IMPLEMENTATION OF LDPC CODES USING THE IEEE 802.16e STANDARD

By

TIMOTHY WEYLIN GERKEN

Problem Report

Submitted to
College of Engineering and Mineral Resources
At
West Virginia University

In Partial Fulfillment of the Requirements
For the Degree of
Master of Science in Electrical Engineering

Committee Members

Matthew Valenti, Ph.D., Chair
Daryl Reynolds, Ph.D.
Don Adjeroh, Ph.D.

Morgantown, West Virginia, 2008

Implementation of LDPC codes using the IEEE 802.16e standard

Timothy Weylin Gerken

(Abstract)

Channel coding enables the effective transmission of data over a noisy channel. Low density parity check codes (LDPC) are an old class of efficient channel codes that have been ignored until recently. When LDPC codes were originally introduced in the 1960's, they exceeded the hardware capabilities at the time, and were considered a theoretical novelty. However, over the past ten years, there has been a renewed interest in LDPC codes due to technological advancements that finally allowed efficient implementations to be realized. LDPC codes have become part of several state-of-the-art wireless standards, including the IEEE 802.16e standard that is behind mobile-WiMAX. In this problem report, an encoder for encoding the LDPC codes for 802.16e is developed. Using this encoder and an off-the-shelf decoder, simulations of this code were conducted and used to produce bit and frame error rate plots.

Chapter 1: Introduction

Communications presents a simple problem. Some information needs to be sent from one entity to another through some channel. This information that is received must match the original information. The challenge is to find how to recover the original information if some sort of noise or errors occurs during the transmission process. There are simple ways of ensuring data is correctly heard such as repeating the transmission. The problem with this is it takes twice as long to convey the information than if the information was heard without errors. Error control coding seeks to maximize the throughput while ensuring the information is transmitted correctly.

In order for communication systems to function, the receiver must obtain the data the transmitter had sent. To achieve this, some type of redundancy must be added to combat errors that occur across a channel. The errors can be caused by interference from other signals as well as noise in the channel itself. The challenge is that it is difficult to obtain an acceptable error rate while not compromising the data rate. The careful balance between error rate and data transmission rate is the focus of most research in coding theory. The goal is to achieve maximal data rates while maintaining inconsequential error rates. There are a variety of different methods for obtaining these goals.

1.1 Shannon Capacity

The Shannon channel capacity theorem states that there exists a rate at which data can be sent with arbitrarily low error probability [Sh48]. The theorem assumes that the signal

power is finite and the noise in the channel is Gaussian. However, this theorem does not provide a means for achieving capacity. The theorem simply shows that a bound exists which cannot be exceeded while maintaining successful data transmission. This gives researchers a goal to pursue but does not show how to arrive at the capacity limit. The Shannon capacity can be depicted as a tradeoff curve for which as the signal to noise ratio (SNR) increases, so does the rate. Researchers have been able to transmit close to the Shannon capacity. Some have used unreasonable techniques to get there such as extremely large redundancy codes, but others have improved upon existing more functional codes. Turbo codes have achieved great success but LDPC codes are gaining popularity.

1.2 LDPC Codes

Low Density Parity-check (LDPC) codes were first proposed in the early 1960's [Ga63]. These codes are called low density because of the sparse nature of the parity-check matrix. The parity-check matrix contains a small number of ones which lends to the name. These early codes were widely neglected because they were too complex to implement using the hardware that was available at the time. Therefore until decades later, research did not focus on LDPC codes. Recently, LDPC codes have proven to efficiently transmit data. The code has a great deal of flexibility since the code can support a range of rates and accuracy based on the needs of the system. The code uses a sparse matrix which allows only storing the non-zero entries and locations instead of storing all the entries in the matrix. Different ways of implementing the LDPC codes allows them to outperform the turbo code in some cases. Most notably, LDPC codes were

chosen over turbo codes in the DVB-S2 standard for satellite coding. LDPC codes have been shown to come very close to the Shannon capacity. The outstanding performance that can achieve rates close to the Shannon channel capacity, as well as its greater acceptance, are motivation for further study of LDPC codes.

1.3 IEEE 802.16e-2005 Standard

The IEEE 802.16 Working Group on Broadband Wireless Access Standards was charged with the task of standardizing the development of broadband wireless metropolitan area networks. The original 802.16 standard was not developed with mobility in mind; however, since its creation, there has been increased demand for mobile services. In response to this demand, the 802.16e standard was developed. This standard is the focus of this project, as it uses LDPC codes. The 802.16e standard expands upon the base 802.16 standard by providing wireless broadband access to vehicles moving at high speed. It operates in licensed bands below 6 GHz. This provides enhancement to the mobility of the previous standard. This is to take advantage of wireless media to provide broadband Internet access to create greater mobility. It will support fixed and mobile services.

1.4 WiMAX

WiMAX, Worldwide Interoperability for Microwave Access, is an industry consortium with the goal of promoting technologies based on the IEEE 802.16 standard for the transmission of wireless data over long distances. The goal of WiMAX is to

standardize the way that broadband wireless systems exchange information and use that information in IEEE 802.16 wireless networks. The relationship between WiMAX and 802.16 is analogous to the relationship between the WiFi consortium and 802.11.

Most WiFi LANs have ranges of no more than 300 feet. WiMAX seeks to expand this range. WiMAX has capabilities of transmitting up to 31 miles. The transmit data rate of WiMAX is also an improvement having up to 75 Mbps [La07]. In order to have greater coverage area, WiMAX operates on lower frequencies in the 2-11 GHz band. This provides for less expensive service rates for a larger number of customers but does hurt transfer rates [Pr06]. With this ability, one would be able to travel and have complete network mobility. The user would no longer have to be tapped into one wireless router or Ethernet connection and consistently enter and reenter wireless LANs every 300 feet. He would be able to travel using a series of hotspots to provide high-speed Internet access.

