The multipath loss in dB and the delay in ms are listed for indoor channel environment. There are 18 taps and delay coefficients in the channel. Three different Doppler frequencies of 5 Hz, 10 Hz, and 15 Hz were considered; the corresponding velocities of these frequencies were 0.29, 0.58 and 0.87 m/s, respectively. They represent walking speeds in an indoor environment.

Loss (dB)	[0, 2.17, 4.34, 6.51, 8.69, 10.86, 13.03, 15.20, 17.37, 19.54, 21.71, 23.89, 26.06, 28.23, 30.4, 32.57, 34.74, 36.92]
Delay (msec)	[0, 0.05, 0.10, 0.15, 0.20, 0.25, 0.30, 0.35, 0.40, 0.45, 0.50, 0.55, 0.60, 0.65, 0.70, 0.75, 0.80, 0.85]
Doppler (Hz)	5
	10
	15

Table 1. Multipath Fading Parameters for Channels 2 and 3

The outdoor channel parameters of Channel 4 are shown in Table 2. Within this channel, two different cases each with different loss, delay and Rician factors are implemented.

	Channel 4A	**Channel 4B**
Loss (dB)	0.0, 1.0, 9.0, 10.0, 15.0, 20.0	2.5, 0.0, 12.8, 10, 25.2, 16
Delay (msec)	0.0, 0.25, 0.5, 1.0, 1.9, 2.2	0.0, 0.25, 9.0, 13.0, 17.0, 20.0
Rician Factors (%)	0, 0, 0, 0, 0, 0	0.5, 0.5, 0, 0, 0, 0

Table 2. Outdoor Channel Parameters of Channel 4

In order to analyze the effects of using RM codes in the COFDM system for both indoor and outdoor channel characteristics, a step-by-step simulation methodology was followed. The performance of the COFDM system with RM codes is tested under different conditions in the order presented in Table 3.

Test No.	Channel	Channel Description
1	Channel 0	Noise-free Channel
2	Channel 1	AWGN effect
3	Channel 2	Multipath effect
4	Channel 3	AWGN + Multipath effect
5	Channel 4	Severe mobile outdoor channel

Table 3. General Test Plan.

C. SIMULATION RESULTS

Channel 0 was used to test whether the system was configured properly and working correctly. In this case, the received message is the same as the source message. Numerous simulations performed for different types of RM and convolutional codes demonstrated that the code ran correctly. Figure 7 shows the transmitted and received QPSK constellations.

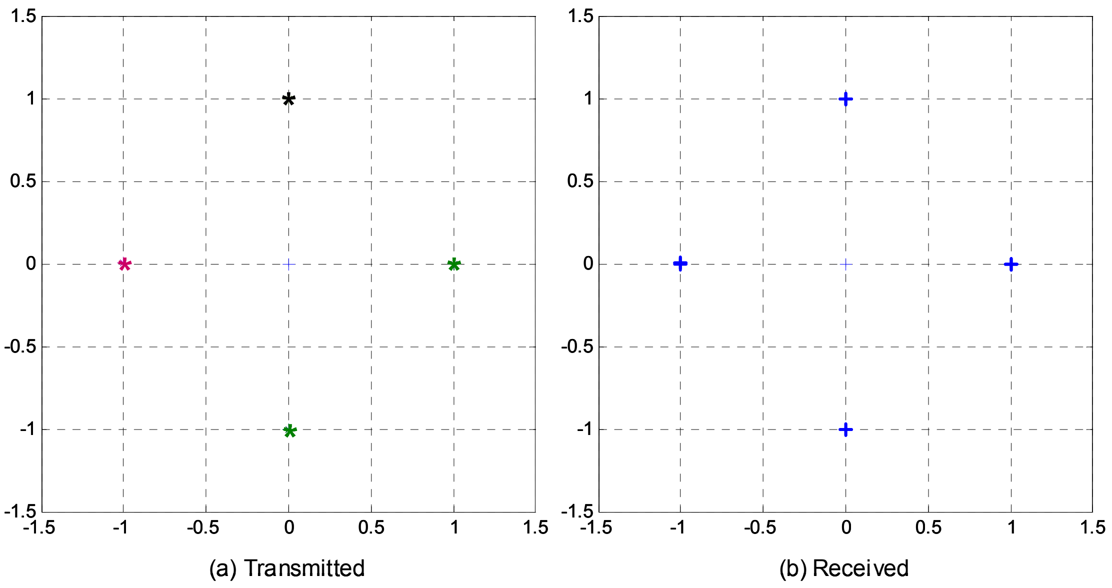


Figure 7. Transmitted and Received QPSK Signal Constellations for Channel 0.

The performance of the COFDM system is next tested using Channel 1, which includes AWGN without multipath fading effects. The received QPSK constellations for different levels of noise are shown in Figure 8.

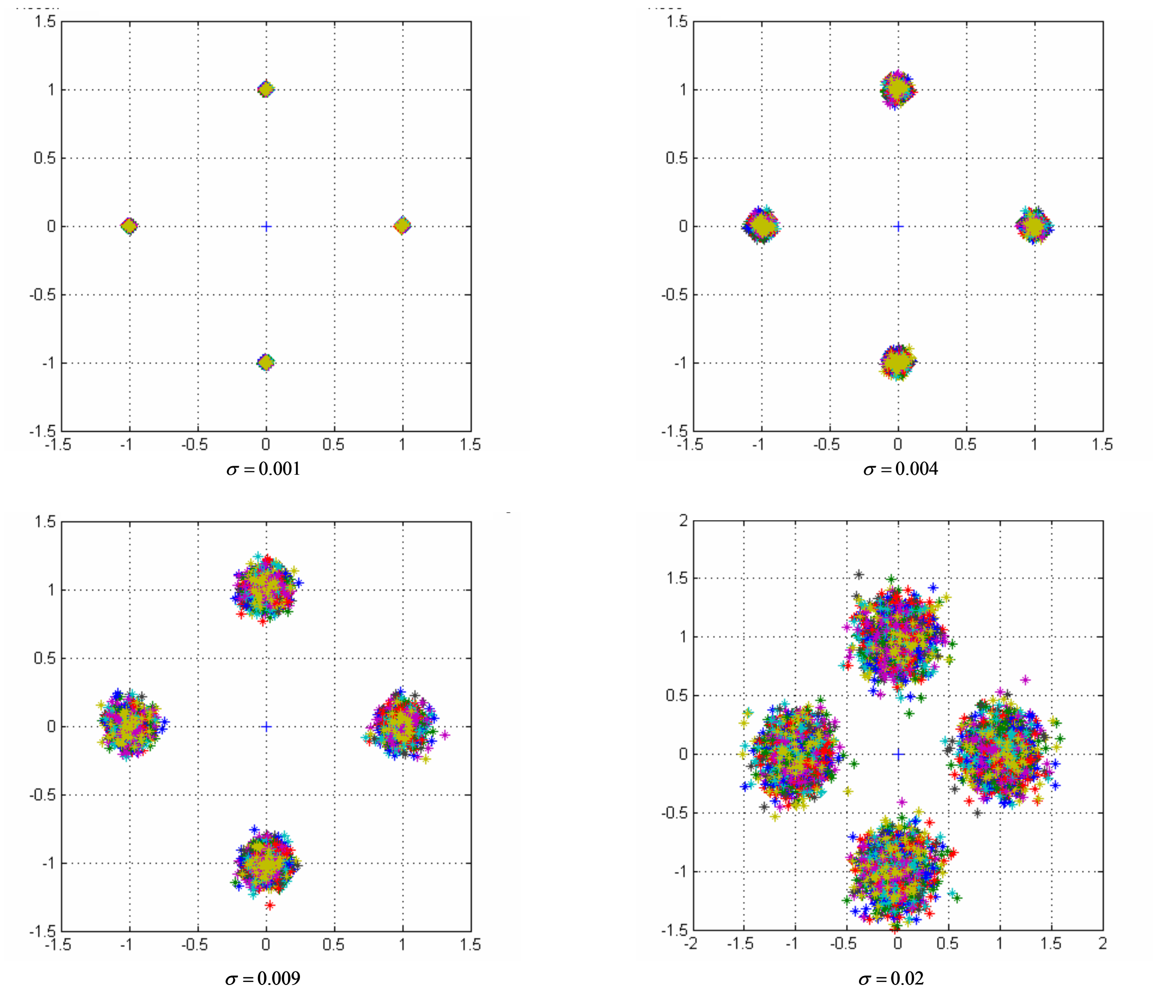


Figure 8. The Effects of AWGN over QPSK Signal Constellation.

The BER plots were obtained for different types of RM codes and convolutional codes. The number of errors that an $R(r,m)$ code can correct depends on the r and m values. The number of the errors that different types of RM codes can correct is tabulated in Table 4. The bit-error plots for three different RM codes and a rate $1/2$ convolutional code along with the QPSK theoretical plots are shown in Figure 9.

Type of code	**Number of the rows (number of the bits in a block)**	**Number of the error(s) that can be corrected**	**Rate**
$R(r,m)$	$k=1+\binom{m}{1}+\binom{m}{2}+....+\binom{m}{r}$	$2^{m-r-1}-1$	$\frac{2^{m-r-1}-1}{k}$
R (1,3)	4	1	$\frac{1}{4}=0.25$
R (2,4)	11	1	$\frac{1}{11}=0.0909$
R (2,5)	16	3	$\frac{3}{16}=0.1875$

Table 4. The Number of the Error(s) that Different Types of RM Codes Can Correct.

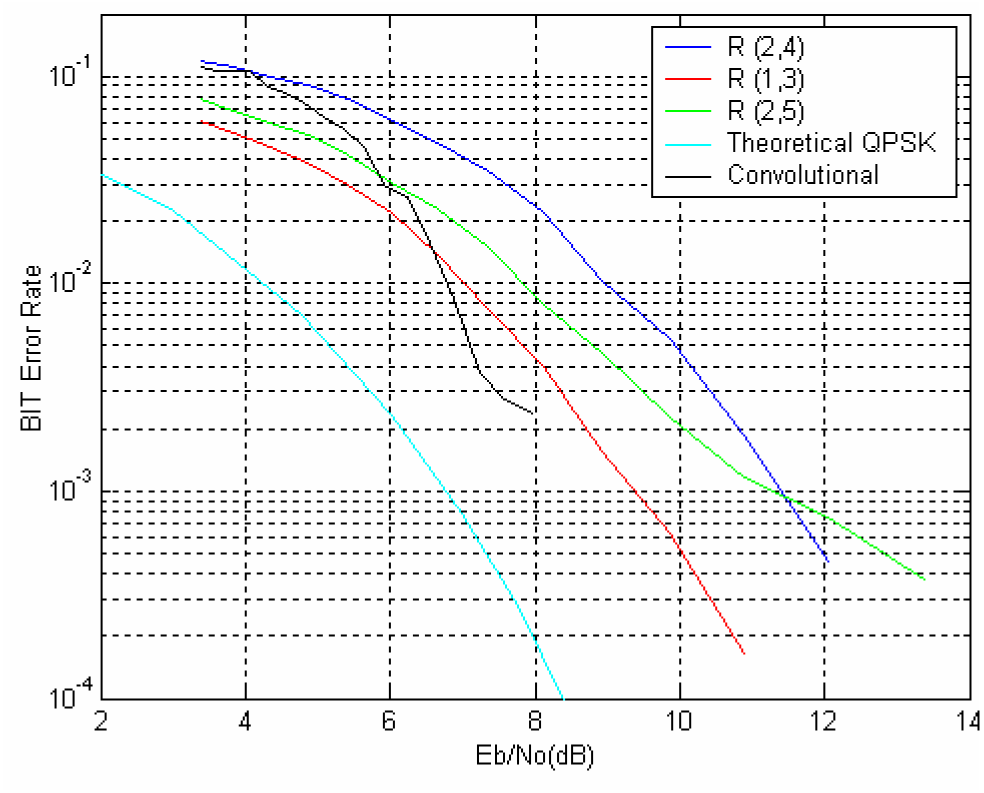


Figure 9. BER Performance of the COFDM system under Channel 1 (AWGN only) Conditions.

From Figure 9 and Table 4, the performance of an RM code depends on the number of error(s) that it can correct in a block. Since this rate is worst for R (2,4), the corresponding BER performance curve was also the worst. Compared to the theoretical, there is 2.5 dB difference for convolutional code, 2.7 dB for R(1,3), 3.6 dB for R(2,5) and 4.8 dB for R(2,4) at $P_b = 10^{-2}$.

1. Multipath Effects

The performance of the COFDM system was studied by adding multipath effects to the channel. First, multipath effects without AWGN were simulated using Channel 2. Figure 10a shows the received QPSK constellation. The constellation points are scattered from their original position due to the effects of multipath fading. Figure 10b shows that, with the addition of differential decoding, the constellation points realigned somewhat within their respective spaces.

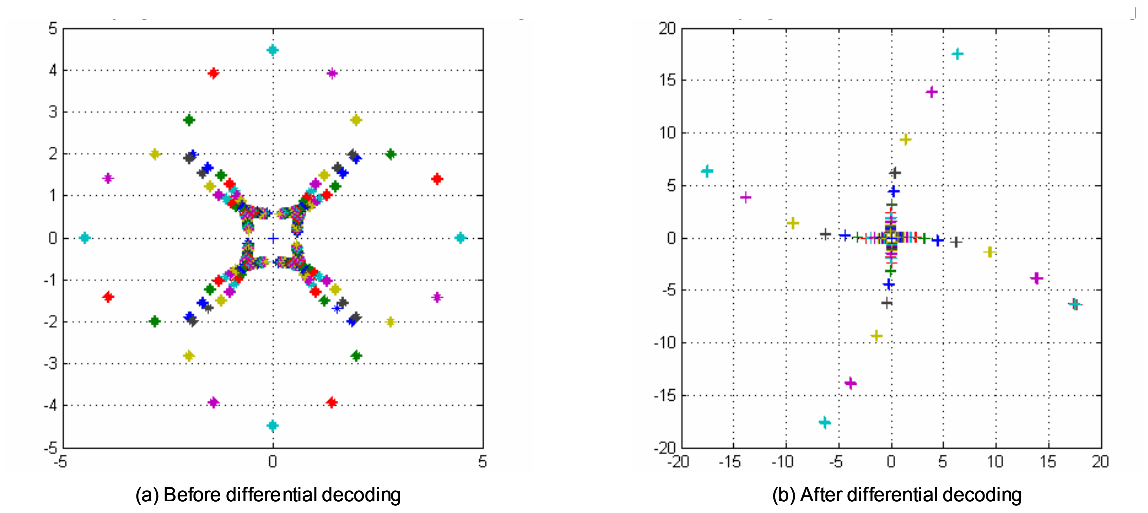
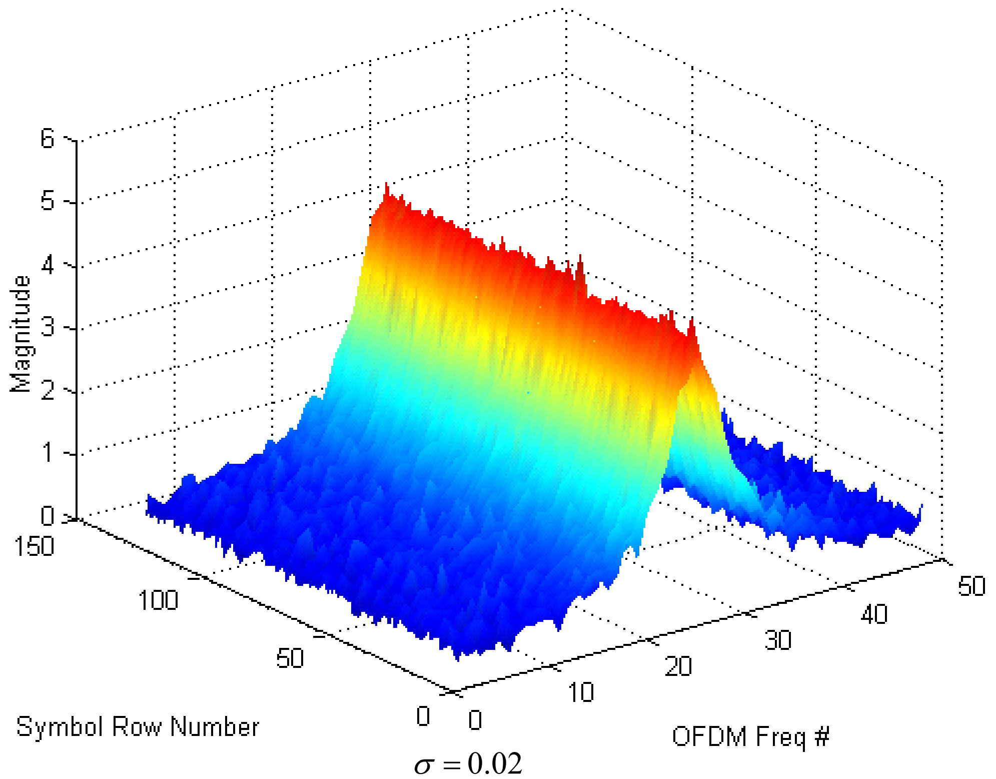


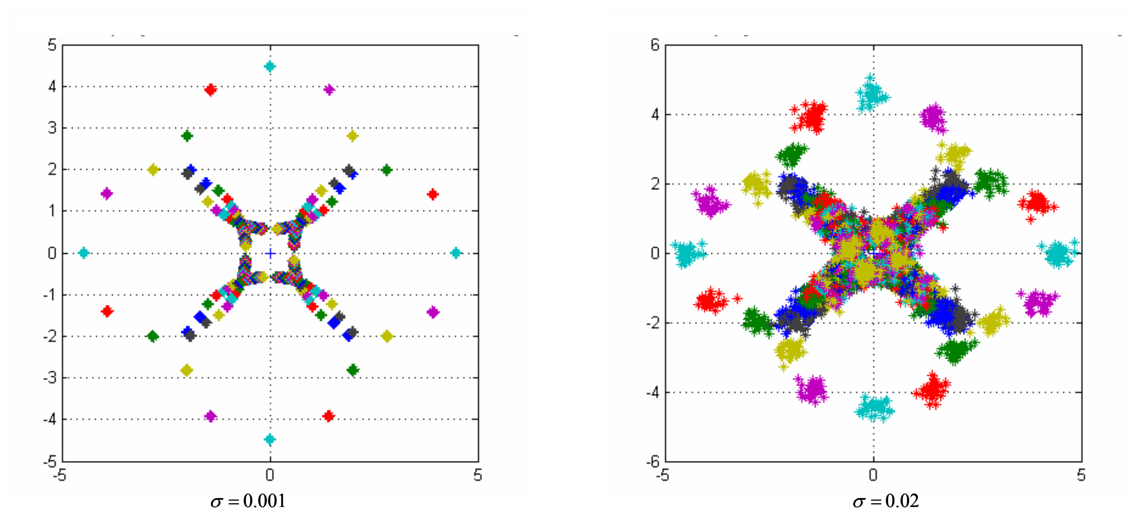
Figure 10. The Effects of Multipath on QPSK Signal Constellation

Channel 3 takes AWGN, multipath effects, and Doppler shift into account, hence more realistic than Channel 2. Channel 3 may be considered a good representation for indoor environments. Figure 11 shows the magnitude of the QPSK signal at the input of the receiver. The magnitude is plotted for different symbols at different subcarriers for an AWGN standard deviation of $\sigma = 0.02$. Note that ideally the magnitude is a constant for all symbols at all subcarriers. Figures 12 and 13 show the QPSK constellation prior to and after the differential decoding. QPSK constellation points are shifted from their original phase sectors because of the effects of multipath fading. The spreading of the constellation points increases with the noise variance. As Figure 13 illustrates, the constellation points can be realigned to their respective phase sectors to some extent by using differential decoding.



$\sigma = 0.02$

Figure 11. The Representation of COFDM Signal Under Multipath and AWGN Effects.



$\sigma = 0.001$ $\sigma = 0.02$

Figure 12. The Effects of AWGN and Multipath on QPSK Signal Constellation Before Differential Decoding

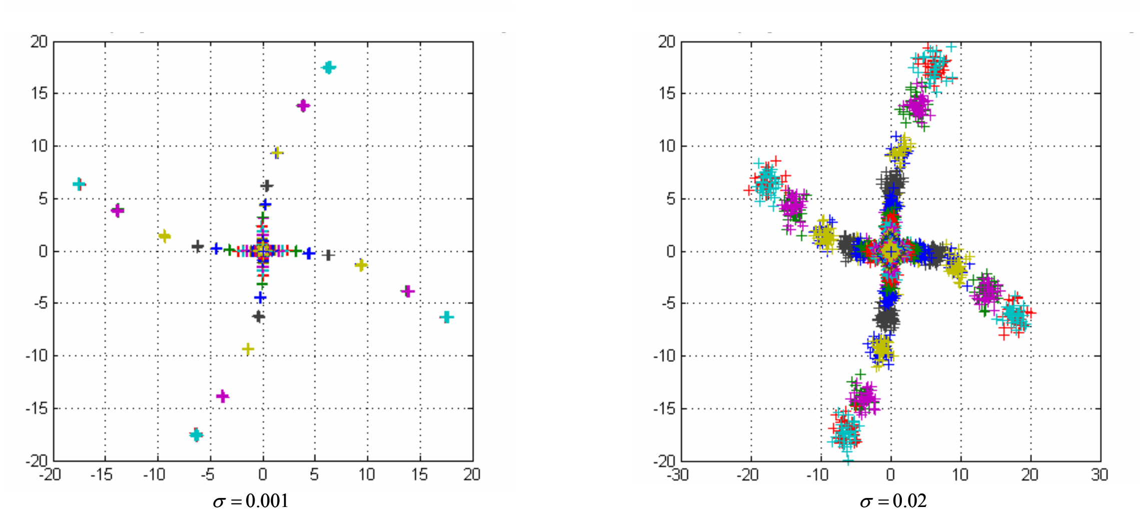


Figure 13. The Effects of AWGN and Multipath on QPSK Signal Constellation After Differential Decoding

Figures 14-16 show the bit-error rate plots for three different Doppler shift values. Each figure contains three different types of RM codes and a rate $1/2$ convolutional code. The BER performance curves show that the effect of the 5-Hz Doppler frequency creates slight difference in the BER performance of the system. As seen, the preformence of the convolutional coder is better than that of the RM codes.

