A

Major Project Report

on

PERFORMANCE ANALYSIS OF TURBO CODED OFDM BROADBAND TRANSMISSION OVER CHANNEL WITH NOISE

Submitted in partial fulfillment of the requirement
for the award of Degree of

MASTER OF ENGINEERING

(Electronics & Communication Engineering**)**

Submitted By

Saurabh Khare

College Roll No: 05/EC/05

University Roll No. 2803

Under the supervision and guidance of:

Dr. M. Kulkarni

Department of Information Technology

DEPARTMENT OF ELECTRONICS &COMMUNICATION ENGINEERING

DELHI COLLEGE OF ENGINEERING

DELHI UNIVERSITY

2005-2007

CERTIFICATE

This is to certify that the work contained in this major project report entitled **"Performance analysis of Turbo Coded OFDM Broadband Transmission over Channel with Noise"** submitted by **Saurabh Khare** in the requirement for the partial fulfillment for the award of the degree of Master of Engineering in Electronics & Communication, Delhi College of Engineering is an account of his work carried out under my guidance and supervision in the academic year 2006-2007.

The work embodied in this major project has not been submitted for the award of any other degree to the best of my knowledge.

Guided by:

Dr. M. Kulkarni
H.O.D
Department of Information Technology
Delhi College of Engineering
University of Delhi, Delhi-42

ACKNOWLEDGEMENT

It is a great pleasure to have the opportunity to extend my heartiest felt gratitude to everybody who helped me throughout the course of this project.

It is distinct pleasure to express my deep sense of gratitude and indebtedness to my learned supervisors Dr. M. Kulkarni, H.O.D, Information Technology, for his invaluable guidance, encouragement and patient reviews. With his continuous inspiration only, it becomes possible to complete this project. He kept on boosting me with time, to put an extra ounce of effort to realize this work.

I am also thankful to Prof. Asok Bhattacharyya, H.O.D, Electronics and Communication, for his valuable suggestions and constant support.

I would also like to take this opportunity to present my sincere regards to all the faculty members of the Department for their support and encouragement.

I am grateful to my parents for their moral support all the time; they have been always around to cheer me up, in the odd times of this work. I am also thankful to my classmates for their unconditional support and motivation during this work.

Saurabh Khare
M.E. (Electronics & Communication)
College Roll No. 05/EC/05
University Roll No. 2803
Department of Electronics & Communication
Delhi College of Engineering, Delhi-110042

ABSTRACT

Error control codes have become a vital part of modern digital wireless systems, enabling reliable transmission to be achieved over noisy channels. Over the past decade, turbo codes have been widely considered to be the most powerful error control code of practical importance.
In the same time-scale, mixed voice/data networks have advanced further and the concept of global wireless networks and terrestrial links has emerged. Such networks present the challenge of optimizing error control codes for different channel types, and for the different qualities of service demanded by voice and data.
Orthogonal Frequency Division Multiplexing (OFDM) has become a popular modulation method in high speed wireless communications. By partitioning a wideband fading channel into flat narrowband channels, OFDM is able to mitigate the detrimental effects of multi path fading using a simple one- tap equalizer. There is a growing need to quickly transmit information wirelessly and accurately.
Engineers have already combine techniques such as OFDM suitable for high data rate transmission with forward error correction (FEC) methods over wireless channels. In this thesis, we enhance the system throughput of a working OFDM system by adding turbo coding.
The smart use of coding and power allocation in OFDM will be useful to the desired performance at higher data rates.
Simulation is done over AWGN and impulsive noise (which is produced in broadband transmission) channels. The wideband system has 48 data sub-channels, each is individually modulated according to channel state information acquired during the previous burst. The end goal is to increase the system throughput while maintaining system performance under a desired bit error rate (BER).

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

The telecommunications' industry is in the midst of a veritable explosion in wireless technologies. Once exclusively military, satellite and cellular technologies are now commercially driven by ever more demanding consumers, who are ready for seamless communication from their home to their car, to their office, or even for outdoor activities. With this increased demand comes a growing need to transmit information wirelessly, quickly, and accurately. To address this need, communications engineer have combined technologies suitable for high rate transmission with forward error correction techniques. The latter are particularly important as wireless communications channels are far more hostile as opposed to wire alternatives, and the need for mobility proves especially challenging for reliable communications. For the most part, Orthogonal Frequency Division Multiplexing (OFDM) is the standard being used throughout the world to achieve the high data rates necessary for data intensive applications that must now become routine.

Orthogonal Frequency Division Multiplexing (OFDM) is a Multi-Carrier Modulation technique in which a single high rate data-stream is divided into multiple low rate data-streams and is modulated using sub-carriers which are orthogonal to each other. Some of the main advantages of OFDM are its multi-path delay spread tolerance and efficient spectral usage by allowing overlapping in the frequency domain. Also one other significant advantage is that the modulation and demodulation can be done using IFFT and FFT operations, which are computationally efficient.

In this thesis forward error correction is performed by using turbo codes. The combination of OFDM and turbo coding and recursive decoding allows these codes to achieve near Shannon's limit performance in the turbo cliff region.

This thesis presents the simulation of turbo-coded OFDM system and analyzes the performance of this system under noisy environment. It is presented as follows:

Chapter 2 introduces the theory behind OFDM as well as some of its advantages and functionality issues. We discuss basic OFDM transceiver architecture, cyclic prefix, intersymbol

interference, intercarrier interference and peak to average power ratios. We also present a few results in both Additive White Gaussian Noise, and impulsive noise environments.

Chapter 3 focuses on turbo codes. We explore encoder and decoder architecture, and decoding algorithms (especially the maximum a posteriori algorithm). We elaborate on the performance theory of the codes and find out why they perform so well.

Chapter 4 ties both technologies together. Here we introduce some theory about broadband transmission on power line network and also explain noises which usually occur in broadband transmission.

Chapter 5 consist simulated results of our work and a few suggestions are made on how to improve our system. Then, we present our results on the combination of turbo coding and OFDM. The core of our simulation results are found here.

CHAPTER 2

BASIC OF OFDM

Orthogonal Frequency Division Multiplexing (OFDM) is very similar to the well-known and used technique of Frequency Division Multiplexing (FDM). OFDM uses the principles of FDM to allow multiple messages to be sent over a single radio channel. It is however in a much more controlled manner, allowing an improved spectral efficiency. A simple example of FDM is the use of different frequencies for each FM (Frequency Modulation) radio stations. All stations transmit at the same time but do not interfere with each other because they transmit using different carrier frequencies. Additionally they are bandwidth limited and are spaced sufficiently far apart in frequency so that their transmitted signals do not overlap in the frequency domain. At the receiver, each signal is individually received by using a frequency tunable band pass filter to selectively remove all the signals except for the station of interest. This filtered signal can then be demodulated to recover the original transmitted information.

OFDM is different from FDM in several ways. In conventional broadcasting each radio station transmits on a different frequency, effectively using FDM to maintain a separation between the stations. There is however no coordination or synchronization between each of these stations. With an OFDM transmission such as DAB, the information signals from multiple stations are combined into a single multiplexed stream of data. This data is then transmitted using an OFDM ensemble that is made up from a dense packing of many subcarriers. All the subcarriers within the OFDM signal are time and frequency synchronized to each other, allowing the interference between subcarriers to be carefully controlled. These multiple subcarriers overlap in the frequency domain, but do not cause Inter-Carrier Interference (ICI) due to the orthogonal nature of the modulation. Typically with FDM the transmission signals need to have a large frequency guard-band between channels to prevent interference. This lowers the overall spectral efficiency. However with OFDM the orthogonal packing of the subcarriers greatly reduces this guard band, improving the spectral efficiency. All wireless communication systems use a modulation scheme to map the information signal to a form that can be effectively transmitted over the communications channel. A wide range of modulation schemes has been developed, with the

most suitable one, depending on whether the information signal is an analogue waveform or a digital signal. Some of the common analogue modulation schemes include Frequency Modulation (FM), Amplitude Modulation (AM), Phase Modulation (PM), Single Side Band (SSB), Vestigial Side Band (VSB), Double Side Band Suppressed Carrier (DSBSC) [5]. Common single carrier modulation schemes for digital communications include, Amplitude Shift Keying (ASK), Frequency Shift Keying (FSK), Phase Shift Keying (PSK) and Quadrature Amplitude Modulation (QAM). Each of the carriers in a FDM transmission can use an analogue or digital modulation scheme. There is no synchronization between the transmission and so one station could transmit using FM and another in digital using FSK. In a single OFDM transmission all the subcarriers are synchronized to each other, restricting the transmission to digital modulation schemes. OFDM is symbol based, and can be thought of as a large number of low bit rate carriers transmitting in parallel. All these carriers transmit in unison using synchronized time and frequency, forming a single block of spectrum. This is to ensure that the orthogonal nature of the structure is maintained. Since these multiple carriers form a single OFDM transmission, they are commonly referred to as 'subcarriers', with the term of 'carrier' reserved for describing the RF carrier mixing the signal from base band. There are several ways of looking at what make the subcarriers in an OFDM signal orthogonal and why this prevents interference between them.

2.1 Orthogonality

Signals are orthogonal if they are mutually independent of each other. Orthogonality is a property that allows multiple information signals to be transmitted perfectly over a common channel and detected, without interference. Loss of orthogonality results in blurring between these information signals and degradation in communications. Many common multiplexing schemes are inherently orthogonal. Time Division Multiplexing (TDM) allows transmission of multiple information signals over a single channel by assigning unique time slots to each separate information signal. During each time slot only the signal from a single source is transmitted preventing any interference between the multiple information sources. Because of this TDM is orthogonal in nature. In the frequency domain most FDM systems are orthogonal as each of the separate transmission signals are well spaced out in frequency preventing

interference. Although these methods are orthogonal the term OFDM has been reserved for a special form of FDM. The subcarriers in an OFDM signal are spaced as close as is theoretically possible while maintain orthogonality between them. OFDM achieves orthogonality in the frequency domain by allocating each of the separate information signals onto different subcarriers [1]. OFDM signals are made up from a sum of sinusoids, with each corresponding to a subcarrier. The baseband frequency of each subcarrier is chosen to be an integer multiple of the inverse of the symbol time, resulting in all subcarriers having an integer number of cycles per symbol. As a consequence the subcarriers are orthogonal to each other. Sets of functions are orthogonal to each other if they match the conditions in equation (2.1). If any two different functions within the set are multiplied, and integrated over a symbol period, the result is zero, for orthogonal functions. Another way of thinking of this is that if we look at a matched receiver for one of the orthogonal functions, a subcarrier in the case of OFDM, then the receiver will only see the result for that function. The results from all other functions in the set integrate to zero, and thus have no effect.

$$\int_0^T s_i(t)s_j(t)dt = \begin{cases} c & i = j \\ 0 & i \neq j \end{cases} \qquad (2.1)$$

Equation (2-2) shows a set of orthogonal sinusoids, which represent the subcarriers for an unmodulated real OFDM signal [1].

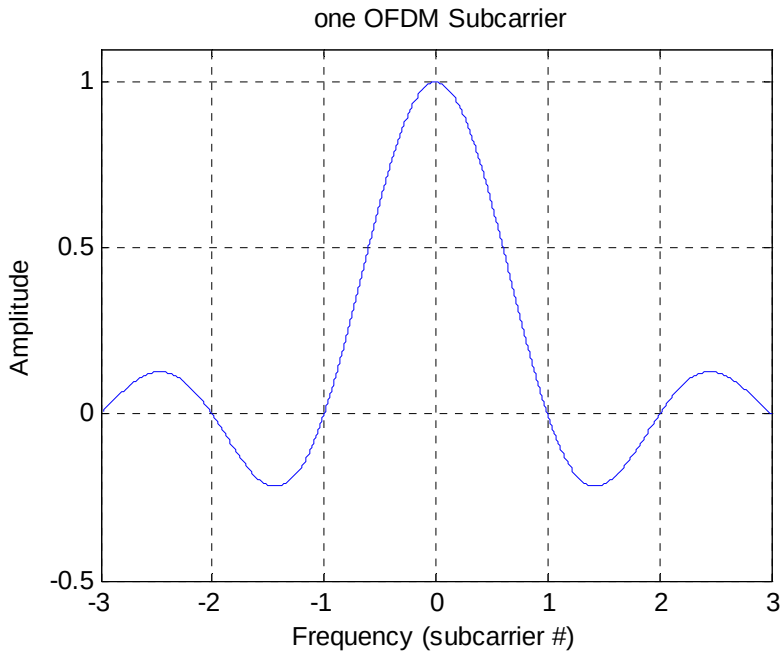
$$s_k(t) = \begin{cases} \sin 2\Pi k f_0 t & 0 < t < T \\ 0 & otherwise \end{cases} \qquad k = 1,2,3...N \qquad (2.2)$$

where f_0 is the carrier spacing, N is the number of carriers, T is the symbol period. Since the highest frequency component is N f_0 the transmission bandwidth is also N f_0. These subcarriers are orthogonal to each other because when we multiply the waveforms of any two subcarriers and integrate over the symbol period the result is zero. Multiplying the two sine waves together is the same as mixing these subcarriers. This results in sum and difference frequency components, which will always be integer subcarrier frequencies, as the frequency of the two mixing subcarriers has integer number of cycles. Since the system is linear we can integrate the result by taking the integral of each frequency component separately then combining the results by adding the two sub-integrals. The two frequency components after the

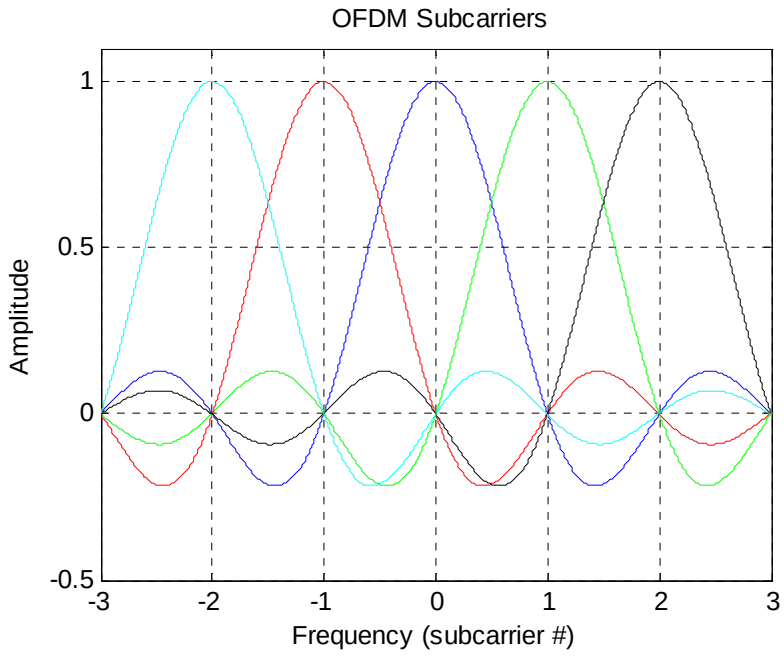
mixing have an integer number of cycles over the period and so the sub-integral of each component will be zero, as the integral of a sinusoid over an entire period is zero. Both the sub-integrals are zeros and so the resulting addition of the two will also be zero, thus we have established that the frequency components are orthogonal to each other.

2.2 Frequency Domain Orthogonality

Another way to view the orthogonality property of OFDM signals is to look at its spectrum. In the frequency domain each OFDM subcarrier has a sinc, sin(x)/x, frequency response, as shown in Figure 2.1. This is a result of the symbol time corresponding to the inverse of the carrier spacing. As far as the receiver is concerned each OFDM symbol transmitted for a fixed time (T_{FFT}) with no tapering at the ends of the symbol. This symbol time corresponds to the inverse of the subcarrier spacing of $1/T_{FFT}$ Hz . This rectangular, waveform in the time domain results in a sinc frequency response in the frequency domain. The sinc shape has a narrow main lobe, with many side-lobes that decay slowly with the magnitude of the frequency difference away from the centre. Each carrier has a peak at the centre frequency and nulls evenly spaced with a frequency gap equal to the carrier spacing. The orthogonal nature of the transmission is a result of the peak of each subcarrier corresponding to the nulls of all other subcarriers. If the DFT is time synchronized, the frequency samples of the DFT correspond to just the peaks of the subcarriers, thus the overlapping frequency region between subcarriers does not affect the receiver [6]. The measured peaks correspond to the nulls for all other subcarriers, resulting in orthogonality between the subcarriers.



(a)Single Carrier of OFDM Signal



(b) Multiple Carriers of OFDM

Figure 2.1 Frequency domain representation of OFDM

2.3 OFDM Generation and Reception

Figure 2-2 shows the block diagram of a typical OFDM transceiver. The transmitter section converts digital data to be transmitted, into a mapping of subcarrier amplitude and phase. It then transforms this spectral representation of the data into the time domain using an Inverse Discrete Fourier Transform (IDFT). The Inverse Fast Fourier Transform (IFFT) performs the same operations as an IDFT, except that it is much more computationally efficiency, and so is used in all practical systems. In order to transmit the OFDM signal the calculated time domain signal is then mixed up to the required frequency. The receiver performs the reverse operation of the transmitter, mixing the RF signal to base band for processing, then using a Fast Fourier Transform (FFT) to analyse the signal in the frequency domain. The amplitude and phase of the subcarriers is then picked out and converted back to digital data. The IFFT and the FFT are complementary function and the most appropriate term depends on whether the signal is being received or generated. In cases where the signal is independent of this distinction then the term FFT and IFFT is used interchangeably.

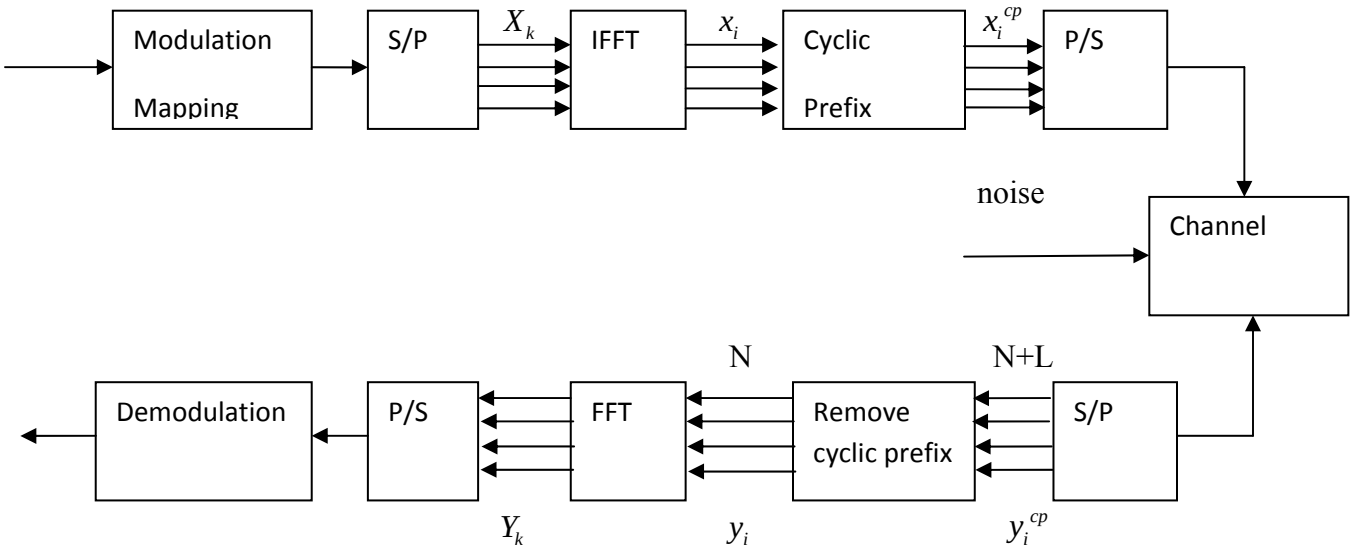


Figure 2.2- Block diagram of a basic OFDM transceiver

2.3.1 Subcarrier Modulation

A large number of modulation schemes are available allowing the number of bits transmitted per carrier per symbol to be varied. Digital data is transferred in an OFDM link by using a modulation scheme on each subcarrier. A modulation scheme is a mapping of data words to a real (In phase) and imaginary (Quadrature) constellation, also known as an IQ constellation. For example 256-QAM (Quadrature Amplitude Modulation) has 256 IQ points in the constellation constructed in a square with 16 evenly spaced columns in the real axis and 16 rows in the imaginary axis. The number of bits that can be transferred using a single symbol corresponds to log2(M), where M is the number of points in the constellation, thus 256-QAM transfers 8 bits per symbol. Increasing the number of points in the constellation does not change the bandwidth of the transmission, thus using a modulation scheme with a large number of constellation points, allows for improved spectral efficiency. For example 256-QAM has a spectral efficiency of 8 b/s/Hz, compared with only 1 b/s/Hz for BPSK. However, the greater the number of points in the modulation constellation, the harder they are to resolve at the receiver.

2.3.2 Serial to Parallel Conversion

Data to be transmitted is typically in the form of a serial data stream. In OFDM, each symbol typically transmits 40 - 4000 bits, and so a serial to parallel conversion stage is needed to convert the input serial bit stream to the data to be transmitted in each OFDM symbol. The data allocated to each symbol depends on the modulation scheme used and the number of subcarriers. For example, for a subcarrier modulation of 16-QAM each subcarrier carries 4 bits of data, and so for a transmission using 100 subcarriers the number of bits per symbol would be 400. At the receiver the reverse process takes place, with the data from the subcarriers being converted back to the original serial data stream.