The WiMAX standard has a variety of different uses. As indicated before, WiMAX provides for a large range of Internet connections in metropolitan areas or other heavily congested places. It also provides high speed connections and allows for truly mobile connection. It accomplishes this without dropping connections since the network has a much larger range than the LAN wireless routers currently in use. WiMAX is a wide range network which is not as restricted as the WiFi networks which only work in small ranges. Those networks require access points such as restaurants and coffee shops which transmits the signals. WiMAX has the capability to broadcast over several kilometers which is a great deal larger to the meters broadcast range of WiFi. The ability

to send over long ranges also allows the broadcast to penetrate buildings without significant loss. WiMAX provides a great opportunity to have complete coverage to metropolitan areas without the extensive web of small WiFi LANs. This technology with also help to extend to rural areas which currently do not have the luxury of broadband Internet access since cable and DSL will have expand to those places.

1.5 Communication Systems

A communication system consists of several parts, but in this report the focus is on a basic model. As shown in Figure 1.1, the model is made up of encoder, modulator, channel, demodulator, and decoder.

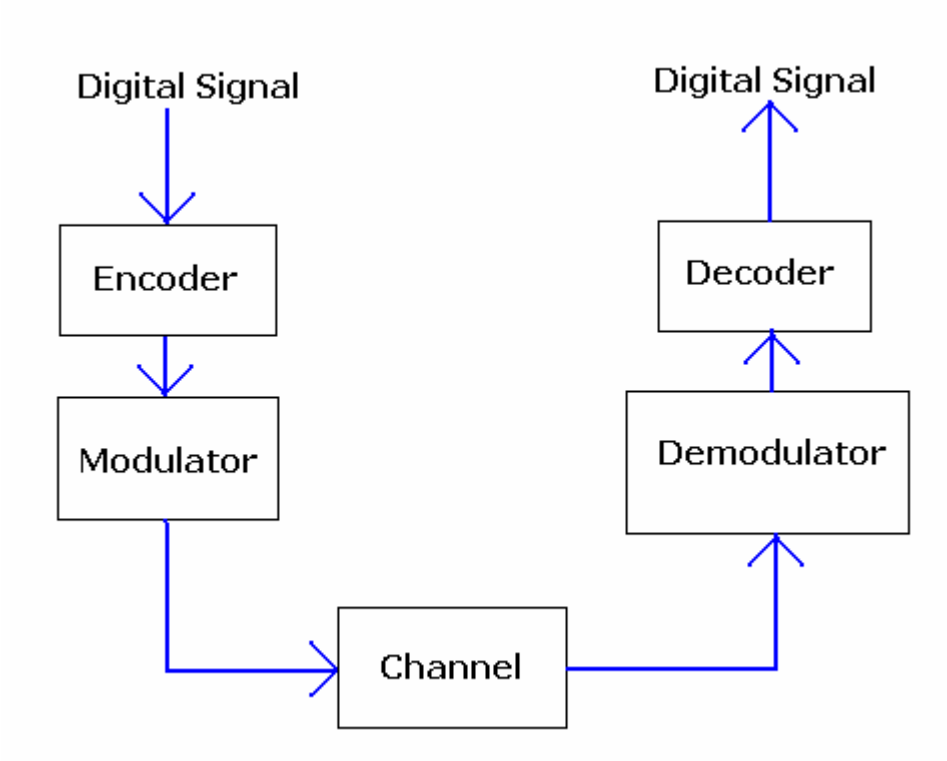

Figure 1.1 Simple communications system

1.5.1 Modulation

Modulation is the process of mapping groups of N bits to one of 2^N symbols from a multidimensional space. When the space is two dimensional, as is the case with many common modulation formats such as PSK (Phase Shift Keying) and QAM (Quadrature Amplitude Modulation), then the alphabet may consist of complex-valued scalar values. These are arranged and pictured in a constellation diagram.

The constellation diagram is drawn on the complex plane. Signals are arranged by predetermined locations according to the type of modulation technique. The symbols are commonly labeled using Gray coding. This method allows the mapping of adjacent symbols to only vary by one bit. A Gray code minimizes the number of bit errors when a symbol error occurs.

PSK modulation transmits using variation in phase to convey information. As shown in Figure 1.2, BPSK in set up such that a representation of bit 0 is at 0º and bit 1 is at 180º.

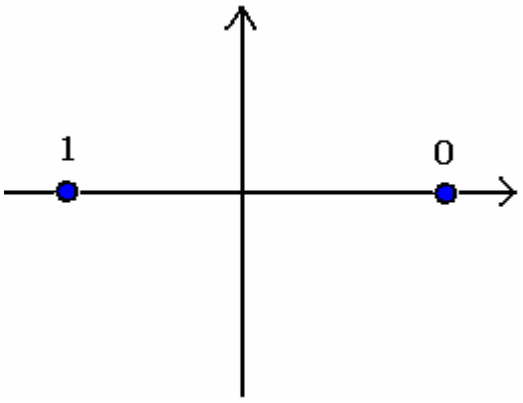

Figure 1.2 BPSK signal constellation

BPSK is defined as having the following signal set:

$$s_0(t) = \sqrt{2E_b/T_b}\cos 2\pi f_0 t,\ 0 \le t \le T_b, \qquad \text{Bit "0"} \qquad (1.1)$$

$$s_1(t) = -\sqrt{2E_b/T_b}\cos 2\pi f_0 t,\ 0 \le t \le T_b, \qquad \text{Bit "1"} \qquad (1.2)$$

Considering the interval $(0, T_b)$ assume that

$$\int_0^{T_b} |s_1(t)|^2\, dt = \int_0^{Tb} |s_2(t)|^2\, dt = E_b \qquad (1.3)$$

In AWGN (Additive White Gaussian Noise) and with perfect synchronization, the bit error probability is:

$$P_b = Q(\sqrt{2E_b/N_o}), \qquad (1.4)$$

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_0^x \exp(-t^2/2)dt \qquad (1.5)$$

This defines the BPSK modulation which is used in the simulation techniques that follow in later chapters.

1.5.2 Encoding

Encoding allows for the successful transmission of information over a noisy

channel. In practice, most channels are viewed as AWGN. This is a simplified view of a typical channel since interference usually plays a role too, but AWGN channels usually just treat the interference as noise. The key to producing a successful signal is to introduce redundancy to the information which is being sent. Redundancy gives the receiver a way of recovering the original signal even if some of the bits are altered because of the noisy channel. The type of redundancy is what gives different encoding procedures different results in terms of rate and SNR.