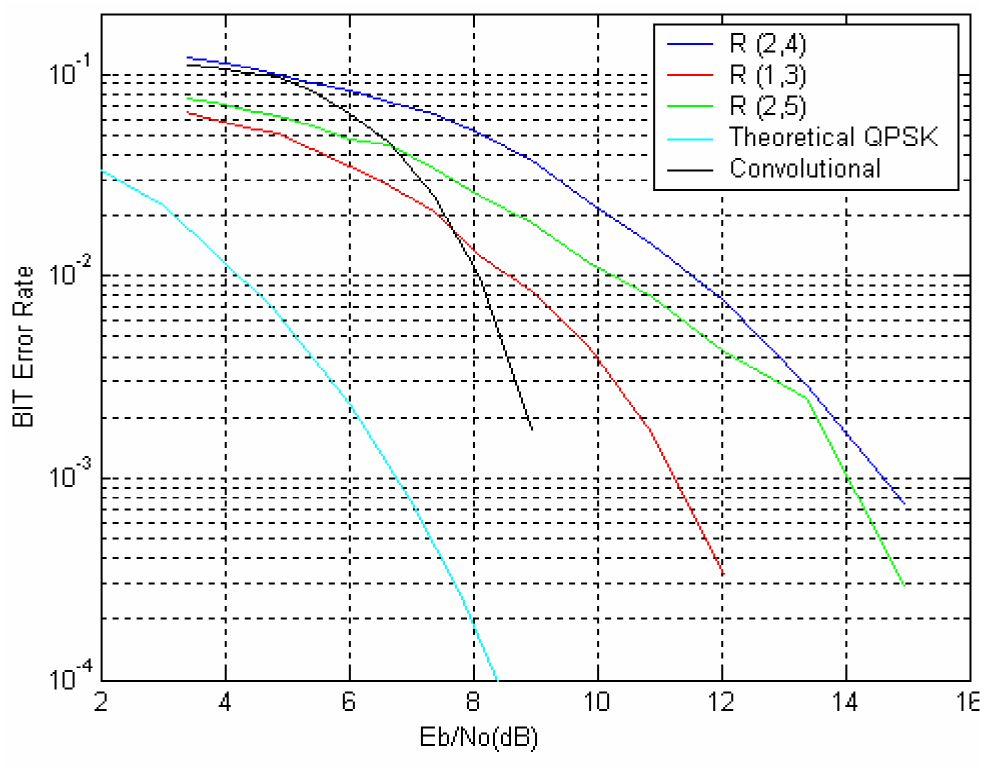


Figure 14. BER Performance of the COFDM system under Channel 3 (AWGN + Multipath Effect) with 5-Hz Doppler Conditions

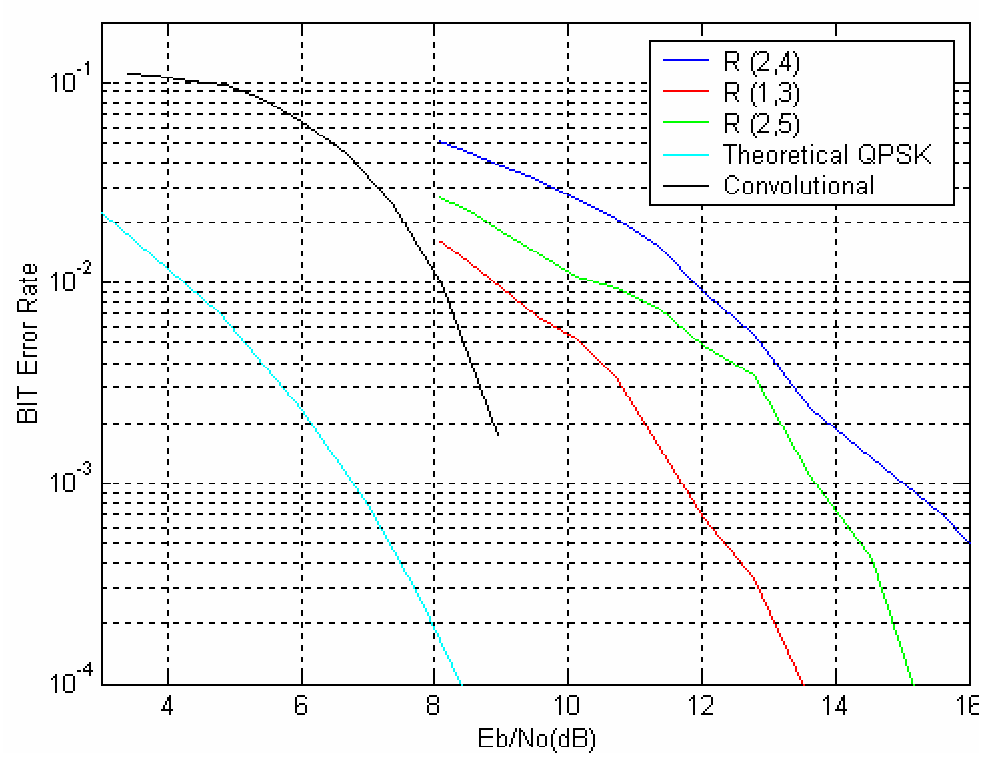


Figure 15. BER Performance of the COFDM system under Channel 3 (AWGN + Multipath Effect) with 10-Hz Doppler Conditions

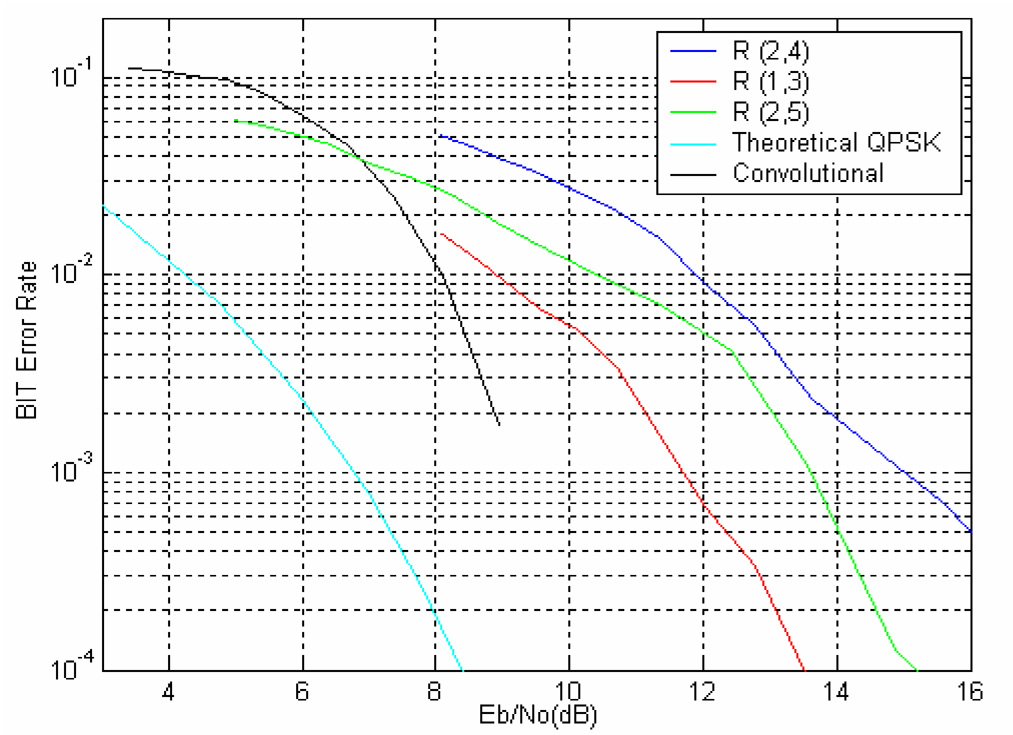
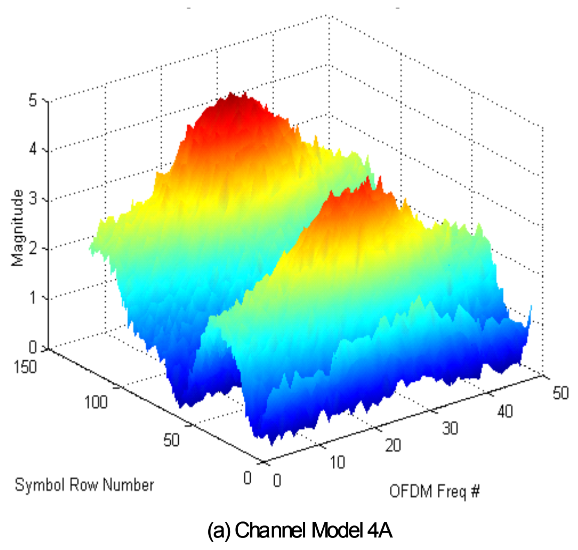


Figure 16. BER Performance of the COFDM system under Channel 3 (AWGN + Multipath Effect) with 15-Hz Doppler Conditions

2. Outdoor Environment

We now study the COFDM system performance under mobile outdoor channel conditions. Two channels were considered for this purpose: Channels 4A and 4B.

Figure 17 shows the COFDM signal magnitude for Channels 4A and 4B for $\sigma = 0.02$. Although the transmitted COFDM signal magnitude was at unity, the multipath fading, delay, and Rician factors in the channels resulted the noticeable signal variations in received signal power levels. The corresponding QPSK constellations are shown in Figure 18. Figure 18 shows that constellation points were scattered from their original points due to the multipath fading, delay, and Rician factors of the channels. Figure 19 shows the QPSK constellations after the differential decoding for Channels 4A and 4B. Compared to the constellation points in Figure 12, which includes multipath effects in AWGN, the signal constellation points in Figure 18 underwent more severe multipath fading.



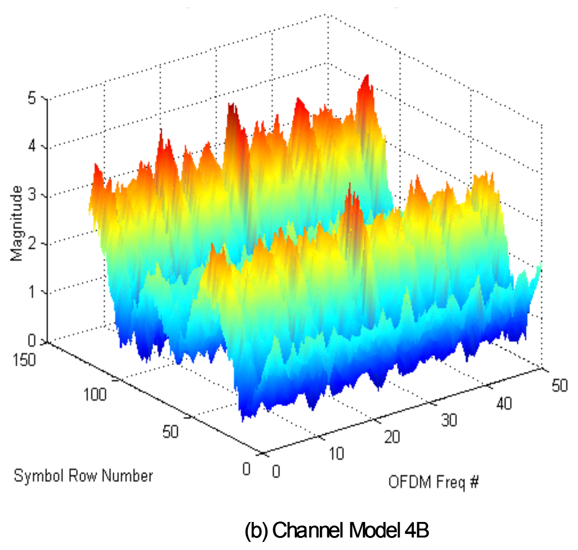
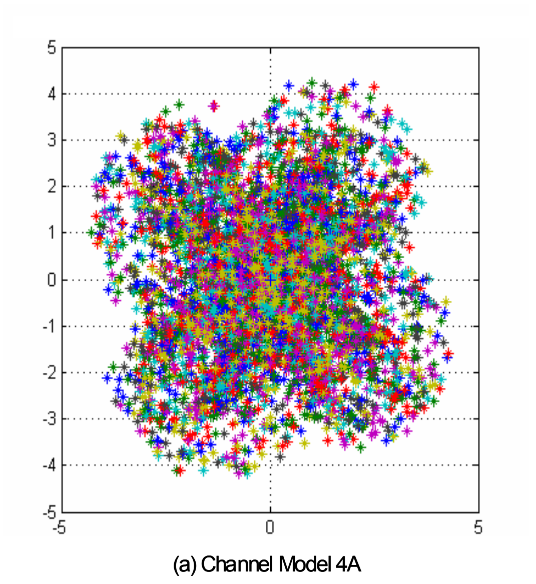


Figure 17. The Effects of Channel 4 on Signal Magnitude.



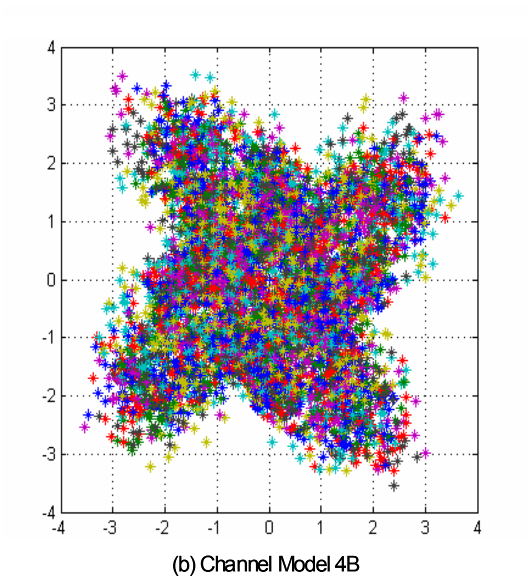
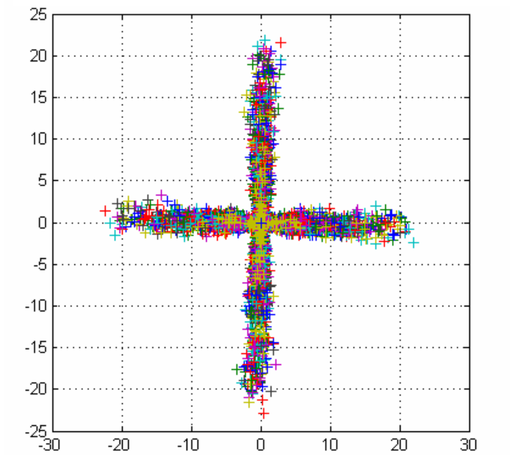


Figure 18. The Effects of Channel 4 on Signal Constellation (Before Differential Decoding).


(a) Channel Model 4A

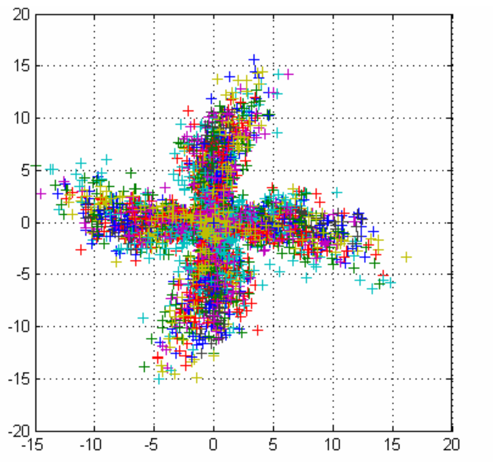

(b) Channel Model 4B

Figure 19. The Effects of Channel 4 over Signal Constellation (After Differential Decoding).

Figures 20 and 21 show the BER plots for Channel 4. In order to obtain a BER less than 10^{-2}, the E_b/N_o value should be higher than about 10 dB for RM codes and 5 dB for convolutional codes. This result shows that the COFDM system requires high signal-to-noise ratio in the outdoor environments.The necessity of high transmission power for Channels 4A and 4B creates the PAPR problem. The results of the PAPR values for the system are presented in the next section.

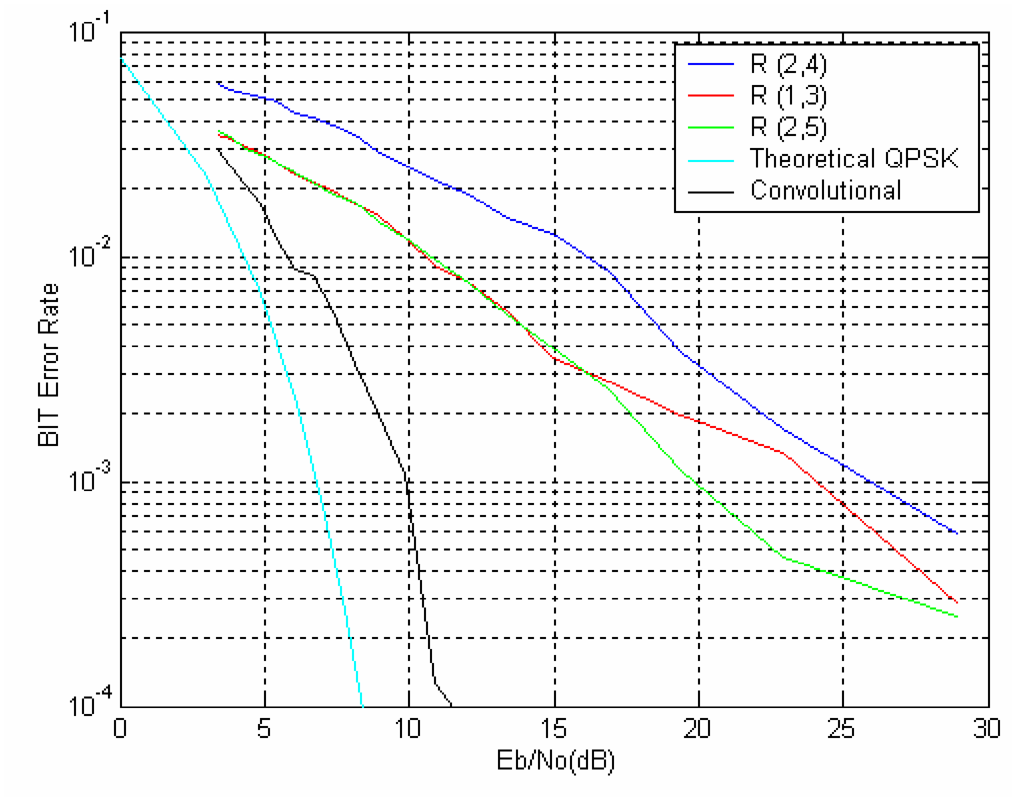


Figure 20. BER Performance Curves for Channel 4A Simulation.

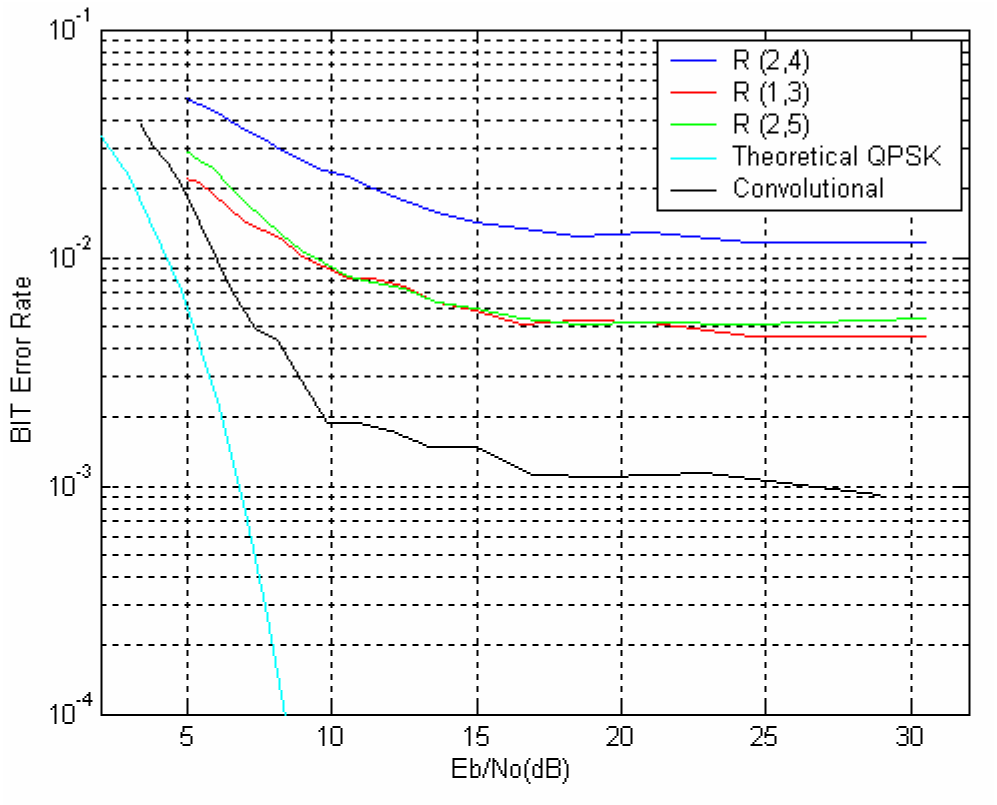


Figure 21. BER Performance Curves for Channel 4B Simulation.

3. PAPR Reduction

The PAPR values of the COFDM system simulated using different seeds to generate the random messages are tabulated in Table 5. The RM codes used in this work are R(1,3), R(2,4), and R(2,5). The RM codes in general are not consistent in reducing the PAPR in COFDM. This result does not agree with the claim in [15].

FEC Code Type	**PAPR (dB)**							
	Seed = 13		**Seed = 23**		**Seed = 33**		**Average**	
	Max	**Ave**	**Max**	**Ave**	**Max**	**Ave**	**Max**	**Ave**
R(1,3)	9.4734	6.7845	10.3031	6.6575	10.7564	9.9686	10.1776	7.8035
R(2,4)	9.2566	6.7392	9.5706	6.9064	9.5816	6.8251	9.4696	6.8235
R(2,5)	11.4281	6.7306	9.6601	6.7231	9.9839	6.6464	10.3573	6.7000
CONV.	10.0090	6.9200	9.4433	6.5059	10.0320	6.7182	9.8281	6.5347

Table 5. PAPR Values without Windowing and Peak Clipping.

Peak clipping and Hanning windowing were used to reduce the PAPR. It can be seen that peak clipping reduced PAPR by 3 dB. However, the reduced PAPR value introduced a high BER as can be seen in Figure 22. This presents a tradeoff between PAPR reduction and BER performance.

Table 6 shows the results when the COFDM signal was clipped at 18 through 12. From Figure 22, the BER performance of the system at CL = 18 is very close to no-clipping performance. However, as the peak clipping level is decreased, the required E_b/N_o to achieve the same BER performance increased because of the increasing probability of existence of the COFDM signal magnitudes higher than the clipping level.

	PAPR(dB)	
	Seed=33	
	Max	**average**
No Clipping	10.7564	9.9686
CL=18	10.4895	6.9636
CL=16	9.5959	6.9440
CL=14	8.5536	6.8727
CL=12	7.3191	6.4491

Table 6. Peak Clipping Results.

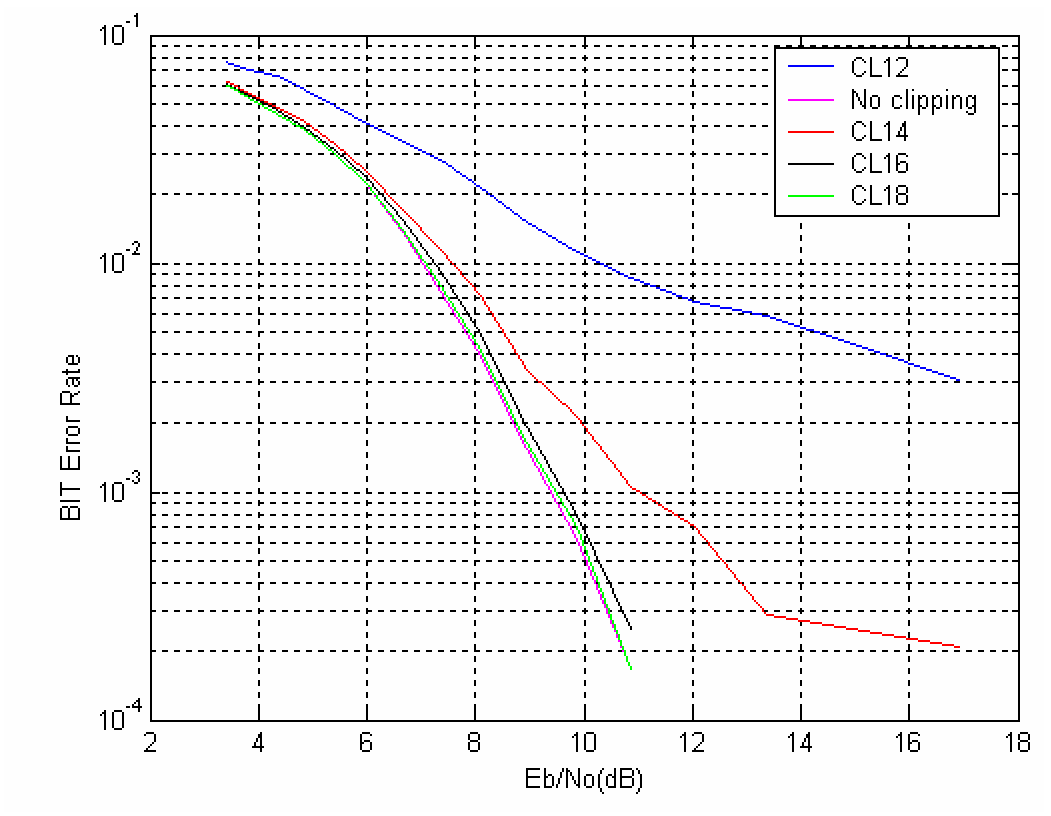


Figure 22. BER Performance of the System with Peak Clipping.