2.3.3 Frequency to Time Domain Conversion

The OFDM message is generated in the complex baseband. Each symbol is

modulated onto the corresponding subcarrier using variants of phase shift keying (PSK) or different forms of quadrature amplitude modulation (QAM). The data symbols are converted from serial to parallel before data transmission. The frequency spacing between adjacent subcarriers is Nπ/2, where N is the number of subcarriers. This can be achieved by using the inverse discrete Fourier transform (IDFT), easily implemented as the inverse fast Fourier transform (IFFT) operation [6]. As a result, the OFDM symbol generated for an N-subcarrier system translates into N samples, with the ith sample being

$$x_i = \sum_{n=0}^{N-1} C_n \exp\left\{ j \frac{2\Pi in}{N} \right\}, 0 \le i \le N-1 \qquad (2.3)$$

At the receiver, the OFDM message goes through the exact opposite operation in the discrete Fourier transform (DFT) to take the corrupted symbols from a time domain form into the frequency domain. In practice, the baseband OFDM receiver performs the fast Fourier transform (FFT) of the receive message to recover the information that was originally sent

2.3.4 Guard Period

For a given system bandwidth the symbol rate for an OFDM signal is much lower than a single carrier transmission scheme. For example for a single carrier BPSK modulation, the symbol rate corresponds to the bit rate of the transmission. However for OFDM the system bandwidth is broken up into N subcarriers, resulting in a symbol rate that is N times lower than the single carrier transmission. This low symbol rate makes OFDM naturally resistant to effects of Inter-Symbol Interference (ISI) caused by multipath propagation. Multipath propagation is caused by the radio transmission signal reflecting off objects in the propagation environment, such as walls, buildings, mountains, etc. These multiple signals arrive at the receiver at different times due to the transmission distances being different. This spreads the symbol boundaries causing energy leakage between them. The effect of ISI on an OFDM signal can be further improved by the addition of a guard period to the start of each symbol. This guard period is a cyclic copy that extends the length of the symbol waveform. Each subcarrier, in the data section of the symbol, (i.e. the OFDM symbol with no guard period added, which is equal to the length

of the IFFT size used to generate the signal) has an integer number of cycles. Because of this, placing copies of the symbol end-to-end results in a continuous signal, with no discontinuities at the joins [4]. Thus by copying the end of a symbol and appending this to the start results in a longer symbol time. Figure 2.3 shows the insertion of a guard period.

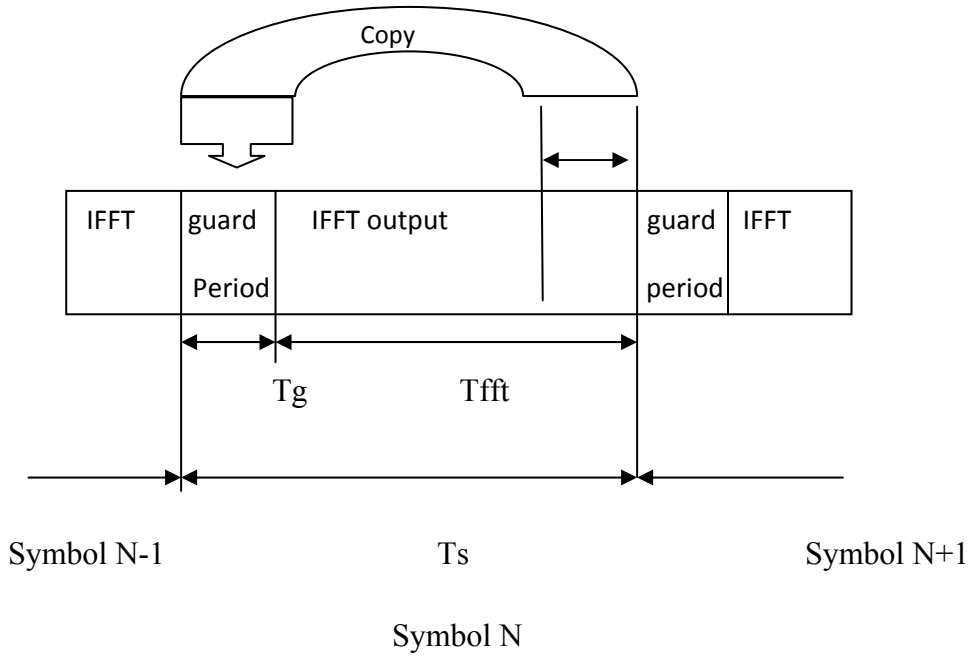


Figure 2.3 Addition of Guard Period to an OFDM Signal

The total length of the symbol is $T_S = T_G + T_{FFT}$, where T_S is the total length of the symbol in samples, T_G is the length of the guard period in samples, and T_{FFT} is the size of the IFFT used to generate the OFDM signal. In addition to protecting the OFDM from ISI, the guard period also provides protection against time-offset errors in the receiver.

2.3.5 Effect of Additive White Gaussian Noise on OFDM

Noise exists in all communications systems operating over an analog physical channel, such as radio. The main sources are thermal background noise, and electrical noise in the receiver amplifiers, and inter-cellular interference. In addition to this noise can also be generated internally to the communications system as a result of Inter-Symbol Interference (ISI),

Inter-Carrier Interference (ICI), and Inter- Modulation Distortion (IMD). These sources of noise decrease the Signal to Noise Ratio (SNR), ultimately limiting the spectral efficiency of the system. Noise, in all its forms, is the main detrimental effect in most radio communication systems. It is therefore important to study the effects of noise on the communications error rate and some of the tradeoffs that exists between the level of noise and system spectral efficiency. Most types of noise present in radio communication systems can be modeled accurately using Additive White Gaussian Noise (AWGN). This noise has a uniform spectral density (making it white), and a Gaussian distribution in amplitude (this is also referred to as a normal distribution). Thermal and electrical noise from amplification, primarily have white Gaussian noise properties, allowing them to be modeled accurately with AWGN. Also most other noise sources have AWGN properties due to the transmission being OFDM. OFDM signals have a flat spectral density and a Gaussian amplitude distribution provided that the number of carriers is large (greater than about 20 subcarriers), because of this the inter-cellular interference from other OFDM systems have AWGN properties. For the same reason ICI, ISI, and IMD also have AWGN properties for OFDM signals.

2.4 Simulation Setup

The effect of AWGN on OFDM was simulated for a wide range of subcarrier modulation schemes. The lower density modulation schemes show the data mapping used. The results presented show the BER performance as a function of the channel SNR. Other simulations in this thesis, present the performance of OFDM under a range of detrimental effects. These simulations measure the performance by finding the effective SNR of the channel instead of the BER. These simulations show the communication performance that corresponds to an equivalent AWGN SNR. The results presented in this section can then be used to predict the BER for the particular modulation scheme used. The symbol error rate of most common modulation schemes has been derived in algebraic form [5]. However, the derivation of the BER is difficult due to possibility of multiple bit errors per symbol. Additionally the BER for only square QAM modulation schemes, such as 16-QAM, 64-QAM, 256-QAM, etc, can be calculated directly. To overcome this problem the BER performance of OFDM was obtained by using

simulation. There are two main categories of modulation presented, which are, QAM and QPSK. These both are simulated in MATLAB and their performances are shown in figure 2.4.

2.5 Simulation Results

The simulated performance for the modulation schemes tested is shown in Figure 2.4. These show the BER as a function of the Energy per Bit to Noise Ratio (EBNR). This is a measure of the energy efficiency of a modulation scheme. If a higher EBNR is needed to transfer data for a given modulation scheme, then it means that more energy is required for each bit transfer. Low spectral efficiency modulation schemes, such as BPSK and QPSK, require a lower EBNR, and hence are more energy efficient. For a power limited system, with unbounded bandwidth, the maximum data rate could be achieved using BPSK or QPSK. However, in most applications the available bandwidth is the limiting factor and so the data rate is maximized by using a more spectrally efficient modulation schemes such as 256 QAM. The BER verses the SNR can be calculated from EBNR shown on the plots in Figure 2.4. The SNR for each modulation takes into account the number of bits per symbol, and so the signal power corresponds to the energy per bit times the number of bits per symbol. In log scale the SNR for a given EBNR can be found with:

$$SNR = 10.\log_{10}(N_b) + EBNR_{db} \qquad (2.4)$$

where SNR is in dB, N_b is the number of bits per symbol for the modulation scheme and $EBNR_{db}$ is the EBNR in dB.

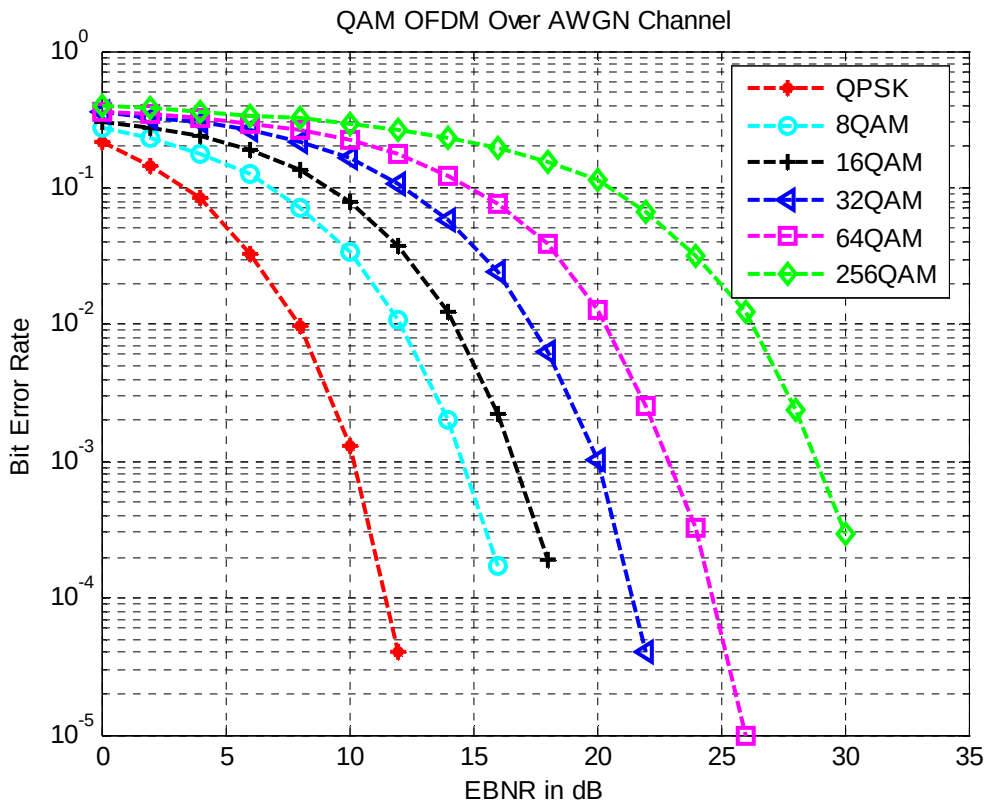


Figure2.4 QPSK OFDM, 8 QAMOFDM, 16 QAMOFDM, 32 QAMOFDM, 64 QAMOFDM and 256 QAMOFDM Performances

Note: BPSK and QPSK have the same performance

2.6 Advantages of OFDM

OFDM has several advantages over single carrier modulation systems and these make it a viable alternative for CDMA in future wireless networks. In this section, some of these advantages are discussed.

2.6.1 Multipath Delay Spread Tolerance

OFDM is highly immune to multipath delay spread that causes inter-symbol interference in wireless channels. Since the symbol duration is made larger (by converting a high

data rate signal into 'N' low rate signals), the effect of delay spread is reduced by the same factor. Also by introducing the concepts of guard time and cyclic extension, the effects of inter-symbol interference (ISI) and inter-carrier interference (ICI) is removed completely.

2.6.2 Immunity to Frequency selective fading Channels

If the channel undergoes frequency selective fading, then complex equalization techniques are required at the receiver for single carrier modulation techniques. But in the case of OFDM the available bandwidth is split among many orthogonal narrowly spaced sub-carriers. Thus the available channel bandwidth is converted into many narrow flat- fading sub-channels. Hence it can be assumed that the sub-carriers experience flat fading only, though the channel gain/phase associated with the sub-carriers may vary. In the receiver, each sub-carrier just needs to be weighted according to the channel gain/phase encountered by it. Even if some sub-carriers are completely lost due to fading, proper coding and interleaving at the transmitter can recover the user data.

2.6.3 High Spectral Efficiency

OFDM achieves high spectral efficiency by allowing the sub-carriers to overlap in the frequency domain. At the same time, to facilitate inter-carrier interference free demodulation of the sub-carriers, the sub-carriers are made orthogonal to each other. If the number of sub-carriers in 'N', the total bandwidth required is

$$BW_{total} = \frac{(N+1)}{T_s} \qquad (2.5)$$

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

$$BW_{total} \approx \frac{N}{T_s} \qquad (2.6)$$

On the other hand, the bandwidth required for serial transmission of the same data is

$$BW_{total} = \frac{2N}{T_s} \qquad (2.7)$$

Thus we achieve very high spectral gain in OFDM compared to the single carrier serial transmission case.

2.6.4 Efficient Modulation and Demodulation

Modulation and Demodulation of the sub-carriers is done using IFFT and FFT methods respectively, which are computationally efficient. By performing the modulation and demodulation in the digital domain, the need for highly frequency stable oscillators is avoided.

2.7 Applications of OFDM

Orthogonal Frequency Division Multiplexing is a new technology whose applications just being explored. The primary applications are in multimedia push technology and in wireless LAN.

2.7.1 Wireless LAN Applications

The IEEE 802.11 committee has its OFDM parameters are as shown below [4]:

Data Rate	6,9,12,18,24,36,48,54 Mbps
Modulation	BPSK, QPSK, 16-QAM, 64 QAM
Coding Rate	1/2, 2/3,3/4
No of Sub-Carriers	52
No of pilots	4
OFDM Symbol Duration	4 us
Guard Interval	800 ns
Sub-Carrier Spacing	312.5 kHz
3 dB bandwidth	16.56 MHz
Channel Spacing	20 MHz

Table 1: Parameters of IEEE 802.11

2.7.2 Digital Audio Broadcasting (DAB)

Digital Audio Broadcasting is a new multimedia push technology, with a good sound quality and better spectrum efficiency. This is achieved by the use of OFDM technology.

The DAB system samples audio at a sample rate of 48 kHz and a resolution of 22bits [7]. Then the data is compressed to between 32 and 384 KBPS. A rate ¼ convolution code is used with constraint length 7. The total data rate is about 2.2Mbps. The frame time is 24ms. QPSK modulation is performed at the transmitter.

The advantage of using OFDM for DAB is that the OFDM suffers very little from delay spread and also that the OFDM system has high spectral efficiency.

2.7.3 Digital Video Broadcasting (DVB)

Digital Video Broadcasting (DVB) is an ETSI standard for broadcasting Digital Television over satellites, cables and thorough terrestrial (wireless) transmission [20].

Terrestrial DVB operates in either of 2 modes called 2k and 8k modes with 1705 carriers and 6817 carriers respectively. It uses QPSK, 16 QAM or 64 QAM subcarrier modulations. It also uses pilot subcarriers for recovering amplitude and phase for coherent demodulation.

CHAPTER 3

TURBO CODES

Turbo codes were first presented at the International Conference on Communications in 1993. Until then, it was widely believed that to achieve near Shannon's bound performance, one would need to implement a decoder with infinite complexity or close. Parallel concatenated codes, as they are also known, can be implemented by using either block codes (PCBC) or convolutional codes (PCCC). PCCC resulted from the combination of three ideas that were known to all in the coding community:

- The transforming of commonly used non-systematic convolutional codes into systematic convolutional codes.
- The utilization of soft input soft output decoding. Instead of using hard decisions, the decoder uses the probabilities of the received data to generate soft output which also contain information about the degree of certainty of the output bits.
- Encoders and decoders working on permuted versions of the same information. This is achieved by using an interleaver.

An iterative decoding algorithm centered around the last two concept would refine its output with each pass, thus resembling the turbo engine used in airplanes. Hence, the name Turbo was used to refer to the process.

3.1 Encoders for Turbo Codes

The encoder for a turbo code is a parallel concatenated convolutional code. Figure 3.1 shows a block diagram of the encoder first presented by Berrou et al [10]. The binary input data sequence is represented by $d_k = (d_1, ..., d_N)$. The input sequence is passed into the input of a convolutional encoder [8], ENC_1, and a coded bit stream, $x_{k_1}^p$ is generated. The data sequence is then interleaved. That is, the bits are loaded into a matrix and read out in a way so as to spread the positions of the input bits. The bits are often read out in a pseudo-random manner. The interleaved data sequence is passed to a second convolutional encoder ENC_2, and a second coded

bit stream, $x_{k_2}^p$ is generated. The code sequence that is passed to the modulator for transmission is a multiplexed (and possibly punctured) stream consisting of systematic code bits x_k^s and parity bits from both the first encoder $x_{k_1}^p$ and the second encoder $x_{k_2}^p$.

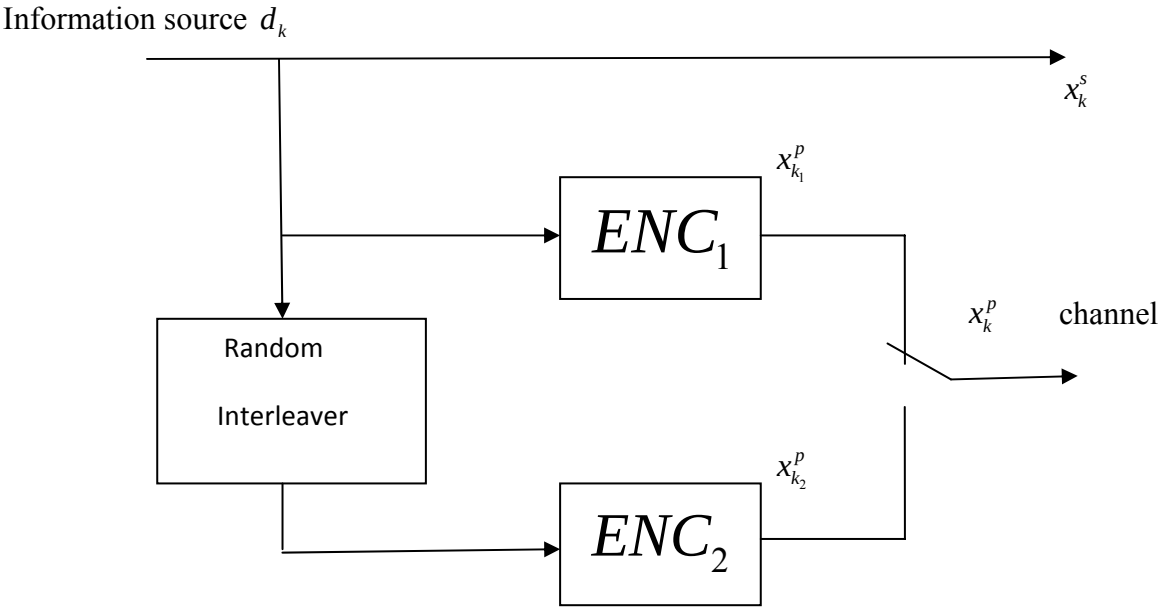
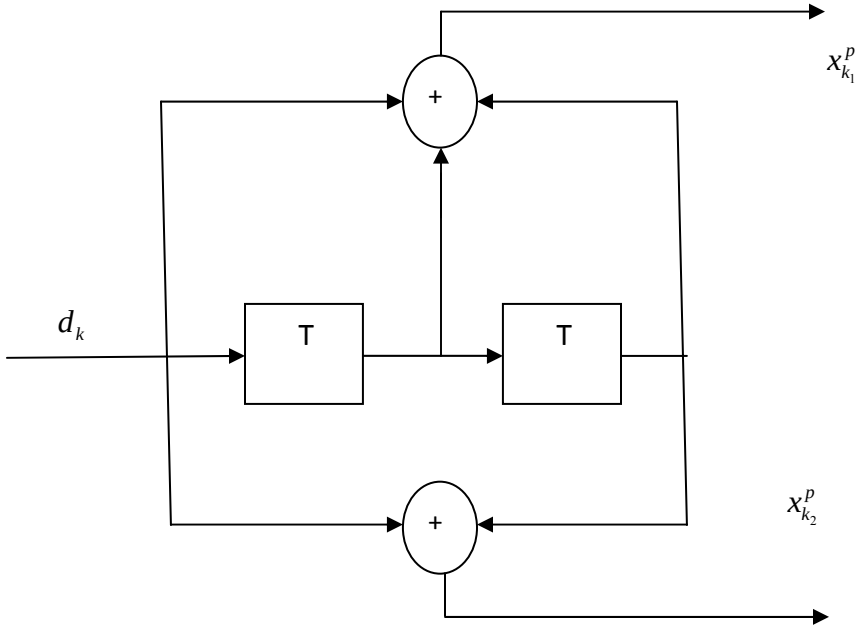