Using LDPC codes allows for a variety of rates and creates a simple encoding procedure. The encoder performs a one-to-one mapping of messages of information bits to codewords. Often, the codeword will contain message bits in the front of the codeword and parity bits at the end of the codeword. Such codes are referred to as *systematic codes*. Even though redundancy is added to the information, this does not ensure correct transmission data to the receiver. The amount of redundancy added as well as the amount of noise in the channel determines the success of the transmissions. As long as the code rate does not exceed the Shannon capacity, then one can feel confident that the error rate will be arbitrarily low according to the theorem.

1.5.3 Decoding

Decoding is essentially the opposite of encoding. Instead of starting with the information and adding redundancy, the decoder takes the received signal, which contains information, redundancy information and/or errors, and tries to find the most likely message. The challenge for the decoder is to discriminate between the correct

information and the errors created by the channel. The decoder takes the received signal and tries to match it to a valid codeword. The decoder can use different means for matching the received signal to the codeword. It may depend upon the probability of a codeword being transmitted, the distance to the closest codeword, or use a lookup table to determine the error and codeword which needs to be recovered. Section 4.3 will show the different methods for decoding.

1.5.4 Demodulation

Demodulation is the reverse function of modulation much like decoding is to encoding. The signal which was sent through the channel is received. This signal is recovered through the decoder by matching the symbols to the proper bit pattern.

1.6 Problem Statement

This problem report deals with implementing the 802.16e WiMAX standard using LDPC encoding. The performance of the code using the standard is tested over noisy channels. This encoding procedure will be implemented and the signals will be recovered using an existing decoding procedure in the Coded Modulation Library (CML). The CML library is an open source toolbox developed at West Virginia University (WVU) and used to simulate codes. The outputs will be used to produce a sequence of charts that plot the error rate versus the signal-to-noise ratio. The CML library contained a generic LDPC encoder but did not have WiMAX support. The goal of the project was to implement the encoder by adding support for the specific LDPC code in the mobile-

WiMAX 802.16e standard. The challenge came from the implementation of the encoder since the encoding of this code was not straightforward.

Chapter 2: Block Codes

A *block code* maps messages of length k to codewords of length n, where $n > k$. The k information bits are represented as the message block vector **m**, where **m** = (m_0, $m_1 \ldots m_{k-1}$). There are 2^k possible different messages which can exist for the block code.

A *systematic code* is a code where the first bits in the codeword are all the information bits. The encoder appends one or more *parity bits* to the end of the message. The parity bits are carefully selected to protect the message against channel errors.

The *codeword* is represented by **c** = (c_0, $c_1 \ldots c_{n-1}$), where n is the length of the codeword. The *code rate* is the ratio $R = k/n$ which is the number of information bits by the number of codeword bits. The encoder is strictly independent for each message and has no memory. The block code is designated as (n, k) to show the size of the message and the codeword. The output codeword only depends on the input message. In dealing with the message vector and codewords, these vectors are binary.

2.1 Linear Block Codes

When the code is systematic, the first k bits of the codeword **c** are exactly the information bits. The additional bits are used to add redundancy to the codeword. With these extra bits, the recovery of the information bits can be achieved even if some errors may occur in the channel due to noise.

In order to determine the parity bits, the generator matrix and parity-check matrix are used. The codeword vector can be represented as:

$$\mathbf{c} = \mathbf{mG}, \tag{2.1}$$

where **G** is the generator matrix (defined in section 2.1.2)

In order to create a meaningful code, the number of message bits must be less than the number of code bits for each message. In this, the inequality $k < n$ must hold therefore, the ratio of k/n is less than or equal to one. Determining the redundancy is the issue involved in creating any particular block code.

2.1.1 Parameters for Linear Block Code

The block code has information bits, length k, which is used to form the codeword, length n. The code is representing as (n, k) with rate $R = k / n$.

The *Hamming weight* is found by counting the number of ones (since the codeword is binary) in a codeword. For example, the Hamming weight of codeword $\mathbf{c_i}$ is:

$$w_j = \sum_{i=0}^{n-1} \mathbf{c}_{j,i} \tag{2.2}$$

The *Hamming distance* is the number of bit positions that one codeword differs from another. It is the number of substitutions needed for one codeword to be the same as the other. So the Hamming distance between $\mathbf{c_i}$ and $\mathbf{c_j}$ is:

$$d_{ij}(\mathbf{c_i}, \mathbf{c_j}) = \sum_{m=0}^{n-1} (\mathbf{c}_{j,m} + \mathbf{c}_{i,m}) \qquad (2.3)$$

Since no two code words can be the same, $d_{ij} > 0$, if $i \neq j$. The Hamming distance calculation is performed in GF(2). The Hamming distance is used with decoding. The maximum likelihood decoding requires the use of Hamming distances. These distances show the most probable codeword based on the additive noise which the channel adds to the original signal. The Hamming distances help determine codewords based on these values. By finding the Hamming distances for all codewords relative to the received vector, the most likely codeword to have been sent is the one with the lowest Hamming distance.

Through using the Hamming distances as a visual tool, the distances will provide a graphical representation of how decision of the received signal should be made. The decision regions make up a mapping where each codeword occupies an area based on its distance to other codewords. If the transmitted signal falls in that decision region, the signal is assigned to that codeword. This makes a clear cut determination for each codeword. Decision regions can be different depending on the type of decoding method used. Either, they can be equally spaced if the probability of each codeword is the same. Or, the highest probability codewords can have the largest regions since they are most likely to be transmitted.

2.1.2 Parity-check Matrix and Generator Matrix

The *generator matrix* contains rows which generate all code words. The generator matrix G is multiplied by the message m, as shown in equation (2.1). The generator matrix is of size $k \times n$. The standard systematic form of the generator matrix is:

$$\mathbf{G} = [\mathbf{I}_k \mid \mathbf{P}], \tag{2.4}$$

where $\mathbf{I}_k$ is the $k \times k$ identity matrix, **P** is a matrix $k \times (n-k)$

A systematic code has the identity matrix contained somewhere in the generator matrix. A standard systematic code has the identity matrix in the beginning as shown in the above formula. Using a systematic code makes calculations quicker since the code contains the original message.