Hanning windowing was also implemented to reduce the PAPR. The values of $k_c = 0.1$ and $k_a = 0.2$ were assigned as Hanning windows coeffiecients during the simulations to achieve the best performance.

When the windows were applied to the COFDM signal, the resulting spectrum was the spectrum of the windowed signal. The windowing process was tested with 3-, 5- and 9-point Hanning windows. The PAPR values are shown in Table 7 and BER performance plots are shown in Figures 23 through 25 for 3-, 5-, and 9-point windows, respectively. Table 7 shows that the improvement in PAPR reduction is limited with Hanning windowing. Figures 23 through 25 show that as the width of the window is increased, the bit errors increased as well.

	PAPR(dB)							
	Seed = 33						**Seed = 23**	
	3-point Hanning Windowing		**5-point Hanning Windowing**		**9-point Hanning Windowing**		**9-point Hanning windowing**	
	Max	**Ave**	**Max**	**Ave**	**Max**	**Ave**	**Max**	**Ave**
No win-dowing	10.7564	6.9686	10.7564	6.9686	10.7564	6.9686	10.3031	6.6575
CR=1,4	9.9869	6.4219	10.0097	6.5186	10.0486	6.4939	9.6236	6.6020
CR=1,4 & B=1	9.9144	6.4307	9.8565	6.4740	9.7377	6.5390	9.1957	6.2795
CR=1,4 & B=1.5	9.9004	6.4102	9.8284	6.4701	9.6647	6.5507	11.4741	6.3601
CR=1,4 & B=2	9.8716	6.4346	9.7694	6.5206	9.7151	6.6559	11.3221	6.4451

Table 7. The Results of 3, 5, 9-Point Hanning Windowing.

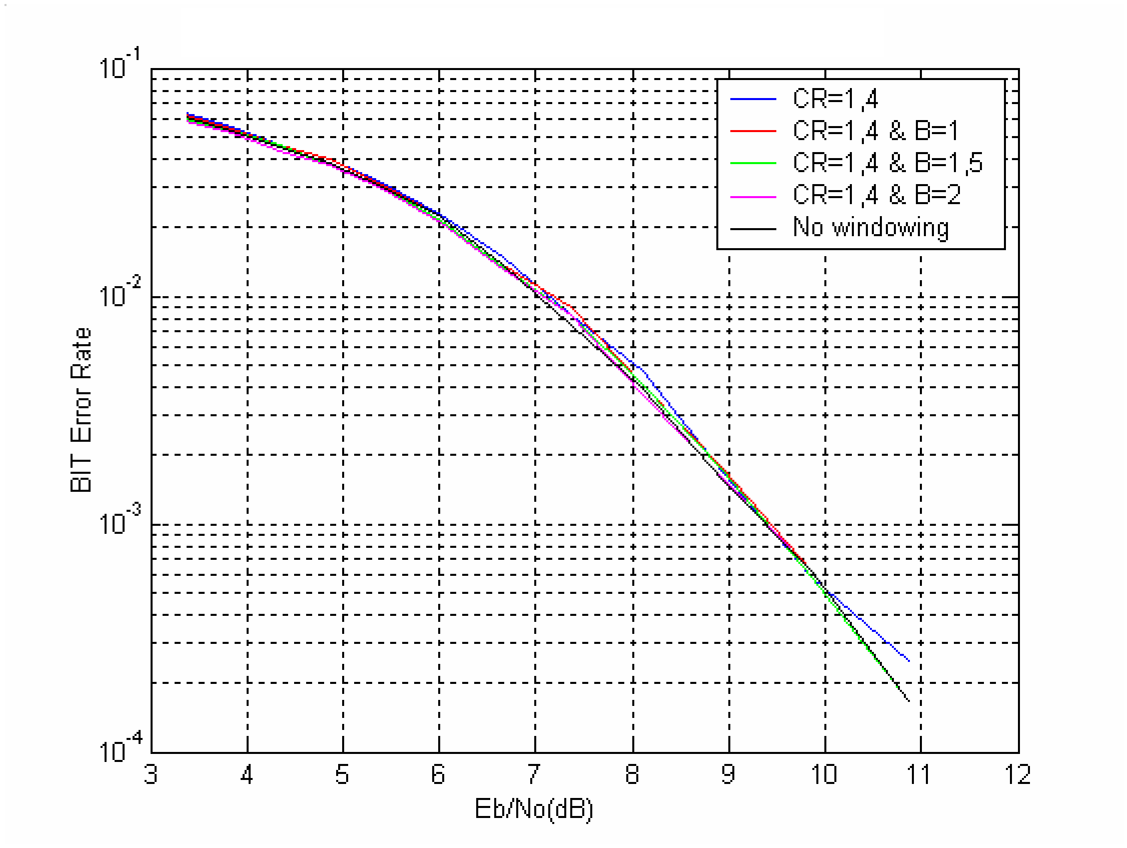


Figure 23. BER Performance of the System with 3-Point Hanning Windowing (CR: Clipping Ratio, B: Bottom Level)

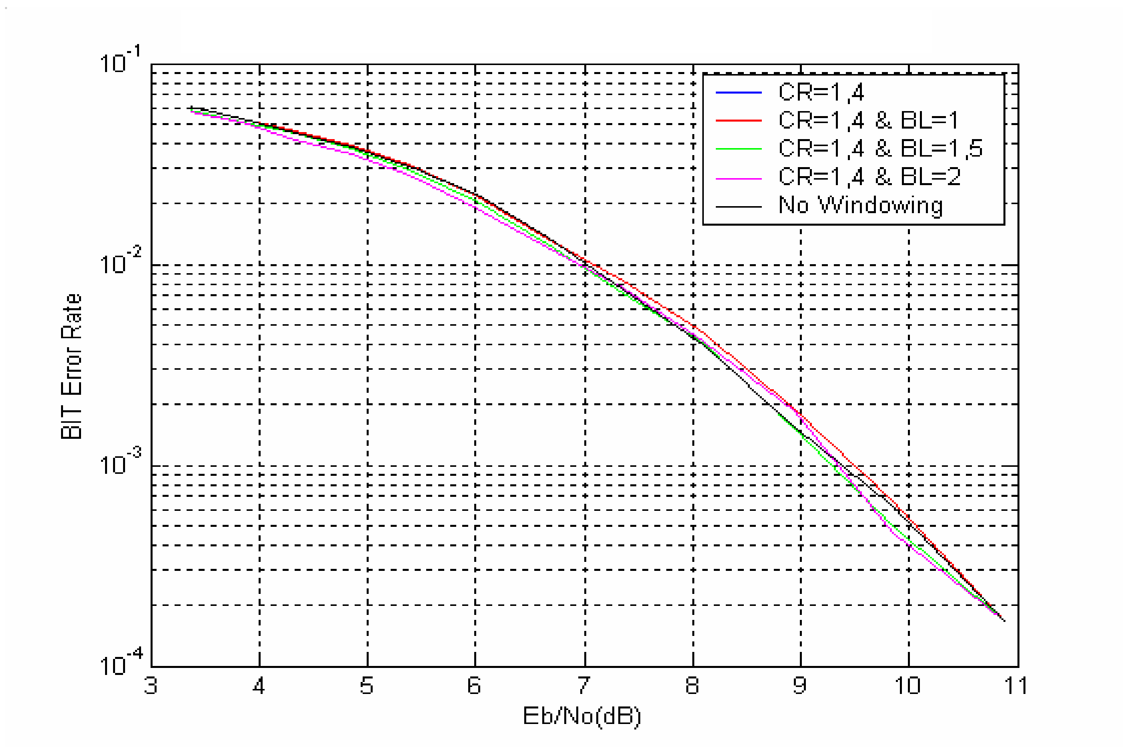


Figure 24. BER Performance of the System with 5-Point Hanning Windowing (CR: Clipping Ratio, B: Bottom Level)

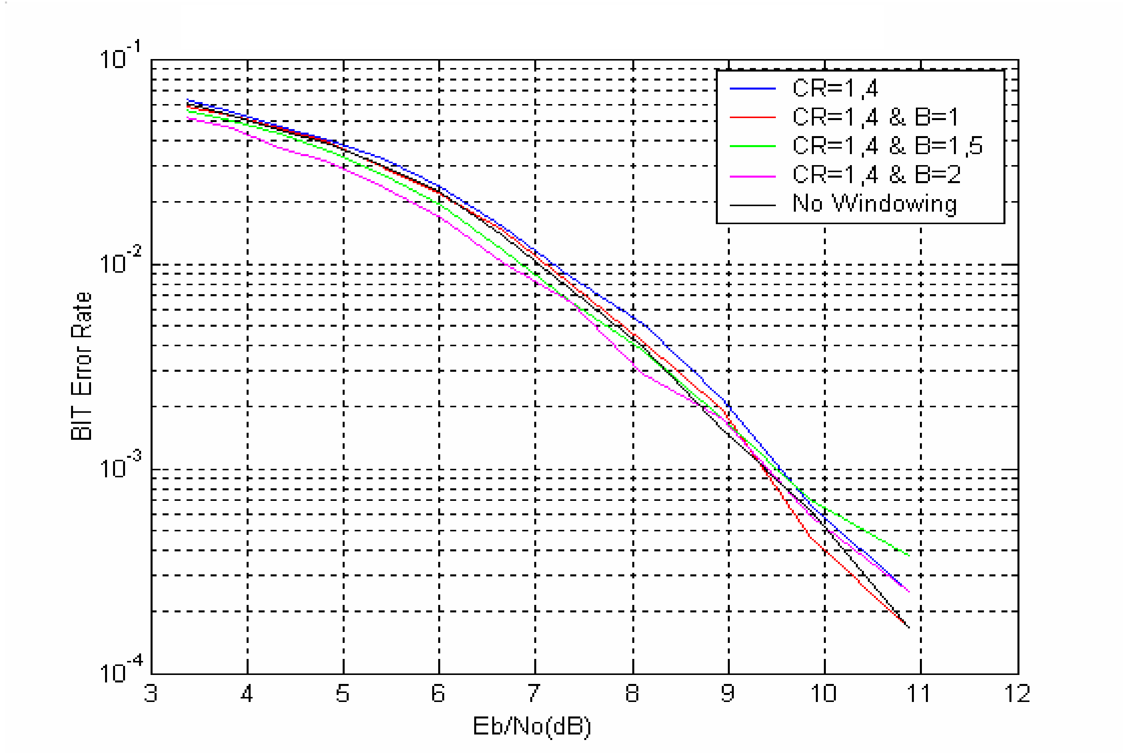


Figure 25. BER Performance of the System with 9-Point Hanning Windowing (CR: Clipping Ratio, B: Bottom level)

This chapter studied the simulation performance of the COFDM system and presented the BER performances of the system under four channels. The effects of AWGN, multipath, and AWGN and multipath effects over QPSK constellations were examined. As the main of goal of this thesis, the chapter also presented the performance of the COFDM system under mobile outdoor channel conditions. The roles played by RM codes, peak clipping, and Hanning windowing on PAPR reduction were also reported.

V. CONCLUSION

In this thesis, a COFDM based digital communication system with Reed-Muller error correction coding was successfully simulated. Three different types of RM codes and, for comparison, a rate one-half convolutional code were investigated through simulation studies using QPSK modulation type.

In order to test the COFDM system in both indoor and outdoor channel environments, we created a step-by-step test methodology. Using four different simulation steps, both indoor and outdoor channel characteristics were studied. The bit-error performance curves for each of the RM codes and the convolutional code were obtained.

Also, Hanning windowing and peak clipping techniques were investigated for reducing the PAPR in OFDM systems.

A. CONCLUSIONS

The results showed that the COFDM system is robust in indoor channel environments. However, for the outdoor channel environments, the system requires higher signal-to-noise ratios (SNR) to achieve the same bit-error rate (BER) performance as in the case of the indoor channel. Also, BER performance curves showed that the RM codes provide better performance in indoor channel environment at low SNR. However, convolutional codes provide better performance in the outdoor channel environments.

Reed-Muller codes are straightforward to implement, and they provide a wide range of coding options. However, RM codes did not provide the expected improvement in PAPR reduction as claimed by [15]. The results showed that they have almost the same performance in PAPR reduction as convolutional codes.

The addition of Hanning windowing and peak clipping improve PAPR reduction, improvement with Hanning windowing is limited. The results showed that, as the width of the window is increased, the bit-errors increased as well. Peak clipping provides considerable reduction in PAPR but at the cost of increased bit error. This presents a tradeoff between the PAPR reduction and the BER performance.

B. FUTURE WORK

The performance of a COFDM based digital communication system with three different Reed-Muller codes has been successfully examined in this thesis. However, simulating the system with different (r, m) pairs would be of interest. We recommend that the RM codes be further investigated to find optimal (r,m) pairs to provide improvement in PAPR reduction.

In the simulations, we used QPSK modulation technique. In future work, 16- and 64-QAM modulation techniques may be used. Therefore, M-ary QAM can be examined in a future work.

As described in Chapter II, COFDM based digital communication systems suffer from frequency offset and are sensitive to time and frequency synchronization. The effects of frequency errors in a COFDM system are analyzed in [6]. The work in this thesis and that in [6] can be combined to analyze the effects of Reed-Muller error correction coding on frequency errors.

It is well known that the performance of a decoder is significantly increased with soft decision decoding. Therefore, in addition to the hard decision decoding technique used in this thesis, a COFDM system can be analyzed with a decoder utilizing a soft decision decoding technique.

APPENDIX A. SIMULATION-ARCHITECTURE

The MATLAB m-files used in the simulation are described in this chapter. In order to achieve the purpose of this thesis, the COFDM simulation code in [4] was modified and adopted. Since some of the subroutines in [4] were modified and some were directly used in the MATLAB programming, citation to [4] was intentionally omitted in the text in this appendix.

A. GENERAL STRUCTURE OF MATLAB PROGRAMMING

Since the main goal of this thesis was to simulate the COFDM system by utilizing RM codes, all signal processing including forward error correction coding (FEC) and different types of channel characteristics were performed at the baseband.

1. COFDM Transmitter

The transmitter part of the COFDM system is shown in Figure 26. The subblocks of the transmitter are explained in the following discussion.

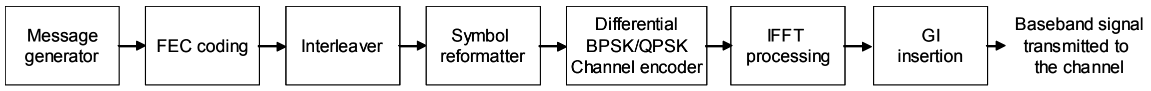


Figure 26. COFDM Transmitter Block Diagram.

a. Message Generator

This block generates a random bit sequence representing the data bits. The length of the message is variable and selected by the user. In order to use the same random message in different simulations, a seed parameter is used in this subblock. As long as the seed value remains the same, the message generator generates the same random message and makes it possible to run the simulation with different parameters and compare the results for the same information bit sequence.

b. *FEC Coding*

As described in Chapter III, in the IEEE 802.11a standard, data are encoded with a convolutional encoder with the coding rate $r = 1/2, 2/3$, or $3/4$. The convolutional encoder uses the half-rate industry-standard generator polynomials, $g_0 = 133_8$ and $g_1 = 171_8$ [7]. In addition to convolutional codes used in the simulation in [4], RM codes are also used in this thesis. The simulation allows the user to choose either one of these two FEC code types or no FEC code. If the convolutional codes are chosen, the *n* and *k* of the code are chosen by the user. If the RM codes are desired, the *r* and *m* pair is chosen by the user.

c. *Interleaver*

After determining the length of the message sequence, the simulation program presents all possible interleaver pairs on the screen. According to the interleaver dimension selected by the user, the block interleaver takes a block of encoded bits and permutes them as described in Chapter II.

d. *Symbol Reformatter*

Since binary phase shift keying (BPSK) or quaternary phase shift keying (4-PSK or QPSK) can be used in the simulation, the symbol reformatter block resizes the symbol lengths as 1-bit or 2-bit in order to form the BPSK or QPSK signal format.

e. *Differential Encoder*

Before the symbols are mapped into the signal, differential encoding is performed. Differential encoding is based on a cumulative summation. Cumulative summations are performed by adding in the modulo-*N* format. If the frequency differential encoding is performed, then the differential encoding is accomplished along the row vectors of the complex-valued modulation matrix. The first element is added to the second element in the row, and the result is replaced as the second element. The first, second and third elements are added and replaced as the third element and this operation is repeated until all elements in the row are completed. If the time differential encoding is performed, this operation is accomplished along the column vectors of the complex-valued modulation matrix.

f. IFFT Processing

The *N*-point IFFT operation is performed in order to convert the frequency domain data into the time domain. Although $N = 64$ is used for all the tests in this thesis, this value can be changed in the *cofdmsim.m* macro file.

g. Guard Interval Insertion

In order to eliminate the ISI and ICI effects, 16 time domain samples are added to the output of the IFFT processor in the IEEE 802.11a standard. In this simulation, this value remains fixed at 16 as in the standard. However, if it is necessary to extend this value in order to eliminate the more severe ISI or ICI effects, this value can be changed in the program.

2. COFDM Receiver

The operations in the receiver part of the system are the inverse operations of the transmitter. The receiver part of the system is shown in Figure 27. The subblocks of the receiver are explained below.

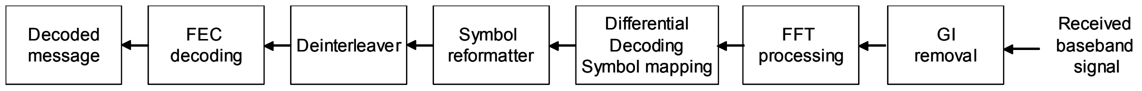


Figure 27. COFDM Receiver Block Diagram.

a. Guard Interval Removal

The time domain samples added in the transmitter to reduce or remove the ISI and ICI effects are removed in this block in order to pass the actual information block to the sunsequent subblocks for processing.

b. FFT Processing

The N-point FFT operation is performed to convert the time domain data into the frequency domain. Due to the multipath effects of the transmission channel, the orthogonality of the carriers may be distorted. Therefore, the output of the FFT processor can be an array of complex values.

c. Differential Decoding and Symbol Reformatting

According to the selection of frequency domain or time domain encoding, the correponding differential decoding is performed in the receiver. After decoding, the rows or columns of ones added by the differential coding operation in the transmitter are removed and symbols are reconstructed according to the magnitude and the phase of received complex values, which are the output of the FFT processor.

d. Deinterleaver

The output of the symbol reformatter is deinterleaved to obtain the actual message order. Thus, the burst errors caused by the noisy channel are distributed, and these randomly distributed errors are sent to the FEC decoder.

e. FEC Decoder

Depending on the coding type used in the transmitter, either the Viterbi decoder or RM decoder decodes the signal received from the deinterleaver.

f. Decoded Message

The output of the FEC decoder is the received information message. At this point, depending on the channel characteristics and the capability of the FEC decoding, some symbol errors may still occur. In order to present the performance of the entire system, the BER is calculated and illustrated by the BER versus E_b/N_0 graphs. Additionally, the PAPR values are also calculated and tabulated.

3. Channel Models

In this simulation, five different channels are used. Channel 0 is a noise-free channel and is performed to verify whether or not the MATLAB coding works appropriately. Channel 1 is an AWGN channel. Channel 2 is the multipath channel that is characterized by frequency selective fading in dB and multipath delays in microseconds. For this channel, Doppler shifts of 5 Hz, 10 Hz and 15 Hz were used. Additionally, a custom channel can be configured in accordance with the loss, frequency shifting and delay values entered by the user. Channel 3 is the combination of Channels 1 and 2, i.e., it has both AWGN and multipath channel characteristics. Channel 4 represents a mobile outdoor channel that includes AWGN and severe multipath effects. For this channel, two sets of parameters are used having different delays, fading, and Ricean factors. The characteristics of the channels are listed in Table 8.

Channel	**Link Number**	**Noise**	**Multipath Effects**		
			Fading	**Doppler**	**Delay**
#0	-	-	-	-	-
#1	X	-	-	-	-
#2	1	-	X	5 Hz	X
	2			10 Hz	
	3			15 Hz	
	4		Customizable		
#3	1	X	X	5 Hz	X
	2			10 Hz	
	3			15 Hz	
	4		Customizable		
#4	A	X	X		
	B	X	X		

Table 8. Channel Characteristics.

B. MATLAB PROGRAMMING DETAILS

As described in the previous section, the simulation consists of three main blocks which are the receiver, channel and transmitter. The main subroutine of the simulation is *chancdl.m*, which first calls *cdrcdlft.m* for the transmission processing and then calls a number of subroutines for the different types of channels and finally calls *decdrcdl.m* for the receiver processing. The schematic arrangement of the simulation is shown in Figure 28.

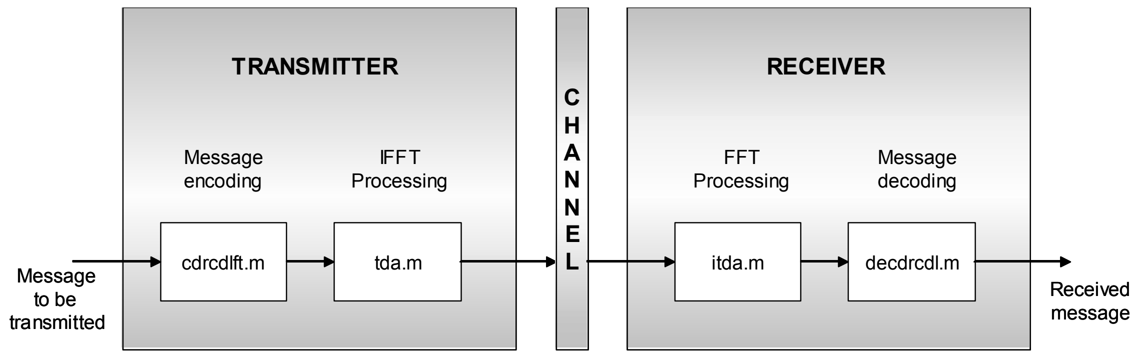


Figure 28. General Arrangement of the COFDM System.

In the transmitter part of the system, message encoding is performed by *cdrcdlft.m* and conversion from frequency domain to time domain and guard interval insertion is performed by *tda.m*. The inverse operations are performed by *itda.m* and *decdrcdl.m* in the receiver. The m-file *itda.m* is responsible for the guard interval removal and conversion to the frequency domain. The message decoding is performed by *decdrcdl.m*. The details of these m-files are given in the next subsection.

1. COFDM Transmitter

The arrangement of the m-files that interact with *cdrcdlft.m* are shown in Figure 29.