Figure 3.1 Structure of a turbo encoder

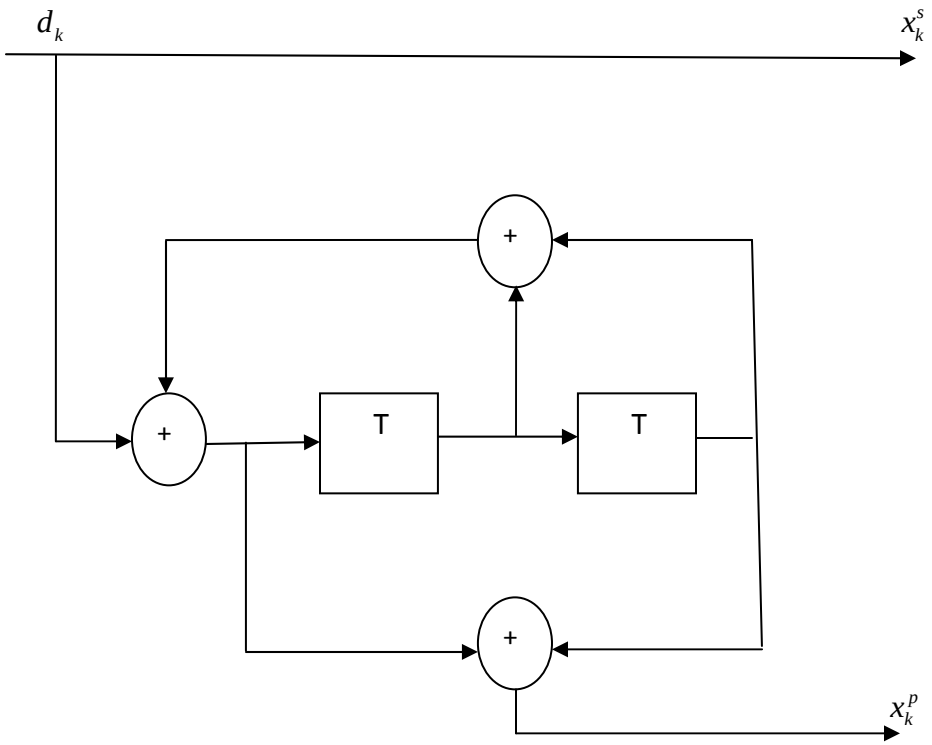
3.1.1 RSC Component Codes

ENC_1 and ENC_2 are Recursive Systematic Convolutional (RSC) codes - that is, convolutional codes which use feedback (they are 'recursive') and in which the uncoded data bits appear in the transmitted code bit sequence (they are 'systematic'). A simple RSC encoder is shown in Figure 3.2 along with a non-systematic (NSC) encoder, for comparison. The RSC encoder is rate 1/2, with constraint length K=3, and generator polynomial G= $\{g_1, g_2\}$ = {7, 5}, where g_1 is the feedback connectivity and g_2 is the output connectivity, in octal notation. An

RSC component encoder has two output sequences. One is the data sequence, $x_k^s = (x_1^s, ..., x_N^s)$. The other is the parity sequence $x_k^p = (x_1^p, ..., x_N^p)$. To understand why RSC component codes are used in a turbo code encoder, rather than the conventional NSC codes, it is necessary to first discuss the structure of error control codes. The minimum number of errors an error control code can correct is determined by the minimum Hamming distance of the code - the minimum number of bit positions in which any two codewords differ. The linear nature of turbo codes (at least, those using BPSK/QPSK modulations) means that the minimum Hamming distance of the code can be determined by comparing each possible codeword with the all-zeroes codeword. This process simplifies analysis of the code somewhat, and the minimum Hamming distance is then equal to the minimum codeweight (number of '1's) which occurs in any codeword. The minimum Hamming distance tends to determine the BER performance of the code at high SNR - the asymptotic performance. A high minimum Hamming distance results in a steep rate of fall of BER as SNR becomes large, whereas a low value results in a slow rate of fall. In the case of a turbo code, this rate of fall at high SNR is so slow; it is termed an error floor. At low SNR, however, codewords with codeweights larger than the minimum value must be considered. It is then that the overall distance spectrum of the code becomes important.



(a)NSC



(b)RSC

Figure 3.2 Example of (a) non-systematic convolutional (NSC) and (b) recursive systematic convolutional (RSC) encoders

This is the relationship between the codeweight and the number of codewords with that codeweight. Now, RSC codes have an infinite impulse response. That is, if a data sequence consisting of a '1' followed by a series of '0's enters the RSC encoder, a code sequence will be generated containing both ones and zeroes for as long as the subsequent data bits remain zero. This property means that RSC encoders will tend to generate high weight code sequences for groups of data bits spread far apart in the input sequence. An NSC code, however, will return to the all-zeroes state after K-1 input zeroes, where K is the constraint length of the encoder. The infinite impulse response property of RSC codes is complemented in turbo codes by the interleaver between component encoders. An interleaver is a device for permuting a sequence of bits (or symbols) at its input into an alternate sequence with a different ordering at the output. Turbo codes tend to make use of pseudo-random interleavers, whose role is to ensure that most groups of data bits which are close together when entering one RSC encoder are spread far apart

before entering the other RSC encoder. The result is a composite codeword which will often have a high codeweight. The details of interleaving will be discussed in more detail in the next section. This does not, however, mean that turbo codes tend to exhibit high minimum Hamming distances, and therefore good asymptotic performance. In fact, the opposite is usually true. We shall see in the following chapter that the pseudo-random nature of most turbo code interleavers tends to result in a mapping such that a few combinations of input bit positions which cause low codeweight sequences in one RSC component code are permuted into combinations of positions which generate low codeweight sequences in the second RSC code. The result in such a case is a low composite codeweight. Such pseudo-random mappings often lead to turbo codes having a low minimum codeweight compared to, say, NSC-based convolutional codes, resulting in a marked error floor at high SNR. It is clear from this brief discussion that interleaver design is crucial in ensuring that a turbo code/interleaver combination has the lowest possible error floor. It was mentioned earlier that at low SNR, the distance spectrum of the code as a whole becomes significant in determining BER performance, and that the combination of RSC code and pseudo-random interleaving produces codewords with higher codeweights most of the time. This results in there being fewer codewords with relatively low codeweights than a comparable convolutional code. It shall be shown later in constructing theoretical bounds for turbo codes that it is the number of codewords at each weight, as well as the actual codeweight, which determines the error probability of a code. The low multiplicity of low codeweight sequences associated with turbo codes, sometimes referred to as spectral thinning, leads to their good BER performance at low SNR. A full analysis of the turbo code characteristics described here is given in [8].

3.1.2 Interleaving

It was mentioned in the previous section that an interleaver is a device for reordering a sequence of bits or symbols. A familiar role of interleavers in communications is that of the symbol interleaver which is used after error control coding and signal mapping to ensure that fading bursts affecting blocks of symbols transmitted over the channel are broken up at the receiver by a de-interleaver, prior to decoding. Most error control codes work much better when errors in the received sequence are spread far apart. Another role is that of the interleaver between component codes in a serially concatenated code scheme; for example, between a Reed

Solomon outer code and a convolutional inner code. The trellis decoding nature of most convolutional codes means that uncorrected errors at the output of the decoder will tend to occur in bursts. The interleaving between the two component codes then ensures that these bursts are adequately spread before entering the outer decoder. In both these examples, the interleaver is typically implemented as a block interleaver. This is a rectangular matrix such that bits or symbols are written in one row at a time, and then read out one column at a time. Thus bits or symbols which were adjacent on writing are spaced apart by the number of rows when reading. The de-interleaving process is simply the inverse of this; writing in column by column and reading out row by row, to achieve the original bit or symbol ordering. Block interleaving is simple to implement, and suitable where the objective is to spread bursts of errors evenly by as large a distance as possible. Block interleavers are not suitable as turbo code interleavers, because they tend to generate large numbers of codewords with a relatively low weight, and therefore with a relatively low Hamming distance between them, due to the regularity of the spreading process. Berrou and Glavieux introduced pseudo-random interleaving into turbo codes to solve this problem. A pseudo-random interleaver is a random mapping between input and output positions, generated by means of some form of pseudo-random number generator. Figure 3.3 shows a simple illustration of pseudo-random interleaving. The original data sequence is represented by the sequence of white squares, and the interleaved data sequence is represented by the grey squares. Turbo code BER performance improves with interleaver length - the so-called interleaver gain - but the loading and unloading of the interleaver adds a considerable delay to the decoding process. This would make a 256x256 interleaver unsuitable for, say, real time speech applications, which are delay sensitive.

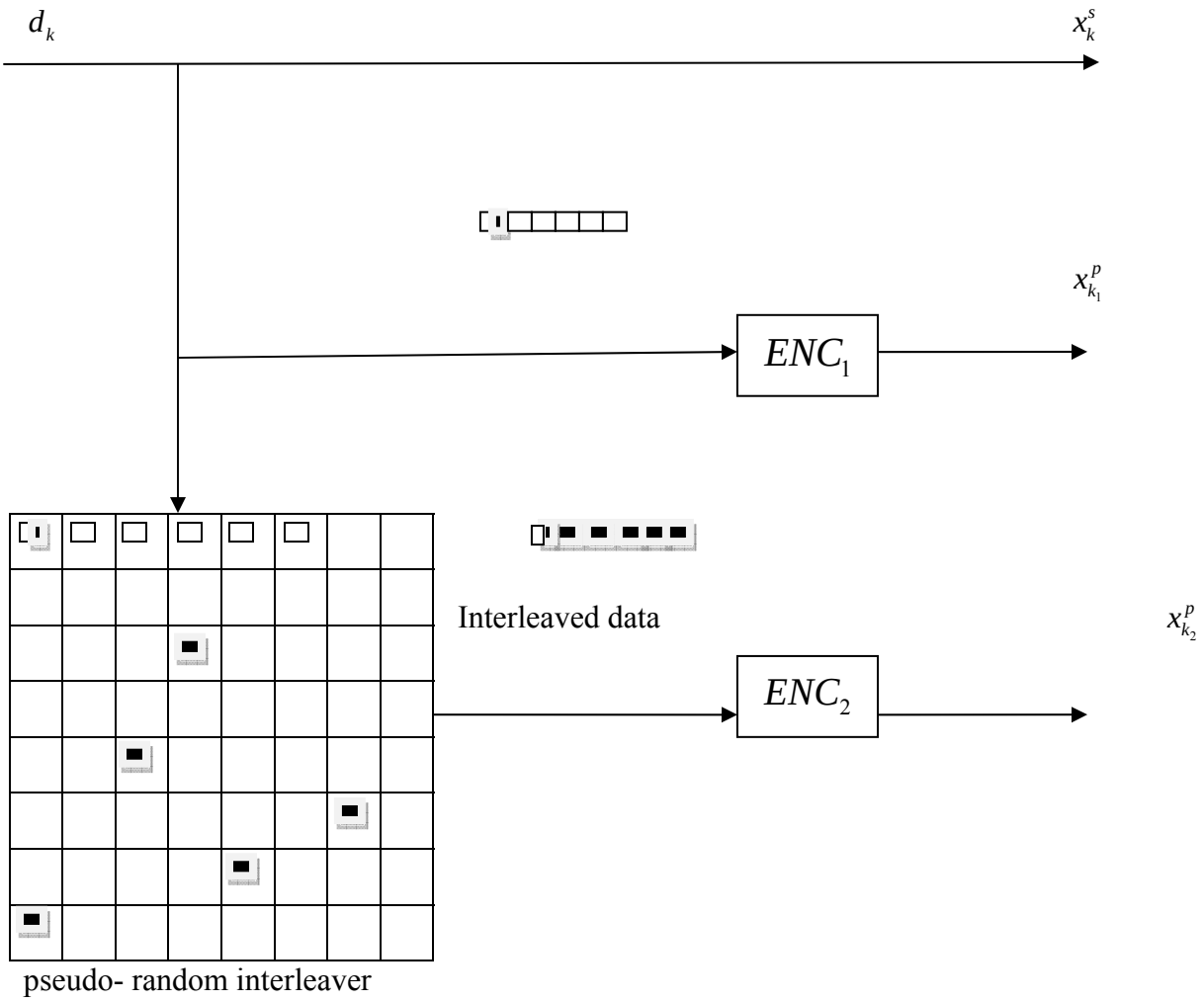


Figure 3.3 Pseudo- Random Interleaving in a Turbo Encoder

3.1.3 Puncturing

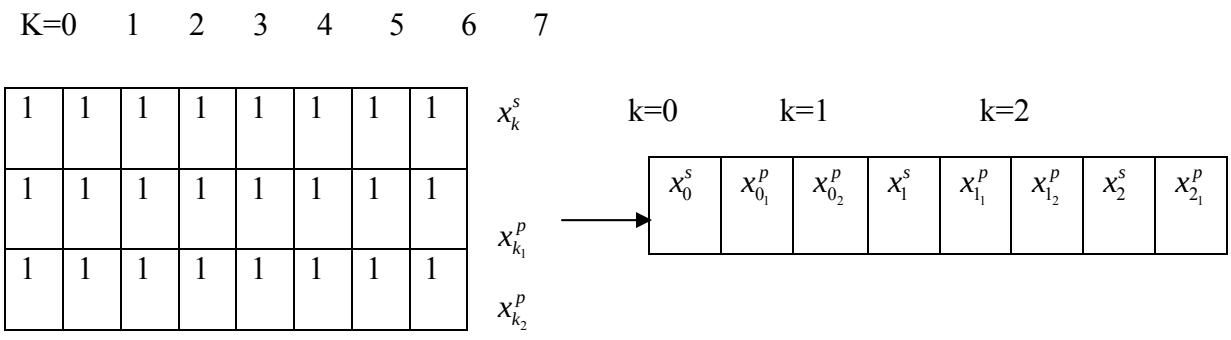
Different code rates are achieved by puncturing the parity bit sequences $x_{k_1}^p$ and $x_{k_2}^p$. Puncturing the data bit sequence x_k^s leads to a severe degradation in turbo code performance. Figure 3.4 illustrates the puncturing process. A number '1' in the tables represents a code bit that is included in the transmitted code bit sequence, and a number '0' represents a code bit that is excluded, or punctured. On the right of each table is the list of code bits which are included in the transmitted code sequence. In a), the code is unpunctured and is of code rate 1/3 whereas in

b), alternate parity bits from each component encoder are punctured at each time interval k. The result is a rate ½ code. In c), the code is more heavily punctured, to form a rate ¾ code.

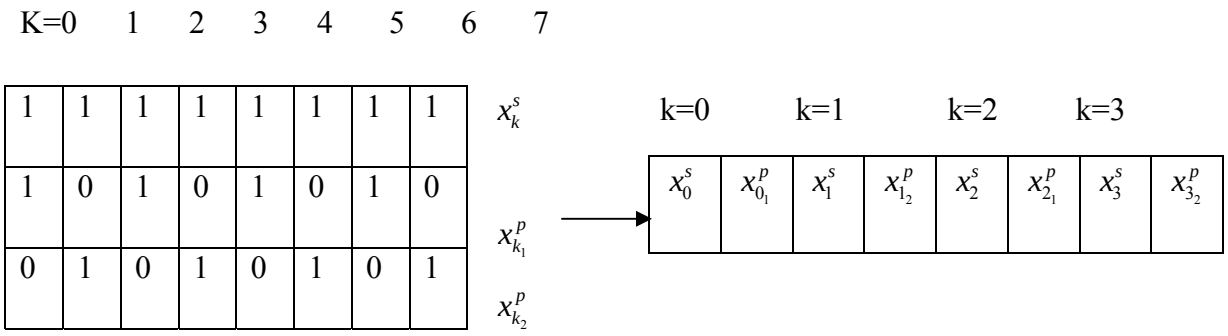
3.1.4 Termination

In contrast to block codes, convolutional codes do not have a fixed block length. Convolutional coding is a continuous process, and could span an entire message, rather than a small group of bits. Turbo codes, however, do have a fixed block length, determined by the length of the interleaver. Tail bits are usually appended to each block of data bits entering one or other of the component encoders, to return it to the all-zeroes state at the end of the trellis. This process is called termination, and allows the MAP algorithm to make assumptions about the start and end trellis states. This yields better BER performance. Termination of both component encoders is more difficult, because the terminating sequence for the first encoder is interleaved and may well not, by itself, terminate the second encoder. Interleaver designs have been devised [9] which interleave a terminating sequence in the first encoder into a terminating sequence in the second encoder. This tends to yield better BER performance than a single code terminating process and is another promising area for investigation.

a) Rate 1/3 Turbo Code



b) Rate 1/2 Turbo Code



c) Rate 1/3 Turbo Code

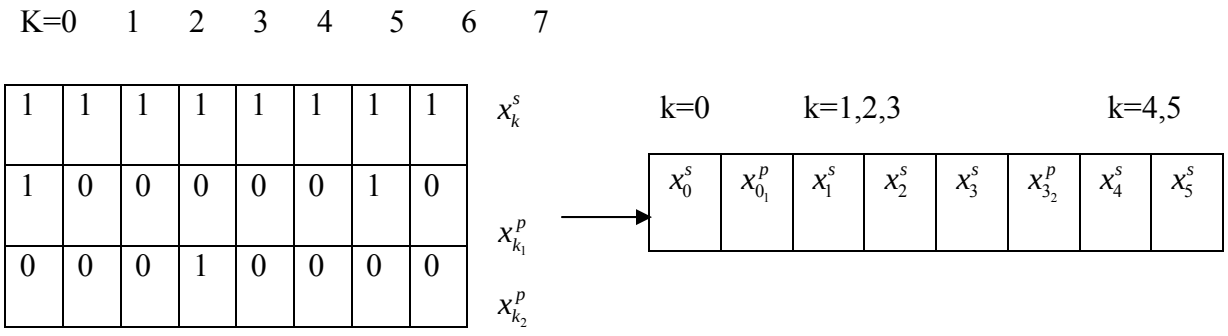


Figure 3.4 Puncture Patterns for Turbo Codes

3.2 Turbo Decoding

A block diagram of a turbo decoder is shown in Figure 3.5. The input to the turbo decoder is a sequence of received code values $R_k = \{y_k^s, y_k^p\}$ from the demodulator. The turbo decoder consists of two component decoders - DEC_1 to decode sequences from ENC_1, and DEC_2 to decode sequences from ENC_2. Each of these decoders is a Maximum A Posteriori (MAP) decoder. DEC_1 takes as its input the received sequence of systematic values y_k^s and the received sequence of parity values $y_{k_1}^p$ belonging to the first encoder ENC_1. The output of DEC_1 is a sequence of soft estimates EXT_1 of the transmitted data its d_k. EXT_1 is called extrinsic data,

in that it does not contain any information which was given to DEC_1 by DEC_2. This information is interleaved, and then passed to the second decoder DEC_2. The interleaver is identical to that in the encoder (Figure 3.1). DEC_2 takes as its input the (interleaved) systematic received values y_k^s and the sequence of received parity values $y_{k_2}^p$ from the second encoder ENC_2, along with the interleaved form of the extrinsic information EXT_1, provided by the first decoder. DEC_2 outputs a set of values, which, when de-interleaved using an inverse form of the interleaver, constitute soft estimates EXT_2 of the transmitted data sequence d_k. This extrinsic data, formed without the aid of parity bits from the first code, is fedback DEC_1. This procedure is repeated in a iterative manner. The iterative decoding process adds greatly to the BER performance of turbo codes. However, after several iterations, the two decoders' estimates of d_k will tend to converge. At this point, DEC_2 outputs a value $\Lambda(d_k)$; a log-likelihood representation of the estimate of d_k. This log likelihood value takes into account the probability of a transmitted '0' or '1' based on systematic information and parity information from both component codes. More negative values of $\Lambda(d_k)$ represent a strong likelihood that the transmitted bit was a '0' and more positive values represent a strong likelihood that a '1' was transmitted. $\Lambda(d_k)$ is de-interleaved so that its sequence coincides with that of the systematic and first parity streams. Then a simple threshold operation is performed on the result, to produce hard decision estimates, d_k, for the transmitted bits.

The decoding estimates EXT_1 and EXT_2, do not necessarily converge to a correct bit decision.

If a set of corrupted code bits form a pair of error sequences that neither of the decoders is able to correct, then EXT_1 and EXT_2 may either diverge, or converge to an incorrect soft value. In the next sections, we will look at the algorithms used in the turbo decoding process, within DEC_1 and DEC_2.