The *parity-check matrix*, **H**, is found through transformations of the generator matrix. Through matrix calculations the generator matrix becomes the parity-check matrix by the following:

$$\mathbf{H} = [-\mathbf{P}^T \mid \mathbf{I}_{n-k}] \tag{2.5}$$

The use of the parity-check matrix allows the decoder to figure out whether the codeword is valid. The *syndrome* of a received vector **x** is $\mathbf{s} = \mathbf{x}\mathrm{H}^T$. If **s** = 0, then **x** is a valid codeword. On the other hand, if $s \neq 0$, then **x** is not a valid codeword.

2.2 Syndrome Decoding

Syndrome decoding can be used in order to effectively decode a signal through a noisy channel. By using minimum distances a code of length n can be corrected up to t, where $t = \left\lfloor \frac{d-1}{2} \right\rfloor$, where d is the minimum distance of the codeword.

A standard array table can be used to determine the correct codeword. A standard array is a table containing $2^{(n-k)}$ rows, one for each syndrome. The first row corresponds to the all-zeroes syndrome and contains the entire set of codewords. When a vector **x** is received, it is located in the standard array and the most likely codeword is read from the top of the table.

Chapter 3: 802.16e LDPC Code

3.1 Code Description

LDPC codes are linear block codes, and they are often in systematic form. The fundamental codes can by used for different code rates and packet sizes. The **H** matrix defines each set of LDPC codes. The **H** matrix is of size $m \times n$, where n is the length of the code and $m = n - k$ is the number of parity bits. Many LDPC codes will have parity-check matrices of the form:

$$\mathbf{H} = \begin{vmatrix} \mathbf{P}_{0,0} & \mathbf{P}_{0,1} & \mathbf{P}_{0,2} & \cdots & \mathbf{P}_{0,n_b-2} & \mathbf{P}_{0,n_b-1} \\ \mathbf{P}_{1,0} & \mathbf{P}_{1,1} & \mathbf{P}_{1,2} & \cdots & \mathbf{P}_{1,n_b-2} & \mathbf{P}_{1,n_b-1} \\ \mathbf{P}_{2,0} & \mathbf{P}_{2,1} & \mathbf{P}_{2,2} & \cdots & \mathbf{P}_{2,n_b-2} & \mathbf{P}_{2,n_b-1} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \mathbf{P}_{m_b-1,0} & \mathbf{P}_{m_b-1,1} & \mathbf{P}_{m_b-1,2} & \cdots & \mathbf{P}_{m_b-1,n_b-2} & \mathbf{P}_{m_b-1,n_b-1} \end{vmatrix}$$

This matrix is made up of many matrices. $\mathbf{P}_{i,j}$ is a $z \times z$ permutation of the identity matrix or a $z \times z$ all zeroes matrix. The 802.16e standard defines a set of base matrices H_{bm}, which are expanded to form the **H** matrix. Since each permutation is specified by a single right circular shift, the H_{bm} matrix contains the permutation information in a compact form. H_{bm} has the dimensionality $m_b \times n_b$ where $n = z * n_b$ and $m = z * m_b$. The base matrix is expanded by replacing each -1 with a z by z zeros matrix, each 0 with a z by z identity matrix, and each positive number by a circularly shifted z by z identity matrix.

The permutation matrices are simple shifts of the identity matrix. All the permutation matrices are right circular shifts of the identity matrix. Since it tends to be rather large, the **H** matrix can be condensed into a smaller form. By labeling the circular shifts by the number of shift the matrix contained, the H_{bm} matrix completely specifies the **H** matrix in a compact manner.

The H_{bm} matrix is divided up into two matrix sections H_{b1} and H_{b2}, where H_{b1} contains the systematic bits and H_{b2} has the parity bits:

$$H_{bm} = [(H_{b1})_{mb \times kb} \mid (H_{b2})_{mb \times mb}]$$

The matrix H_{b2} may be partitioned as:

$$H_{b2} = [h_b \mid H'_{b2}]$$

where h_b is an odd weight vector and H'_{b2} is a matrix with a dual diagonal structure.

A dual diagonal structure is one that has a main diagonal of ones like the identity matrix and a second diagonal of ones one column above the main diagonal. The vector h_b has a structure where h_b (0) = 1, h_b (m_b-1) = 1 and a third value h_b (j) = 1, where $0 < j < m_b - 1$.

The base model matrix is defined for n = 2304, the largest code length for each code rate. The shift sizes are determined for each code rate. Each base model has 24 columns, n_b, and the expansion factor $z_f = n/24$. For code length 2304, the expansion

factor is designated $z_O = 96$. The shift sizes {p(i,j)} corresponding to expansion factor are derived from {p(i,j)} by scaling according to equation 3.1 and 3.2. These are used when z0 is less than 96 as designated above. $p(i, j)$ is the ith, jth entry of the H_{bm} matrix. To determine the shift the following is used:

$$p(f, i, j) = p(i, j) \quad \text{, if } p(i, j) \le 0 \tag{3.1}$$

$$p(f, i, j) = p(i, j) * z_f / z_O \quad \text{, if } p(i, j) > 0 \tag{3.2}$$

3.2 Code Rate H_{bm} Matrices

The following are the H_{bm} matrices corresponding to particular code rates and different subtypes.