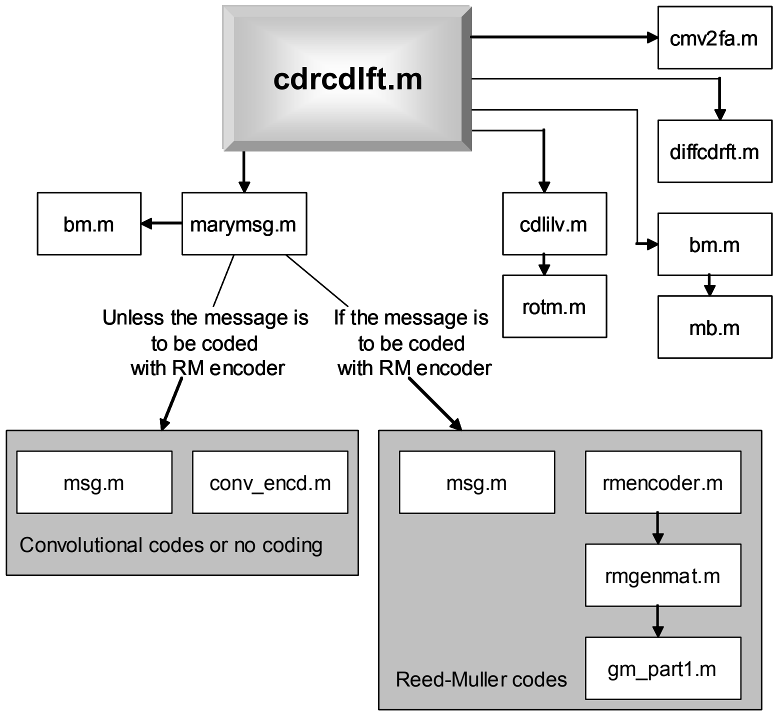


Figure 29. The Arrangement of the Macro Files within the cdrcdlft.m.

A random message of length *k* is generated by *msg.m*. The length *k* depends on the number of the symbols chosen by the user and the parameter *q*, which represents the number of bits that construct each M-ary symbol, where $M = 2^q$. The seed parameter *s* makes it possible to test the system with a different parameter but the same random message for comparing the results. After the random message sequence is created, it is encoded by *cnv_encd.m* or *rmencoder.m* depending on whether convolutional codes or RM codes are chosen by the user. If the FEC coding is not performed, *cnv_encd.m* is performed with the same *n* and *k* values. (Note that the parameter *k* here is not the length of the message, but represents the input block length of the convolutional encoder.) If RM codes are selected, the monomial vectors of the RM encoding matrix are created by *gm_part1.m* and the remaining rows of the encoding matrix are created by *rmgenmat.m*, using the monomial vectors. After that, the message sequence is coded by *rmencoder.m*. If convolutional codes are selected, the message sequence is coded by *conv_encd.m*. After FEC coding, the coded bit sequence is returned back to *marymsg.m*, which then calls *bm.m* to form symbols by grouping *q*-bits together to create M-ary symbols. The output is a vector of equivalent decimal numbers. The macro *marymsg.m* calculates these values using the other subroutines according to the input parameters *q, n, m, ecc, r_rm*, and *m_rm* where *n* is the number of the rows and *m* is the number of the columns of the overall output message matrix. The parameter *ecc* represents the FEC coding option, that is, the convolutional codes or RM codes, and *r_rm* and *m_rm* parameters are the *r* and *m* values of the $R(r,m)$ code. (The parameter *m* also represents the number of the COFDM frequency tones. It must be an even positive integer in order to completely fill the assigned bandwidth.) After the coded symbol values are returned back to *cdrcdlft.m*, this function calls the subroutine *cdlilv.m* to interleave the array. The subroutine parameters are *l, k, dcase, s* and *sync*, where *l* represents the number of the rows and *k* represents the number of the columns in the interleaver matrix. The parameter *dcase* is a variable indicating the interleaving method to be used. Although there are nine cases in the simulation program, only *dcase 0*, which indicates the simple interleaving, is performed in this thesis and the others are beyond the scope of this thesis. The parameter *s* represents the input message to be interleaved and *sync* represents the frame synchronization bits. However, this parameter was also not used in this thesis.

The interleaved message array is converted from the M-ary to N-ary format for the N-PSK modulation. Although QPSK is used in this thesis, this value remains as a variable for different PSK modulation types. This conversion is conducted by *bm.m* and *mb.m*. The signal in the M-ary format is first converted to binary by *mb.m* and then converted to the N-ary format by *bm.m*.

The message array that is interleaved and converted from M-ary to N-ary is then differentially encoded since the message array symbols are represented in decimal notation. The transition of symbols to complex values is performed by *difcdrft.m*. The output of this subroutine is an array that represents the differentially encoded complex values. The symbol-to-complex modulation value mapping is performed in two ways depending on whether the differential encoding is processed in time or frequency.

The last subroutine used within *cdrcdlft.m* is *cmv2fa.m*, which takes the complex valued array *M* and the number of the FFT points *N* as inputs and arranges *M* into a frequency array. The output is the shifted frequency array, which has padded zeros in the middle of the array. The number of these zeros is equal to the difference between the number of FFT points and the number of the complex values. Although RF conversion and filtering is not included in this simulation, the padded zeros are considered as a guard band for the filter slopes. At this point, *cdrcdlft.m* completes its task and sends the frequency array to *chancdl.m*.

In order to generate the COFDM frequencies, the macro file *tda.m* is called by *chancdl.m*. The IFFT operation is performed on complex modulation frequency array by *tda.m*, and the periodic guard interval is also inserted. The input parameter N_g is the number of the additional time domain samples added to the beginning of the information symbol interval as described in Chapter II. The output of this subroutine is the time domain samples prepared to be transmitted through the channel.

The input and output parameters of the subroutines that performed in the transmitter part of the COFDM simulation are listed in Table 9.

The name of the subroutine	Which program calls	Input parameters	Output parameters
cdrcdlft.m	chancdl.m	picy_n, pic, dcase, s, freqno, rintlv, N, mary,nary,fort	Fa, MD, B_ce, B_random, nsymno
marymsg.m	cdrcdlft.m	q, n, m, ecc, r_rm, m_rm	vmary_ce, random_bit, enc_output
bm.m	cdrcdlft.m, marymsg.m	q, v	m
msg.m	marymsg.m	k	u
cnv_encd.m	marymsg.m	ce_g, ce_k0,ce_input	ce_output
rmencoder.m	marymsg.m	r, m, msg1	encd_msg
rmgenmat.m	rmencoder.m	r, m	GM, k
gm_part1.m	rmgenmat.m	m	bb
cdlilv.m	cdrcdlft.m	l, k, dcase, s, SYNC	si
rotm.m	cdlilv.m	v, m	vp, vn
mb.m	cdrcdlft.m	q, m	b
diffcdrft.m	cdrcdlft.m	q, m, fort	MD
cmv2fa	cdrcdlft.m	N, M	X

Table 9. The Input and Output Parameters of the Subroutines Run within the Transmitter Part of the COFDM Simulation.

2. COFDM Receiver

The various m-files used in the simulations of the COFDM receiver and their interactions are shown in Figure 30. The receiver operations are the inverse of those in the transmitter.

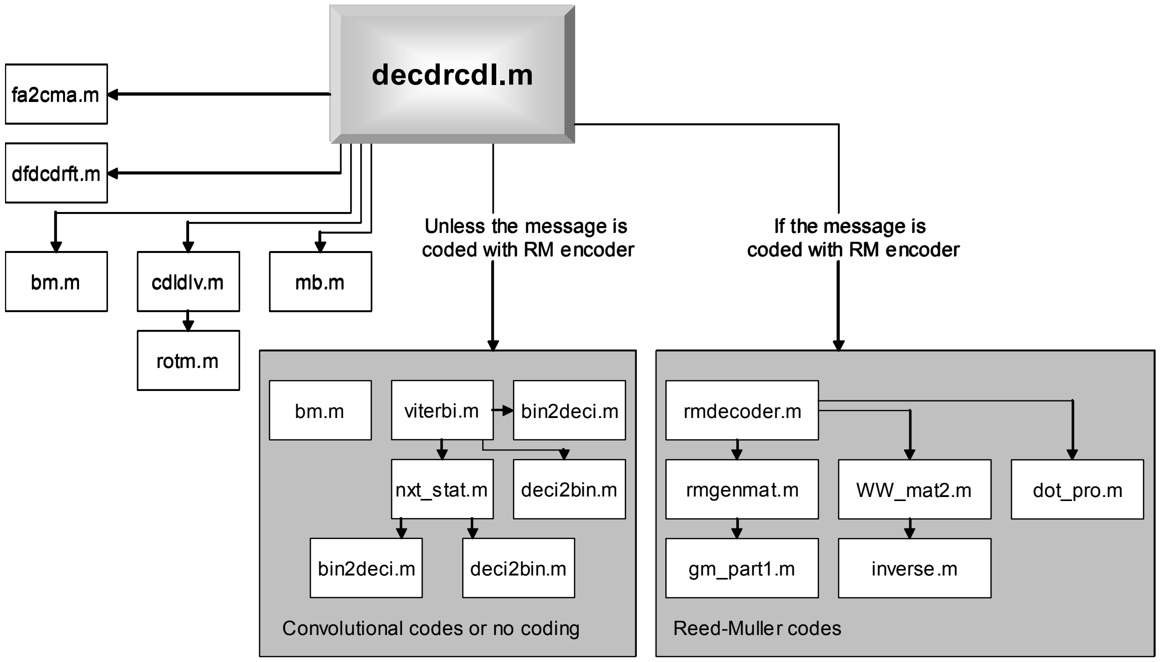


Figure 30. The Arrangement of the Macro Files within the decdrcl.m.

The complex modulation array is obtained from the frequency array by using *fa2cma.m*. The input parameters are *X* and *K* where *X* is the complex frequency array to be rearranged and *K* is half the number of COFDM frequency tones. The output parameter *Mn* is the complex modulation array of actual interleaved frequencies. After the frequency array is reconstructed into the complex array, the complex modulation values are differentially decoded into N-ary symbols by *dfdcdrft.m*. All reference ones added in the transmitter are removed, and the remaining received message matrix is decoded. The input parameters *MD, fort, q* and *qp*, where *MD* is the complex modulation values as the output of the m-file *fa2cma.m* and *q* is the number of bits in a symbol according to the N-ary format. The parameter *fort* represents whether the differential coding is performed in frequency or time. The output parameters of this m-file are *s* and *M*, where *s* represents the corresponding mapped symbols in decimal format and *M* is the differentially decoded modulation array. In order to convert the modulation array values from N-ary to M-ary, values are converted from N-ary to binary by *mb.m* and binary to M-ary by *bm.m*. The m-file *cdldlv* deinterleaves. The input parameter *si* represents the received interleaved message array and the other parameters are the same on the ones in *cdlilv.m*. The m-file *rotm.m* is called by *cdlilv.m* to rotate the array.

The input and output parameters of the subroutines used in the receiver part of the COFDM simulation are listed in Table 10.

The name of the subroutine	**Which program calls**	**Input parameters**	**Output parameters**
decdrcdl.m	Chancdl.m	picy_n, pic, dcase, K, Fa, nsymno, freqno, rdintlv, cdintlv, mary, nary, fort, B_random, ecc, r_rm, m_rm	outmsg, decoder_output,_bit, random_msg, random_bit, M, MM, binary_value
fa2cma.m	decdrcdl.m	K, X	Mm
dfdcdrft.m	decdrcdl.m	qp, q, MD, fort	s, M
bm.m	decdrcdl.m	q, v	M
cdldlv.m	decdrcdl.m	l, k, dcase, si, SYNC	s
rotm.m	cdldlv.m	v, m	vp, vn
mb.m	decdrcdl.m	q, m	b
viterbi.m	decdrcdl.m	v_G, v_k, channel_output	decoder_output, survivor_state, cumulated_metric
nxt_stat.m	viterbi.m	current_state, input, v_L, v_k	next_state, memory_contents
bin2deci.m	nxt_stat.m, viterbi.m	v_x	v_y
deci2bin.m	nxt_stat.m, viterbi.m	x, l	y
rmdecoder.m	decdrcdl.m	Me, r, m	Msg_decoded_line
rmgenmat.m	rmdecoder.m	r, m	GM, k
gm_part1	rmgenmat.m	m	bb
WW_mat2.m	rmdecoder.m	GM, r, m ,k	WW1
inverse.m	WW_mat2.m	D	d
dot_pro.m	rmdecoder.m	X, S	z

Table 10. The Input and Output Parameters of the Subroutines Run within the Receiver Part of the COFDM Simulation.

3. COFDM Channel

Figure 31 shows the m-files used to simulate the channel.

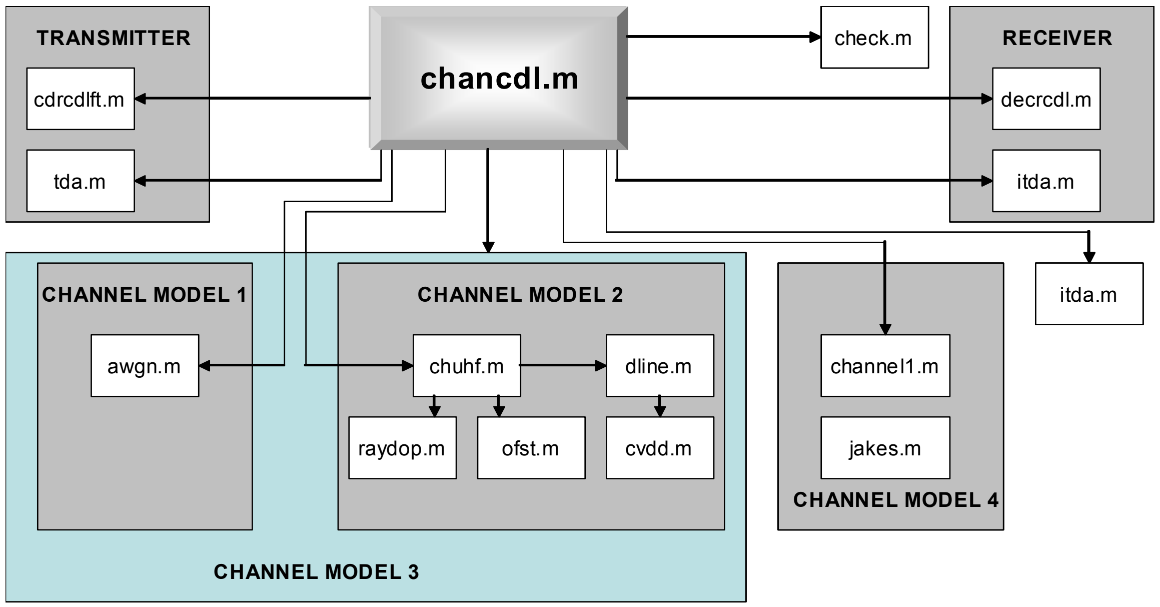


Figure 31. The Arrangement of the Macro Files within the chancdl.m.

The input parameters of the macro file *chuhf.m* are *s, x, loss, dly, dop, N* and *freqspace*, where *loss* is the signal loss in dB, *dly* is the time delays in microseconds and *dop* is the Doppler frequency shifting in Hz. Multipath delay effects are created by *dline.m* and *cvdd.m*. The input array is filtered by a FIR filter whose coefficients are created by *cvdd.m* according to the values in the array *dly*. As a next step, the Doppler shift frequency is calculated by *ray_dop.m* according to the seed value *s*, *freqspace* and the number of the FFT points *N*. The channel offset is calculated by *ofst.m*. Finally, the power losses are calculated by multiplying each loss value in the array *loss* with the corresponding message value. The output of *chuhf.m* is a complex valued time domain array including multipath effects.

Channel 3 is realized by using *awgn.m* and *chuhf.m*. Channel 4 is simulated by *channel1.m* and *jakes.m*. The first subroutine calls the second and provides two severe channels having different multipath effects. The selection of these two channels must be conducted before running the simulation.

The input and output parameters of the subroutines simulating the channel part of the COFDM simulation are shown in Table 11.

The name of the subroutine	**Which program calls**	**Input parameters**	**Output parameters**
awgn.m	chancdl.m	X, N, sigma	Y
chuhf.m	chancdl.m	s, x, loss, dly, dop, N, freqspace	y
raydop.m	chuhf.m	s, M, N, es	c
ofst.m	chuhf.m	e, N, x	xo
dline.m	chuhf.m	x, d	xd
cvdd.m	dline.m	x, alpha	y
channel1.m	chancdl.m	xt	yr
jakes.m	channel1.m	Fd, Fs, N	g
itda.m	chancdl.m	Ng, y	y

Table 11. The Input and Output Parameters of the Subroutines Run within the Channel Part of the COFDM Simulation.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. COFDM MATLAB CODE

This appendix presents the MATLAB m-files used in the simulations. Descriptions of the m-files here can be found in Appendix A.

AWGN

```
%AWGN

%TITLE           : ADDITIVE WHITE GAUSSIAN NOISE

%--------------------------------------------------------------------------------
% Thesis Advisor      : Prof M. Tummala, Naval Postgraduate School
% Author              : Prof. Paul H. Moose, Naval Postgraduate School
% Modified by         : TEZEREN,Serdar Umit,LTJG,TURKISH NAVY
%--------------------------------------------------------------------------------
function Y = awgn(X,N,sigma)
%Find dimensions of the input array
[rr,cc]=size(X);
%Generate a random real part
seed=14;
randn('state',seed);
wreal=randn(rr,cc);
%Generate a random imaginary part
seed=seed+6;
randn('state',seed);
wimg=i*randn(rr,cc);
%An array of random complex entries chosen from a normal distribution with
%mean 0.0 and variance 1.0. Array dimension is the same as X.
W=wreal+wimg;
%Random noise multiplied by the sigma factor and added to the signal.
Y=X+(sigma.*W);
%--------------------------------------------------------------------------------
```

BIN2DECI

```
%BIN2DECI

%TITLE             : BINARY TO DECIMAL CONVERSION

%--------------------------------------------------------------------------------
% Thesis Advisor      : Prof M. Tummala, Naval Postgraduate School
% Author              : Tan Kok Chye, Naval Postgraduate School
%----------------------------------------------------------------------------------
function v_y=bin2deci(v_x)
v_l=length(v_x);
v_y=(v_l-1:-1:0);
v_y=2.^v_y;
v_y=v_x*v_y';
%----------------------------------------------------------------------------------
```

BM

```
%BM

%TITLE             : BINARY TO M-ARY CONVERTER

%-------------------------------------------------------------------------------
% Thesis Advisor      : Prof M. Tummala, Naval Postgraduate School
% Author              : Prof. Paul H. Moose, Naval Postgraduate School
% Modified by         : Tan Kok Chye, Naval Postgraduate School
%---------------------------------------------------------------------------------
function m=bm(q,v)
%Find the length of input vector,v,and determine if there is a remainder
%after dividing by q
n=length(v);
r=rem(n,q);
%If there is no remainder,don't pad v input vector. Otherwise add the appropriate
%number of zeros to generate a code word with an exact multiple of q bits.
if r==0
v=v;
else
v=[v zeros(1,q-r)];
end
%Place least significant bit of the symbol on the left end.
map=1;
for j=1:q-1
map=[map 2^j];
end
%Remove q bits at a time from v to generate m-ary symbol values.
n=length(v);
p=round(n/q);
A=zeros(q,p);
A(:)=v;
m=map*A;
m_ary_msg=m;
%---------------------------------------------------------------------------------
```

CDLDLV

```
%CDLDLV

%TITLE           : BLOCK DEINTERLEAVER

%------------------------------------------------------------------------
% Thesis Advisor      : Prof M. Tummala, Naval Postgraduate School
% Author              : Prof. Paul H. Moose, Naval Postgraduate School
% Modified by         : Tan Kok Chye, Naval Postgraduate School
%--------------------------------------------------------------------------
function s=cdldlv(l,k,dcase,si,SYNC)
si(length(si)+l-length(SYNC):length(si))=zeros(l,length(SYNC));
N=length(si);
if l*k==N
x=zeros(l,k);
x(:)=si;
K=(1:k)-1;
CR=K.*(K+1)/2;
L=(1:l)-1;
RR=L.*(L+1)/2;
if dcase==1
for kk=1:k
x(:,kk)=rotm(x(:,kk),CR(kk));
end
elseif dcase==2
for kk=1:k
[z,x(:,kk)]=rotm(x(:,kk),CR(kk));
end
elseif dcase==3
for kk=1:l
x(kk,:)=rotm(x(kk,:),RR(kk));
end
elseif dcase==4
for kk=1:l
[z,x(kk,:)]=rotm(x(kk,:),RR(kk));
end
elseif dcase==5
for kk=1:k
x(:,kk)=rotm(x(:,kk),CR(kk));
end
for ll=1:l
x(ll,:)=rotm(x(ll,:),RR(ll));
end
elseif dcase==6
for kk=1:k
```

```
[z,x(:,kk)]=rotm(x(:,kk),CR(kk));
end
for ll=1:l
x(ll,:)=rotm(x(ll,:),RR(ll));
end
elseif dcase==7
for kk=1:k
x(:,kk)=rotm(x(:,kk),CR(kk));
end
for ll=1:l
[z,x(ll,:)]=rotm(x(ll,:),RR(ll));
end
elseif dcase==8
for kk=1:k
[z,x(:,kk)]=rotm(x(:,kk),CR(kk));
end
for ll=1:l
[z,x(ll,:)]=rotm(x(ll,:),RR(ll));
end
end
x=x';
s=x(:);
s=s';
end
%-------------------------------------------------------------------------------------
```

CDLILV

```
%CDLILV

%Title : BLOCK INTERLEAVER

%-----------------------------------------------------------------------
% Thesis Advisor       : Prof M. Tummala, Naval Postgraduate School
% Author               : Prof. Paul H. Moose, Naval Postgraduate School
% Modified by          : Tan Kok Chye, Naval Postgraduate School
%-------------------------------------------------------------------------
function si = cdlilv(l,k,dcase,s,SYNC)
N=length(s);
if l*k==N
x=zeros(l,k);
x=x';
x(:)=s;
x=x';
Intermediate_mx=x;
K=(1:k)-1;
CR=K.*(K+1)/2;
L=(1:l)-1;
RR=L.*(L+1)/2;
if dcase==1
for kk=1:k
[z,x(:,kk)]=rotm(x(:,kk),CR(kk));
end
elseif dcase==2
for kk=1:k
x(:,kk)=rotm(x(:,kk),CR(kk));
end
elseif dcase==3
for kk=1:l
[z,x(kk,:)]=rotm(x(kk,:),RR(kk));
end
elseif dcase==4
for kk=1:l
x(kk,:)=rotm(x(kk,:),RR(kk));
end
elseif dcase==5
for ll=1:l
[z,x(ll,:)]=rotm(x(ll,:),RR(ll));
end
for kk=1:k
[z,x(:,kk)]=rotm(x(:,kk),CR(kk));
end
```

```
elseif dcase==6
for ll=1:l
[z,x(ll,:)]=rotm(x(ll,:),RR(ll));
end
for kk=1:k
x(:,kk)=rotm(x(:,kk),CR(kk));
end
elseif dcase==7
for ll=1:l
x(ll,:)=rotm(x(ll,:),RR(ll));
end
for kk=1:k
[z,x(:,kk)]=rotm(x(:,kk),CR(kk));
end
elseif dcase==8
for ll=1:l
x(ll,:)=rotm(x(ll,:),RR(ll));
end
for kk=1:k
x(:,kk)=rotm(x(:,kk),CR(kk));
end
end
si=x(:);
si=si';
end
si(length(si)-length(SYNC)+1:length(si))=SYNC;
%-----------------------------------------------------------------------------------------
```