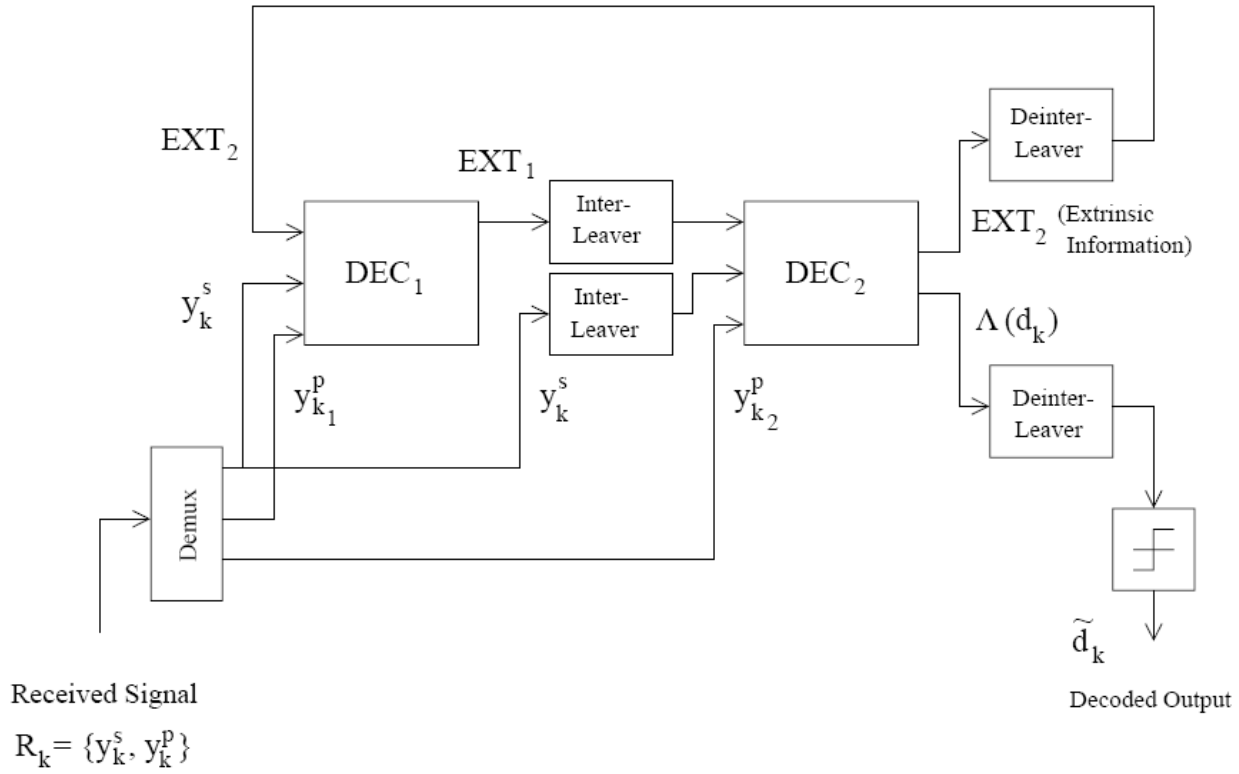


Figure 3.5 Turbo Decoder Structure.

3.3 The MAP Algorithm

We review now the decoding algorithms used within DEC_1 and DEC_2 to implement the soft-input, soft-output processing needed for iterative decoding. We begin with the modified Bahl, or Maximum A Posteriori (MAP), algorithm presented in Berrou et al.'s original paper [10].

3.3.1 The Need for a Soft Input/Soft Output Algorithm

Decoding of convolutional codes is most frequently achieved using the Viterbi algorithm [11], which makes use of a decoding trellis to record the estimated states of the

encoder at a set of time instants. The Viterbi algorithm works by rejecting the least likely path through the trellis at each node, and keeping the most likely one. The removal of unlikely paths leaves us, usually, with a single source path further back in the trellis. This path selection represents a 'hard' decision; on the transmitted sequence. The Viterbi decoder estimates a maximum likelihood sequence. Making hard decisions in this way, at an early point in the decoding process, represents a loss of valuable information. It is frequently advantageous to retain finely-graded probabilities, 'soft decisions', until all possible information has been extracted from the received signal values. The turbo decoding relies on passing information about individual transmitted bits from one decoding stage to the next. The interleaving of the received information sequence between decoders limits the usefulness of estimating maximum likelihood sequences. So, an algorithm is required that can output soft-decision maximum likelihood estimates on a bit-by-bit basis. The decoder should also be able to accept soft decision inputs from the previous iteration of the decoding process. Such a decoder is termed a Soft Input-Soft Output (SISO). Berrou and Glavieux used two such decoders in each stage of their turbo decoder. They implemented the decoders using a modified version of an SISO algorithm proposed by Bahl, Cocke, Jelinek and Raviv [12]. Their modified Bahl algorithm is commonly referred to as the Maximum A Posteriori or MAP algorithm, and achieves soft decision decoding on a bit-by-bit basis by making two passes of a decoding trellis, as opposed to one in the case of the Viterbi algorithm. One pass is made in the forward direction and one in the backward direction, as shown in Figure 3.6. $m = 0...(M-1)$ represents the states of each decoder (DEC_1 or DEC_2), where the total number of states $M = 2^{K-1}$ and K is the constraint length of the component codes ENC_1 and ENC_2. $\alpha_k(m)$ and $\beta_k(m)$ are the state probabilities that the component code is in state m at time instant k, in the forward and backward directions respectively.

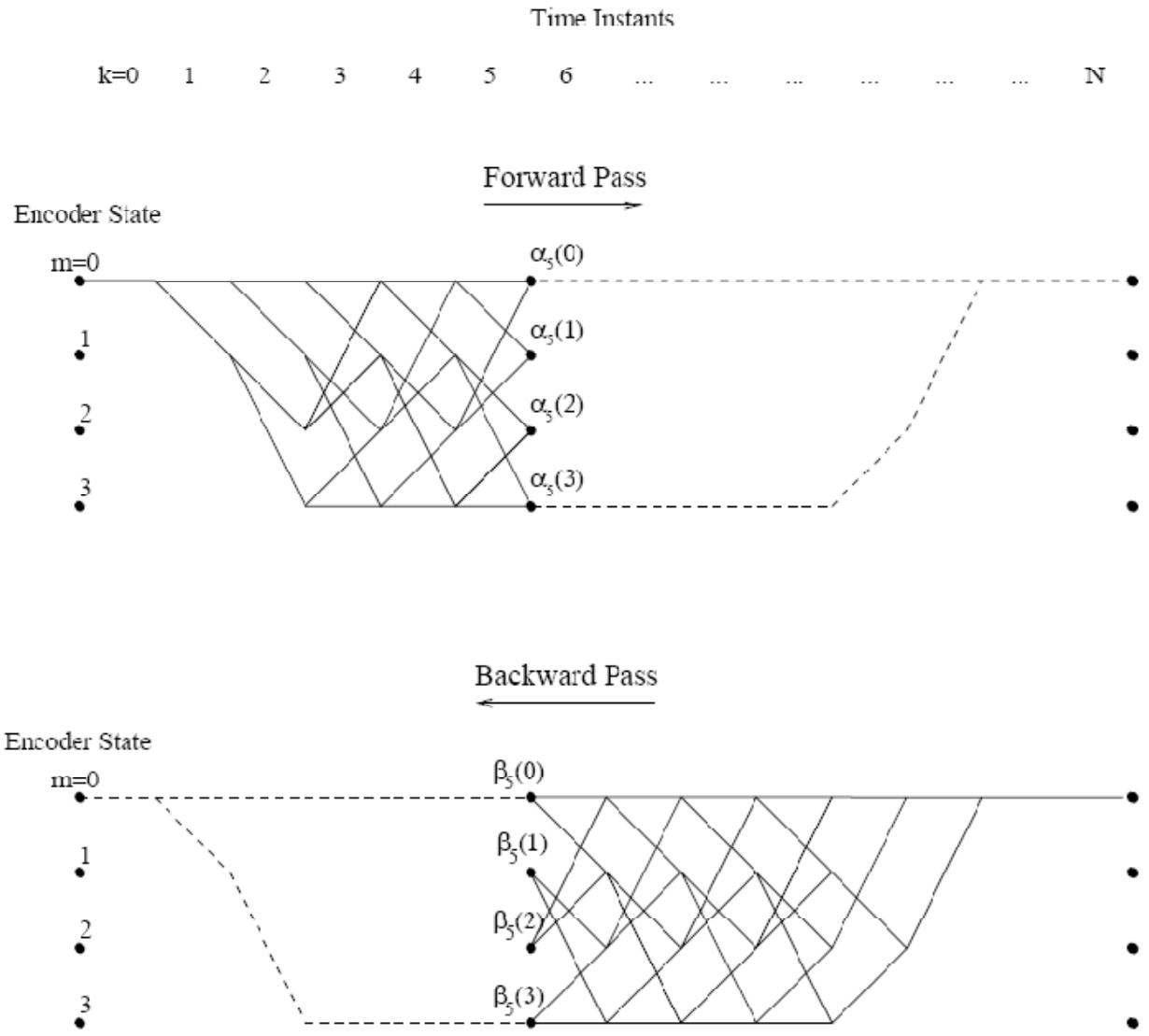


Figure3.6 Modified Bahl Algorithm.

3.3.2 Derivation of the MAP Algorithm

The MAP algorithm has been described in [13] and is repeated here:

Forward Pass - Calculation of α State Probabilities

Figure 3.7 illustrates the calculations made at each time interval k, for the simple four states RSC

code with trellis connectivity defined by the generator polynomialG={7,5} First, the trellis is traversed in the forward direction. At each node, the current state probability, α is calculated by multiplying the state probability at the previous node $\alpha_{k-1}(m')$ by the branch transition probability, $\gamma_{k-1}(m',m)$ (Equation 3.3), given the received code pair $R_k = \{y_k^s, y_k^p\}$. This is expressed as follows:

$$\alpha_k(m) = \frac{\sum_{m'}\sum_{i=0}^{1}\gamma_i((y_k^s, y_k^p), m', m)\alpha_{k-1}(m')}{\sum_{m}\sum_{m'}\sum_{i=0}^{1}\gamma_i((y_k^s, y_k^p), m', m)\alpha_{k-1}(m')} \qquad (3.1)$$

where m is the current state, m' is the previous state and i is the data bit ('0' or '1') corresponding to each branch exiting a node.

Backward Pass - Calculation of β State **Probabilities**

Then the trellis is traversed in the reverse direction. Again the probability of each branch being taken is calculated. The current state probability $\beta_k(m)$ is found by multiplying the probability of arriving in the previous state $\beta_{k+1}(m)$ by the probability of taking the current state transition $\gamma_{k+1}(m',m)$ given the current received values $R_{k+1} = \{y_{k+1}{}^s, y_{k+1}{}^p\}$. This is expressed as follows:

$$\beta_k(m) = \frac{\sum_{m'}\sum_{i=0}^{1}\gamma_i((y_{k+1}^s, y_{k+1}^p), m', m)\beta_{k+1}(m')}{\sum_{m}\sum_{m'}\sum_{i=0}^{1}\gamma_i((y_{k+1}^s, y_{k+1}^p), m', m)\alpha_k(m')} \qquad (3.2)$$

where the symbols have the same meaning as before, but β is the backward state probability. The transition probability for each branch between nodes is given by the equation:

$$\gamma_i((y_k^s, y_k^p), m', m) = p((y_k^s, y_k^p)\,|\,d_k = i, m', m).q(d_k = i\,|\,m', m).\pi(m\,|\,m') \qquad (3.3)$$

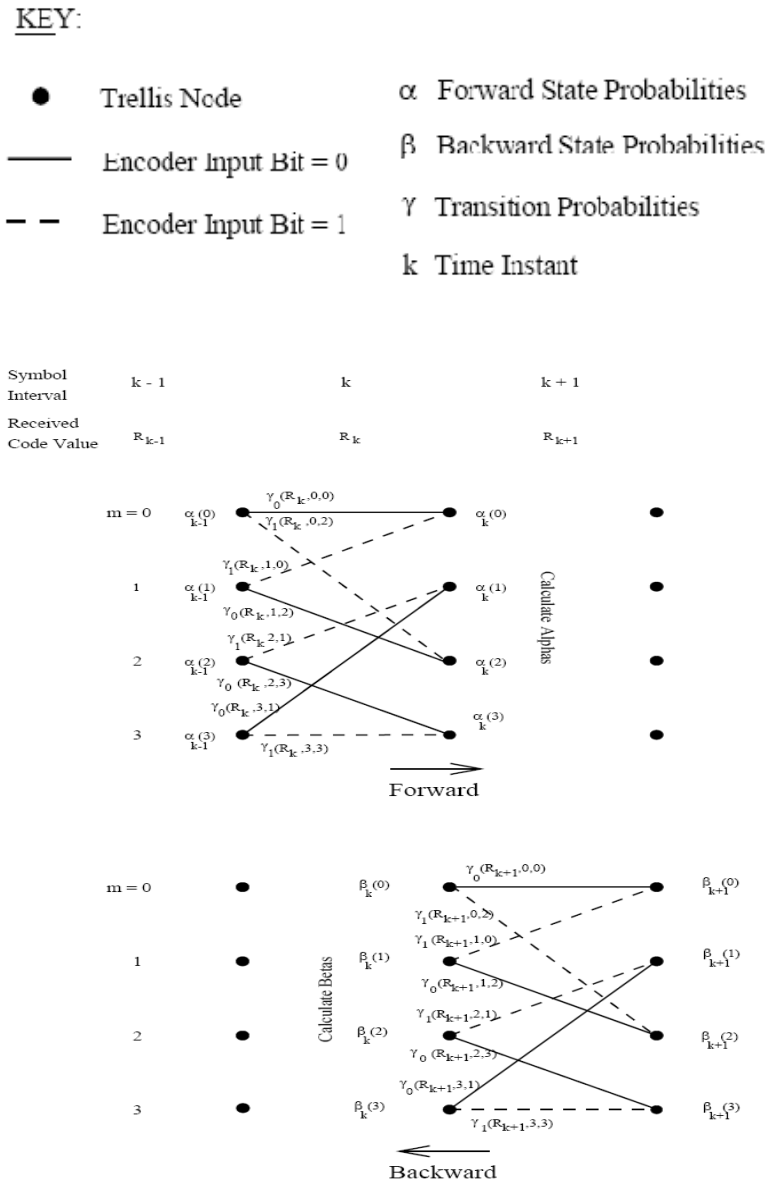


Figure 3.7 Calculation of State Probabilities in the Modified Bahl Algorithm

$p(...)$ is the transition probability of the channel; that is, the probability that a given received symbol $\{y_k^s, y_k^p\}$, will result when symbol $\{x_k^s, y_k^p\}$ is transmitted. This function is defined by the pdf of the channel; for example, a Gaussian pdf in the case of the AWGN channel. $q(...)$ is the probability that any branch from a node can be taken, given the previous state m',

current state m and the data bit i associated with the branch. q(…)is either '0' or '1', depending on the generator polynomial of the RSC encoder. $\pi(m|m')$ represents the a priori information which forms the input to each component MAP decoder from the other MAP decoder, within the iterative decoding process.

Calculation of Log Likelihood Probabilities, $\Lambda(d_k)$

Finally, the forward and backward probabilities at each time interval k of the trellis are used to provide a soft estimate of whether the transmitted data bit d_k was a '1' or a '0'. This process is illustrated in Figure 3.8, again for the RSC code with generator polynomial G={7,5}. The soft estimate is represented as a log likelihood ratio (LLR), $\Lambda(d_k)$ in Figure 3.5, as this is a convenient form for representing a probability which can have a wide dynamic range. It is calculated as follows:

$$\Lambda(d_k) = \ln \frac{\sum_m \sum_{m'} \gamma_1((y_k^s, y_k^p), m', m)\alpha_{k-1}(m')\beta_k(m)}{\sum_m \sum_{m'} \gamma_0((y_k^s, y_k^p), m', m)\alpha_{k-1}(m')\beta_k(m)} \qquad (3.4)$$

$\Lambda(d_k)$ represents the probability that the current data bit is a '0' (if $\Lambda(d_k)$ is negative) or a '1' (if $\Lambda(d_k)$ is positive). After a number of iterations, typically 8…18, the de-interleaved value of $\Lambda(d_k)$ from DEC_2 is converted to a hard decision estimate, d_k, of the transmitted data bit. This forms the output of the final turbo decoder stage. The MAP algorithm, as described in part so far, is optimal for estimating the maximum likelihood data sequence on a bit-by-bit basis. However, the MAP algorithm is not usually implemented for reasons of computational complexity. Specifically, although the soft output of the algorithm, $\Lambda(d_k)$, is represented as a log likelihood probability, intermediate values α, β and γ are represented as actual probabilities in the range0…1,($10^{-\infty}...10^{0}$) The dynamic range of these pure probabilities is high, which means that, in terms of implementation, memory needs are also high. Also, Equations 3.1 through 3.4 require a high number of multiplications - operations which are computationally intensive. To simplify the implementation, it is better to work with the logarithmic values of α, β , γ and Λ, rather than their actual probabilities. To save space here, the full implementation of a turbo decoder is

now described in terms of these simplified algorithms; the Max Log-MAP and Log- MAP algorithms.

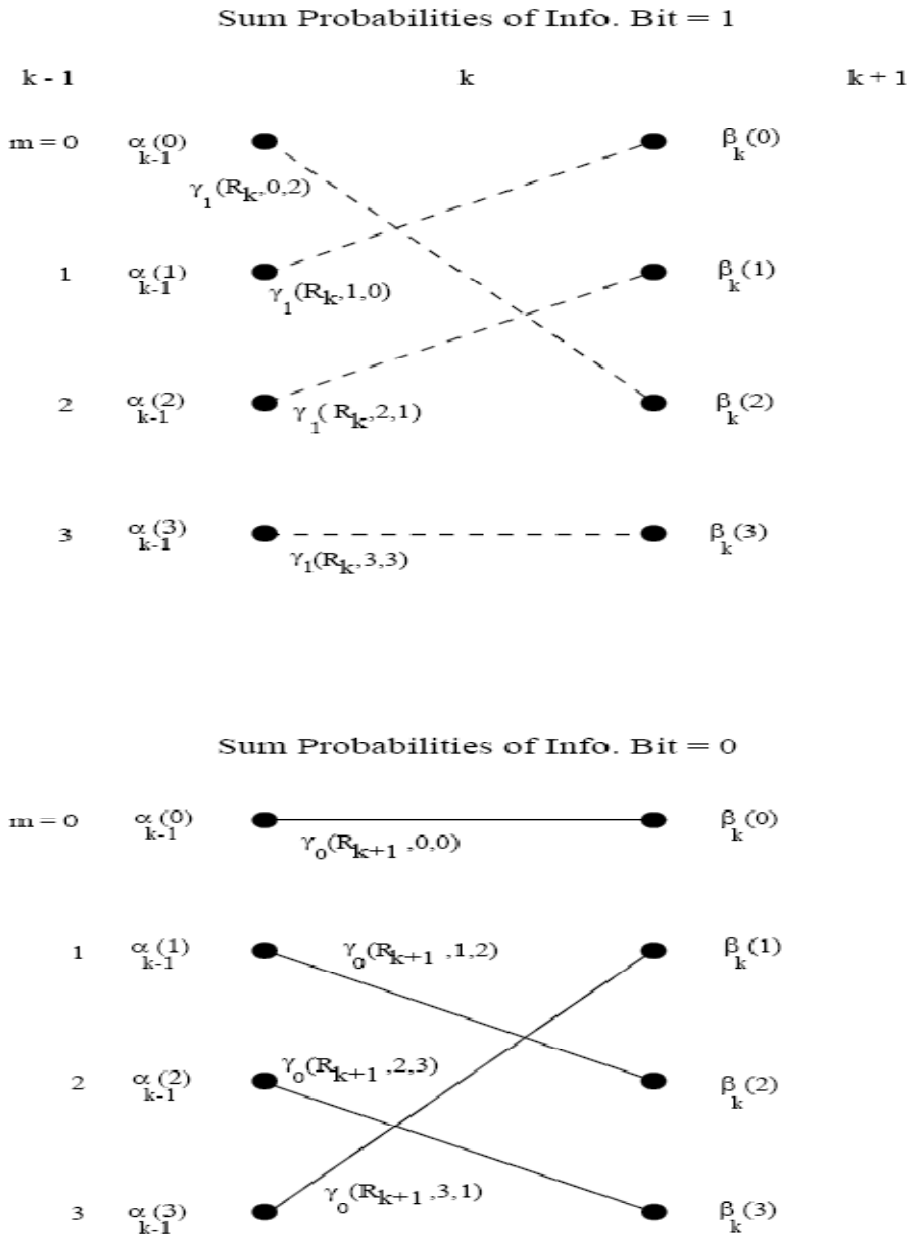


Figure 3.8 Calculation of Log Likelihood Ratio (LLR) for Each Time Instant, *k*

3.4 The Max Log-MAP Algorithm

As stated above, working in the logarithmic domain compacts the dynamic range of all the values we are working with. It also converts the multiplication operations in Equations 3.1 through 3.4 to additions. Let us take the logarithm of $\gamma_i((y_k^s, y_k^p), m', m)$ derived in Equation 3.3 and, for the AWGN channel, insert:

$$p(y_k^p \,|\, d_k = i, m', m) = \frac{1}{\sqrt{\pi N_0}} . \exp(-\frac{1}{N_0}[y_k^p - x_k^p(i, m', m)]^2)$$

$$p(y_k^p \,|\, d_k = i) = \frac{1}{\sqrt{\pi N_0}} . \exp(-\frac{1}{N_0}[y_k^s - x_k^s(i)]^2) \qquad (3.5)$$

Where x_k^s and x_k^p are the transmitted systematic and parity bits, respectively, corresponding to the current branch at time instant k, y_k^s and y_k^p are the received systematic and parity values, and N_0 is the noise power spectral density. Again m' and m represent the previous and current states in the trellis, respectively. We can now write the logarithm of Equation 3.3 for $q(.) = 1$ as:

$$\ln \gamma_i[(y_k^s, y_k^p), m', m] = \frac{2 y_k^s x_k^s(i)}{N_0} + \frac{2 y_k^p x_k^p(i)}{N_0} + \ln P_\tau(m \,|\, m') + K \qquad (3.6)$$