Rate 1/2:

-1	94	73	-1	-1	-1	-1	-1	55	83	-1	-1	7	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	27	-1	-1	-1	22	79	9	-1	-1	-1	12	-1	0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	24	22	81	-1	33	-1	-1	-1	0	-1	-1	0	0	-1	-1	-1	-1	-1	-1	-1	-1
61	-1	47	-1	-1	-1	-1	-1	65	25	-1	-1	-1	-1	-1	0	0	-1	-1	-1	-1	-1	-1	-1
-1	-1	39	-1	-1	-1	84	-1	-1	41	72	-1	-1	-1	-1	-1	0	0	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	46	40	-1	82	-1	-1	-1	79	0	-1	-1	-1	-1	0	0	-1	-1	-1	-1	-1
-1	-1	95	53	-1	-1	-1	-1	-1	14	18	-1	-1	-1	-1	-1	-1	-1	0	0	-1	-1	-1	-1
-1	11	73	-1	-1	-1	2	-1	-1	47	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0	-1	-1	-1
12	-1	-1	-1	83	24	-1	43	-1	-1	-1	51	-1	-1	-1	-1	-1	-1	-1	-1	0	0	-1	-1
-1	-1	-1	-1	-1	94	-1	59	-1	-1	70	72	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0	-1
-1	-1	7	65	-1	-1	-1	-1	39	49	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0
43	-1	-1	-1	-1	66	-1	41	-1	-1	-1	26	7	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0

Rate 2/3 A code:

3	0	-1	-1	2	0	-1	3	7	-1	1	1	-1	-1	-1	-1	1	0	-1	-1	-1	-1	-1
-1	-1	1	-1	36	-1	-1	34	10	-1	-1	18	2	-1	3	0	-1	0	0	-1	-1	-1	-1
-1	-1	12	2	-1	15	-1	40	-1	3	-1	15	-1	2	13	-1	-1	-1	0	0	-1	-1	-1
-1	-1	19	24	-1	3	0	-1	6	-1	17	-1	-1	-1	8	39	-1	-1	-1	0	0	-1	-1
20	-1	6	-1	-1	10	29	-1	-1	28	-1	14	-1	38	-1	-1	0	-1	-1	-1	0	0	-1
-1	-1	10	-1	28	20	-1	-1	8	-1	36	-1	9	-1	21	45	-1	-1	-1	-1	-1	0	0
35	25	-1	37	-1	21	-1	-1	5	-1	-1	0	-1	4	20	-1	-1	-1	-1	-1	-1	-1	0
-1	6	6	-1	-1	-1	4	-1	14	30	-1	3	36	-1	14	-1	1	-1	-1	-1	-1	-1	0

Rate 2/3 B code:

2	-1	19	-1	47	-1	48	-1	36	-1	82	-1	47	-1	15	-1	95	0	-1	-1	-1	-1	-1	-1
-1	69	-1	88	-1	33	-1	3	-1	16	-1	37	-1	40	-1	48	-1	0	0	-1	-1	-1	-1	-1
10	-1	86	-1	62	-1	28	-1	85	-1	16	-1	34	-1	73	-1	-1	-1	0	0	-1	-1	-1	-1
-1	28	-1	32	-1	81	-1	27	-1	88	-1	5	-1	56	-1	37	-1	-1	-1	0	0	-1	-1	-1
23	-1	29	-1	15	-1	30	-1	66	-1	24	-1	50	-1	62	-1	-1	-1	-1	-1	0	0	-1	-1
-1	30	-1	65	-1	54	-1	14	-1	0	-1	30	-1	74	-1	0	-1	-1	-1	-1	-1	0	0	-1
32	-1	0	-1	15	-1	56	-1	85	-1	5	-1	6	-1	52	-1	0	-1	-1	-1	-1	-1	0	0
-1	0	-1	47	-1	13	-1	61	-1	84	-1	55	-1	78	-1	41	95	-1	-1	-1	-1	-1	-1	0

Rate 3/4 A code:

6	38	3	93	-1	-1	-1	30	70	-1	86	-1	37	38	4	11	-1	46	48	0	-1	-1	-1	-1
62	94	19	84	-1	92	78	-1	15	-1	-1	92	-1	45	24	32	30	-1	-1	0	0	-1	-1	-1
71	-1	55	-1	12	66	45	79	-1	78	-1	-1	10	-1	22	55	70	82	-1	-1	0	0	-1	-1
38	61	-1	66	9	73	47	64	-1	39	61	43	-1	-1	-1	-1	95	32	0	-1	-1	0	0	-1
-1	-1	-1	-1	32	52	55	80	95	22	6	51	24	90	44	20	-1	-1	-1	-1	-1	-1	0	0
-1	63	31	88	20	-1	-1	-1	6	40	56	16	71	53	-1	-1	27	26	48	-1	-1	-1	-1	0

Rate 3/4 B code:

-1	81	-1	28	-1	-1	14	25	17	-1	-1	85	29	52	78	95	22	92	0	0	-1	-1	-1	-1
42	-1	14	68	32	-1	-1	-1	-1	70	43	11	36	40	33	57	38	24	-1	0	0	-1	-1	-1
-1	-1	20	-1	-1	63	39	-1	70	67	-1	38	4	72	47	29	60	5	80	-1	0	0	-1	-1
64	2	-1	-1	63	-1	-1	3	51	-1	81	15	94	9	85	36	14	19	-1	-1	-1	0	0	-1
-1	53	60	80	-1	26	75	-1	-1	-1	-1	86	77	1	3	72	60	25	-1	-1	-1	-1	0	0
77	-1	-1	-1	15	28	-1	35	-1	72	30	68	85	84	26	64	11	89	0	-1	-1	-1	-1	0

Rate 5/6 code:

$$\begin{bmatrix} 1 & 25 & 55 & -1 & 47 & 4 & -1 & 91 & 84 & 8 & 86 & 52 & 82 & 33 & 5 & 0 & 36 & 20 & 4 & 77 & 80 & 0 & -1 & -1 \\ -1 & 6 & -1 & 36 & 40 & 47 & 12 & 79 & 47 & -1 & 41 & 21 & 12 & 71 & 14 & 72 & 0 & 44 & 49 & 0 & 0 & 0 & 0 & -1 \\ 51 & 81 & 83 & 4 & 67 & -1 & 21 & -1 & 31 & 24 & 91 & 61 & 81 & 9 & 86 & 78 & 60 & 88 & 67 & 15 & -1 & -1 & 0 & 0 \\ 50 & -1 & 50 & 15 & -1 & 36 & 13 & 10 & 11 & 20 & 53 & 90 & 29 & 92 & 57 & 30 & 84 & 92 & 11 & 66 & 80 & -1 & -1 & 0 \end{bmatrix}$$

Chapter 4: Encoding / Decoding Methods

4.1 LDPC Direct Encoding

The LDPC code allows for many different code rates as described in Chapter 3. The process of encoding involves taking the original message $\mathbf{m} = (m_0, \ldots, m_{k-1})$ and generating parity-check bits $\mathbf{p} = (p_0, \ldots, p_{m-1})$ to add redundancy. These two blocks are sent and since the information block is contained in the codeword, the codeword is systematic. With the LDPC code, the encoder takes the information block and uses the H_{bm} matrix to find the parity bits. By expanding the H_{bm} matrix into the H matrix, the encoding can be done through matrix operations in the Galois field, GF(2).