CDRCDLFT

```
%CDRCDLFT

%Title              : COFDM ENCODER WITH BLOCK INTERLEAVER

%-----------------------------------------------------------------------
% Thesis Advisor      : Prof M. Tummala, Naval Postgraduate School
% Author              : Prof. Paul H. Moose, Naval Postgraduate School
% Modified by         : TEZEREN,Serdar Umit,LTJG,TURKISH NAVY
%-----------------------------------------------------------------------
func-
tion[Fa,MD,B_ce,B_random,nsymno,enc_output]=cdrcdlft(picy_n,pic,dcase,s,freqno,rint
lv,cintlv,N,mary,nary,fort,ecc,r_rm,m_rm);
% Determine whether the number of OFDM frequencies are even (# of matrixcol-
umns),indicated
% by the "freqno" parameter. If odd go to error message. Odd frequencies are not allowed
% since the formation of the frequency array is symmetrical and even.
if rem(freqno,2)~=0
disp('ERROR: The number of matrix columns,freqno,representing OFDM frequencies
must be an even number!')
elseif rem(freqno,2)==0
% Determine if the row and column interleave parameters are greater than freqno when
% multiplied together. If not, then display error message and stop.
if (rintlv*cintlv)<(freqno)
disp('')
disp('ERROR: The row and column interleave parameters are not compatible with # of
OFDM frequencies!')
disp('')
else
% Calculate the row symbol number
symno=rintlv*cintlv/freqno;
% Display error message if symno and freqno not compatible with rintlv and cintlv and
stop.
% If not compatible,the interleaver function does not work correctly.
if rem(symno,1)~=0
disp('')
disp('ERROR: The row and column interleave parameters are not compatible with # of
OFDM frequencies!')
disp(' For the entered rintlv, cintlv, and freqno parameters, the calculated symno is:')
disp(symno)
multiesall=mltpl(rintlv,cintlv);
multies=multiesall(1,(2:length(multiesall)-1));
disp(' Possible choices for freqno based upon rintlv and cintlv are:')
disp('')
disp(multies)
```

```
elseif rem(symno,1)==0
if freqno >= N;
disp('')
disp('ERROR: The number of frequency points, N, needs to be increased !')
disp('N must be larger than:')
disp('')
disp(freqno)
disp('')
elseif freqno < N;
% Generate a random message matrix of m-ary symbols,based upon parameter,mary.
Nmbr_of_symbols =symno*freqno;
[B_ce,B_random,enc_output]=marymsg(mary,symno,freqno,ecc,r_rm,m_rm);
Rndm_m_ary_msg=B_random;
% Perform a block interleaving function on the matrix, B, with rintlv rows
% and cintlv columns.
SYNC=[];
[Br Bc]=size(B_ce);
Bt=B_ce';
Bvect=Bt(:)';
si=cdlilv(rintlv,cintlv,dcase,Bvect,SYNC);
Bi=reshape(si,Bc,Br)';
Intrlvd_array=Bi;
m1=bm(nary,mb(mary,Bi));
lengthm1=length(m1);
nsymno=lengthm1;
remm1=rem(lengthm1,freqno);
if remm1==0;
m1=m1;
else
zero=zeros(freqno-remm1);
m1=[m1 zero(1,:)];
end
length2m1=length(m1);
m=(reshape(m1,freqno,length2m1/freqno))';
N_ary_msg=m;
% Generate a differentially encoded matrix of complex
% values with unit magnitude and one of (2^n) equal phases.
MDD=difcdrft(nary,m,fort);
[MDm MDn]=size(MDD);
MD=MDD;Cmplx_mod_array=MDD;
% Form the frequency array of modulation values that include guard interval.
Fa=cmv2fa(N,MD);
Freq_array=Fa;
end;end
end;end
%---------------------------------------------------------------------
```

CHANCDL

```
%CHANCDL

% TITLE           : SIMULATIONS FOR AWGN & MULTIPATH FADING CHANNEL

%----------------------------------------------------------------------------------
% Thesis Advisor      : Prof M. Tummala, Naval Postgraduate School
% Author              : Dave Roderick, Naval Postgraduate School
% Modified by         : TEZEREN,Serdar Umit,LTJG,TURKISH NAVY
%-------------------------------------------------------------------------------------
func-
tion[errmax,errors,freqerrs,papr]=chancdl(chnmdl,wait,prnt,picy_n,pic,dcase,s,freqno,rint
lv,cintlv,N,mary,nary,n,k,blklgth,Ng,sigs,loss,dly,dop,freqspace,fort,ecc,r_rm,m_rm)
sigvect=sigs;
klgth=length(k);
chklp=1;
errvect=[];
bervect=[];
freqerrmx=[];
errsperpr=[];
Es_No=[];
Eb_No=[];
sermx=[];
bermx=[];
rowerrmx=[];
symno=rintlv*cintlv/freqno;
for lp=1:length(sigvect);
%
[xmt,modvals,B_ce,B_random,nsymno,enc_output]=cdrcdlft(picy_n,pic,dcase,s,freqno,ri
ntlv,cintlv,N,mary,nary,fort,ecc,r_rm,m_rm);
xmtifft=tda(Ng,xmt);
[ppp]=papr3(xmtifft,N,freqno,pic);
xmtifft=ppp;
size_modvals=size(modvals)

xmtpts=1:length(xmtifft);
   if chnmdl==0
   sandn=xmtifft;
   elseif chnmdl==1
   disp(['Sigma=',num2str(sigvect(lp))]);
   sandn=awgn(xmtifft,N,sigvect(lp));
   elseif chnmdl==2
   sandn=chuhf(s+1,xmtifft,loss,dly,dop,N,freqspace);
   elseif chnmdl==3
   sandmltpth=chuhf(s+1,xmtifft,loss,dly,dop,N,freqspace);
   disp(['Sigma=',num2str(sigvect(lp))]);
   sandn=awgn(sandmltpth,N,sigvect(lp));
   elseif chnmdl==4
   sandnmlt=channel1(xmtifft);
   sandn=awgn(sandnmlt,N,sigvect(lp));
   end

sandnfft=itda(Ng,sandn);

K=(length(modvals(1,:)))/2;
[rcvd,rcvd_bit,random_msg,random_bit,M,MM,binary_value]=decdrcdl(picy_n,pic,dcase
,K,sandnfft,nsymno,freqno,rintlv,cintlv,mary,nary,fort,B_random,ecc,r_rm,m_rm);
Transmitted_msg=random_msg;
Received_msg=rcvd;

[er-
rors,bit_error,freqerrs,errmx,rowerrs]=check(pic,random_msg,random_bit,rcvd,rcvd_bit,
n,k(chklp),blklgth,ecc,r_rm,m_rm,enc_output,binary_value);
errvect=[errvect,errors];
bervect=[bervect,bit_error];
freqerrmx=[freqerrmx;freqerrs];
rowerrmx=[rowerrmx;rowerrs];
crntEs_No=1/(2*N*(sigvect(lp)^2));
crntEb_No=crntEs_No;
Es_No=[Es_No,crntEs_No];
Eb_No=[Eb_No,crntEb_No];
Es_Nodb=10*log10(Es_No);
Eb_Nodb=10*log10(Eb_No);
end
ser=errvect/(symno*freqno);
ber=bervect/(2*symno*freqno);
sermx=[sermx;ser];
bermx=[bermx;ber];
errsum=sum(errvect);
errsperpr=[errsperpr,errsum];
errmax=max(rowerrmx');
if picy_n==1
figure(pic+1)
plot(modvals,'*')
hold on;
plot(0,0,'+')
hold off;
title(['Transmitted Signal ',int2str(2^nary), '-ary Constellation Plot'])
xlabel(['Magnitude=1'])
axis('square');
orient tall
```

```
grid
   if prnt==1;
   print
   pause(10);
   end
pause(wait);
end
%

if picy_n==1
figure(pic+2)
plot([0:N-1],abs(xmt),'*')
title(['Frequency Array Plot (The number of FFT points are ',int2str(N),')'])
xlabel(['Guard interval length is ',int2str(Ng)])
axis('square');
orient tall
grid
   if prnt==1;
   print
   pause(10)
   end
pause(wait);
end
%
if picy_n==1
figure(pic+3)
surf(abs(modvals));
shading interp
grid
orient tall
title(['Magnitude of Transmitted Signal'])
xlabel('OFDM Freq #')
ylabel('Symbol Row Number')
zlabel(['Magnitude'])
   if prnt==1;
   print
   pause(10)
   end
   %pause(wait);
end
%
if picy_n==1
figure(pic+6)
plot(M,'*')
hold on;
plot(0,0,'+')
```

```
hold off;
title(['Received ', int2str(2^nary),'-ary Signal Constellation Plot, BEFORE Differential
Decoding'])
orient tall
axis('square');
grid
   if prnt==1;
   print
   pause(10)
   end
pause(wait);
end
%
if picy_n==1
figure(pic+7)
plot(MM,'+')
hold on;
plot(0,0,'+')
hold off;
title(['Received ',int2str(2^nary),'-ary Signal Constellation Plot, AFTER Differential De-
coding'])
orient tall
axis('square');
grid
   if prnt==1;
   print
   pause(10)
   end
pause(wait);
end
%
%
if picy_n==1
%roty_n=input('Do you want to rotate 3-D plot?(yes=1,no=0):');
roty_n=0;
figure(pic+8)
surf(abs(M));
grid on
shading interp
%grid
orient tall
title(['Magnitude Variation of Received Signal'])
xlabel('OFDM Freq #')
ylabel('Symbol Row Number')
zlabel(['Magnitude'])
   if roty_n==1
```

```
%Rotate the 3-D plot
        for k=1:5
        view(-70+10*k,15+10*k)
        disp('');
        disp('Press "enter" to rotate plot...');
        pause
        end
        %elseif roty_n==0
    end
    if prnt==1;
    print
    pause(10)
    end
    %pause(wait);
end
%
if errsum~=0
%2-D Error Performance Curve showing BER vs. Es/No.

    figure(pic+12)
    semilogy(Eb_Nodb,ber)
    grid
    if fort==1
        if dop==[5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5]
       title(['Link 1 : Performance graph: BIT Error Rate vs. Eb/No (Freq. Diff.Enc.)(Total
errors=',int2str(sum(errvect)),')'])
        elseif dop==[10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10]
       title(['Link 2 : Performance graph: BIT Error Rate vs. Eb/No (Freq. Diff.Enc.)(Total
errors=',int2str(sum(errvect)),')'])
        elseif dop==[15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15]
       title(['Link 3 : Performance graph: BIT Error Rate vs. Eb/No (Freq. Diff.Enc.)(Total
errors=',int2str(sum(errvect)),')'])
        else
       title(['Custom Link Performance graph: BIT Error Rate vs. Eb/No (Freq.
Diff.Enc.)(Total errors=',int2str(sum(errvect)),')'])
        end
    elseif fort==0
        if dop==[5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5]
       title(['Link 1 : Performance graph: BIT Error Rate vs. Eb/No (Freq. Diff.Enc.)(Total
errors=',int2str(sum(errvect)),')'])
        elseif dop==[10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10]
       title(['Link 2 : Performance graph: BIT Error Rate vs. Eb/No (Freq. Diff.Enc.)(Total
errors=',int2str(sum(errvect)),')'])
        elseif dop==[15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15]
       title(['Link 3 : Performance graph: BIT Error Rate vs. Eb/No (Freq. Diff.Enc.)(Total
errors=',int2str(sum(errvect)),')'])
```

```
       else
      title(['Custom Link Performance graph: BIT Error Rate vs. Eb/No (Freq.
Diff.Enc.)(Total errors=',int2str(sum(errvect)),')'])
       end
   end
   xlabel(['Eb/No(dB)(# of
OFDM=',int2str(freqno),')(dcase=',int2str(dcase),')(Interleaverpair=',int2str(rintlv),',',int2s
tr(cintlv),') M-ary=',int2str(2^mary),',Nary=',int2str(2^nary)]);
   ylabel(['Sigma Range:(',num2str(min(sigs)),'-
',num2str(max(sigs)),')(RS=',int2str(floor((n-
k)/2)),')(Symbol#=',int2str(symno*freqno),')(Seed=',num2str(s),')']);
   orient tall
%2-D Error Performance Curve showing SER vs. Eb/No.
   figure(pic+13)
   semilogy(Es_Nodb,ser)
   grid
   if fort==1
       if dop==[5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5]
      title(['Link 1 : Performance graph: Symbol Error Rate vs. Es/No (Freq.
Diff.Enc.)(Total errors=',int2str(sum(errvect)),')'])
       elseif dop==[10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10]
      title(['Link 2 : Performance graph: Symbol Error Rate vs. Es/No (Freq.
Diff.Enc.)(Total errors=',int2str(sum(errvect)),')'])
       elseif dop==[15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15]
      title(['Link 3 : Performance graph: Symbol Error Rate vs. Es/No (Freq.
Diff.Enc.)(Total errors=',int2str(sum(errvect)),')'])
       else
      title(['Custom Link Performance graph: Symbol Error Rate vs. Es/No (Freq.
Diff.Enc.)(Total errors=',int2str(sum(errvect)),')'])
       end
   elseif fort==0
       if dop==[5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5]
      title(['Link 1 : Performance graph: Symbol Error Rate vs. Es/No (Freq.
Diff.Enc.)(Total errors=',int2str(sum(errvect)),')'])
       elseif dop==[10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10]
      title(['Link 2 : Performance graph: Symbol Error Rate vs. Es/No (Freq.
Diff.Enc.)(Total errors=',int2str(sum(errvect)),')'])
       elseif dop==[15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15]
      title(['Link 3 : Performance graph: Symbol Error Rate vs. Es/No (Freq.
Diff.Enc.)(Total errors=',int2str(sum(errvect)),')'])
       else
      title(['Custom Link Performance graph: Symbol Error Rate vs. Es/No (Freq.
Diff.Enc.)(Total errors=',int2str(sum(errvect)),')'])
       end
   end
   text(min(ceil(Es_Nodb)),.18,['Loss=[',num2str(loss),']']);
   text(min(ceil(Es_Nodb)),.12,['Delay=[',num2str(dly),']']);
   text(min(ceil(Es_Nodb)),.08,['Doppler=[',num2str(dop),']']);
   xlabel(['Es/No(dB)(# of OFDM=',int2str(freqno),')(dcase=',int2str(dcase),')(Interleaver
pair=',int2str(rintlv),',',int2str(cintlv),') M-ary=',int2str(2^mary),',Nary=',int2str(2^nary)]);
   ylabel(['Sigma Range:(',num2str(min(sigs)),'-
',num2str(max(sigs)),')(RS=',int2str(floor((n-
k)/2)),')(Symbol#=',int2str(symno*freqno),')(Seed=',num2str(s),')']);
   orient tall
end
if prnt==1
print
pause(10)
end
%---------------------------------------------------------------------
```

CHANNEL1

```
%CHANNEL1

% TITLE          : SEVERE CHANNEL

%--------------------------------------------------------------------------------
% Thesis Advisor      : Prof M. Tummala, Naval Postgraduate School
% Author              : Prof R. Cristi, Naval Postgraduate School
% Modified by         : TEZEREN,Serdar Umit,LTJG,TURKISH NAVY
%----------------------------------------------------------------------------------
function  yr=channel1(xt)
[asd asn]=size(xt);
xt=xt.';
xt=xt(:).';
%  yr=channel(xt)
%  with xt=transmitted vector
%       yr=received vector
%       length(yr)=length(xt)
% if the program gives you an error, just increase the size of the
% transmitted vector "xt"
% Created by: Roberto Cristi, November 2003

Fs=1.25*10^6;           % sampling frequency (Hz)
Fd=200;                 % doppler frequency (Hz)

% Parameters for Channel A
 %Td=[0.0, 0.25,0.5,1.0,1.9,2.2];                  % time delays (in microsec)
 %PdB=[0.0,-1.0,-9.0,-10.0,-15.0,-20.0]; % Powers (in dB)
 %K=[0,0,0,0,0,0];                                         % Ricean Factor

% Parameters for Channel B
Td=[0.0, 0.25,9.0,13.0, 17.0, 20.0];    % time delays (in microsec)
PdB=[-2.5,0.0,-12.8,-10, -25.2, -16];  % Powers (in dB)
K=[0.5,0.5,0,0,0,0];                             % Ricean Factors

Td=Td*(10^(-6));        % time delays (in sec)
nd=round(Td*Fs);        % time delay in samples
P=10.^(PdB/10);                 % Powers (Linear)
Np=length(xt);

yr=zeros(size(xt));
for k=1:length(Td)
g=jakes(Fd, Fs, Np);
s=sqrt(P(k)/(K(k)+1));                      % random path
m=sqrt(P(k)*K(k)/(K(k)+1)); % direct path
```

```
total=s*g+m*exp(j*2*pi*Fd/Fs*(0:length(xt)-1));
yr=yr+[zeros(1,nd(k)), xt(1:length(xt)-nd(k))].*total;
end
yr=reshape(yr,asn,asd).';
%--------------------------------------------------------------------------------------
```