Note that evaluation of the branch transition γ assumes knowledge of the noise power spectral Density N_0. Research has been done into the performance of the MAP algorithm for noisy channel estimates [14], and shows that there is a spread of values around the actual value of N_0 which will provide good turbo decoding performance. Constant K cancels out in the calculation of $\alpha_k(S_k)$ and $\beta_k(S_k)$. For the logarithm of the α term we get:

$$\ln \alpha_k(m) = \ln\left[\sum_{m'} \sum_{i=0}^{1} \exp(\ln \gamma_i[(y_k^s, y_k^p), m', m]) + \ln \alpha_{k-1}(m'))\right]$$

$$- \ln\left[\sum_{m} \sum_{m'} \sum_{i=0}^{1} \exp(\ln \gamma_i[(y_k^s, y_k^p), m', m]) + \ln \alpha_{k-1}(m'))\right] \qquad (3.7)$$

So we have lost the multiplications from Equation 3.1, but we have gained exponential terms, which are themselves computationally intensive. However, we can approximate the sum of a series of log terms by considering only the maximum log value:

$$\ln\left[\exp(\delta_1)+\ldots+\exp(\delta_n)\right] \approx \max_{i\in(1..n)} \delta_i \tag{3.8}$$

The simplification provides us with the so-called Max Log-MAP algorithm. If we use this approximation in Equation 3.7, we can write $\bar{\alpha}_k(m)$, the log version of $\alpha_k(m)$, as:

$$\begin{aligned}\bar{\alpha}_k(m) \approx & \max_{(m',i)}\left\{\bar{\gamma}_i\left[(y_k^s, y_k^p), m', m\right]+\bar{\alpha}_{k-1}(m')\right\} \\ & -\max_{(m,m',i)}\left\{\bar{\gamma}_i\left[(y_k^s, y_k^p), m', m\right]+\bar{\alpha}_{k-1}(m')\right\}\end{aligned} \tag{3.9}$$

Similarly:

$$\begin{aligned}\bar{\beta}_k(m) \approx & \max_{(m',i)}\left\{\bar{\gamma}_i\left[(y_{k+1}^s, y_{k+1}^p), m, m'\right]+\bar{\beta}_{k+1}(m')\right\} \\ & -\max_{(m,m',i)}\left\{\bar{\gamma}_i\left[(y_{k+1}^s, y_{k+1}^p), m, m'\right]+\bar{\alpha}_k(m)\right\}\end{aligned} \tag{3.10}$$

The log-likelihood probability of each bit, $\Lambda(d_k)$, is then given approximately by:

$$\begin{aligned}\Lambda(d_k) \approx & \max_{(m,m')}\left\{\bar{\gamma}_1\left[(y_k^s, y_k^p), m', m\right]+\bar{\alpha}_{k-1}(m')+\bar{\beta}_k(m)\right\} \\ & -\max_{(m,m')}\left\{\bar{\gamma}_0\left[(y_k^s, y_k^p), m', m\right]+\bar{\alpha}_{k-1}(m')+\bar{\beta}_k(m)\right\}\end{aligned} \tag{3.11}$$

In accordance with the findings in [10], $\Lambda(d_k)$ must be divided into three components to allow iterative decoding to occur. These components are termed the extrinsic component (that part of $\Lambda(d_k)$ which each decoder derived independently of the other decoder), the a priori component (that part which was derived solely by the other decoder) and the systematic component (that part which derives solely from the systematic part of the received signal). In order to separate the components of $\Lambda(d_k)$ in this way, we must start by defining that part of the branch transition

probability γ which is derived purely from the parity component of the received signal. In log form, from Equation 3.3, we can write:

$$\bar{\gamma}_i'(y_k^p, m', m) = \ln p(y_k^p \,|\, d_k = i, m, m') + \ln q(d_k = i \,|\, m, m') \tag{3.12}$$

We can then insert this value into Equation 2.11, to obtain:

$$\begin{aligned}\Lambda(d_k) \approx & \left\{ \max_{(m,m')} \left(\bar{\gamma}_1 \left[(y_k^s, y_k^p), m', m \right] + \bar{\alpha}_{k-1}(m') + \bar{\beta}_k(m) \right) \right. \\ & \left. + \ln p(y_k^s \,|\, d_k = 1) + \ln P_\tau(d_k = 1) \right\} \\ & - \left\{ \max_{(m,m')} \left(\bar{\gamma}_0 \left[(y_k^s, y_k^p), m', m \right] + \bar{\alpha}_{k-1}(m') + \bar{\beta}_k(m) \right) \right. \\ & \left. + \ln p(y_k^s \,|\, d_k = 0) + \ln P_\tau(d_k = 0) \right\}\end{aligned} \tag{3.13}$$

Finally, this can be arranged as:

$$\begin{aligned}\Lambda(d_k) \approx & \max_{(m,m')} \left(\bar{\gamma}_1' \left[y_k^p, m', m \right] + \bar{\alpha}_{k-1}(m') + \bar{\beta}_k(m) \right) \\ & - \max_{(m,m')} \left(\bar{\gamma}_0' \left[y_k^p, m', m \right] + \bar{\alpha}_{k-1}(m') + \bar{\beta}_k(m) \right) + \frac{4 y_k^s}{N_0} + L(d_k)\end{aligned} \tag{3.14}$$

The first two terms form the extrinsic component, the third term is the systematic component and the fourth is the a priori component. It is the extrinsic component which is passed from one MAP decoder to the other, where, after interleaving, it forms the a priori input term of the new decoder. In Equation 3.6, the log probability of a branch transition was calculated. The unknown term so far in this is the a priori term. We can determine this term, $L(d_k)$, as follows. If $q(d_k = 1 \,|\, m, m') = 1$, then:

$$L(d_k) = \ln\left(\frac{P_\tau(d_k = 1)}{P_\tau(d_k = 0)} \right) = \ln\left[\frac{P_\tau(m \,|\, m')}{1 - P_\tau(m \,|\, m')} \right] \tag{3.15}$$

Hence $\ln P_\tau(m \,|\, m') = L(d_k) - \ln(1 + \exp(L(d_k)))$. An approximation for $P_\tau(m \,|\, m')$ can be found using Equation 3.8:

$$\ln P_\tau(m|m') \approx L(d_k) - \max[0, L(d_k)] \qquad (3.16)$$

Alternatively, if $q(d_k = 0|m, m') = 1$, then:

$$L(d_k) = \ln\left(\frac{P_\tau(d_k = 1)}{P_\tau(d_k = 0)}\right) = \ln\left[\frac{1 - P_\tau(m|m')}{P_\tau(m|m')}\right] \qquad (3.17)$$

Hence $P_\tau(m|m') = -\ln(1 + \exp(L(d_k)))$. Again, an approximation for $P_\tau(m|m')$ can be found using Equation 3.8:

$$\ln P_\tau(m|m') \approx -\max[0, L(d_k)] \qquad (3.18)$$

The Max Log-MAP algorithm described above allows us to greatly simplify the implementation of the MAP algorithm. However, the approximation in Equation 3.8 makes it sub-optimal and its BER performance is therefore poorer than that of the MAP algorithm. A solution to the problem is called the Log MAP algorithm.

3.5 The Log-MAP Algorithm

Equation 3.8 estimated $\ln\left(\exp(\delta_1) + ... + \exp(\delta_n)\right)$ by considering only the maximum exponential term. Robertson et al. used the Jacobian logarithm [16] to improve the approximation:

$$\ln\left(\exp(\delta_1) + \exp(\delta_2)\right) = \max(\delta_1, \delta_2) + \ln\left(1 + \exp(-|\delta_2 - \delta_1|)\right) = \max(\delta_1, \delta_2) + f_c(|\delta_2 - \delta_1|) \qquad (3.19)$$

Where $f_c(.)$ is a correction function. In fact, the expression $\ln\left(\exp(\delta_1) + ... + \exp(\delta_n)\right)$ can be computed exactly using the Jacobian logarithm, by finding the maximum term and applying the correction recursively through a sequence of δ values. The recursion is initialised with Equation 3.19. We assume $\delta = \ln\left(\exp(\delta_1) + ... + \exp(\delta_n)\right)$ is known. Hence:

$$\ln\left(\exp(\delta_1) + ... + \exp(\delta_n)\right) = \ln(\Delta + \exp(\delta_n)) \qquad (3.20)$$

Where

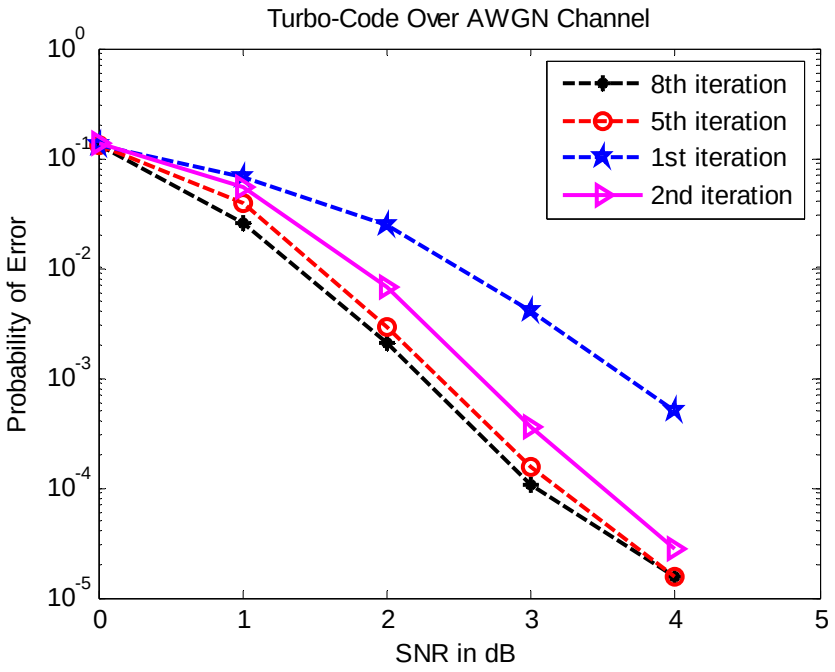
$$\begin{aligned}\Delta &= \exp(\delta_1) + \ldots + \exp(\delta_{n-1}) \\ &= \exp(\delta) \\ &= \max(\ln\Delta, \delta_n) + f_c(|\ln\Delta - \delta_n|) \\ &= \max(\delta, \delta_n) + f_c(|\delta - \delta_n|) \end{aligned} \quad (3.21)$$

When implementing the Log-MAP algorithm, all maximizations over two values are augmented by the correction function $f_c(.)$ as defined by Equation 3.19. By correcting the Max Log-MAP algorithm in this way, the precision of the MAP algorithm has been preserved. The incorporation of the correction function increases the complexity slightly relative to the Max Log-MAP algorithm. However, since $f_c(.)$ depends only on $|\delta_2 - \delta_1|$, this effect can be minimized by storing the correction values in a simple look-up table. It was found in that good performance can be achieved with as few as eight correction values.

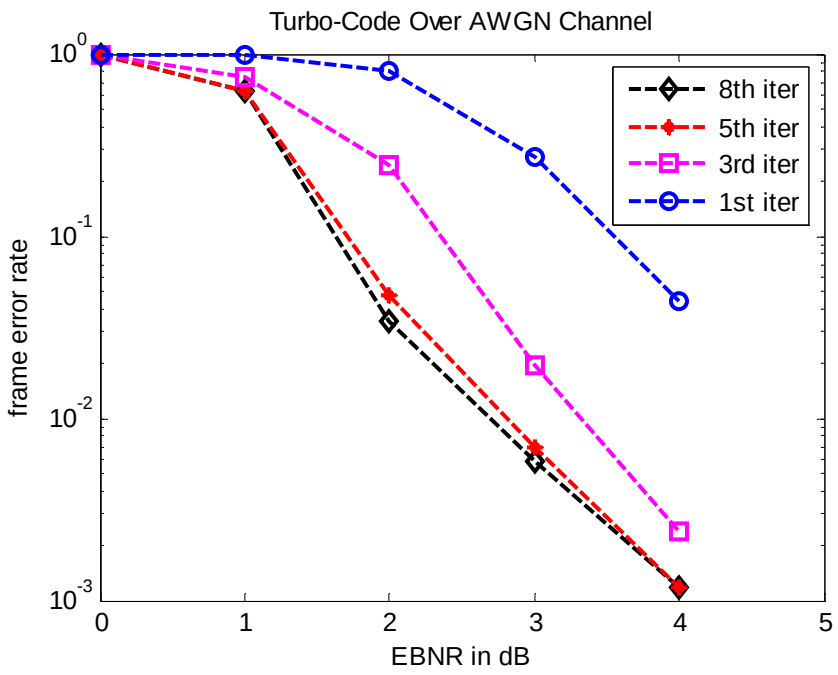
3.6 Turbo Code Error-Performance Example

Performance results using MATLAB simulations have been presented in for a rate 1/2, K =3 encoder implemented with generators **G1** = {1 1 1 } and **G2** = {1 0 1}, using parallel concatenation and a pseudo-random interleaver. The log-MAP algorithm was used with a data block length of 200 bits. After each frame the encoders are forced to the zero state. The corresponding termination tail - 4 information bits and 4 parity bits for each encoder, a total of 16 bits - is appended to the transmitted frame and used in the decoder. In principle the termination reduces the rate, but for large frames this has no practical influence. For low signal-to-noise ratios the main problem is lack of convergence in the iterated decoding process, resulting in frames with a large number of errors. In this region we are far from optimal decoding. This means that we may benefit from more iteration. As we see from the figure there is a considerable gain by going from 1 to 8 iterations, and with more iterations the performance might be even better. After 8 decoder iterations, the bit-error probability P_e was less than 10^{-4} at Eb/N0 = 4 dB. The error-performance improvement as a function of the number of decoder iterations is seen in Figure 3.9. Note that, as the Shannon limit of -1.6 dB is approached, the

required system bandwidth approaches infinity, and the capacity (code rate) approaches zero. Therefore, the Shannon limit represents an interesting theoretical bound, but it is not a practical goal. For binary modulation, several authors use $P_e = 10^{-5}$ and Eb/N0 = 0.2 dB as a pragmatic Shannon limit reference for a rate ½ code. Thus, with parallel concatenation of RSC convolutional codes and feedback decoding, the error performance of a turbo code at $P_e = 10^{-5}$ is within 0.5 dB of the (pragmatic) Shannon limit. For high signal-to-noise ratios the decoding is almost optimal, and the main problem is codewords of low weight. This region is usually referred to as the error-floor since the improvement for increasing signal-to-noise ratio is very small. In spite of the name it is not a true floor, since the BER and FER is constantly decreasing - although not nearly as fast as for the low signal-to-noise ratios. Notice that when the signal to noise ratio is high a small number of iterations is sufficient.



(a) Bit-error probability (BER) of turbo code



(b) frame error rate (FER) of turbo code

Figure 3.9 (a) Bit-error probability (BER) and (b) frame error rate (FER) as a function of Eb/N0 and multiple iterations.

CHAPTER 4

ANALYSIS OF TURBO CODED OFDM

The combination of turbo codes with the OFDM transmission is so called Turbo Coded OFDM (TC-OFDM) can yield significant improvements in terms of lower energy needed to transmit data, a very improvement issue in personal communication devices. Unfortunately, the majority of existing papers treating the TC-OFDM assumes that the channel estimation using only the pilot symbols is sufficient (or even that the channel is perfectly known). It is shown, however, that there is a large potential gain in using the iterative property of turbo decoders where soft bit estimates are used together with the known pilot symbols. The performance of such an iterative estimation scheme proves to be of particular interest when the channel is strongly frequency- and time- selective.

Similar to every other communications scheme, coding can be employed to improve the performance of overall system. Several coding schemes, such as block codes, convolutional codes and turbo codes have been investigated within OFDM systems. Moreover, the deep fades in the frequency response of the channel cause some groups of subcarriers to be less reliable than other groups and hence cause bit errors to occur in bursts rather than, independently. The burst errors can extensively degrade the performance of coding. To solve this problem, several ways are considered. The easiest method is to use stronger codes, in fact an interleaving technique along with coding can guarantee the independence among errors by affecting randomly scattered errors. We use turbo code to improve the performance . For analysis of the OFDM system, first we examine the uncoded situation and then we will analyze the effect of coding under turbo coded OFDM condition.

From simulation results, it is demonstrated that Max-Log-MAP algorithm, described in chapter 3, is promising in terms of performance and complexity. It is shown that performance is substantially improved by increasing the number of iterations. It is expected that an application of turbo coding to an OFDM system leads to considerable coding gain. Performance of a turbo-coded OFDM system is analyzed and simulated. **As** a modulation scheme, 64-QAM is employed. The performance is evaluated in terms of bit error probability. It has been confirmed that the turbo code achieves near Shannon-limit error correcting capability in

an AWGN channel. The utility of turbo coded OFDM extends to fading channels and also to channel with heavy impulsive noise. One of the example of such channel is low voltage power line channel. When we transmit signals over low voltage power line channel and if any electric device is turned on/off, it raises a surge over the channel (what we called impulsive noise). In such cases turbo coded OFDM will provide greater performance.

4.1 Power Line Communication

The communication flow of today is very high. Many applications are operating at high speed and a fixed connection is often preferred. If the power utilities could supply communication over the power-line to the costumers it could make a tremendous breakthrough in communications. Every household would be connected at any time and services being provided at real-time. Using the power-line as a communication medium could also be a cost-effective way compared to other systems because it uses an existing infrastructure, wires exists to every household connected to the power-line network. During the last years the use of Internet has increased. If it would be possible to supply this kind of network communication over the power-line, the utilities could also become communication providers, a rapidly growing market. On the contrary to power related applications, network communications require very high bit rates and in some cases real time responses are needed (such as video and TV). This complicates the design of a communication system but has been the focus of many researchers during the last years. Systems under trial exist today that claim a bit rate of 1 Mb/s, but most commercially available systems use low bit rates, about 10-100 kb/s, and provides low-demanding services such as meter reading. The power-line was initially designed to distribute power in an efficient way, hence it is not adapted for communication and advanced communication methods are needed. Today's research is mainly focused on increasing the bit rate to support high-speed network applications.

4.2 Power-Line Network

The power-line network is a large infrastructure covering most parts of the

inhabited areas. The power is typically generated by, e.g., a power plant and then transported on high-voltage (e.g., 400kV) cables to a medium-voltage substation, which transforms the voltage into, e.g., 10kV and distributes the power to a large number of low-voltage grids. Each low-voltage grid has one substation, which transforms the voltage into 400 V and delivers it to the connected households, via low-voltage lines. Typically several low-voltage lines are connected to the substation. Each low-voltage line consists of four wires, three phases and neutral. Coupled to the lines are cable-boxes, which are used to attach households to the grid. Many systems today use a topology with a central node (the substation) communicating with clients (the households). All communication is between the substation and the households and there is no communication between households. Because there is a physical connection between every two households it would also be possible to support this kind of communication. As an alternative, this communication could be routed through the substation.

The configuration with a central node and a set of clients may be compared with systems for mobile telephony, e.g., GSM. In GSM a base station (central node) is connected to all mobile phones (clients) within a restricted area. Thus the network topology is not unusual, but used in practice. Power-line communication is based on electrical signals, carrying information, propagating over the power-line. A communication channel is defined as the physical path between two communication nodes on which the communication signal is propagated. In a low-voltage grid there is a lot of different channels, in fact the links between the substation and each household are all different channels with different characteristics and qualities. If the communication system supports communication between households all these links are also different channels. The quality is estimated from how good the communication is on a channel. The quality is mostly a parameter of the noise level at the receiver and the attenuation of the electrical signal at different frequencies. The higher the noise level the harder it is to detect the received signal. If the signal gets attenuated on its way to the receiver it could also make the decision harder because the signal gets more hidden by the noise.

On the power-line the noise is generated from all loads connected to the grid. Also broadcast radio interferes with the communication. The attenuation is a parameter of the physical length of the channel and impedance mismatches in the grid. The power-line is often considered a harsh environment because of the time-variant characteristics of the noise and the attenuation, but this is also the case in most communication systems and only limits the

performance that can be achieved. Advanced communication systems exists today, designed to overcome the problems with such channels as, e.g., GSM.

4.3 The Power-Line as a Communication Channel

In this section we study the power-line as a communication channel and discuss the current research. We defined a channel as a physical path between a transmitter and a receiver. Note that a low-voltage grid consists of many channels each with its own characteristics and quality. Figure 4.1 below shows a digital communication system using the power-line as a communication channel. The transmitter is shown to the left and the receiver to the right. Important parameters of the communication system are the output impedance, Zt, of the transmitter and the input impedance, Zl, of the receiver. A coupling circuit is used to connect the communication system to the power-line. The purpose of the coupling circuits is two-fold. Firstly, it prevents the damaging 50 Hz signal, used for power distribution, to enter the equipment. Secondly, it certifies that the major part of the received/transmitted signal is within the frequency band used for communication. This increases the dynamic range of the receiver and makes sure the transmitter introduces no interfering signals on the channel.