By using the generator matrix **G** transformed from the **H** matrix, where $\mathbf{G}\,\mathbf{H}^T = 0$, the encoder can determine the parity bits. The generator matrix has dimensionality $k \times n$ and the information block of length k. Through the operation **mG**, the codeword **c**, a vector length n is found. The problem with using this method is that the encoding can become very complex. LDPC codes allow for easier encoding using the **H** matrix directly.

4.2 Encoding Method

First we need to define the components to be calculated. The purpose of encoding is to determine what the **p**, parity bits, will be formed from the H matrix. The information block is represented by **m**. The information block is divided into k_b groups of z bits where $k_b = n_b - m_b$. The information block **s** is denoted **u** where,

$$\mathbf{u} = [\mathbf{u}(0)\ \mathbf{u}(1)\ \ldots\ \mathbf{u}(k_b\text{-}1)],$$

where **u**(i) for all i is a column vector

$$\mathbf{u}(i) = [s_{iz}\ s_{iz+1}\ \ldots\ s_{(i+1)z-1}]^T$$

The parity bits, **p**, are denoted as **v** and are also grouped into z bits. The formation of this vector is similar to **u**.

$$\mathbf{v} = [\mathbf{v}(0)\ \mathbf{v}(1)\ \ldots\ \mathbf{v}(m_b\text{-}1)],$$

where each of these elements is also column vectors as follows,

where $\mathbf{v}(i) = [p_{iz}\ p_{iz+1}\ \ldots\ p_{(i+1)z-1}]^T$

4.2.1 Encoding Initialization

The initialization determines **v**(0). Through matrix operations by summing over all of H_{bm}, **v**(0) is found:

$$P_{p(x,kb)}\mathbf{v}(0) = \sum_{j=0}^{kb-1}\sum_{i=0}^{mb-1} P_{p(i,j)}\mathbf{u}(j) \qquad (4.1)$$

where x: $1 \le x \le m_b - 2$, which is the row in which there is an unpaired nonnegative value. P_i is the $z \times z$ identity matrix which is shifted.

4.2.2 Encoding Recursion

The other vectors are found through similar matrix operation. For finding **v**(1):

$$\mathbf{v}(1) = \sum_{j=0}^{kb-1} P_{p(i,j)}\mathbf{u}(j) + P_{p(i,kb)}\mathbf{v}(0),\ i = 0 \qquad (4.2)$$

And for the remainder of the vector, the following equation is used:

$$\mathbf{v}(i+1) = \mathbf{v}(i) + \sum_{j=0}^{kb-1} P_{p(i,j)}\mathbf{u}(j) + P_{p(i,kb)}\mathbf{v}(0),\ i = 1,\ldots, m_b\text{-}2 \qquad (4.3)$$

4.2.3 Back Substitution

Back substitution can be used if the H matrix is in the proper form. This also requires the matrix to be in row-echelon form:

$$H = [A\ L],$$

where A is a low density matrix

L is a lower triangle matrix: nonzero elements exist on and below the main diagonal and ones are on the main diagonal

Back substitution requires altering the parity-check matrix H to change into the proper form. By using simple row calculations, the matrix can be changed into the correct form. First, the last row must be replaced with the sum of all rows. This can be somewhat difficult in the IEEE 802.16e standard. Since the H_{bm} matrices are represented by different integer values, this requires expanding the circular shifts of the identity matrix then adding those matrices and not just adding the integer values. After accounting for

these calculations, the bottom row, which contains matrices, is moved to the top row and all others are shifted down once. Now the matrix is in back substitution form and can be used for finding the parity bits from information bits.

4.3 Decoding

Decoding involves taking the transmitted signal and trying to recover the original information bits. Since, there are different types of encoding procedures, decoding must be adaptive to handle these methods. A decoder can use different types of methods for finding the information.

4.3.1 Maximum Likelihood

Maximum likelihood decoding is one example. This method uses the conditional probability that a certain codeword was sent given a received vector. Using this logic, the decoder maximizes the following probability:

$$P(x \text{ received} \mid y \text{ sent})$$

4.3.2 Syndrome

Syndrome decoding uses the same theory as minimum distance decoding but uses a lookup table instead to find where the errors exist. Syndrome decoding uses the parity-check matrix, H, to help find the error. Using the property of the parity-check matrix,

$$H\mathbf{x} = 0,$$

where $\mathbf{x}$ is a codeword

The received signal can be represented as:

$$H\mathbf{z} = H(\mathbf{x} + \mathbf{e}),$$

where **z** is the received signal

e is the error vector

Through linearity and pre-computed values of H*e, the error values are found for all possible code words. With this table, the location of the error and corresponding codeword can be found and recovered.