CHECK

```
%CHECK

% TITLE            : SOURCE AND SINK MESSAGE CHECKER

%-----------------------------------------------------------------------------------
% Thesis Advisor      : Prof M. Tummala, Naval Postgraduate School
% Author              : Prof. Paul H. Moose, Naval Postgraduate School
% Modified by         : TEZEREN,Serdar Umit,LTJG,TURKISH NAVY
%-----------------------------------------------------------------------------------
func-
tion[error_no,bit_error_total,freqerrs,errmx,rowerrs]=check(pic,x,xbit,y,ybit,n,k,blklgth,e
cc,r_rm,m_rm,enc_output,binary_value)

if ecc~=2;% unless reed muller
if blklgth>n
disp('')
disp('ERROR! The block length, blklgth, must be equal or less than the code word
length,n.')
disp('Please enter a smaller value for blklgth, or change n.')
disp('')
elseif blklgth<=n
   if n<k
   disp('')
   disp('Error! The code word length,n,must be equal or larger than the information
length,k.')
   disp('Please enter a larger value for n, or change k to a smaller number.')
   disp('')
   elseif n>=k
   First_matrix=x;
   Second_matrix=y;
   [rx cx]=size(x);
%Compare inputs x and y and generate error matrix, "errors"
   errors=(x~=y);
   First=xbit;
   Second=ybit;
   [rx1 cx1]=size(xbit);
%Compare inputs xbit and ybit and generate BIT error matrix, "bit_errors"
   bit_errors=(xbit~=ybit);
%Find the error distribution vs. OFDM frequencies
   freqerrs=sum(errors);
%Find the error location in "errors" where element in x and y differ.
   Error_locations=(find(errors))';
   Error_number=sum(sum(errors));
   Correct_symbl_num=(size(y,1)*size(y,2))-Error_number;
```

```
%Find the bit error location in "errors" where element in x1 and y1 differ.
%bit_Error_locations=(find(bit_errors))';
   bit_error_total=sum(sum(bit_errors));
   symcorr=floor((n-k)/2);
      if blklgth<=(n-k)
      disp('Error!!!The block length is too short for the given n and k values')
      disp('')
      elseif blklgth>(n-k)
      errtrans=errors';
%Reshape the error matrix as a vector of errors
      errvect=errtrans(:)';
      blkrem=rem(length(errvect),blklgth);
         if blkrem~=0;
         zeropad=zeros(blklgth-blkrem);
         errvectpad=[errvect zeropad(1,:)];
         elseif blkrem==0;
         errvectpad=errvect;
         end
         blknos=length(errvectpad)/blklgth;
         errcorct=[];
         errblksum=[];
         for lp=1:blknos;
         errblk=errvectpad(((blklgth*(lp-1))+1):(blklgth*lp));
         errblklgth=length(errblk);
         if sum(errblk)<=symcorr;
         noerr=zeros(errblklgth);
         errblk=noerr(1,:);
         elseif sum(errblk)>symcorr;
         errblk=errblk;
         end
         errcorct=[errcorct errblk];
         errblksum=[errblksum sum(errblk)];
      end
      newerrvect=errcorct(1:length(errvect));
      errtot=sum(newerrvect);
      RSerrs=(reshape(newerrvect,size(errors,2),size(errors,1)))';
%Find the error distribution vs. OFDM Frequencies
      freqerrs=sum(RSerrs);
      errindex=(find(RSerrs))';
      RSerrtot=sum(errblksum);
      RSerrdif=Error_number-RSerrtot;
      errperblk=[(1:blknos);errblksum];
%Check to see if x and y are the same. If not, display error message
      if x==y;
      disp('GREAT!!!there are no errors.')
      error_no=0;
```

```
        errmx=errors;
        rowerrs=sum(errors');
        else
        disp('WARNING!:Errors were detected!')
        disp('')
        if n==k
        disp('WARNING!: Since n=k,Conv. coding has no error correcting possible')
        disp('')
        end
        disp(['For the given input parameters:n=',int2str(n),'and k=',int2str(k),',the convolu-
tional code is capable'])
        disp(['of correcting ',int2str(symcorr),'errors.'])
        disp('')
%Check to see if xbit and ybit are the same. If not, display error message
        if xbit==ybit;
        disp('GREAT!!!there are no bit errors.')
        bit_error_no=0;
        bit_errmx=bit_errors;
        bit_rowerrs=sum(bit_errors');
        else
        %disp('WARNING!:Errors were detected!')
        disp('')
        if n==k
        disp('WARNING!: Since n=k,Conv. coding has no error correcting possible')
        disp('')
        end
    end
%CONV code was able to correct all errors
    if errtot==0
    Pre_RS_error_matrix=errors;
    disp('EXCELLENT: Convolutional code corrected all detected errors!')
    disp(['Originally the error total was:',int2str(Error_number)])
    disp('')
    error_no=0;
    [rx cx]=size(x);
    errmx=zeros(rx,cx);
    rowerrs=sum(errmx');
%CONV code was able to correct some errors but not all of them
    elseif errtot<Error_number
    Pre_RS_error_matrix=errors;
    Post_RS_error_matrix=RSerrs;
    errmx=RSerrs;
    rowerrs=sum(errmx');
    disp('OOOPS: Convolutional code corrected some detected errors, but not all.')
    disp(['Originally the error total was : ',int2str(Error_number)])
    disp('')
```

```
   disp(['After Convolutioanl decoding , the error number was reduced
to:',int2str(RSerrtot)])
   disp('')
   error_no=RSerrtot;
   disp(['The total number of correct symbols are:',int2str((size(y,1)*size(y,2))-RSerrtot)])
   disp('')
   %disp('The error number distribution per block number is :')
   %disp(errperblk)
%figure(pic+3)
%bar((1:blknos),errblksum)
%axis([0.5(blknos+0.5)0(max(errblksum)+1)])
%title(['Simulation#',int2str(pic),':Error Distribution Per Message Block (Error
count=',int2str(error_no),')'])
%xlabel(['Message Block Number(block size:',int2str(blklgth),'symbols)'])
%conv code did not correct any errors
   elseif errtot==Error_number
   Error_matrix=errors;
   errmx=errors;
   rowerrs=sum(errors');
   disp('OOOPS!:The convolutional code did not correct any errors.')
   disp('Perhaps a more powerful code is required.')
   disp('')
   disp(['The total number of error occurrences is:',int2str(Error_number)])
   disp('')
   error_no=errtot;
   %disp('The error number distribution per block number is :')
   %disp(errperblk)
%figure(pic+4)
%bar((1:blknos),errblksum)
%axis([0.5 (blknos+.5) 0 (max(errblksum)+1)])
%title(['Simulation#',int2str(pic),':Error Distribution Per Message Block. (Error
count=',int2str(error_no),')'])
%xlabel(['Message Block Number (block size:',int2str(blklgth)',symbols)'])
%
   end
end
end
end
end
disp('____________________________________________');
elseif ecc==2;
   k=1;
   for i=1:r_rm;
   k=k+prod(1:m_rm)/prod(1:m_rm-i)/prod(1:i);
   end
   disp(['The rate of this code is ',int2str(k),' / ',int2str(2^m_rm),' =
',num2str(k/(2^m_rm))])
y=y(1,[1:size(x,2)]);
ybit=ybit(1,[1:size(xbit,2)]);
%Compare inputs x and y and generate error matrix, "errors"
   errors=(x~=y);
%Compare inputs xbit and ybit and generate BIT error matrix, "bit_errors"
   bit_errors=(xbit~=ybit);
%Find the error distribution vs. OFDM frequencies
   freqerrs=sum(errors);
   Error_number=sum(sum(errors));
   bit_error_total=sum(sum(bit_errors));
   e=2^(m_rm-r_rm-1)-1;
   %Reed-Muller symbol correction for k symbols
   if m_rm-r_rm<=1
     disp('WARNING! This code does NOT have error correction capability!!!')
  elseif m_rm-r_rm>1
     disp(['This code can correct up to ',int2str(e),' error(s) in a blocklength of
',int2str(2^m_rm),' bits'])
  end
  errtrans=errors';
%Reshape the error matrix as a vector of errors
      errvect=errtrans(:)';
      blkrem=rem(length(errvect),k);
         if blkrem~=0;
         zeropad=zeros(k-blkrem);
         errvectpad=[errvect zeropad(1,:)];
         elseif blkrem==0;
         errvectpad=errvect;
         end
      blknos=length(errvectpad)/k;
      errcorct=[];
      errblksum=[];
      for lp=1:blknos;
         errblk=errvectpad(((k*(lp-1))+1):(k*lp));
         errblklgth=length(errblk);
         if sum(errblk)<=e;
         noerr=zeros(errblklgth);
         errblk=noerr(1,:);
         elseif sum(errblk)>e;
         errblk=errblk;
         end
         errcorct=[errcorct errblk];
         errblksum=[errblksum sum(errblk)];
      end
      newerrvect=errcorct(1:length(errvect));
```

```
        errtot=sum(newerrvect);
        RMerrs=(reshape(newerrvect,size(errors,2),size(errors,1)))';
%Find the error distribution vs. OFDM Frequencies
        freqerrs=sum(RMerrs);
        errindex=(find(RMerrs))';
        RMerrtot=sum(errblksum);
        RMerrdif=Error_number-RMerrtot;
        errperblk=[(1:blknos);errblksum];
%Check to see if x and y are the same. If not, display error message
        if x==y;
        disp('GREAT!!!there are no errors.')
        error_no=0;
        errmx=errors;
        rowerrs=sum(errors');
        else
        disp('WARNING!:Errors were detected!')
        disp('')
%Check to see if xbit and ybit are the same. If not, display error message
        if xbit==ybit;
        disp('GREAT!!!there are no bit errors.')
        bit_error_no=0;
        bit_errmx=bit_errors;
        bit_rowerrs=sum(bit_errors');
        else
        %disp('WARNING!:Errors were detected!')
        disp('')
        %RM code was able to correct all errors
    %if Error_number==0
    if errtot==0
    disp('EXCELLENT: The Reed-Muller code corrected all detected errors!')
    disp(['Originally the error total was:',int2str(Error_number)])
    disp('')
    error_no=0;
    [rx cx]=size(x);
    errmx=zeros(rx,cx);
    rowerrs=sum(errmx');
    %RM code was able to correct some errors but not all of them
    elseif errtot<Error_number
    Pre_RM_error_matrix=errors;
    Post_RM_error_matrix=RMerrs;
    errmx=RMerrs;
    rowerrs=sum(errmx');
    disp('The Reed-Muller code corrected some detected errors, but not all.')
    disp(['Originally the error total was : ',int2str(Error_number)])
    disp('')
    disp(['After R-M decoding , the error number was reduced to:',int2str(RMerrtot)])
```

```
   disp('')
   error_no=RMerrtot;
   disp(['The total number of correct symbols are:',int2str((size(y,1)*size(y,2))-
RMerrtot)])
   disp('')
   %disp('The error number distribution per block number is :')
   %disp(errperblk); %degisme degisme ; koyulacak
%figure(pic+3)
%bar((1:blknos),errblksum)
%axis([0.5(blknos+0.5)0(max(errblksum)+1)])
%title(['Simulation#',int2str(pic),':Error Distribution Per Message Block (Error
count=',int2str(error_no),')'])
%xlabel(['Message Block Number(block size:',int2str(blklgth),'symbols)'])
%RM code did not correct any errors
   elseif errtot==Error_number
   Error_matrix=errors;
   errmx=errors;
   rowerrs=sum(errors');
   disp('The Reed-Muller code did not correct any errors.')
   disp('Perhaps a more powerful R-M code is required.')
   disp('')
   disp(['The total number of error occurrences is:',int2str(Error_number)])
   disp('')
   error_no=errtot;
   %disp('The error number distribution per block number is :')
   %disp(errperblk)
%figure(pic+4)
%bar((1:blknos),errblksum)
%axis([0.5 (blknos+.5) 0 (max(errblksum)+1)])
%title(['Simulation#',int2str(pic),':Error Distribution Per Message Block. (Error
count=',int2str(error_no),')'])
%xlabel(['Message Block Number (block size:',int2str(blklgth)',symbols)'])
disp('_____________________________________________');
   end
end
end
end
%-----------------------------------------------------------------------------------
```

CHUHF

```
%CHUHF

% TITLE            : UHF CHANNEL MODEL (MULTIPATH CHANNEL)

%--------------------------------------------------------------------------------
% Thesis Advisor      : Prof M. Tummala, Naval Postgraduate School
% Author              : Prof. Paul H. Moose, Naval Postgraduate School
% Modified by         : Tan Kok Chye, Naval Postgraduate School
%--------------------------------------------------------------------------------
function y=chuhf(s,x,loss,dly,dop,N,freqspace)
c=10.^(-loss./20);
deltat=1/(N*freqspace);
d=(dly.*.000001)./deltat;
e=dop./freqspace;
[L,Nt]=size(x);
D=length(d);
x=x.';
x=x(:).';
%D path with delays from d. (Uses macro dline.m)
xd=dline(x,d);
[rr,cc]=size(xd);
x=xd(1,:);
% Offsets direct path by .7 of max doppler freq. (uses macro ofst.m)
xo=ofst(.7*e(1),N,x);
% First path with no fading. (uses macro ray_dop.m)
for l=1:D
a=ray_dop(s,cc,N,e(1));
xd(1,:)=a.*xd(1,:);
end
%Sums the fading paths
y=c*xd;
%Adds in the First path without fading for the GSM-Ricean.
%
y=y+xo;
y=y(1:L*Nt);
y=reshape(y,Nt,L).';
%--------------------------------------------------------------------------------
```

CMV2FA

```
%CMV2FA

% TITLE            : COMPLEX FREQUENCY ARRAY GENERATOR

%----------------------------------------------------------------------------------
% Thesis Advisor      : Prof M. Tummala, Naval Postgraduate School
% Author              : Dave Roderick, Naval Postgraduate School
% Modified By         : Tan Kok Chye, Naval Postgraduate School
%-----------------------------------------------------------------------------------
function X=cmv2fa(N,M)
[m n]=size(M);
if rem(n,2)==0;
M=M;
else
M=[zeros(m,1) M];
end
[m n]=size(M);
K=round(n/2);
%Generate a matrix of zeros with m row and N columns.
X=zeros(m,N);
X(:,1:K)=M(:,K+1:2*K);
X(:,N-K+1:N)=M(:,1:K);
%------------------------------------------------------------------------------------
```

CNV_ENCD

```
%CNV_ENCD

% TITLE            : CONVOLUTIONAL ENCODING

%--------------------------------------------------------------------------------
% Thesis Advisor      : Prof M. Tummala, Naval Postgraduate School
% Author              : Tan Kok Chye, Naval Postgraduate School
%--------------------------------------------------------------------------------
function ce_output=cnv_encd(ce_g,ce_k0,ce_input)
% cnv_encd(ce_g,ce_k0,ce_input)
% determines the output sequence of a binary convolutional encoder
% ce_g is the generator matrix of the convolutional code
% with ce_n0 rows and ce_l*ce_k0 columns. Its rows are ce_g1,ce_g2,....ce_gn.
% ce_k0 is the number of bits entering the encoder at each clock cycle.
% check to see if extra zero padding is necessary
if rem(length(ce_input),ce_k0)>0
ce_input=[ce_input,zeros(size(1:ce_k0-rem(length(ce_input),ce_k0)))];
end
ce_n=length(ce_input)/ce_k0;
%check the size of matrix ce_g
if rem(size(ce_g,2),ce_k0)>0
error('Error, ce_g is not of the right size.')
end
% determine ce_l and ce_n0
ce_l=size(ce_g,2)/ce_k0;
%disp(['The value of ce_l is:',int2str(ce_l)]);
ce_n0=size(ce_g,1);
%disp(' ')
%disp(['The value of ce_n0 is:',int2str(ce_n0)]);
%add extra zeros
ce_u=[zeros(size(1:(ce_l-1)*ce_k0)),ce_input,zeros(size(1:(ce_l-1)*ce_k0))];
%generate ce_uu, a matrix whose column are the contents of
%conv. encoder at various cycles.
ce_u1=ce_u(ce_l*ce_k0:-1:1);
for ce_i=1:ce_n+ce_l-2
ce_u1=[ce_u1,ce_u((ce_i+ce_l)*ce_k0:-1:ce_i*ce_k0+1)];
end
ce_uu=reshape(ce_u1,ce_l*ce_k0,ce_n+ce_l-1);
%determine the ce_output
ce_output=reshape(rem(ce_g*ce_uu,2),1,ce_n0*(ce_l+ce_n-1));
%--------------------------------------------------------------------------------
```

COFDMSIM

```
%COFDMSIM

% TITLE          : SIMULATION OF COFDM

%--------------------------------------------------------------------------------
% Thesis Advisor       : Prof M. Tummala, Naval Postgraduate School
% Author               : Dave Roderick, Naval Postgraduate School
% Modified By          : TEZEREN,Serdar Umit,LTJG,TURKISH NAVY
%--------------------------------------------------------------------------------
clc,close all;clear all;
disp('________________________________________________________________');
disp('This batch m-file runs COFDM simulations using different channel models.')
fort=input('To run the frequency version, enter 1(one), To run the time version, enter
0(zero), or to run both enter 2(two):');
freqno=input('Enter the # of OFDM frequencies (note : must be even):');
%N=input('Enter the number of FFT points (Note : This number must be larger than # of
OFDM frequencies):');
N=64;
chnmdl=input('Do you want to run channel model 0, 1, 2, 3 or 4? (Enter 0,1,2,3 or 4):');
if chnmdl==0
disp('Code check simulation performed.');
sigs=0;
loss=0;
dop=0;
dly=0;
elseif chnmdl==1
disp('AWGN Channel simulation performed.');
sigs=input('Enter the sigma noise parameter range or single value. (Ex lin-
space(0,0.02,20)or .003):');
loss=0;
dop=0;
dly=0;
elseif chnmdl==2
disp('Multipath Channel simulation performed.');
sigs=0;
pthno=input('Do you want to run link1 ,link2, link3 or a custom link ? (Enter 1,2,3 or 4
for custom):');
%
%Link parameters
%
   if pthno==3
   %my link 3
```

```
loss=[0,2.17,4.34,6.51,8.69,10.86,13.03,15.20,17.37,19.54,21.71,23.89,26.06,28.23,30.4,
32.57,34.74,36.92];
   dop=[15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15];

dly=[0,0.05,0.10,0.15,0.20,0.25,0.30,0.35,0.40,0.45,0.50,0.55,0.60,0.65,0.70,0.75,0.80,0.
85];
   elseif pthno==2
   %my link 2

loss=[0,2.17,4.34,6.51,8.69,10.86,13.03,15.20,17.37,19.54,21.71,23.89,26.06,28.23,30.4,
32.57,34.74,36.92];
   dop=[10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10];

dly=[0,0.05,0.10,0.15,0.20,0.25,0.30,0.35,0.40,0.45,0.50,0.55,0.60,0.65,0.70,0.75,0.80,0.
85];
   elseif pthno==1
   %my link 1

loss=[0,2.17,4.34,6.51,8.69,10.86,13.03,15.20,17.37,19.54,21.71,23.89,26.06,28.23,30.4,
32.57,34.74,36.92];
   dop=[5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5];

dly=[0,0.05,0.10,0.15,0.20,0.25,0.30,0.35,0.40,0.45,0.50,0.55,0.60,0.65,0.70,0.75,0.80,0.
85];
   elseif pthno==4
   disp('Custom link simulation...')
   loss=input('Enter the path loss in db (Ex
[0,2.17,4.34,6.51,8.69,10.86,13.03,15.20,17.37,19.54,21.71,23.89,26.06,28.23,30.4,32.57
,34.74,36.92]):');
   dop=input('Enter the doppler frequency in Hertz (Ex
[5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5]):');
   dly=input('Enter the time delays of the multipaths in microsecs (Ex
[0,0.05,0.10,0.15,0.20,0.25,0.30,0.35,0.40,0.45,0.50,0.55,0.60,0.65,0.70,0.75,0.80,0.85]):'
);
   end
elseif chnmdl==3
disp('AWGN+Multipath Channel simulation performed');
sigs=input('Enter the sigma noise parameter range or single value. (Ex lin-
space(0,0.02,20) or .003):');
pthno=input('Do you want to run link1, link2, link3 or a custom link ?(Enter 1,2,3 or 4
for custom):');
%
%Link parameters
%
   if pthno==3
   %my link 3

loss=[0,2.17,4.34,6.51,8.69,10.86,13.03,15.20,17.37,19.54,21.71,23.89,26.06,28.23,30.4,
32.57,34.74,36.92];
   dop=[15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15];

dly=[0,0.05,0.10,0.15,0.20,0.25,0.30,0.35,0.40,0.45,0.50,0.55,0.60,0.65,0.70,0.75,0.80,0.
85];
   elseif pthno==2
   %my link 2

loss=[0,2.17,4.34,6.51,8.69,10.86,13.03,15.20,17.37,19.54,21.71,23.89,26.06,28.23,30.4,
32.57,34.74,36.92];
   dop=[10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10];

dly=[0,0.05,0.10,0.15,0.20,0.25,0.30,0.35,0.40,0.45,0.50,0.55,0.60,0.65,0.70,0.75,0.80,0.
85];
   elseif pthno==1
   %my link 1

loss=[0,2.17,4.34,6.51,8.69,10.86,13.03,15.20,17.37,19.54,21.71,23.89,26.06,28.23,30.4,
32.57,34.74,36.92];
   dop=[5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5];

dly=[0,0.05,0.10,0.15,0.20,0.25,0.30,0.35,0.40,0.45,0.50,0.55,0.60,0.65,0.70,0.75,0.80,0.
85];
   elseif pthno==4
   disp('Custom link simulation...')
   loss=input('Enter the path loss in db (Ex
[0,2.17,4.34,6.51,8.69,10.86,13.03,15.20,17.37,19.54,21.71,23.89,26.06,28.23,30.4,32.57
,34.74,36.92]):');
   dop=input('Enter the doppler frequency in Hertz (Ex
[5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5]):');
   dly=input('Enter the time delays of the multipaths in microsecs (Ex
[0,0.05,0.10,0.15,0.20,0.25,0.30,0.35,0.40,0.45,0.50,0.55,0.60,0.65,0.70,0.75,0.80,0.85]):'
);
   end
elseif chnmdl==4;
   sigs=input('Enter the sigma noise parameter range or single value. (Ex lin-
space(0,0.02,20)or .003):');
   disp('Mobile channel simulation')
   %sigs=0;
   loss=0;
   dop=0;
   dly=0;
```

```
end
%%%%allcase=input('Simulate all interleaver cases (yes) or specific ones(no)?
(1=yes,0=no):');
allcase=0;
if allcase==1
disp('All cases,(0-8),will be tested.');
cases=[0:8];
elseif allcase==0
%%%%cases=input('Enter specific case numbers from (0 to 8)(Ex [0 4 5 8]):');
cases=0;
end
if fort~=2
   if length(cases)~=1
      casey_n=input('Do you want to find optimal interleaver case(s) ? (1=yes, 0=no):');
   else
      casey_n=0;
   end
end
totsym=input('Enter the total minimum number of symbols to simulate (Ex 10000):');
rowno=ceil(totsym/freqno);
if totsym~=(rowno*freqno)
disp(['Note:Based on the parameters thus far, the actual total number of symbol to be
simulated will be :',int2str(rowno*freqno)]);
end
%pry_n=input('For the interleaver, do you want to calculate all possible intermediate ma-
trix dimension pairs?(1=yes,0=no):');
pry_n=1;
pair1=1;
pair2=rowno*freqno;
if pry_n==1
Intrlvr_pairs=intlvprs(rowno,freqno);
intlvrprs=Intrlvr_pairs;
%ee=round(size(intlvrprs,1)/3);
%int(:,1)=intlvrprs([1:ee],1);
%int(:,2)=intlvrprs([1:ee],2);
%int(:,3)=intlvrprs([ee+1:2*ee],1);
%int(:,4)=intlvrprs([ee+1:2*ee],2);
%int(:,5)=intlvrprs([2*ee+1:end],1);
%int(:,6)=intlvrprs([2*ee+1:end],2);
disp('')
disp('For these input parameters, all possible acceptable interleaver dimension pairs are: ')
disp(Intrlvr_pairs)
%disp(int)
end
pairs=input(['Desired interleaver pair? (Ex [row # col #] = [20 50] ):']);
rintlv=pairs(1);
```

```
cintlv=pairs(2);
%mary=input('Enter the number of M-ary bits, q (i.e. for 256-ary, q=8):');
mary=1;
%nary=input('Enter the number of N-ary bits,q(i.e. for 16-ary, q=4):');
nary=2;
freqspace=round(16600000/freqno);
%Ng=input('Enter the guard interval length (Number of sample points):');
Ng=16;
ecc=input('Enter the type of error correction coding ? (0=No FEC code, 1=Convolutional
Code, 2=Reed-Muller Code):');
if ecc==1
code=input('Enter n, k and error correction block length (Ex: [240 200 240]):');
n=code(1);
k=code(2);
blklgth=code(3);
r_rm=0;m_rm=0;
elseif ecc==2;
pair_rm=input('Enter the r,m pair for the Reed-Muller code (Ex: [2 4]):');
r_rm=pair_rm(1,1);m_rm=pair_rm(1,2);
if r_rm>m_rm
display('ERROR!!! m must be equal to or larger than r, and both must be integer larger
than zero!')
pair_rm=input('Enter the r,m pair for the Reed-Muller code (Ex: [2 4]):');
r_rm=pair_rm(1,1);m_rm=pair_rm(1,2);
end
n=0;k=0;blklgth=0;
elseif ecc==0
n=freqno;
k=freqno;
blklgth=freqno;
r_rm=0;m_rm=0;
end

%svals=input('Enter specific seed values, or 0 for a random seed (ex [103 22, 60] or
[0]):');
svals=33;
%picy_n=input('Do you want signal plots? (1=yes, 0=no):');
picy_n=1;
if picy_n==1
%wait=input('How many seconds of delay between pictures?');
wait=3;
wait=round(wait);
elseif picy_n==0
wait=0;
end
%prnty_n=input('Do you want print outs? (1=yes, 0=no):');
```

```
prnty_n=0;
pic=0;
svect=[];
for run=1:length(svals);
   errvect=[];
   errcase=[];
   errtot=[];
   if min(svals)==0
      rand('seed',sum(100*clock));
      s=round(abs(rand(1)*pi*10*(pic+1)*run));
   elseif min(svals)~=0
   s=svals(run);
   end
   svect=[svect,s];
   for l=1:length(cases);
   disp('___________________________________________________________')
   disp(['Run#:',int2str(run)]);
   if chnmdl==0
      disp(['Channel Model 0 : Code Check Simulation Performed'])
   elseif chnmdl==1
      disp(['Channel Model 1 : AWGN Channel with  Performed'])
   elseif chnmdl==2
      disp(['Channel Model 2 Link ',int2str(pthno),' : Multipath Channel Performed'])
   elseif chnmdl==3
      disp(['Channel Model 3 Link ',int2str(pthno),' : AWGN + Multipath Channel Per-
formed'])
   elseif chnmdl==4
      disp(['Channel Model 4 : Severe Mobile Outdoor Channel Performed']);
   end
   %disp(['Seed=',int2str(s)]);
   disp(['Interleaver case=',int2str(cases(l))]);
   disp(['Interleaver Pair=[',int2str(pairs),']']);
   disp(['The # of OFDM subcarriers=',int2str(freqno)]);
   if ecc==0;
      disp(['FEC code is not used for this run']);
   elseif ecc==1
      disp(['n=',int2str(n),'  k=',int2str(k),'  convolutional code is used for this run']);
   elseif ecc==2
   disp(['R(',int2str(r_rm),',',int2str(m_rm),') Reed-Muller code is used for this run']);
end
      if fort<=1
[errmax,errors,freqerrs,papr]=chancdl(chnmdl,wait,prnty_n,picy_n,pic,cases(l),s,freqno,ri
ntlv,cintlv,N,mary,nary,n,k,blklgth,Ng,sigs,loss,dly,dop,freqspace,fort,ecc,r_rm,m_rm);
      elseif fort==2
      disp('Frequency differential encoding/decoding simulation...')
      disp('___________________________________________________________')
```

```
[errmax,errors,freqerrs,papr]=chancdl(chnmdl,wait,prnty_n,picy_n,pic,cases(l),s,freqno,ri
ntlv,cintlv,N,mary,nary,n,k,blklgth,Ng,sigs,loss,dly,dop,freqspace,1,ecc,r_rm,m_rm);
        disp('*********************************************************')
        disp('Time differential encoding/decoding simulation....')

disp('______________________________________________________________________')
[errmax,errors,freqerrs,papr]=chancdl(chnmdl,wait,prnty_n,picy_n,pic+12,cases(l),s,freq
no,rintlv,cintlv,N,mary,nary,n,k,blklgth,Ng,sigs,loss,dly,dop,freqspace,0,ecc,r_rm,m_rm)
;
        end
        errtot=[errtot sum(errors)];
        errvect=[errvect errtot];
        errcase=[errcase sum(errmax)];
    end
    if fort~=2
    casearry=[cases;errcase];
        if casey_n==1
        figure(pic+14)
        bar(cases,errcase)
        grid
        orient tall
            if fort==1
            title([int2str(pic),':Maximum Error Total Vs. Interleaver CASE Number
(Freq.Diff. Enc.) (OFDM Freq.#=',int2str(freqno),')'])
            elseif fort==0
             title([int2str(pic),':Maximum Error Total Vs. Interleaver CASE Number (Time
Diff. Enc.) (OFDM Freq.#=',int2str(freqno),')'])
            end
            xlabel(['CDL Interleaver CASE Number'])
            ylabel(['Maximum Error Count For Any Symbol Row (Seed=',int2str(s),')'])
            axis([-.5 8.5 0 (max(errcase)+1)])
            if prnty_n==1;
              print
              pause(10)
            end
            pause(wait);
            figure(pic+15)
            bar(cases,errtot)
            grid
            orient tall
            title([int2str(pic),':Error Totals Vs. Interleaver CASE Number'])
            xlabel(['CDL Interleaver CASE Number'])
            ylabel(['Sigma:(',num2str(min(sigs)),'-',num2str(max(sigs)),') Error Total'])
            axis([-.5 8.5 (min(errtot)-1) (max(errtot)+1)])
            if prnty_n==1;
            print
            pause(10)
            end
            pause(wait);
        end
     pic=pic+1;
    end
end
disp('**************************************************************')
disp('')
disp('Channel model batch run is finished!')
%Seed=svect
%-----------------------------------------------------------------------------------
```