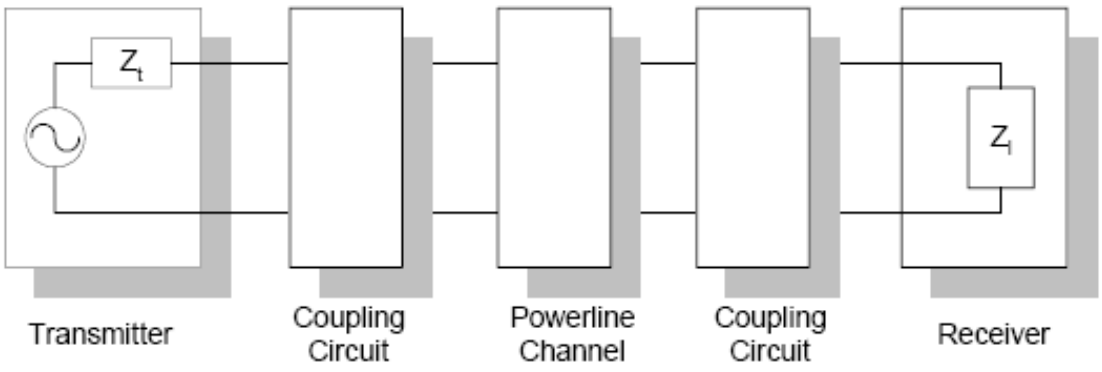


Figure 4.1 A digital communication channel for the powerline channel

a) Bandwidth Limitations

As described above the bandwidth is proportional to the bit rate, thus a large bandwidth is needed in order to communicate with high bit rates.

b) Radiation of the Transmitted Signal

When transmitting a signal on the power-line the signal is radiated in the air. One can think of the power-line as a huge antenna, receiving signals and transmitting signals. It is important that the signal radiated from the power-line does not interfere with other communication systems. When using the frequency interval 1-20 MHz for communication the radiation is extremely important because many other radio applications are assigned in this frequency interval. It is not appropriate for a system to interfere with, e.g., airplane navigation or broadcast systems. Recent research has studied this problem and tries to set up a maximum power level of transmission. It is important that this work is finished in the near future since it limits the use of this bandwidth and the development of communication systems for the power-line channel. When the cables are below ground the radiation is small. Instead it is the radiation from the households that makes the major contribution. Wires inside households are not shielded and thus radiate heavily. A solution might be to use filters to block the communication signal from entering the household.

c) Impedance Mismatches

Normally, at conventional communication, impedance matching is attempted, such as the use of 50 ohm cables and 50 ohm transceivers. The power-line network is not matched. The input (and output) impedance varies in time, with different loads and location. It can be as low as milli Ohms and as high as several thousands of Ohms and is especially low at the substation. Except the access impedance several other impedance mismatches might occur in the power-line channel. E.g., cable-boxes do not match the cables and hence the signal gets attenuated. Recent research has suggested the use of filters stabilizing the network. The cost of these filters might be high and they must be installed in every household and perhaps also in every cable-box.

d) **Signal-to-Noise-Ratio**

A key parameter when estimating the performance of a communication system, is the signal-to-noise power ratio, SNR:

$$SNR = \frac{\text{Re}\,ceived \quad power}{Noise \quad power}$$

This parameter is related to the performance of a communication system. The noise power on the power-line is a sum of many different disturbances. Loads connected to the grid, such as TV, computers and vacuum cleaners generate noise propagating over the power-line. Other communication systems might also disturb the communication, thus introducing noise at the receiver. When the signal is propagating from the transmitter to the receiver the signal gets attenuated. If the attenuation is very high the received power gets very low and might not be detected. The attenuation on the power-line has shown to be very high (up to 100 dB) and puts a restriction on the distance from the transmitter to the receiver. An option might be to use repeaters in the cable-boxes, thus increasing the communication length. The use of filters could improve the signal-to-noise ratio. If a filter is placed at each household blocking the noise generated indoors from entering the grid, the noise level in the grid will decrease, but the cost is a higher complexity.

e) **The Time-variant Behavior of the Grid**

A problem with the power-line channel is the time-variance of the impairments. The noise level and the attenuation depend partly on the set of connected loads, which varies in time. A channel which is time-variant complicates the design of a communication system. At some time instants the communication might work well but at other times a strong noise source could be inherent on the channel, thus blocking the communication. To solve this, a possible solution is to let the communication system adapt to the channel. At any time the characteristics of the channel are estimated, e.g., through measurements, and the effect is evaluated to make a better decision. The cost of this is higher complexity.

f) **A Channel Model of the Power-Line Communication Channel**

In the previous section we have seen some impairments that reduce the performance of a power-line communication system:

• Impedance mismatches at the transmitter

• Channel attenuation

• Disturbances (noise)

• Impedance mismatches at the receiver

• Time-variations of the impairments

Figure 4.2 shows a model of the power-line channel with the parameters above. All impairments except the noise are shown as time-variant linear filters, characterized by its frequency response.

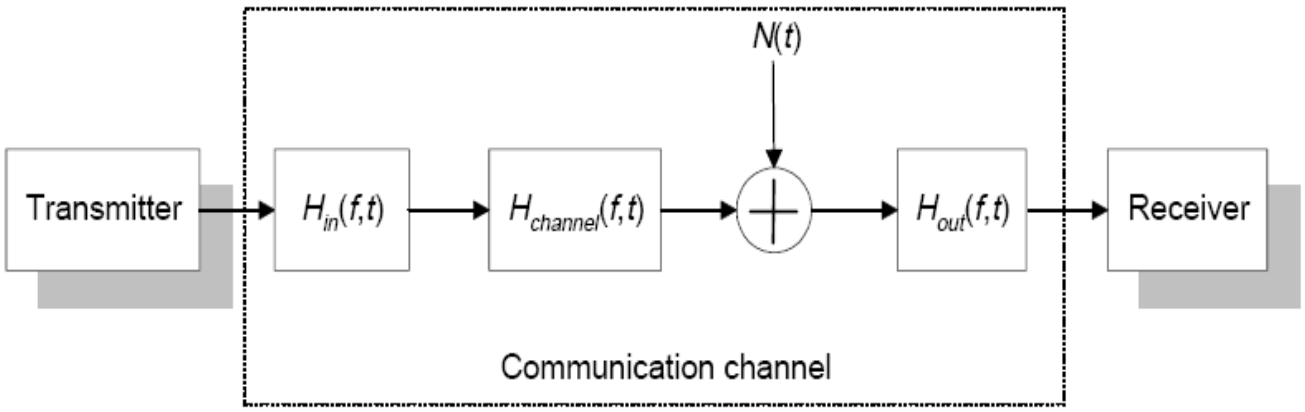


Figure 4.2 Impairments present on the powerline

The disturbance is shown as an additive interfering random process. All the impairments above can be incorporated into a single filter model, shown in Figure 4.3, consisting of a time-variant filter and additive noise.

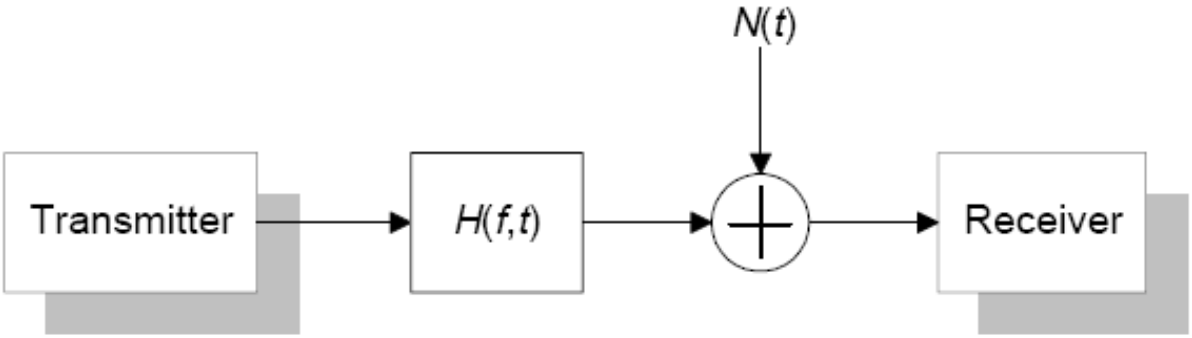


Figure 4.3 A simplified model of the poweline channel

Despite of its simple form this model captures a whole range of properties essential to communication system design and to the corresponding performance. The transfer function and the noise can either be estimated through measurements or derived by theoretical analysis. Still a lot of measurements and modeling are needed to get a thorough understanding of the grid due to the variance of the characteristics.

4.4 **Noise in Powerline Channel**

Five general classes of noise can be defined in the powerline channel.

1) Colored background noise with a relatively low power spectral density (PSD), varying with frequency. This type of noise is mainly caused by summation of numerous noise sources with low power.

2) Narrow-band noise, mostly sinusoidal signals, with modulated amplitudes caused by ingress of broadcast stations. The level is generally varying with daytime.

3) Periodic impulsive noise asynchronous to the mains frequency with a repetition rate between 50 and 200 kHz, with a discrete line spectrum spaced according to the impulse repetition rate. This type of noise is mostly caused by switched power supplies.

4) Periodic impulsive noise synchronous to the mains frequency with a repetition rate of 50 or 100 Hz (in Europe). The impulses are of short duration (some microseconds) and have a PSD decreasing with frequency. This type of noise is caused by power supplies, mainly by switching of rectifier diodes, which occurs synchronously with the mains cycle.

5) Asynchronous impulsive noise is caused by switching transients in the network. The impulses have durations of some microseconds up to a few milliseconds with random occurrence. The PSD of this type of noise can reach values of more than 50 dB above the background noise.

Two types of modeling for the noise on PLC channel

1. In the first type background noise is modeled as the white Gaussian process and arrival of impulsive noise follows the poisson pdf. We assume the background noise as additive white Gaussian noise (AWGN) w_k with mean zero and variance σ_w^2, and the impulsive noise i_k is given by:

$$i_k = b_k * g_k \qquad (4.1)$$

Where b_k is the Poisson process which is the arrival of the impulsive noise and g_k is the white Gaussian process with mean zero and variance σ_i^2. This model can be physically thought of as each transmitted data symbol being hit independently by an impulsive noise b_k with a probability distribution and with a random amplitude g_k.

Let a_k be the transmitted signal, then the received signal can be expressed as

$$r_k = a_k + n_k \qquad (4.2)$$

Where n_k is the noise given by

$$n_k = w_k + i_k = w_k + b_k * g_k \tag{4.3}$$

Since the power-line network is not designed for communications purposes, the channel exhibits an unfavorable frequency selective transfer function. Furthermore, this channel is distorted by impulsive noise and by severe narrowband interference. Unlike many other communication channels, power-line channel does not represent an additive white Gaussian noise (AWGN) environment. Noise in LV power-line is characterized within two categories: background and impulsive noise [18, 19]. Many electric appliances frequently cause man-made electromagnetic noise on power-line channels. Such man-made noise produces an impulsive distortion on channel causing a burst of noise. A large impulse often causes the entire transmitted symbol to be corrupted and it can be devastating to the overall system performance. Enhancement techniques, such as coding can help an OFDM system to tackle impulsive noise burst. Here, the bit error rate (BER) performance of the OFDM system under impulsive noise and frequency fading is analyzed. The modeling of impulsive noise over broadband power-line channels has been a challenge for researchers since early 1980s. Several different modeling methods are available. Zimmerman and Dostert in [19] propose one of the first modeling methods for this kind of noise at high frequencies. In their method, they use partitioned Markov models to characterize the nature of the impulsive noise. Here we use simplified Markov model to represent the burst errors caused by the impulsive noise.

A Marcov process X (t) is a random process whose past has no influence on the future if its present is specified; that is, if $t_n > t_{n-1}$, then

$$P\left[X(t_n) \le x_n \left| X(t), \quad t \le t_{n-1} \right.\right] = P\left[X(t_n) \le x_n \left| X(t_{n-1}) \right.\right] \tag{4.4}$$

From this definition, it follows that if $t_1 < t_2 < ... < t_n$, then

$$P\left[X(t_n) \le x_n \left| X(t_{n-1}), X(t_{n-2}), ...X(t_1) \right.\right] = P\left[X(t_n) \le x_n \left| X(t_{n-1}) \right.\right] \tag{4.5}$$

And Guass-Marcov process X (t) is a marcov process whose probability density function is guassian [3]. The simplest method for generating a Marcov process is by means of the simple recursive formula

$$X_n = \rho X_{n-1} + w_n \tag{4.6}$$

When w_n is a zero-mean i.i.d. random variables and ρ is a parameter that determines the degree of correlation between X_n and X_{n-1}; that is,

$$E(X_n X_{n-1}) = \rho E(X_{n-1}^2) = \rho \sigma_{n-1}^2 \tag{4.7}$$

If the sequence { w_n } is Guassian, then the resulting process X(t) is a Guass-Marcov process.

2. A statistical analysis method is employed. Although several PDFs have been suggested in the literature, it is found that the probability distribution of the time-domain noise amplitudes resembles the Nakagami-m distribution function. The Nakagami-m PDF can be written as

$$p(r) = \frac{2}{\Gamma(m)} \left(\frac{m}{\Omega} \right)^m r^{2m-1} e^{\frac{-mr^2}{\Omega}} \tag{4.8}$$

where r is the random variable, $p(r)$ is the probability of the corresponding random variable, $\Gamma(m)$ is the Gamma function, m is defined as the ratio of moments, and Ω is the mean power of the random variable.

CHAPTER 5

SIMULATION RESULTS

This chapter describes the simulation model used in this thesis and presents the simulation results. Based on the discussion in Chapters 2, 3 and 4, the software implementation of the model is straightforward. The software tool that was used to code was Matlab. Matlab is a powerful mathematical tool that allows us to model complicated systems with ease. Simulations were performed to study the system performance with turbo code under a variety of channel conditions.

5.1 Simulation Model

Since the main goal of this thesis was to simulate the COFDM system by utilizing turbo code. The block diagram of the entire system is shown in Figure 5.1.

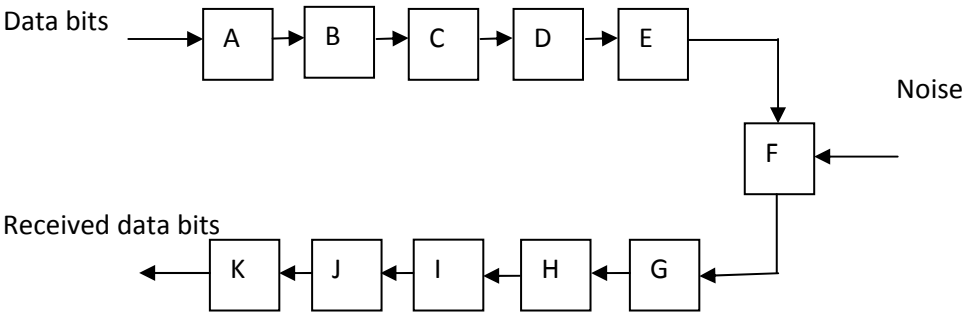


Figure 5.1 Simulation model of turbo coded OFDM

Here A = turbo encoder, B = QAM/QPSK modulation, C = serial to parallel convertor, D = IFFT, E = parallel to serial convertor, F = channel with noise, G = serial to parallel convertor, H = FFT, I = parallel to serial convertor, J = QAM/QPSK demodulation and K = turbo decoder.

5.2 Simulation Parameters

During the simulations, in order to compare the results, the same random message were generated. For that we use randint function in MATLAB.

Parameters	**Values**
Digital Modulation	QPSK, 16- QAM, 64- QAM
Turbo code rates	1/2
SISO Decoder	Log-MAP
Code Generator	{111, 101}
Interleaver Size	1 × 100

Table 2: Simulation parameters

We measured the performance of the turbo coded OFDM through MATLAB simulation. The simulation follows the procedure listed below:

1. Generate the information bits randomly.

2. Encode the information bits using a turbo encoder with the specified generator matrix.

3. Use QPSK or different QAM modulation to convert the binary bits, 0 and 1, into complex signals (before these modulation use zero padding)

4. Performed serial to parallel conversion.

5. Use IFFT to generate OFDM signals, zero padding is being done before IFFT.

6. Use parallel to serial convertor to transmit signal serially.

7. Introduce noise to simulate channel errors. We assume that the signals are transmitted over an AWGN (Additive White Gaussian Noise) channel. The noise is modeled as a Gaussian random variable with zero mean and variance σ^2 . The variance of the noise is obtained as

$$\sigma^2 = \frac{1}{2 * E_b / N0} ,$$

We use a built-in MATLAB function randn to generate a sequence of normally distributed random numbers, where randn has zero mean and 1 variance. Thus the received signal at the decoder side is:

X' = noisy(X)

Where noisy (X) is the signal corrupted by noise.

8. At the receiver side, perform reverse operations to decode the received sequence.

9. Count the number of erroneous bits by comparing the decoded bit sequence with the original one.

10. Calculate the BER and plot it.

All the simulations are done to achieve a desired BER 10^{-3}. For simulation results, two noise models were considered: the AWGN and the time-Markov model. Both models are utilized by the parameters defined above. The BER performance of TCOFDM system is compared with the respective uncoded system under the AWGN channel. No other channel codes are considered in this paper as the iterative decoding scheme easily outperforms conventional codes, or in other words non-iterative decoded codes.

As mentioned before, bursty errors deteriorate the performance of the any communications system. The burst errors can happen either by impulsive noise or by deep frequency fades. Powerline channels suffer from both of these deficiencies. Figure 5.2 shows the performance of uncoded OFDM system with AWGN and impulsive noise (which is modeled as marcov noise).

In this figure it is shown that, for the required BER 10^{-3} AWGN channel gives better performance as compared with marcov channel. AWGN gives a gain of approximately 22 db over marcov channel. We observe a little gain at lower SNR between 0 to <10dB, and more gain at higher SNR <40dB.

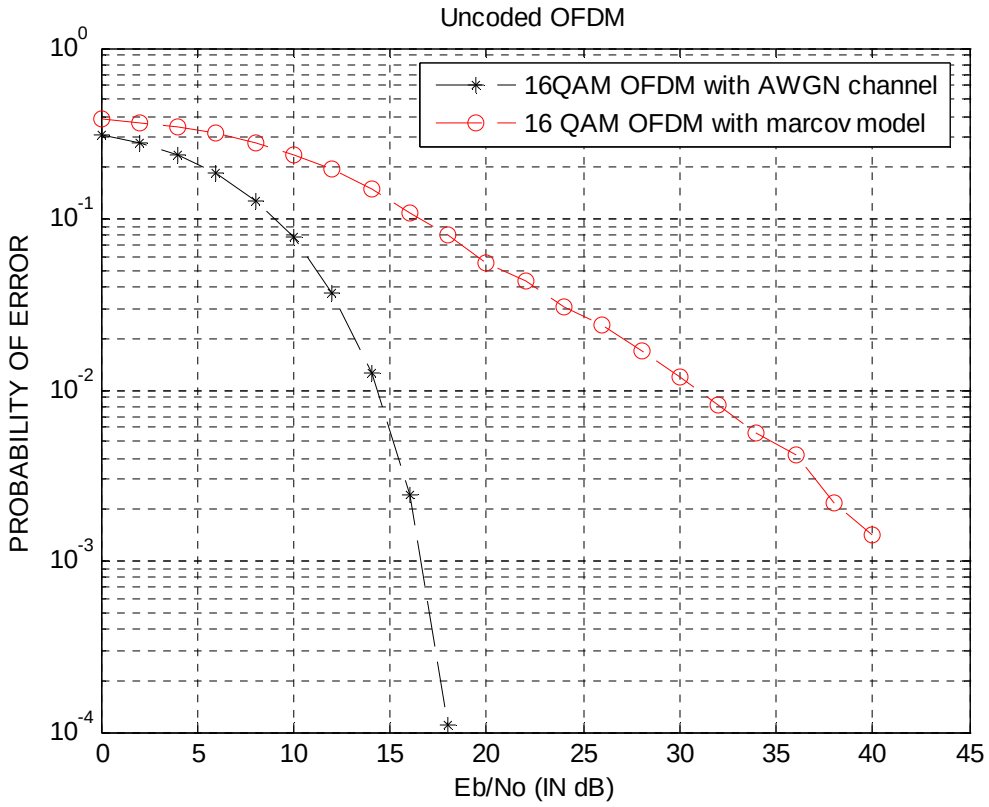


Figure 5.2 Performance of uncoded OFDM system in channel with impulsive noise.

To improve the performance of this system FEC code can be used. Convolutional code is a good example of FEC code. It is shown in figure 5.3 that convolutional coding in OFDM can give performance improvement of some 5 db on AWGN channel over the uncoded OFDM system at required BER. Here the convolutional codes are based on the rate ½, constraint length 3 and (7, 5) generators matrix convolutional code.