4.3.3 Suboptimal Decoding Based On Graphs

The Tanner graph is a graphical representation of the parity-check matrix. It is a bipartite graph which consists of two sets of nodes and edges which connect the nodes. Considering the following block code:

$$H = \begin{vmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{vmatrix}$$

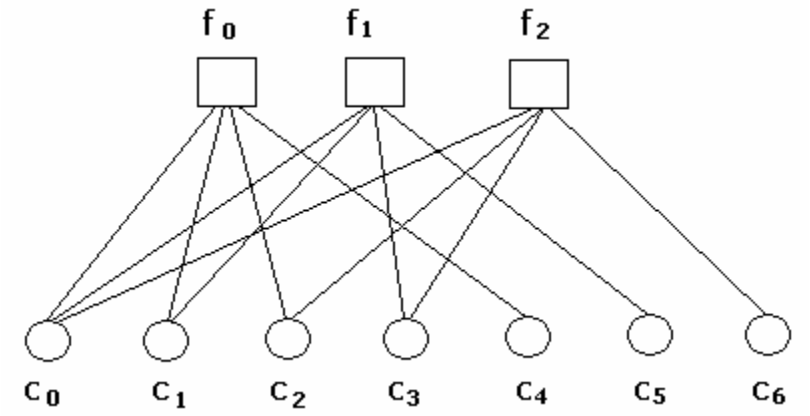

Figure 4.1 Tanner graph of (6, 4) block code

Each fi, check node, is a row and each ci, variable node, is a column in H. The check nodes are connected to the variable nodes wherever there are ones in the H matrix. For example, h0,0 = h0,1 = h0,2 = h0,4 = 1, so f0 is connected to c0, c1, c2, and c4. Also, h0,0 = h1,0 = h2,0 = 1 where c0 is connected to f0 and f1. From this you can see that the sum over a check node in GF (2) must be zero.

A *cycle* is compromised of l edges where a path begins and ends on the same node. For Figure 4.1, the length of the block code is 4 (ie. = f_0-c_0-f_1-c_1-f_0). The *girth* of the Tanner graph is the shortest possible path. A small girth has a negative impact on decoding. It is undesirable to have girth of length 4.

For decoding, you can take a look at the check nodes. For each codeword received calculate the check nodes in GF (2), the check nodes that sum to zero are correct and thus the variable nodes connecting to them are correct. From the remaining variable nodes, an elimination process must be done in order to find which code bit is in error. This will

show that one variable node is in error and needs to be flipped. This variable flipped will cause all the check nodes to sum to zero and thus the codeword is now correct.

Decoding on a Tanner graph is iterative, with messages being passed back and forth between the variable and check nodes. Such a decoder is called a message-passing decoder, and is implemented using the sum-product algorithm. The message passing algorithm is implemented in CML by the function MpDecode, which was used to produce the simulation results given in the next chapter.

Chapter 5: Simulation Results

5.1 Simulation Description

In order to find differences in code rates, we must first test the error rates of each of the H matrices using the different code rates. The rates as described by the IEEE 802.16e standard are as follows: rate = 1/2, 2/3 A, 2/3 B, 3/4 A, 3/4 B, 5/6. These correspond to the H_{bm} matrices listed in Section 3.2.

The model is based on a code length of n = 2304 for all code rates. Each of the numbers in the H_{bm} matrix identifies the number of circular shifts required to convert to the proper encoding H matrix.

With these simulations, BPSK modulation is used. Also the channel has additive white Gaussian noise. The simulations were produced using MATLAB and the CML library function MpDecode, which implements sum-product decoding.

5.1.1 Initial Encoding Testing

The encoding process initially used an all zeroes codeword to test the H_{bm} to H matrix conversion as well as testing the decoding procedure. First, the H_{bm} matrix was converted to the expanded H matrix. This process took the circular shifts of the p(i,j) values and converted them into circular shifts of the identity matrix. After the expanded

H matrix was found, the ones were identified. The ones' locations were compiled into a vector in both columns and rows labeled as variables h_rows and h_cols.

The initial test took an all zeroes codeword and modulated that codeword using BPSK modulation. This modulated codeword then is put through the channel by adding noise. With the use of the MpDecode function, using 30 iterations of the sum-product algorithm, in the CML library, an output of the number of errors is found. Using this number, the bit error rate (BER) is calculated. A break is created at BER $\leq 10^{-4}$ to prevent the code from running indefinitely.

Through running all the code rates through this process, the test for the H matrix expansion and the output of bit error rate and frame error rate were found without problems.

5.1.2 Encoding Random Codewords

After testing the initial code, the encoder needed to be expanded to test all possible codewords. To do this, the MpDecode function was again used. This required the location of the ones in the H_{bm} matrix. Similarly to the initial testing, this was found by expanding the H_{bm} matrix to the H matrix, where the circular shifts of the identity matrices are expanded. After H is created, the locations of the ones in H are stored in the variables h_rows and h_cols, which store the locations only in rows and columns respectively.

Next the back substitution form of the H_{bm} matrix must be determined. This is found through manipulation of the H_{bm} matrix. First, the last row must be found through matrix addition. So the last row is replaced by the sum of all rows. This addition is done in the expanded form. The circular shift indices are not added but the expanded identity matrices of those indices are added. Next, the last row is moved to the top of the matrix and all others are shifted down. Now, similar to the previous technique, the location of the ones is store in the variables h_rows_expanded and h_cols_expanded. This now is in back substitution form and can be used for encoding.

The encoder is run to create parity bits for the codeword. To find the length of the codeword, the message bits are divided by the rate of the code. Next the message bits are placed as the beginning of the codeword. By modulo 2 operations, only the locations the current codeword and the h_rows_expanded are the same are where the ones going to appear, all others will be zeroes. Using this method and solving for the unknown code bit will yield the value of that code bit. This process is repeated for every parity bit until the proper length of the codeword is achieved.

Through using this encoding method, the simulation can be achieved. With the use of MpDecode from the CML library, the encoded signal with noise can be decoded. The signal is sent with through an AWGN channel. The signal spans a range of SNR values. A simulation plot through finding the error rates can be found.

5.2 Simulation Results

From Figure 5.1 and Figure 5.2, as the code rate increases, the performance worsens. The figures were plotted using a logarithmic scale on the y axis, BER or FER. The number of decoder iterations was 30. Also there were 100 total errors per SNR point. The SNR points ranged from -1 to 8 or until the bit error rate became lower than 10^{-4}. The codewords were randomly generated MATLAB's rand function and rounded to zero or one. The AWGN channel was created using MATLAB's randn function multiplied by the square root of the variance. All of the H_{bm} matrices are tested with the different code rates are run through this simulation:

BER of all rates described by IEEE 802.16e standard: Figure 5.1

FER of all rates described by IEEE 802.16e standard: Figure 5.2

5.3 Conclusion

The LDPC code in 802.16e is described in terms of the parity-check matrix instead of its generator matrix. The generator matrix method for solving LDPC codes can become quite complicated. The LDPC codes are defined for much less complication by directly encoding from the parity-check matrix. Some manipulations had to be performed in order to encode the message. This project demonstrates how the parity-check matrix can be used to directly encoding a message. Also the variation of code rates described by the Hbm matrices show the strength of the 1/2 rate code compared to 5/6 rate code.