CVDD

```
%CVDD

% TITLE          : CONTINUOUS VARIABLE DIGITAL DELAY ELEMENT

%--------------------------------------------------------------------------------
% Thesis Advisor       : Prof M. Tummala, Naval Postgraduate School
% Author               : Tan Kok Chye, Naval Postgraduate School
%---------------------------------------------------------------------------------
function [y]=cvdd(x,alpha)
if ((nargin~=2)|(nargout~=1))
error('ERROR:usage:y=y=cvdd(x,alpha);');
return;
end
if (size(x)~=size(alpha))
error('ERROR:x and alpha must be the same size');
return;
end
if (abs(alpha)>0.5)
error('ERROR:alpha must be within -0.5 and 0.5');
return;
end
%----------------------------------------------------
% Initialization
%----------------------------------------------------
% Initialize FIR filter coefficients are in [1] (0,0.328 pass band)
C0=[-0.013824 0.054062 -0.157959 0.616394 0.616394 -0.157959 0.054062 -0.013824];
C1=[0.003143 -0.019287 0.1008 -1.226364 1.226364 -0.1008 0.019287 -0.003143];
C2=[0.055298 -0.216248 0.631836 -0.465576 -0.465576 0.631836 -0.216248 0.055298];
C3=[-0.012573 0.077148 -0.403198 0.905457 -0.905457 0.403198 -0.077148 0.012573];
%------------------------------------------------------
% 4 parallel FIR and add together based on [1]
%------------------------------------------------------
y0=filter(C0,[1],x);
y1=filter(C1,[1],x);
y2=filter(C2,[1],x);
y3=filter(C3,[1],x);
%
y=alpha.*y3;
y=alpha.*(y+y2);
y=alpha.*(y+y1);
y=y+y0;
%---------------------------------------------------------------------------------
```

DECDRCDL

```
%DECDRCDL

% Title : COFDM DECODER

%-----------------------------------------------------------------------------------
% Thesis Advisor      : Prof M. Tummala, Naval Postgraduate School
% Author              : Dave Roderick, Naval Postgraduate School
% Modified By         : TEZEREN,Serdar Umit,LTJG,TURKISH NAVY
%-----------------------------------------------------------------------------------
func-
tion[outmsg,decoder_output_bit,random_msg,random_bit,M,MM,binary_value]=decdrcd
l(picy_n,pic,dcase,K,Fa,nsymno,freqno,rdintlv,cdintlv,mary,nary,fort,B_random,ecc,r_rm
,m_rm)
%to generate BER vs Eb/No.
%to provide bit outputs
M=fa2cma(K,Fa);
Cmplx_mod_vals=M;
%
naryp=nary;
[s,MM]=dfdcdrft(naryp,nary,M,fort);
[L,cc]=size(s);
strans=s';
svect=strans(:).';
corrs=svect(1:nsymno);
%
nsymno;
Br=bm(mary,mb(nary,corrs));
lengthBr=length(Br);
rmndr=rem(length(Br),freqno);
if rmndr==0;
Br=Br;
elseif rmndr~=0;
Br=Br(1:(lengthBr-rmndr));
end
rcvd=(reshape(Br,freqno,length(Br)/freqno))';
Rcvd_Intlv_Ary=rcvd;
%
[Br Bc]=size(rcvd);
SYNC=[];
sr=rcvd';
si=sr(:)';
sd=cdldlv(rdintlv,cdintlv,dcase,si,SYNC);
binary_value=mb(mary,sd);
if ecc~=2;
```

```
viter_G=[1 0 1 1 0 1 1;1 1 1 1 0 0 1];
viter_k=1;
[viterbi_output,survivor_sta,cumul_metrix]=viterbi(viter_G,viter_k,binary_value);
decoder_output=viterbi_output;
%mary_dec=bm(mary,decoder_output);
%decoder_output_bit=decoder_output;
%outmsg=reshape(sd,Bc,Br)';
%
%random_bit=B_random;
%random_msg=bm(mary,random_bit);
%[Brow Bcol]=size(random_msg);
%
%outmsg=reshape(mary_dec,Bcol,Brow)';bu zaten yuzdeliydi kaldirma
%Sink_Msg=outmsg;
elseif ecc==2;
[rmdecoder_output]=rmdecoder(binary_value,r_rm,m_rm);
decoder_output=rmdecoder_output;
end
mary_dec=bm(mary,decoder_output);
decoder_output_bit=decoder_output;
%outmsg=reshape(sd,Bc,Br)';  bu zaten yuzdeliydi kaldirma
%
random_bit=B_random;
random_msg=bm(mary,random_bit);
[Brow Bcol]=size(random_msg);
%
outmsg=reshape(mary_dec,size(mary_dec,1)*size(mary_dec,2),1)';
Sink_Msg=outmsg;
%-----------------------------------------------------------------------------------
```

DECI2BIN

```
%DECI2BIN

% TITLE          :  DECIMAL TO BINARY CONVERSION

%--------------------------------------------------------------------------------
% Thesis Advisor       : Prof M. Tummala, Naval Postgraduate School
% Author                   : Tan Kok Chye, Naval Postgraduate School
%--------------------------------------------------------------------------------
function y=deci2bin(x,l)
y=zeros(1,l);
vi=1;
while x>=0 & vi<=l
y(vi)=rem(x,2);
x=(x-y(vi))/2;
vi=vi+1;
end
y=y(l:-1:1);
%--------------------------------------------------------------------------------
```

DFDCDRFT

```
%DFDCDRFT

% TITLE          : COMPLEX NUMBER DEMODULATOR & FREQUENCY / TIME
DIFFERENTIAL DECODING

%-----------------------------------------------------------------------------------
% Thesis Advisor       : Prof M. Tummala, Naval Postgraduate School
% Author               : Prof. Paul H. Moose, Naval Postgraduate School
% Modified by          : Tan Kok Chye, Naval Postgraduate School
%-----------------------------------------------------------------------------------
function [s,M]=dfdcdrft(qp,q,MD,fort)
if fort==0 %Time Differential decoding
MD=MD';
[m n]=size(MD);
% Perform a looping routine to find the phase differences between adjacent values in the
% array,MD,and put these calculated values into array,M.
for l=1:m
for j=1:n-1
M(l,j)=MD(l,j+1)*conj(MD(l,j));
end
end
%Transpose the array back to its original form
M=M';
% Calculate the number of M-ary symbols based upon the exponent qp,then use this
number
% to find the number of equally spaced phases in a unit circle.
N=2^qp;
dph=2*pi/N;
% Divide the phase arguments of elements in M, by the equal phases generated by dph.
phn=angle(M)./dph;
% Calculate the phase sector number by finding the remainders.
s=rem(round(phn)+N,N);
elseif fort==1 % Frequency Differential decoding
% Transpose the modulation array, and find the dimensions
[m,n]=size(MD);
MD=MD(:,2:n);
[m n]=size(MD);
% Perform a looping routine to find the phase differences between
% adjacent values in the array, MD, and put these calculated values into array,M.
for l=1:m
for j=1:n-1
M(l,j)=MD(l,j+1)*conj(MD(l,j));
end
end
```

```
N=2^qp;
dph=2*pi/N;

% Calculate the phase sector number by finding the remainders.
phn=angle(M)./dph;
s=rem(round(phn)+N,N);
end
%------------------------------------------------------------------------------------
```

DIFCDRFT

```
%DIFCDRFT

% TITLE         : COMPLEX NUMBER MODULATOR & FREQUENCY / TIME
DIFFERENTIAL ENCODING

%---------------------------------------------------------------------------------
% Thesis Advisor      : Prof M. Tummala, Naval Postgraduate School
% Author              : Prof. Paul H. Moose, Naval Postgraduate School
% Modified by         : Tan Kok Chye, Naval Postgraduate School
%---------------------------------------------------------------------------------

function MD=difcdrft(q,m,fort)
if fort==0 %Time differential encoding
% M-ary alphabet size
N=2^q;
% Determine the number of equal phases based upon the m-ary symbol length
dph=2*pi/N;
% Find the size of the input symbol matrix (# of row & # of columns)
[rr n]=size(m);
% Perform the time differential encoding of phase values by cumulative summing matrix,
% m, down one column at a time across the entire matrix. This function generates a ma-
trix.
for k=1:n
md=cumsum(m(:,k));
% Generate the complex numbers with correspondiing phase values.
MD(:,k)=exp(i*dph.*md);
end
% Inject the reference row of ones (zero phase) at top of output matrix for
% differential encoding synchronization
MD=[ones(1,n); MD];
elseif fort==1 % Frequency Differential encoding
% M-ary alphabet size
N=2^q;dph=2*pi/N;
% Find the size of the input symbol matrix (# of row & # of columns)
[rr n]=size(m);
md=cumsum(m');
md=md';
% Generate the complex numbers with corresponding phase values.
MD=exp(i*dph.*md);
% Inject the reference row of ones (zero phase) at top of output matrix for
% differential encoding synchronization.
MD=[ones(rr,2) MD];
end
%---------------------------------------------------------------------------------
```

DLINE

```
%DLINE

% TITLE          : UHF CHANNEL DELAY LINE GENERATOR

%---------------------------------------------------------------------------------
% Thesis Advisor       : Prof M. Tummala, Naval Postgraduate School
% Author               : Prof. Paul H. Moose, Naval Postgraduate School
% Modified by          : Tan Kok Chye, Naval Postgraduate School
%----------------------------------------------------------------------------------
function xd=dline(x,d)
x=x.';
dmax=max(d);
dmin=min(d);
nmin=floor(dmin);
nmax=ceil(dmax);
x=[x;zeros(nmax+3,1)];
N=length(x);
Nd=length(d);
%
for n=1:Nd;
di=d(n);
D=floor(di);
deld=di-D;
xd(:,n)=cvdd(x,deld-.5);
xd(:,n)=[zeros(D,1);xd(1:N-D,n)];
end
xd=xd.';
[rr,cc]=size(xd);
xd=xd(:,4+nmin:cc);
%-----------------------------------------------------------------------------------
```

DOT_PRO

```
%DOT_PRO

% TITLE          : DOT PRODUCT OPERATOR

%--------------------------------------------------------------------------------
% Thesis Advisor       : Prof M. Tummala, Naval Postgraduate School
% Author                    : TEZEREN,Serdar Umit, LTJG, TURKISH NAVY
%---------------------------------------------------------------------------------
function z=dot_pro(X,S)
a=size(X);
z=0;
for i=1:a(1,2);
   z=xor(z,(X(i)*S(i)));
end
%----------------------------------------------------------------------------------
```

FA2CMA

```
%FA2CMA

% TITLE          : FREQUENCY ARRAY TO COMPLEX MODULATION ARRAY
CONVERTER

%--------------------------------------------------------------------------------
% Thesis Advisor : Prof M. Tummala, Naval Postgraduate School
% Author         : Prof. Paul H. Moose, Naval Postgraduate School
% Modified by    : Tan Kok Chye, Naval Postgraduate School
%--------------------------------------------------------------------------------
function Mm=fa2cma(K,X)
[m n]=size(X);
Mm(:,1:K)=X(:,n-K+1:n);
Mm(:,K+1:2*K)=X(:,1:K);
Cmplx_mod_vals=Mm;
%--------------------------------------------------------------------------------
```

GM_PART1

```
%GM_PART1

% TITLE          : RM ENCODING MATRIX FIRST PART GENERATOR

%--------------------------------------------------------------------------------
% Thesis Advisor : Prof M. Tummala, Naval Postgraduate School
% Author         : TEZEREN,Serdar Umit, LTJG, TURKISH NAVY
%--------------------------------------------------------------------------------
function [bb]=gm_part1(m);
AA=zeros(2^m,m);
B=eye(m);
for i=2:2^m;
   AA(i,:)=xor(AA(i-1,:),B(m,:));
   for j=1:m;
      if rem(i-1,2^j)==0;
        AA(i,:)=xor(AA(i,:),B(m-j,:));
      end
   end
end
AA=AA';
for i=1:size(AA,1);
   for j=1:size(AA,2);
      if AA(i,j)==0;
         AA(i,j)=1;
      elseif AA(i,j)==1;
         AA(i,j)=0;
      end
   end
end
bb=zeros(size(AA,1)+1,size(AA,2));
bb(1,:)=ones(1,size(AA,2));
bb([2:end],:)=AA;
%--------------------------------------------------------------------------------
```

INTLVPRS

```
%INTLVPRS

%TITLE        : INTERMEDIATE MATRIX INTERLEAVER DIMENSION PAIRS

%-----------------------------------------------------------------------------------
%Thesis Advisor       : Prof J. McEachen, Naval Postgraduate School
%Author               : Dave Roderick, Naval Postgraduate School
%Modified By          : Tan Kok Chye, Naval Postgraduate School
%-----------------------------------------------------------------------------------
function pairs=intlvprs(n,m)
prod=n*m;
multvect=[1];
for i=2:prod;
remdr=rem(prod,i);
if remdr==0
multvect=[multvect i];
else
multvect=multvect;
end
mult=multvect;
end
lngth=length(mult);
nbr=mult(lngth);
result=[1 nbr];
for i=2:lngth;
crntpr=[mult(i) nbr/mult(i)];
result=[result;crntpr];
end
pairs=result;
%-----------------------------------------------------------------------------------
```

INVERSE

```
%TITLE          : INVERSE OF THE BINARY ARRAY

%----------------------------------------------------------------------------------
% Thesis Advisor : Prof M. Tummala, Naval Postgraduate School
% Author         : TEZEREN,Serdar Umit, LTJG, TURKISH NAVY
%----------------------------------------------------------------------------------
function d=inverse(D)
e=size(D);
f=e(1,2);
d=xor(ones(1,f),D);
%----------------------------------------------------------------------------------
```

ITDA

```
%ITDA

%TITLE          : IFFT AND GI REMOVAL OPERATOR

%--------------------------------------------------------------------------------
% Thesis Advisor      : Prof M. Tummala, Naval Postgraduate School
% Author              : Prof. Paul H. Moose, Naval Postgraduate School
% Modified by         : Tan Kok Chye, Naval Postgraduate School
%---------------------------------------------------------------------------------
function Y=itda(Ng,y)
[L Nt]=size(y);
% Remove the guard interval for channel compensation, Ng, precursor.
y=y(:,Ng+1:Nt);
% Take the FFT of array,y
Y=fft(y.').';
%----------------------------------------------------------------------------------
```

```
%JAKES

% TITLE         : RANDOM VECTOR GENERATOR WITH JAKES SPECTRUM

%---------------------------------------------------------------------------------
% Thesis Advisor      : Prof M. Tummala, Naval Postgraduate School
% Author              : Prof R. Cristi, Naval Postgraduate School
%----------------------------------------------------------------------------------
function g=jakes(Fd,Fs,N);
% g=jakes(Fd,Fs,N)
% generate a random vector of length N with jakes spectrum
% Fd=doppler frequency
% Fs=sampling frequency (Fs>10*Fd required)
% N=vector length
% g=generated random vector
% The vector g has unit power, ie g'*g*Fs/N=1
L=round(log2(Fs/(10*Fd)));
F0=Fs/(2^L);            % sampling frequency for generating the random sequence
N0=2*ceil(N/(2^L));
Nf=512;                         % number of frequency components...
                        % ... in spectrum definition
Nh=256;                         % length of FIR filter
% FIR Filter impulse response
w=0.54-0.46*cos(2*pi*(0:Nh-1)/Nh);          % hamming window
fd=Fd/F0;       % digital doppler frequency
f=0:(1/Nf):1-(1/Nf);    % vector of digital frequencies (1/2=Nyquist Freq)
kd=floor(fd*Nf);        % index for doppler frequency fd
H0(1:kd)=sqrt(1-(f(1:kd)/fd).^2);
I=find(H0==0);  H0(I)=0.0001*ones(size(I));
H=zeros(1,Nf);
H(1:length(H0))=(1./H0);
H=H.*exp(-j*2*pi*f*Nh/2);
h0=real(ifft(H));
h=h0(1:length(w)).*w;
% Generate time varying taps at low sampling freq  Fs_ch
seed=14;
randn('state',seed);
a=randn(1,N0+Nh);
seed=seed+6;
b=randn(1,N0+Nh);
%x=randn(1,N0+Nh)+j*randn(1,N0+Nh);
x=a+j*b;

g=filter(h,1,x);
```

```
g=g(Nh+1:Nh+N0);    % steady state response
% Upsample to Fs=(2^L)*F0 in L stages
for m=1:L
F0=2*F0;
g=reshape([g;zeros(size(g))], 1,2*length(g));
omegad=2*pi*Fd/F0;
Domega=(pi/2)-omegad;
omegac=((pi/2)+omegad)/2;
M=ceil(((8*pi/Domega)-1)/2);
Nfilt=2*M+1;                                    % filter order using hamming window
nt=0:Nfilt-1;
w=0.54-0.46*cos(2*pi*nt/Nfilt);       % hamming window
hm(1:M)=sin(omegac*(nt(1:M)-M))./(pi*(nt(1:M)-M));
hm(M+1)=omegac/pi;
hm(M+2:Nfilt)=sin(omegac*(nt(M+2:Nfilt)-M))./(pi*(nt(M+2:Nfilt)-M));
hm=hm(1:Nfilt).*w;
g=filter(hm,1,g);
g=g(Nfilt:length(g));   % steady state
end
% Normalize
g=g(1:N);
K=g*g'/N;
g=g/sqrt(K);
%--------------------------------------------------------------------------------
```

MARYMSG

```
%MARYMSG

% TITLE          : M-ARY RANDOM SIGNAL GENERATOR

%-----------------------------------------------------------------------------------
% Thesis Advisor       : Prof M. Tummala, Naval Postgraduate School
% Author               : Tan Kok Chye, Naval Postgraduate School
% Modified by          : TEZEREN,Serdar Umit, LTJG, TURKISH NAVY
%-----------------------------------------------------------------------------------
function [vmary_ce,random_bit,enc_output]=marymsg(q,n,m,ecc,r_rm,m_rm)
% if reedmuller is desired
if ecc==2;
k=1;   % # of the rows=k;# of the columns=2^m
for i=1:r_rm;
   k=k+prod(1:m_rm)/prod(1:m_rm-i)/prod(1:i);
end
[random_bit]=msg(round(q*n*m*k/(2^m_rm)));
enc_output=rmencoder(r_rm,m_rm,random_bit);

%unless reedmuller is desired
elseif ecc~=2;
[random_bit]=msg(n*m*q*.5-6);
conv_g=[1 0 1 1 0 1 1;1 1 1 1 0 0 1];
conv_k0=1;
enc_output=cnv_encd(conv_g,conv_k0,random_bit);
end
decml=bm(q,enc_output);
deciml=decml(1,1:(n*m));
vmary_ce=(reshape(deciml,m,n))';
%-----------------------------------------------------------------------------------
```

MB

```
%MB

% TITLE          : M-ARY TO BINARY CONVERTER
%------------------------------------------------------------------------------------
% Thesis Advisor       : Prof J. McEachen, Naval Postgraduate School
% Author               : Prof. Paul H. Moose, Naval Postgraduate School
% Modified by          : Tan Kok Chye, Naval Postgraduate School
%------------------------------------------------------------------------------------
function [b]=mb(q,m)
row=size(m,1);
col=size(m,2);
m=reshape(m',1,(row*col));
b0=rem(m,2);
m=(m-b0)./2;
B=b0;
for j=1:q-1
bj=rem(m,2);
m=(m-bj)./2;
B=[B;bj];
end
b=B(:)';
binary=b;
%------------------------------------------------------------------------------------
```

METRIC

```
%METRIC

% TITLE          : VITERBI HARD DECISION DECODING METRIC

%--------------------------------------------------------------------------------
% Thesis Advisor       : Prof M. Tummala, Naval Postgraduate School
% Author               : Tan Kok Chye, Naval Postgraduate School
%--------------------------------------------------------------------------------
function distance=metric(v_x,v_y)
if v_x==v_y
distance=0;
else
distance=1;
end
%--------------------------------------------------------------------------------
```

MLTPL

```
%MLTPL

% TITLE          : COMMON MULTIPLES

%--------------------------------------------------------------------------------
% Thesis Advisor      : Prof M. Tummala, Naval Postgraduate School
% Author              : Tan Kok Chye, Naval Postgraduate School
%---------------------------------------------------------------------------------
function [mult]=mltpl(n,m)
max=n*m;
multvect=[1];
for i=2:max;
   remdr=rem(max,i);
if remdr==0
   multvect=[multvect i];
else
   multvect=multvect;
end
mult=multvect;
end
%----------------------------------------------------------------------------------
```

MSG

```
%MSG

% TITLE          : RANDOM BIT SEQUENCE GENERATOR

%--------------------------------------------------------------------------------
% Thesis Advisor      : Prof M. Tummala, Naval Postgraduate School
% Author              : TEZEREN,Serdar Umit, LTJG, TURKISH NAVY
%--------------------------------------------------------------------------------
function source_msg_bit=msg(k)
seed=33;
source_msg_bit=randint(1,k,[0 1],seed);
%u=zeros(1,k);%u(1,1)=1;
%--------------------------------------------------------------------------------
```

NXT_STAT

```
%NXT_STAT

% TITLE          : NEXT STATE IN VITERBI ALGORITHM

%--------------------------------------------------------------------------------
% Thesis Advisor     : Prof M. Tummala, Naval Postgraduate School
% Author             : Tan Kok Chye, Naval Postgraduate School
%--------------------------------------------------------------------------------
function [next_state,memory_contents]=nxt_stat(current_state,input,v_L,v_k)
binary_state=deci2bin(current_state,v_k*(v_L-1));
binary_input=deci2bin(input,v_k);
next_state_binary=[binary_input,binary_state(1:(v_L-2)*v_k)];
next_state=bin2deci(next_state_binary);
memory_contents=[binary_input,binary_state];
%--------------------------------------------------------------------------------
```