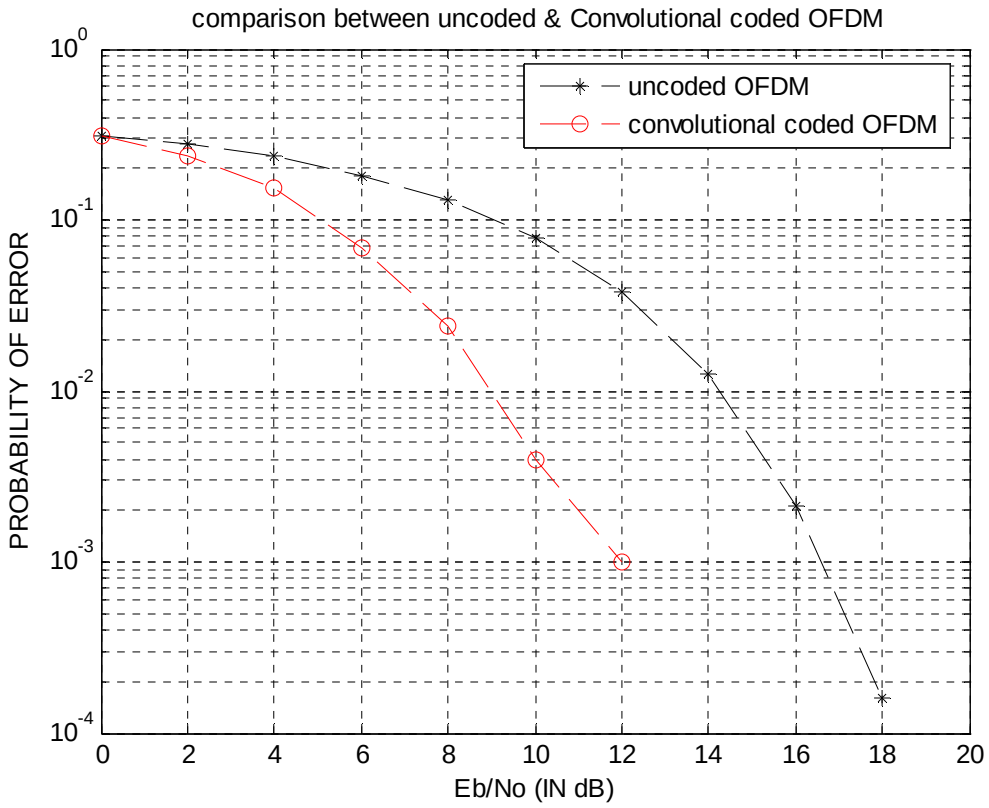


Figure 5.3 Performance analysis between Uncoded and convolutional coded OFDM system

Further improvement in the performance can be obtained by applying turbo coding instead of convolutional code[15]. As described in chapter 3, turbo code gives better performance at low SNR. The BER performance of TCOFDM system is compared with the respective uncoded system under the fading AWGN channel. No other channel codes are considered in this thesis as the iterative decoding scheme easily outperforms conventional codes. We have successfully simulated the turbo codes with polynomial generators, (1, 15 /13)8 and (1, 5 / 7)8 which are iteratively decoded by Log- MAP for a number of decoding iterations. We have simulated the polynomial (1, 5 / 7)8 as a reference. The simulated results are shown in Figure 5.4. From the results, we observe that both turbo codes (1, 15 /13)8 and (1, 5/7)8 give considerably good BER performance. As we compare (1, 15 /13)8 codes with (1, 5 / 7)8, we observe a little gain at higher SNR between 8 to <10dB. The overall performance is considered very well in operation under fading channel which is also efficient in terms of power consumption as compared to the uncoded system.

Code generator	SNR for BER10^{-2}	SNR for BER 10^{-3}
(1,5/7)	~ 7.2 db	~ 8.9 db
(1,15/13)	~ 6.8 db	~ 8.3 db

Table 3: Comparison of SNR for different code generators

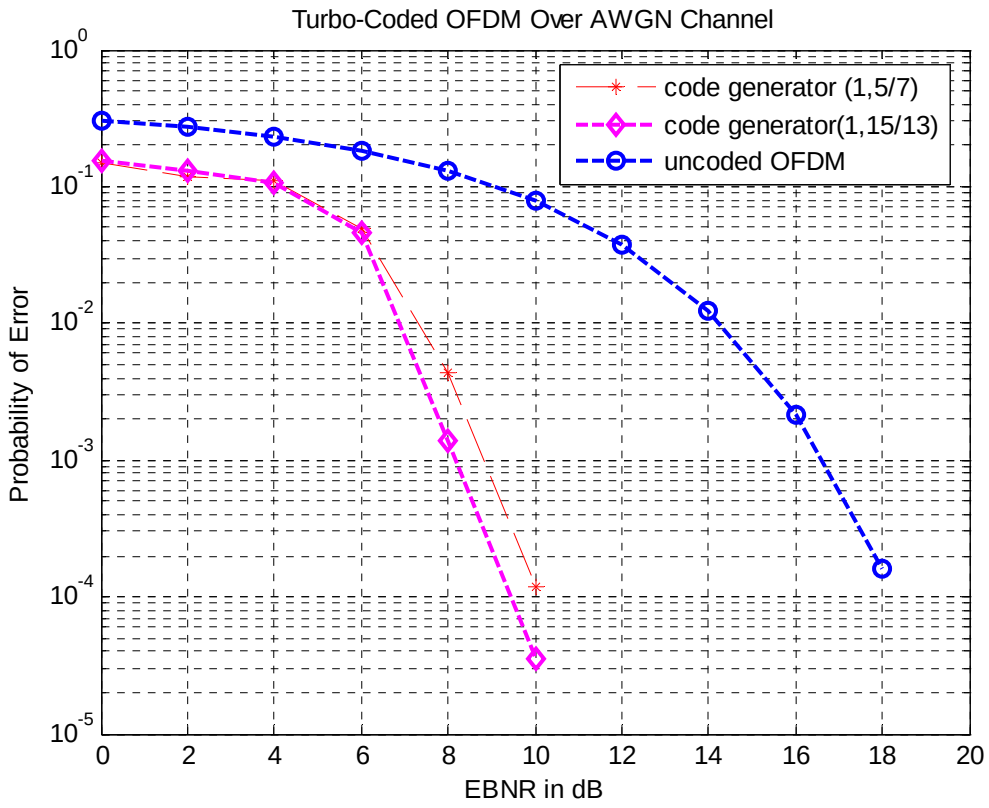


Figure 5.4 Performance of turbo coded OFDM with different generators polynomial

In figure 5.5 it is shown that turbo-codes of length 200, with QPSK modulation, can give performance improvements of some 8 dB on AWGN channel, over the conventional convolutional codes of the same code rate. Results are shown in table

Type of Coded OFDM	Gain at 10^{-2} over Uncoded OFDM	Gain at 10^{-3} over Uncoded OFDM
Convolutional coded OFDM	4.8 db	5.2 db
16 QAM TCOFDM	6.5 db	7.5 db
QPSK TCOFDM	11.5 db	13 db

Table 4: Comparison of gain at BER 10^{-2} and BER 10^{-3} for turbo coded OFDM and convolutional coded OFDM over uncoded OFDM

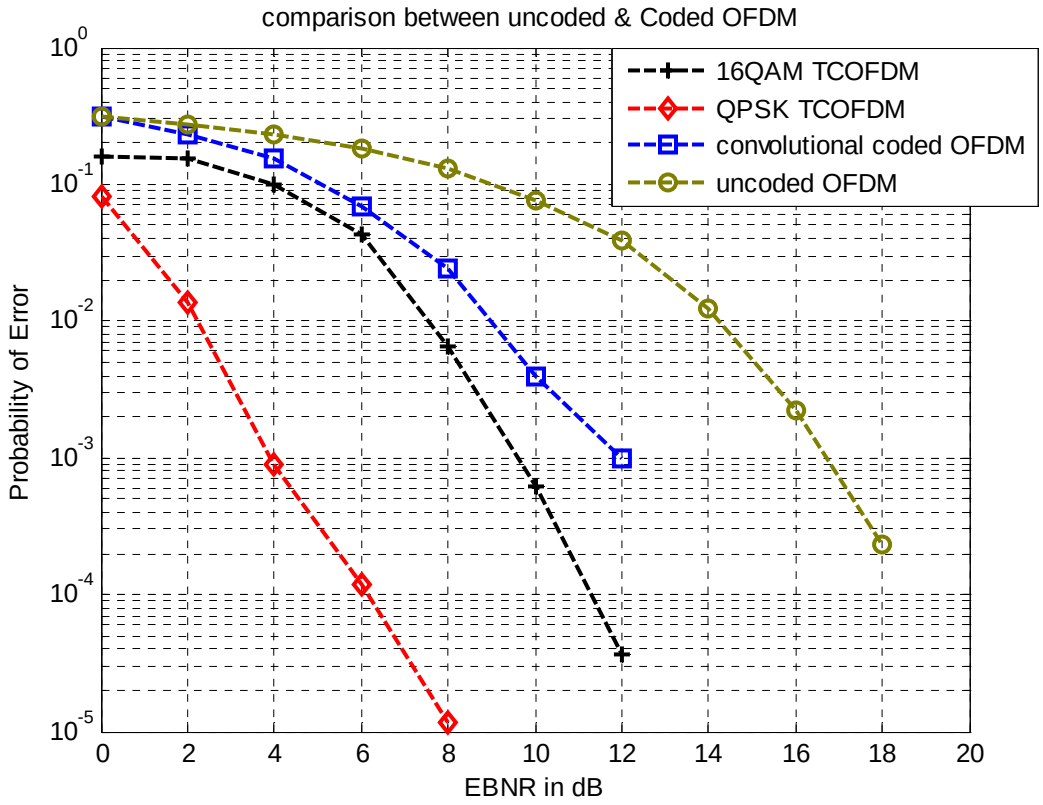


Figure 5.5 Different coded and uncoded OFDM system analysis over AWGN channel

Broadband communications for indoor power-line networks with impulsive noise using orthogonal frequency division multiplexing (OFDM) is considered. This channel is distorted by impulsive noise. A large impulse often causes the entire transmitted symbol to be corrupted and it can be devastating to the overall system performance. Here simulation is done

on two type of impulsive noise model. In which first, marcov [21], is described in chapter 3 and second asynchronous impulsive noise is modeled by the fact that the time domain impulse noise, spreads over the whole carriers by the DFT operation in the receiver. Asynchronous impulsive noise is caused by switching transients in the network. Especially influence of the impulse noise whose amplitude is large is very severe. For simulation we generated a random impulsive noise. Simulation also shows the performance of marcov noise. Simulation results are shown in figure 5.6 and 5.7.

Type of noise in TCOFDM	Gain at 10^{-2} over Uncoded OFDM	Gain at 10^{-3} over Uncoded OFDM
AWGN	7.5 db	7.8 db
Impulsive (marcov)	5.0 db	2.4 db

Table 5: Performance of turbo coded OFDM in noisy channel

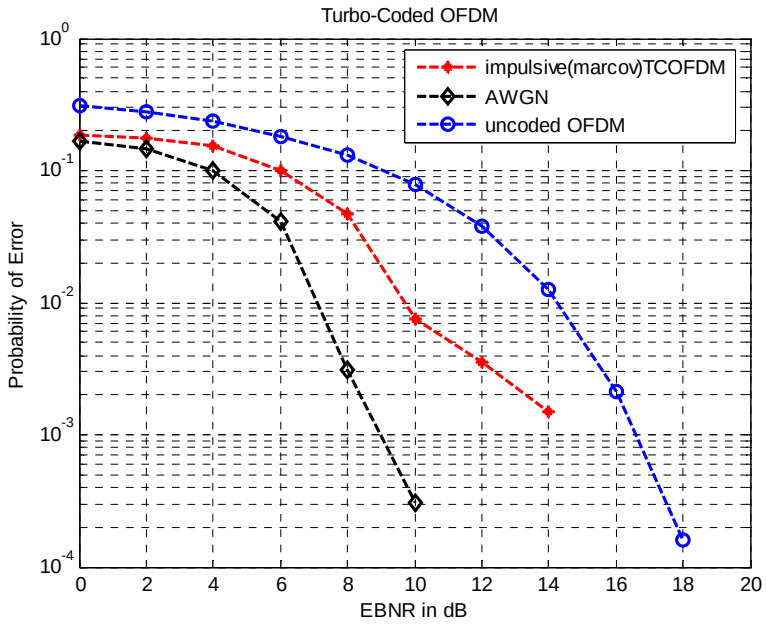


Figure 5.6 Performance of turbo coded OFDM over AWGN and impulsive noise (marcov) channel

Figure 5.7 shows the influence of asynchronous impulsive noise on turbo coded OFDM system. As shown in figure, the influence of the impulse noise is distributed over the whole carriers by applying DFT in the receiver. Therefore, data symbol on each sub-carrier is degraded under the case where large impulse noise is added or many impulses are added to the OFDM symbol whereas small impulse noise affect less data symbols on sub-carrier, hence less affective. If there are so many symbol errors in the symbols, then whole OFDM symbol will be lost.

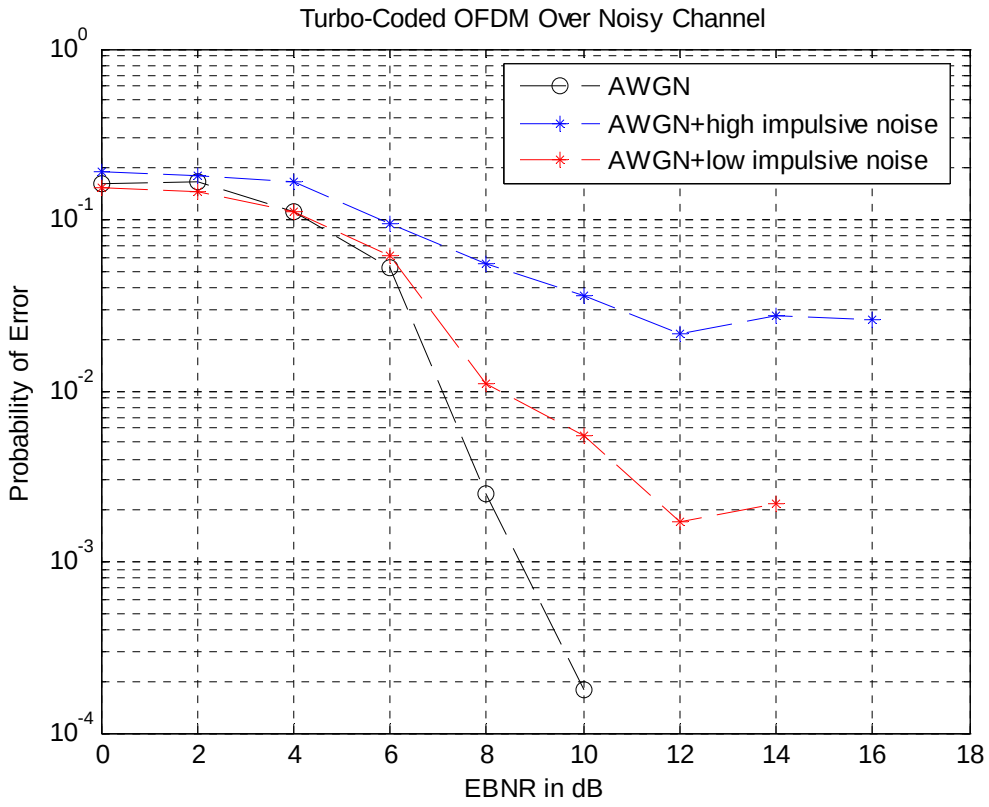


Figure 5.7 Performance of turbo coded OFDM over AWGN and impulsive noise channel

CHAPTER 6

CONCLUSION

To conclude, this major project gives the detail knowledge of a current key issue in the field of communications named Orthogonal Frequency Division Multiplexing (OFDM).

We introduced the theory behind OFDM and discussed basic OFDM transceiver architecture. We identified some factors that could result in the OFDM system not performing to its potential. These factors included ISI caused by a dispersive channel, ICI and its deleterious effects, and the issue of PAPR which is crucial for proper functionality. We explored techniques to combat some of these problems such as the use of a cyclic prefix (longer than the channel delay spread), and equalization made easy thanks to the wideband nature of the OFDM. As long as the subcarrier spacing is kept smaller than the coherence bandwidth, we can take advantage of the high correlation between adjacent subcarriers. We also presented a few results in both AWGN and Rayleigh environments, as we needed to validate our modified, simplified simulator.

We focused our attention on turbo codes and their implementation. We described the encoder architecture. In our case, the code is the result of the parallel concatenation of two identical RSCs. The code can be punctured in order to fulfill bit rate requirements. The decoder succeeded in its duty thanks to the decoding algorithms that it is built around. We focused mainly on the study of the MAP. We discovered that the power of the scheme came from the two individual decoders performing the MAP on interleaved versions of the input. Each decoder used information produced by the other as a priori information and outputted a posteriori information. We elaborated on the performance theory of the codes. Then we tied concepts of OFDM and turbo coding with a target-based, modulation scheme. We also introduced noises, which occur in power line communication networks. And analyze the performance of power line networks.

To fully support my thesis work, I simulated the entire work on MATLAB7. First I developed an OFDM system model then try to improve the performance by applying forward error correcting codes to our uncoded system. From the study of the system, it can be concluded that we are able to improve the performance of uncoded OFDM by convolutional coding scheme. Further

improvement on the performance has been achieved by applying turbo coding to uncoded OFDM system. The system model developed is quite flexible and can be easily modified and/or extended to study the performance of this scheme. The lack of powerful machines has not allowed us to generate more bits and therefore better graphs. For this reason, our results should be considered preliminary.

REFERENCES

[1] Ramjee Prasad, “OFDM for Wireless Communications systems”, Artech House Publishers, 2004.

[2] L. Hanzo, M. Munster, B.J. Choi, T. Keller, “OFDM & MC-CDMA for Broadband Multi-user Communications, WLANs and Broadcasting” John Wiley Publishers, 2003.

[3] John G. Proakis, Masoud Salehi, “communication system using MATLAB” Thomson Asia Pvt. Ltd., Singapore, 2003.

[4] Anibal Luis Intini, “orthogonal Frequency Division Multiplexing For Wireless Networks” Standard IEEE 802.11a, University Of California, Santa Barbara.

[5] B.P. Lathi, “Modern Digital and Analog Communication Systems”, *CBS College Publishing*, 1983.

[6] Weinstein, S. B. and Ebert, P. M., “Data transmission by frequency division multiplexing using the discrete fourier transform,” *IEEE Trans. Comm. Technology*, vol. COM-19, pp. 282-289, Oct. 1971.

[7] K. Taura and H. Kato, “A Digital Audio Broadcasting (DAB) Receiver,” *IEEE Trans. On Consumer Elec.*, vol. 42, Aug. 1996.

[8] L. Perez, J. Seghers, and D. Costello, “A Distance Spectrum Interpretation of Turbo Codes”, *IEEE Transactions on Information Theory*, vol. 42, no. 6, pp. 1698–1709, Nov. 1996.

[9] W. J. Blackert, E. K. Hall, and S. G. Wilson, “Turbo Code Termination and Interleaver Conditions”, *IEE Electronics Letters*, vol. 31, no. 24, pp. 2082–2084, Nov 1995.

[10] C. Berrou, A. Glavieux, and P. Thitimajshima, “Near Shannon Limit Error-Correcting Coding: Turbo Codes”, *Proceedings of the IEEE International Conference on Communications, ICC ’93, Geneva.*, pp. 1064–1070, May 1993.

[11] G. D. Forney, “The Viterbi Algorithm”, *Proceedings IEEE,* vol. 61, no. 3, pp. 268–278, March 1973.

[12] L.Bahl, J.Cocke, F.Jelinek, and J.Raviv, “Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate”, *IEEE Trans. on Information Theory*, vol. 20, pp. 248–287, March 1974.

[13] P. Robertson, “Improving Decoder and Code Structure of Parallel Concatenated Recursive Systematic (Turbo) Codes”, in *IEE Trans. of International Conference on Universal Personal Communications, San Diego*, Sept. 1994, pp. 183–187.

[14] T. A. Summers and S. G. Wilson, “SNR Mismatch and Online Estimation in Turbo Decoding”, *IEEE Trans. on Communications*, vol. 46, no. 4, pp. 421–423, April 1998.

[15] A. G. Burr, G. P. White, “Performance of Turbo-coded OFDM” in *IEE Trans. of International Conference on Universal Personal Communications*, 1999.

[16] J. Erfanian, S. Pasupathy, and G. Gulak, “Reduced Complexity Symbol Detectors with Parallel Structures for ISI Channels”, *IEEE Trans. Communications*, vol. 42, pp. 1661–1671, Feb. Mar. Apr. 1994.

[17] O. G. Hooijen, “On the channel capacity of the residential power circuit used as a digital communications medium,” *IEEE Commun. Lett.*, vol. 2, no. 10, pp. 267–268, Oct. 1998.

[18] M. Zimmerman and K. Dostert, “Analysis and modeling of impulsive noise in broad-band power-line communications,” *IEEE Trans. Electromagn. Compat.*, vol. 44, no. 1, pp. 249–258, Feb. 2002.

[19] M. H. L. Chan and R. W. Donaldson, “Amplitude, width, and interarrival distribution for noise impulses on intrabuilding power line communication networks,” *IEEE Trans. Electromagn. Compat.*, vol. 31, no. 3, pp. 320–323, Aug. 1989.

[20] S. O’Leary and D. Priestly, “Mobile broadcasting of DVB-T signals,” *IEEE Transactions on Broadcasting*, vol. 44, pp. 346–352, September 1998.

[21] Pouyan Amirshahi, S. Mohammad Navidpour, and Mohsen Kavehrad, “Performance Analysis of Uncoded and Coded OFDM Broadband Transmission Over Low Voltage Power-Line Channels With Impulsive Noise” *IEEE Transactions on Power Delivery*, VOL. 21, NO. 4, October 2006.