OFST

```
%OFST

% TITLE          : CHANNEL OFFSET

%-------------------------------------------------------------------------------
% Thesis Advisor       : Prof M. Tummala, Naval Postgraduate School
% Author               : Prof. Paul H. Moose, Naval Postgraduate School
% Modified by          : Tan Kok Chye, Naval Postgraduate School
%-------------------------------------------------------------------------------
function xo=ofst(e,N,x)
[m Nt]=size(x);
xo=x.';
x=x.';
x=x(:);
x=x.';
Nt=length(x);
l=1:Nt;
ex=x.*exp(i*(2*pi/N)*e.*l);
xo(:)=x;
xo=xo.';
%-------------------------------------------------------------------------------
```

PAPR2

```
%PAPR2

% TITLE          : PEAK-TO-AVERAGE POWER RATIO, PEAK CLIPPING AND
HANNING WINDOWING

%-------------------------------------------------------------------------------------
% Thesis Advisor       : Prof M. Tummala, Naval Postgraduate School
% Author               : TEZEREN Serdar Umit, LTJG, TURKISH NAVY
%-------------------------------------------------------------------------------------
function [ppp]=papr2(xmtifft,N,freqno,pic);
window=1; %0 FOR NO WINDOWING.....1 FOR WINDOWING...2 FOR CLIPPING
if window==1
   xmtifft=xmtifft*50;
xmtifft=xmtifft(:,[17:80]);
sig=abs(xmtifft);
figure(30);plot(sig)
han=hanning(9)'; %HANNING WINDOW
kc=0.1; %hanning coefficient
wc=1-kc*han; %window construction
%size_wc=size(wc)
ka=0.2;
wc2=1+ka*han;
rmspow=sqrt(64);
cr=1.4;
A=cr*rmspow;  %clipping level
sig1=sig';
mxm=max(sig1);
for i=1:size(sig,1)
   for j=1:size(sig,2)
      if mxm(1,i)==sig(i,j)
         place(i,1)=j;
      end
   end
end

ss=0;
for i=1:size(sig,1)
   for j=1:size(sig,2)
      if sig(i,j)>A
         if j>=5 & j<=size(sig,2)-4
            xmtifft(i,[j-4:j+4])=xmtifft(i,[j-4:j+4]).*wc;
         elseif j<5
            xmtifft(i,[j:j+4])=xmtifft(i,[j:j+4]).*(1-kc*hanning(5)');
         elseif j>size(sig,2)-4
```

```
                xmtifft(i,[j-4:j])=xmtifft(i,[j-4:j]).*(1-kc*hanning(5)');
            end
        end
    end
end

BB=1;
if BB==1
B=2;    %BOTTOM LEVEL
ss=0;
for i=1:size(sig,1)
    for j=1:size(sig,2)
        if sig(i,j)<B
            if j>=5 & j<=size(sig,2)-4
                xmtifft(i,[j-4:j+4])=xmtifft(i,[j-4:j+4]).*wc2;
            elseif j<5
                xmtifft(i,[j:j+4])=xmtifft(i,[j:j+4]).*(1+ka*hanning(5)');
            elseif j>size(sig,2)-2
                xmtifft(i,[j-4:j])=xmtifft(i,[j-4:j]).*(1+ka*hanning(5)');
            end
        end
    end
end
end

% PEAK CLIPPING
elseif window==2
CL=12; %CLIPPING LEVEL
    xmtifft=xmtifft*50;
xmtifft=xmtifft(:,[17:80]);
sig=abs(xmtifft);
 figure(30);plot(sig);
for i=1:size(sig,1)
    for j=1:size(sig,2)
        if sig(i,j)>CL
            xmtifft(i,j)=CL;
        end
    end
end

end
if window~=0
% insertion guard interval
Ng=16;
xmtifft=[xmtifft(:,N-Ng+1:N) xmtifft];
```

```
%xmtifft_afterGI=size(xmtifft)
ppp=xmtifft*.02;

elseif window==0
 ppp=xmtifft;
end

% PAPR CALCULATION
pow=xmtifft.*conj(xmtifft);
pow1=pow'; %transpose
maxi=max(pow1); %find peak ins. power of OFDM signal
ave=mean(pow1); %find average ins. power of OFDM signal
pap=maxi./ave; %PAPR array (PAPR value of each OFDM symbol)
max_PAPR=10*log10(max(pap)) % maximum PAPR in dB
ave_PAPR=10*log10(sum(pap)/size(pap,2)) %average PAPR in dB
%-----------------------------------------------------------------------------------
```

RAY_DOP

```
%RAY_DOP

% TITLE          : RAYLEIGH DOPPLER

%--------------------------------------------------------------------------------
% Thesis Advisor      : Prof M. Tummala, Naval Postgraduate School
% Author              : Prof. Paul H. Moose, Naval Postgraduate School
% Modified by         : Tan Kok Chye, Naval Postgraduate School
%--------------------------------------------------------------------------------
function c=ray_dop(s,M,N,es)
m=0:M-1;
randn('seed',s+10);
pr1=randn(1,20);
randn('seed',s+20);
pim=i*randn(1,20);
p=pr1+pim;
p=p/(40^.5);
rand('seed',s+30);
e=rand(1,20);
e=es*cos(2*pi*(e-.5));
E=exp(i*2*pi*e'*m/N);
c=p*E;
%--------------------------------------------------------------------------------
```

RMDECODER

```
%RMDECODER

% TITLE          : RM DECODER

%--------------------------------------------------------------------------------
% Thesis Advisor       : Prof M. Tummala, Naval Postgraduate School
% Author               : TEZEREN,Serdar Umit, LTJG, TURKISH NAVY
%--------------------------------------------------------------------------------
function [msg_decoded_line]=rmdecoder(Me,r,m);
k=1;% # of the rows=k;# of the columns=2^m
for i=1:r;
   k=k+prod(1:m)/prod(1:m-i)/prod(1:i);
end
[GM,k]=rmgenmat(r,m);
[WW1]=WW_mat2(GM,r,m,k);
rrr=rem(size(Me,2),2^m);
if rrr~=0;
   %display('ERROR!!! The # of the bits at the encoder output is not equal to the # of the
bits at the decoder input');
   Me=[Me zeros(1,2^m-rrr)];
end
Me=reshape(Me,2^m,length(Me)/2^m)';%RESHAPE Me AS 2^m COLUMNS
cv_row=1;
for i=2:k;
   counter=0;
   for j=1:size(WW1,1);
      dpt=dot_pro(GM(i,:),WW1(j,:));
      if dpt==1;
         dpt2_counter=0;flag=0;
         for z=2:k;
            dpt2=dot_pro(GM(z,:),WW1(j,:));
            if dpt2==0;
               dpt2_counter=dpt2_counter+1;
            end
         end
         if dpt2_counter==k-2;
            CV(cv_row,:)=WW1(j,:);
            check(1,cv_row)=i;
            check(2,cv_row)=j;
            cv_row=cv_row+1;
            counter=counter+1;
            GG(1,i)=counter;
         end
      end
```

```
    end
end

dp0=0;dp1=0;row_dd=1;My=zeros(size(Me,1),length(GM));
ss=0;
for f=1:size(Me,1);
    for i=2:k;
        for j=1:GG(1,i);%(2^(m-r));
            dp=dot_pro(CV(row_dd,:),Me(f,:));
            DP(i,j)=dp;
            if dp==1
               dp1=dp1+1;
            elseif dp==0;
               dp0=dp0+1;
            end
            row_dd=row_dd+1;
        end
        if dp1>dp0
           dotpro=1;
        else %if dp0>dp1
            dotpro=0;
            end
        dp0=0;dp1=0;
        WWW(1,i-1)=dotpro;
        My(f,:)=xor(My(f,:),(dotpro*GM(i,:)));
    end
    MM(f,:)=xor(My(f,:),Me(f,:));
    tt1=0;tt0=0;
    for i=1:size(MM,2);
        if MM(f,i)==0;
           tt0=tt0+1;
        elseif MM(f,i)==1;
           tt1=tt1+1;
        end
    end
    if tt1>tt0
      tt=1;
    elseif tt0>tt1;
      tt=0;
      elseif tt0==tt1
      tt=1;
   end
    msg_decoded(f,1)=tt;
    for i=2:k;
        msg_decoded(f,i)=WWW(1,i-1);
    end
   row_dd=1;
end
Mc=xor(MM,Me);
msg_decoded_line=reshape(msg_decoded',1,size(msg_decoded,1)*size(msg_decoded,2))
;
%---------------------------------------------------------------------------------------
```

RMENCODER

```
%RMENCODER

% TITLE          : RM ENCODER

%--------------------------------------------------------------------------------
% Thesis Advisor       : Prof M. Tummala, Naval Postgraduate School
% Author               : TEZEREN,Serdar Umit, LTJG, TURKISH NAVY
%---------------------------------------------------------------------------------
function [encd_msg]=rmencoder(r,m,msg1);
[GM,k]=rmgenmat(r,m);
len_msg=size(msg1,2);
rrr=rem(len_msg,k);   %check whether padding is necessary
if rrr>0;
   msg=[msg1 zeros(1,k-rrr)];
elseif rrr==0;
   msg=msg1;
end
len=size(msg,2)/k;
msg= reshape(msg,k,len)';%make the message in a matrix form
Mc=msg*GM;
for i=1:len;
   for j=1:2^m;
      if rem(Mc(i,j),2)==0;
         Mc(i,j)=0;
      elseif rem(Mc(i,j),2)>0;
         Mc(i,j)=1;
      end
   end
end
Mc;                                    %ENCODED MESSAGE MATRIX
encd_msg=reshape(Mc',1,(size(Mc,1)*size(Mc,2)));%ENCODED MESSAGE
%----------------------------------------------------------------------------------
```

RMGENMAT

```
%RMGENMAT

% TITLE          : RM ENCODING MATRIX GENERATOR

%--------------------------------------------------------------------------------
% Thesis Advisor       : Prof M. Tummala, Naval Postgraduate School
% Author               : TEZEREN,Serdar Umit, LTJG, TURKISH NAVY
%--------------------------------------------------------------------------------
function [GM,k]=rmgenmat(r,m);
k=1;% # of the rows=k;# of the columns=2^m
for i=1:r;
   k=k+prod(1:m)/prod(1:m-i)/prod(1:i);
end
[R]=gm_part1(m); %creating the monomials of the generator matrix (1,x1,x2,....,xm)
row=m+2;         %creating the second part of the generator matrix (x1x2,x1x3,.....,xm-
1xm)
for i=2:m;
   for j=i+1:m+1;
      R(row,:)=and(R(i,:),R(j,:));
      row=row+1;
   end
end
if k>row-1;  % check whether the third part is necessary
  % if necessary,
  for i=2:m+1;   %creating the third part of the generator matrix (x1x2x3,x1x2x4,.....,xm-
2xm-1xm)
     for j=i+1:m;
        for z=j+1:m+1;
           R(row,:)=R(i,:)&R(j,:)&R(z,:);
           row=row+1;
        end
     end
  end
end
if k>row-1;% check whether the fourth part is necessary
  % if necessary,
  %creating the fourth part of the generator matrix (x1x2x3x4,x1x2x3x5,.....,xm-3xm-
2xm-1xm)
  for i=2:m-2;
     for j=i+1:m-1;
        for z=j+1:m;
           for y=z+1:m+1;
              R(row,:)=R(i,:)&R(j,:)&R(z,:)&R(y,:);
              row=row+1;
```

```
            end
         end
      end
   end
end
if k>row-1;% check whether the fifth part is necessary
   % if necessary, add the fifth part
   for i=2:m-2;
      for j=i+1:m-1;
         for z=j+1:m;
            for y=z+1:m+1;
               R(row,:)=R(i,:)&R(j,:)&R(z,:)&R(y,:);
               row=row+1;
            end
         end
      end
   end
end
if k>row-1;% check whether the sixth part is necessary
   % if necessary, add the sixth part
   for i=2:m-3;
      for j=i+1:m-2;
         for z=j+1:m-1;
            for y=z+1:m;
               for h=m+1;
                  R(row,:)=R(i,:)&R(j,:)&R(z,:)&R(y,:)&R(h,:);
                  row=row+1;
               end
            end
         end
      end
   end
end
if k>row-1
    display('ERROR!!! the generator matrix need to be modified seventh part is necessary')
end
for i=1:k;
    GM(i,:)=R(i,:);
end
GM;% RM ENCODING MATRIX
%------------------------------------------------------------------------------------
```

ROTM

```
%ROTM

% TITLE          : ROTATE VECTOR

%---------------------------------------------------------------------------------
% Thesis Advisor       : Prof M. Tummala, Naval Postgraduate School
% Author               : Prof. Paul H. Moose, Naval Postgraduate School
% Modified by          : Tan Kok Chye, Naval Postgraduate School
%---------------------------------------------------------------------------------
function [vp,vn]=rotm(v,m)
L=length(v);
m=rem(m,L);
ii=(1:L)-1;
isp=rem(ii-m+L,L)+1;
isn=rem(ii+m+L,L)+1;
vp=v(isp);
vn=v(isn);
%---------------------------------------------------------------------------------
```

TDA

```
%TDA

% TITLE         : FFT AND GI INSERTION OPERATOR

%--------------------------------------------------------------------------------
% Thesis Advisor      : Prof M. Tummala, Naval Postgraduate School
% Author              : Prof. Paul H. Moose, Naval Postgraduate School
% Modified by         : Tan Kok Chye, Naval Postgraduate School
%--------------------------------------------------------------------------------
function x=tda(Ng,X)
[m N]=size(X);
% Perform inverse FFT on frequency values in array,X
x=ifft(X.');
% Add precursor of Ng samples to the beginning of the time domain array for channel
% compensation.
x=x.';
if Ng==0
x=x;
else
x=[x(:,N-Ng+1:N) x];
end
%--------------------------------------------------------------------------------
```

VITERBI

```
%VITERBI

% TITLE          : VITERBI DECODER

%--------------------------------------------------------------------------------
% Thesis Advisor       : Prof M. Tummala, Naval Postgraduate School
% Author               : Tan Kok Chye, Naval Postgraduate School
%----------------------------------------------------------------------------------
func-
tion[decoder_output,survivor_state,cumulated_metric]=viterbi(v_G,v_k,channel_output)
v_n=size(v_G,1);
% check the sizes
if rem(size(v_G,2),v_k)~=0
error('Size of v_G and v_k do not agree')
end
if rem(size(channel_output,2),v_n)~=0
error('channel output not of the right size')
end
v_L=size(v_G,2)/v_k;
number_of_states=2^((v_L-1)*v_k);
%generate state transition matrix, output matrix, and input matrix
for v_j=0:number_of_states-1
for v_l=0:2^v_k-1
[next_state,memory_contents]=nxt_stat(v_j,v_l,v_L,v_k);
input(v_j+1,next_state+1)=v_l;
branch_output=rem(memory_contents*v_G',2);
nextstate(v_j+1,v_l+1)=next_state;
output(v_j+1,v_l+1)=bin2deci(branch_output);
end
end
state_metric=zeros(number_of_states,2);
depth_of_trellis=length(channel_output)/v_n;
channel_output_matrix=reshape(channel_output,v_n,depth_of_trellis);
survivor_state=zeros(number_of_states,depth_of_trellis+1);
%start decoding of non-tail channel outputs
for v_i=1:depth_of_trellis-v_L+1
flag=zeros(1,number_of_states);
if v_i<=v_L
step=2^((v_L-v_i)*v_k);
else
step=1;
end
for v_j=0:step:number_of_states-1
for v_l=0:2^v_k-1
```

```
branch_metric=0;
binary_output=deci2bin(output(v_j+1,v_l+1),v_n);
for v_ll=1:v_n
branch_metric=branch_metric+metric(channel_output_matrix(v_ll,v_i),binary_output(v_
ll));
end
if((state_metric(nextstate(v_j+1,v_l+1)+1,2)>state_metric(v_j+1,1)+branch_metric)|flag(
nextstate(v_j+1,v_l+1)+1)==0)
state_metric(nextstate(v_j+1,v_l+1)+1,2)=state_metric(v_j+1,1)+branch_metric;
survivor_state(nextstate(v_j+1,v_l+1)+1,v_i+1)=v_j;
flag(nextstate(v_j+1,v_l+1)+1)=1;
end
end
end
state_metric=state_metric(:,2:-1:1);
end
%start decoding of the tail channel_outputs
for v_i=depth_of_trellis-v_L+2:depth_of_trellis
flag=zeros(1,number_of_states);
last_stop=number_of_states/(2^((v_i-depth_of_trellis+v_L-2)*v_k));
for v_j=0:last_stop-1
branch_metric=0;
binary_output=deci2bin(output(v_j+1,1),v_n);
for v_ll=1:v_n
branch_metric=branch_metric+metric(channel_output_matrix(v_ll,v_i),binary_output(v_
ll));
end
if((state_metric(nextstate(v_j+1,1)+1,2)>state_metric(v_j+1,1)+branch_metric)|flag(next
state(v_j+1,1)+1)==0)
state_metric(nextstate(v_j+1,1)+1,2)=state_metric(v_j+1,1)+branch_metric;
survivor_state(nextstate(v_j+1,1)+1,v_i+1)=v_j;
flag(nextstate(v_j+1,1)+1)=1;
end
end
state_metric=state_metric(:,2:-1:1);
end
%generate the decoder output from the optimal path
state_sequence=zeros(1,depth_of_trellis+1);
state_sequence(1,depth_of_trellis)=survivor_state(1,depth_of_trellis+1);
for v_i=1:depth_of_trellis
state_sequence(1,depth_of_trellis-
v_i+1)=survivor_state((state_sequence(1,depth_of_trellis+2-v_i)+1),depth_of_trellis-
v_i+2);
end
decoder_output_matrix=zeros(v_k,depth_of_trellis-v_L+1);
for v_i=1:depth_of_trellis-v_L+1
```

```
dec_output_deci=input(state_sequence(1,v_i)+1,state_sequence(1,v_i+1)+1);
dec_output_bin=deci2bin(dec_output_deci,v_k);
decoder_output_matrix(:,v_i)=dec_output_bin(v_k:-1:1)';
end
decoder_output=reshape(decoder_output_matrix,1,v_k*(depth_of_trellis-v_L+1));
cumulated_metric=state_metric(1,1);
%-----------------------------------------------------------------------------------
```

WWMAT_2

```
%WWMAT_2

% TITLE          : CHARACTERISTIC VECTOR FINDER
%--------------------------------------------------------------------------------
% Thesis Advisor       : Prof M. Tummala, Naval Postgraduate School
% Author               : TEZEREN,Serdar Umit, LTJG, TURKISH NAVY
%--------------------------------------------------------------------------------
function [WW1]=WW_mat2(GM,r,m,k);
[GM,k]=rmgenmat(r,m);
row_d=1;%rows themselves and inverses
for i=2:m+1;
   WW1(row_d,:)=GM(i,:);
   WW1(row_d,:)=inverse(GM(i,:));
   row_d=row_d+2;
end
%double combinations
for i=2:k;
   for j=i+1:k;
      WW1(row_d,:)=GM(i,:) & GM(j,:);
      WW1(row_d+1,:)=inverse(GM(i,:)) & GM(j,:);
      WW1(row_d+2,:)=GM(i,:) & inverse(GM(j,:));
      WW1(row_d+3,:)=inverse(GM(i,:)) & inverse(GM(j,:));
      row_d=row_d+4;
   end
end
for cc=3:m-1;
   for j=1:size(WW1,1);
      for i=cc+1:m+1;
         WW1(row_d,:)=GM(i,:) & WW1(j,:);
         WW1(row_d+1,:)=inverse(GM(i,:)) & WW1(j,:);
         row_d=row_d+2;
      end
   end
end
WW1(row_d,:)=ones(1,size(WW1,2));%adding all ones matrix at the end of the matrix
         %--------------------------------------------------------------------------------
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

1. Ben Cooke, “Reed-Muller Error Correction Codes,” *The MIT Undergraduate Journal of Mathematics*, Vol. 1, pp. 21-26, 1999.

2. HUT Communications Laboratory, *Lecture Notes on OFDM,* Course Notes for S-72.311 (Postgraduate Course in Communications Engineering), Hut, Finland (Unpublished).

3. David V. Roderick, “A Coded Orthogonal Frequency Division Multiplexing Simulation of a High Data Rate, Line-of-Sight, Digital Radio for Mobile Maritime Communications,” Master’s Thesis, Naval Postgraduate School, Monterey, California, 1997.

4. K. C. Tan, “Development, Simulation and Evaluation of the IEEE 802.11a Physical Layer in a Multipath Environment”, Master’s Thesis, Naval Postgraduate School, Monterey, California, 2001.

5. A. L. Intini, “Orthogonal Frequency Division Multiplexing for Wireless Networks,” Unpublished Project Report, University of California, Santa Barbara, California, 2000.

6. A. Y. Erdogan, “Analysis of the Effects of Frequency Offset in OFDM Systems,” Master’s Thesis, Naval Postgraduate School, Monterey, California, 2004.

7. Institute of Electrical and Electronics Engineers, *802.11a, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications High-Speed Physical Layer in the 5 GHz Band*, Piscataway, New Jersey, 1999.

8. Theodore S. Rappaport, *Wireless Communications*, Prentice Hall, Upper Saddle River, New Jersey, 2002.

9. V. Tarokh and H. Jafarkhani, “On the Computation and Reduction of the Peak-to-Average Power Ratio in Multicarrier Communications,” *IEEE Transactions on Communications*, Vol. 48, No. 1, pp. 37-44, 2000.

10. J. H. Scott, “The How and Why of OFDM,” *EBU Technical Review*, pp. 43-50, Geneva, 22-28 January 1998.

11. Stephen B. Wicker, *Error Correction Systems for Digital Communication and Storage*, Prentice Hall, Upper Saddle River, New Jersey, 1995.

12. Clark Robertson, Notes for EC4580 (Error Correction Coding), Naval Postgraduate School, 2003 (Unpublished).

13. A. Burr, “Turbo Codes: The Ultimate Error Correction Codes?,” *Electronics & Communication Engineering Journal*, pp. 155-165, August 2001.

14. T. A. Wilkinson and A. E. Jones, “Minimization of Peak to Average Envelope Power Ratio of Multicarrier Transmission Schemes by Block Coding,” *IEEE Vehicular Technology Conference*, pp. 825-829, 1995.

15. J. A. Davis and J. Jedwab, “Peak-to-Mean Power Control in OFDM, Golay Complementary Sequences, and Reed-Muller Codes,” *IEEE Transactions on Information Theory,* Vol. 45, No. 7, pp. 2397-2417, 1999.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Chairman, Code EC
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California

4. Proffessor Murali Tummala, Code EC/TU
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California

5. Proffessor Roberto Cristi, Code EC/CX
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California

6. Dr. Richard North
   Deputy Technical Director
   PEO C4I and Space
   San Diego, California

7. Benjamin Cooke
   Duke University Mathematics Department
   Durham, North Carolina

8. Serdar Umit TEZEREN
   Deniz Kuvvetleri Komutanligi
   Bakanliklar, Ankara, TURKEY