APPENDICES

(A) ABBREVIATIONS

ADC : Analog to Digital Converter

AWGN : Additive White Gaussian Noise

BER : Bit Error Rate

BPSK : Binary Phase Shift Keying

BW : Band Width

COFDM : Coded Orthogonal Frequency Division Multiplexing

CDMA : Code Division Multiple Access

DAC : Digital to Analog Converter

DAB : Digital Audio Broadcasting

DFT : Discrete Fourier Transform

DVB-T : DVB- Terrestrial

FEC : Forward Error Correction

FER : Frame Error Rate

FFT : Fast Fourier Transform

FDM : Frequency Division Multiplexing

IFFT : Inverse Fast Fourier Transform

ICI : Inter Carrier Interference

IDFT : Inverse Discrete Fourier Transform

IMD : Inter- Modulation Distortion

ISI : Inter Symbol Interference

LAN : Local Area Network

LLR : Log Likelihood Ratio

MAP : Maximum A Posteriori

NSC : Non-Systematic Convolutional Codes

OFDM : Orthogonal Frequency Division Multiplexing

PAPR : Peak to Average Power Ratio

PCBC : Parallel Concatenated Block Code

PCCC : Parallel Concatenated Convolutional Code

PLC : Power Line Communication

PSD : Power Spectral Density

QAM : Quadrature Amplitude Multiplexing

QPSK : Quadrature Phase Shift Keying

RF : Radio Frequency

RSC : Recursive Systematic Convolutional Codes

SISO : Soft Input-Soft Output

SNR : Signal to Noise Ratio

TC-OFDM : Turbo Coded OFDM

TDM : Time Division Multiplexing

(B) MATLAB CODE

```
clc;
clear all
M = 16; % Size of signal constellation
k = log2(M); % Number of bits per symbol
Kmax=16; %number of subcarriers
X0=0;
rho=0.95;
diary turbo_logmap.txt
% Choose decoding algorithm
dec_alg =0;  % Log MAP
% Frame size
L_total = input(' Please enter the frame size (= info + tail, default: 400)   ');
if isempty(L_total)
  L_total = 400;    % infomation bits plus tail bits
end
while rem(L_total,(k/2))~=0
    L_total=L_total+1;
end
% Code generator
g = input(' Please enter code generator: ( default: g = [1 1 1; 1 0 1 ] )      ');
if isempty(g)
g = [ 1 1 1;
       1 0 1 ];
end
%g = [1 1 0 1; 1 1 1 1];
%g = [1 1 1 1 1; 1 0 0 0 1];
[n,K] = size(g);
m = K - 1;
```

```
nstates = 2^m;
%puncture = 0, puncturing into rate 1/2;
%puncture = 1, no puncturing
puncture = input(' Please choose punctured / unpunctured (0/1): default 0     ');
if isempty(puncture)
    puncture = 0;
end
% Code rate
rate = 1/(2+puncture);
% Fading amplitude; a=1 in AWGN channel
a = 1;
% Number of iterations
niter = input(' Please enter number of iterations for each frame: default 5       ');
if isempty(niter)
   niter = 5;
end
% Number of frame errors to count as a stop criterior
ferrlim = input(' Please enter number of frame errors to terminate: default 10        ');
if isempty(ferrlim)
   ferrlim = 10;
end
EbN0db = input(' Please enter Eb/N0 in dB : default [9.0]    ');
% Enter Eb/N0 in dB like 3.0103:2:19.0103
if isempty(EbN0db)
   EbN0db = [9.0];
end
fprintf('\n\n-----------------------------------------------------\n');
fprintf(' === Log-MAP decoder === \n');
fprintf(' code generator: \n');
for ii = 1:n
    for jj = 1:K
```

```
        fprintf( '%6d', g(ii,jj));
    end
    fprintf('\n');
end
if puncture==0
   fprintf(' Punctured, code rate = 1/2 \n');
else
   fprintf(' Unpunctured, code rate = 1/3 \n');
end
fprintf(' iteration number =  %6d\n', niter);
fprintf(' terminate frame errors = %6d\n', ferrlim);
fprintf(' Eb / N0 (dB) = ');
for i = 1:length(EbN0db)
    fprintf('%10.2f',EbN0db(i));
end
fprintf('\n-------------------------------------------------\n\n');
tic;
ber1=zeros(length(EbN0db),niter);
for jal=1:8
    for nEN = 1:length(EbN0db)
            en = 10^(EbN0db(nEN)/10);      % convert Eb/N0 from unit db to normal numbers
            L_c = 4*a*en*rate;   % reliability value of the channel
            % standard deviation of AWGN noise
            sigma = 1/sqrt(2*en);
            % Clear bit error counter and frame error counter
            errs(nEN,1:niter) = zeros(1,niter);
            nferr(nEN,1:niter) = zeros(1,niter);
            nframe = 0;    % clear counter of transmitted frames
            while nferr(nEN, niter)<ferrlim
            nframe = nframe + 1;
            x = round(rand(1, L_total-m));    % info. bits
```

```
[temp, alpha] = sort(rand(1,L_total));        % random interleaver mapping
en_output = encoderm( x, g, alpha, puncture ) ; % encoder output (+1/-1)
for i=1:length(en_output)
        if en_output(i)==-1
                en_output1(i)=0;
        else
                en_output1(i)=en_output(i);
        end
end
%================================================
%          padding for QAM
zpq=rem(length(en_output1),k);
if (zpq~=0)
 en_output1=[en_output1 zeros(1,(k-zpq))];
end
%       bin to symbol=================================
qamin = bi2de(reshape(en_output1,k,length(en_output1)/k).','left-msb');
% %      ========================
% %       QAM mod
ofdm_in = qammod(qamin,M);
% % ====================
%%        Zero padding for OFDM
zp=rem(length(ofdm_in),Kmax);
 if (zp~=0)
 ofdm_in=[ofdm_in;zeros((Kmax-zp),1)];
end


%=========================================================

 s = reshape(ofdm_in,(length(ofdm_in)/Kmax),Kmax);
```

```
ss=ifft(s);
sss=reshape(ss,size(ofdm_in));
% ================================================
 rxdata = awgn(sss,EbN0db(nEN),'measured'); % AWGN Channel
%   ================================================
%ImpNoise=impulse1(10,0,sss);

%  rxdata = ImpNoise.'+awgn(sss,EbN0db(nEN),'measured');
%      ================================================
%      t5=gaus_mar(X0,rho,length(sss),sigma);    %gauss-markov noise
%      rxdata = t5'+ awgn(sss,EbN0db(nEN),'measured');
%      ================================================
 cho=reshape(rxdata,(length(ofdm_in)/Kmax),Kmax);
fftout=fft(cho);
r=reshape(fftout,size(ofdm_in));
%==============Remove Zero padding of ofdm==
 if zp~=0
r=r(1:(length(r)-(Kmax-zp)),1);
end
%=========================================
% QAM demod
qam_dout = qamdemod(r,M);
% intrger to binary
turbo_din = de2bi(qam_dout,'left-msb');
turbo_din = reshape(turbo_din.',prod(size(turbo_din)),1);
%==============remove Zero padding of QAM=========
if zpq~=0
turbo_din=turbo_din(1:(length(turbo_din)-(k-zpq)),1);
end
%================================================
for i=1:length(turbo_din )
```

```
        if turbo_din(i)==0
        turbo_din1(i)=-1;
        else
        turbo_din1(i)=turbo_din (i);
        end
end
turbo_din1=turbo_din1.*rand(size(turbo_din1,1));
yk = demultiplex(turbo_din1,alpha,puncture);
% demultiplex to get input for decoder 1 and 2
% Scale the received bits
rec_s = 0.5*L_c*yk;
% Initialize extrinsic information
L_e(1:L_total) = zeros(1,L_total);
for iter = 1:niter
% Decoder one
L_a(alpha) = L_e;  % a priori info.
L_all = logmapo(rec_s(1,:), g, L_a, 1);  % complete info.
L_e = L_all - 2*rec_s(1,1:2:2*L_total) - L_a;  % extrinsic info.
% Decoder two
L_a = L_e(alpha);  % a priori info.
L_all = logmapo(rec_s(2,:), g, L_a, 2);  % complete info.
 L_e = L_all - 2*rec_s(2,1:2:2*L_total) - L_a;  % extrinsic info.
% Estimate the info. bits
 xhat(alpha) = (sign(L_all)+1)/2;
% Number of bit errors in current iteration
 err(iter) = length(find(xhat(1:L_total-m)~=x));
% Count frame errors for the current iteration
 if err(iter)>0
 nferr(nEN,iter) = nferr(nEN,iter)+1;
end
end   %iter
```

```
        % Total number of bit errors for all iterations
         errs(nEN,1:niter) = errs(nEN,1:niter) + err(1:niter);

      if rem(nframe,3)==0 | nferr(nEN, niter)==ferrlim
        % Bit error rate
      ber(nEN,1:niter) = errs(nEN,1:niter)/nframe/(L_total-m);
        % Frame error rate
      fer(nEN,1:niter) = nferr(nEN,1:niter)/nframe;

                % Display intermediate results in process
     fprintf('*************** Eb/N0 = %5.2f db ***************\n', EbN0db(nEN));
     fprintf('%d frames transmitted, %d frames in error.\n', nframe, nferr(nEN, niter));
     fprintf('rate 1/%d, averaging counter = %d. \n', 2+puncture, jal);
        for i=1:niter
            fprintf('%8.4e    ', ber(nEN,i));
        end
      fprintf('\n');
        for i=1:niter
            fprintf('%8.4e    ', fer(nEN,i));
        end
      fprintf('\n');
      fprintf('************************************************\n\n');

        % Save intermediate results
      save turbo_sys_demo EbN0db ber fer
    end
  end     %while
end       %nEN
ber1=ber1+ber;
end
toc;
```

```
diary off
if length(EbN0db)>1
   figure(1)
   ber1=ber1/jal;
   ber11=ber1(:,1);
   semilogy(EbN0db+10*log10(1/2),ber11.', 'r*--')
   xlabel('EBNR in dB')
   ylabel('Probability of Error')
   title('Turbo-Coded OFDM Over AWGN Channel')
   grid
end


function bin_state = bin_state( int_state, m )
for j = 1:length( int_state )
  for i = m:-1:1
     stat(j,m-i+1) = fix( int_state(j)/ (2^(i-1)) );
     int_state(j) = int_state(j) - stat(j,m-i+1)*2^(i-1);
  end
end
bin_state = stat;


function subr = demultiplex(r, alpha, puncture);
L_total = length(r)/(2+puncture);
% Extract the parity bits for both decoders
if puncture == 1        % unpunctured
 for i = 1:L_total
    x_sys(i) = r(3*(i-1)+1);
    for j = 1:2
       subr(j,2*i) = r(3*(i-1)+1+j);
    end
```

```
  end
else          % unpunctured
  for i = 1:L_total
     x_sys(i) = r(2*(i-1)+1);
     for j = 1:2
       subr(j,2*i) = 0;
     end
     if rem(i,2)>0
       subr(1,2*i) = r(2*i);
     else
       subr(2,2*i) = r(2*i);
     end
  end
end
% Extract the systematic bits for both decoders
for j = 1:L_total
% For decoder one
  subr(1,2*(j-1)+1) = x_sys(j);
% For decoder two: interleave the systematic bits
  subr(2,2*(j-1)+1) = x_sys(alpha(j));
end


function [output, state] = encode_bit(g, input, state)
[n,k] = size(g);
m = k-1;
% determine the next output bit
for i=1:n
  output(i) = g(i,1)*input;
  for j = 2:k
    output(i) = xor(output(i),g(i,j)*state(j-1));
```

```
  end;
end
state = [input, state(1:m-1)];


function en_output = encoderm( x, g, alpha, puncture )
[n,K] = size(g);
m = K - 1;
L_info = length(x);
L_total = L_info + m;
% generate the codeword corresponding to the 1st RSC coder
% end = 1, perfectly terminated;
input = x;
output1 = rsc_encode(g,input,1);
% make a matrix with first row corresponing to info sequence
% second row corresponsing to RSC #1's check bits.
% third row corresponsing to RSC #2's check bits.
y(1,:) = output1(1:2:2*L_total);
y(2,:) = output1(2:2:2*L_total);
% interleave input to second encoder
for i = 1:L_total
  input1(1,i) = y(1,alpha(i));
end
output2 = rsc_encode(g, input1(1,1:L_total), -1 );
y(3,:) = output2(2:2:2*L_total);
% paralell to serial multiplex to get output vector
% puncture = 0: rate increase from 1/3 to 1/2;
% puncture = 1; unpunctured, rate = 1/3;
if puncture > 0     % unpunctured
  for i = 1:L_total
     for j = 1:3
```

```
          en_output(1,3*(i-1)+j) = y(j,i);
       end
    end
else            % punctured into rate 1/2
   for i=1:L_total
       en_output(1,n*(i-1)+1) = y(1,i);
       if rem(i,2)
      % odd check bits from RSC1
          en_output(1,n*i) = y(2,i);
       else
      % even check bits from RSC2
          en_output(1,n*i) = y(3,i);
       end
    end
end

% antipodal modulation: +1/-1
en_output = 2 * en_output - ones(size(en_output));



function [X]=gaus_mar(X0,rho,N,sigma)
% [X]=gaus_mar(X0,rho,N)
%       GAUS_MAR  generates a Gauss-Markov process of length N.
%           The noise process is taken to be white Gaussian
%           noise with zero mean and unit variance.
for i=1:2:N,
  [Ws(i) Ws(i+1)]=gngauss(sigma);        % Generate the noise process.
end;
X(1)=rho*X0+Ws(1);                % first element in the Gauss--Markov process
for i=2:N,
```

```
  X(i)=rho*X(i-1)+Ws(i);          % the remaining elements
end;


function [gsrv1,gsrv2]=gngauss(m,sgma)
%   [gsrv1,gsrv2]=gngauss(m,sgma)
%   [gsrv1,gsrv2]=gngauss(sgma)
%   [gsrv1,gsrv2]=gngauss
%      GNGAUSS  generates two independent Gaussian random variables with mean
%         m and standard deviation sgma. If one of the input arguments is missing,
%         it takes the mean as 0.
%         If neither the mean nor the variance is given, it generates two standard
%         Gaussian random variables.
if nargin == 0,
  m=0; sgma=1;
elseif nargin == 1,
  sgma=m; m=0;
end;
u=rand;                         % a uniform random variable in (0,1)
z=sgma*(sqrt(2*log(1/(1-u))));      % a Rayleigh distributed random variable
u=rand;                         % another uniform random variable in (0,1)
gsrv1=m+z*cos(2*pi*u);
gsrv2=m+z*sin(2*pi*u);


function [impulse]=impulse1(amp,impulse_impact,sss)
% amp=15;
% impulse_impact=0;
% sss=randn(80,1);
aaa=randn(size(sss));
bbb=randn(size(sss));
ccc=aaa.*bbb;
```

```
if length(sss)>80
if impulse_impact==0          %low impulse impact
   for (ix=1:length(sss))
      if (ccc(ix)>=4.2)
         impulse(ix)=amp;
      else
         impulse(ix)=0;
      end
   end
elseif impulse_impact==1          %medium impulse impact
   for (ix=1:length(sss))
      if (ccc(ix)>=2.5)
         impulse(ix)=amp;
      else
         impulse(ix)=0;
      end
   end
elseif impulse_impact==2          %medium impulse impact
   for (ix=1:length(sss))
      if (ccc(ix)>=1.5)
         impulse(ix)=amp;
      else
         impulse(ix)=0;
      end
   end
end
else
   if impulse_impact==0          %low impulse impact
   for (ix=1:length(sss))
      if (ccc(ix)>=2)
         impulse(ix)=amp;
```

```
        else
            impulse(ix)=0;
        end
    end
elseif impulse_impact==1          %medium impulse impact
    for (ix=1:length(sss))
        if (ccc(ix)>=1.2)
            impulse(ix)=amp;
        else
            impulse(ix)=0;
        end
    end
elseif impulse_impact==2          %medium impulse impact
    for (ix=1:length(sss))
        if (ccc(ix)>=0.7)
            impulse(ix)=amp;
        else
            impulse(ix)=0;
        end
    end
    end
end

function int_state = int_state( state )
 [dummy, m] = size( state );
for i = 1:m
   vect(i) = 2^(m-i);
end
int_state = state*vect';
```

```
function L_all = logmapo(rec_s,g,L_a,ind_dec)
% Total number of bits: Inftyo. + tail
L_total = length(rec_s)/2;
[n,K] = size(g);
m = K - 1;
nstates = 2^m;          % number of states in the trellis
% Set up the trellis
[next_out, next_state, last_out, last_state] = trellis(g);
Infty = 1e10;
% Initialization of Alpha
Alpha(1,1) = 0;
Alpha(1,2:nstates) = -Infty*ones(1,nstates-1);
% Initialization of Beta
if ind_dec==1
  Beta(L_total,1) = 0;
  Beta(L_total,2:nstates) = -Infty*ones(1,nstates-1);
elseif ind_dec==2
  Beta(L_total,1:nstates) = zeros(1,nstates);
else
  fprintf('ind_dec is limited to 1 and 2!\n');
end
% Trace forward, compute Alpha
for k = 2:L_total+1
   for state2 = 1:nstates
     gamma = -Infty*ones(1,nstates);
     gamma(last_state(state2,1)) = (-rec_s(2*k-3)+rec_s(2*k-2)*last_out(state2,2))....
         -log(1+exp(L_a(k-1)));
     gamma(last_state(state2,2)) = (rec_s(2*k-3)+rec_s(2*k-2)*last_out(state2,4))....
         +L_a(k-1)-log(1+exp(L_a(k-1)));

     if(sum(exp(gamma+Alpha(k-1,:)))<1e-300)
```

```
        Alpha(k,state2)=-Infty;
      else
        Alpha(k,state2) = log( sum( exp( gamma+Alpha(k-1,:) ) ) );
      end
    end
    tempmax(k) = max(Alpha(k,:));
    Alpha(k,:) = Alpha(k,:) - tempmax(k);
end
% Trace backward, compute Beta
for k = L_total-1:-1:1
  for state1 = 1:nstates
     gamma = -Infty*ones(1,nstates);
     gamma(next_state(state1,1)) = (-rec_s(2*k+1)+rec_s(2*k+2)*next_out(state1,2))....
          -log(1+exp(L_a(k+1)));
     gamma(next_state(state1,2)) = (rec_s(2*k+1)+rec_s(2*k+2)*next_out(state1,4))....
          +L_a(k+1)-log(1+exp(L_a(k+1)));
     if(sum(exp(gamma+Beta(k+1,:)))<1e-300)
        Beta(k,state1)=-Infty;
     else
        Beta(k,state1) = log(sum(exp(gamma+Beta(k+1,:))));
     end
  end
  Beta(k,:) = Beta(k,:) - tempmax(k+1);
end
% Compute the soft output, log-likelihood ratio of symbols in the frame
for k = 1:L_total
  for state2 = 1:nstates
     gamma0 = (-rec_s(2*k-1)+rec_s(2*k)*last_out(state2,2))....
          -log(1+exp(L_a(k)));
     gamma1 = (rec_s(2*k-1)+rec_s(2*k)*last_out(state2,4))...
          +L_a(k)-log(1+exp(L_a(k)));
```

```
    temp0(state2) = exp(gamma0 + Alpha(k,last_state(state2,1)) + Beta(k,state2));
    temp1(state2) = exp(gamma1 + Alpha(k,last_state(state2,2)) + Beta(k,state2));
  end
  %================================================
aaaa1=sum(temp1);
aaaa2=sum(temp0);
if aaaa1==0
   aaaa1=0.001;
end
if aaaa2==0
   aaaa2=0.001;
end
L_all(k) = log(aaaa1) - log(aaaa2);
%=====================================================
% L_all(k) = log(sum(temp1)) - log(sum(temp0));
end


function y = rsc_encode(g, x, terminated)
 [n,K] = size(g);
m = K - 1;
if terminated>0
  L_info = length(x);
  L_total = L_info + m;
else
  L_total = length(x);
  L_info = L_total - m;
end
% initialize the state vector
state = zeros(1,m);
% generate the codeword
```

```
for i = 1:L_total
   if terminated<0 | (terminated>0 & i<=L_info)
      d_k = x(1,i);
   elseif terminated>0 & i>L_info
      % terminate the trellis
      d_k = rem( g(1,2:K)*state', 2 );
   end
   a_k = rem( g(1,:)*[d_k state]', 2 );
   [output_bits, state] = encode_bit(g, a_k, state);
   % since systematic, first output is input bit
   output_bits(1,1) = d_k;
   y(n*(i-1)+1:n*i) = output_bits;
end



function [next_out, next_state, last_out, last_state] = trellis(g)
 [n,K] = size(g);
m = K - 1;
max_state = 2^m;
% set up next_out and next_state matrices for systematic code
for state=1:max_state
   state_vector = bin_state( state-1, m );
   % when receive a 0
   d_k = 0;
   a_k = rem( g(1,:)*[0 state_vector]', 2 );
   [out_0, state_0] = encode_bit(g, a_k, state_vector);
   out_0(1) = 0;
   % when receive a 1
   d_k = 1;
   a_k = rem( g(1,:)*[1 state_vector]', 2 );
   [out_1, state_1] = encode_bit(g, a_k, state_vector);
```

```
  out_1(1) = 1;
  next_out(state,:) = 2*[out_0 out_1]-1;
  next_state(state,:) = [(int_state(state_0)+1) (int_state(state_1)+1)];
end
% find out which two previous states can come to present state
last_state = zeros(max_state,2);
for bit=0:1
  for state=1:max_state
    last_state(next_state(state,bit+1), bit+1)=state;
    last_out(next_state(state, bit+1), bit*2+1:bit*2+2) ...
      = next_out(state, bit*2+1:bit*2+2);
  end
end
```