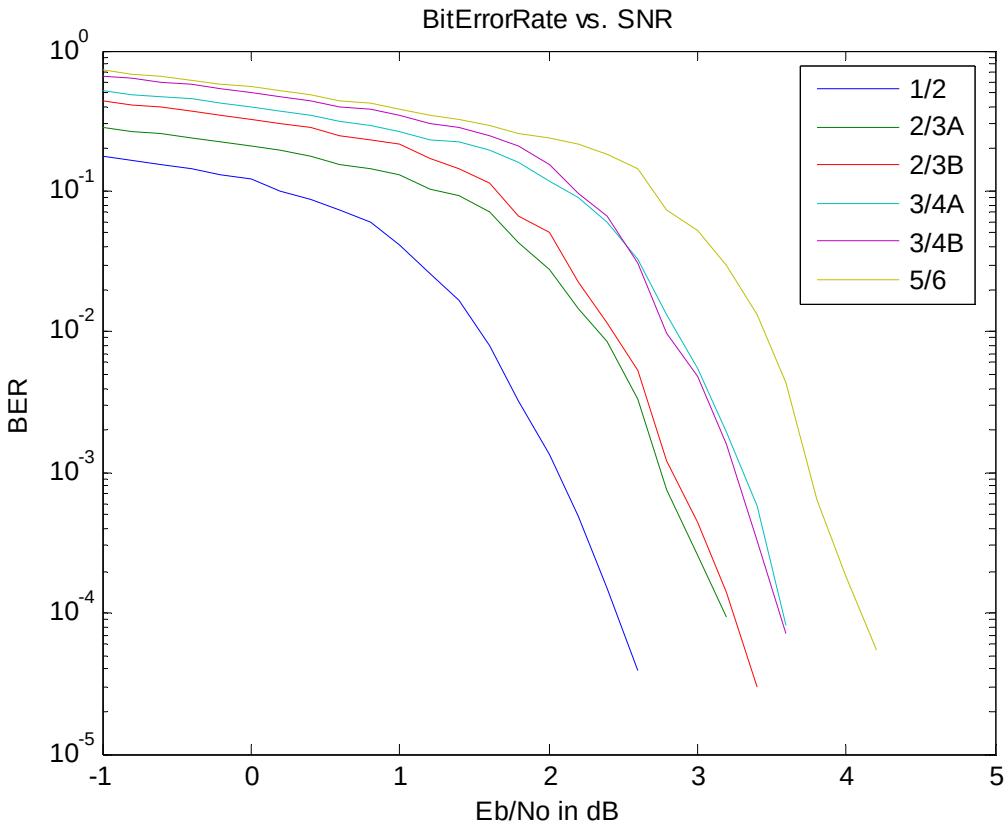

Figure 5.1 – Bit Error Rates vs. SNR using 30 iterations of sum-product decoding for code rates: 1/2, 2/3A, 2/3B, 3/4A, 3/4B, 5/6 (n = 576)

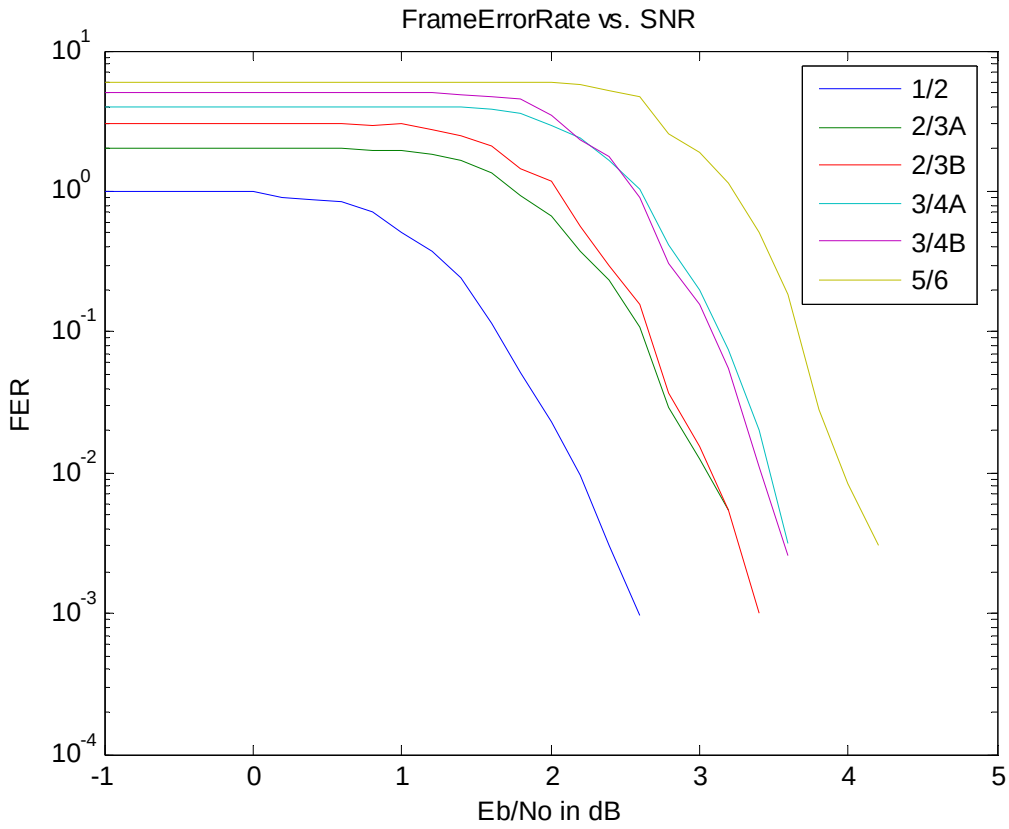

Figure 5.2 – Frame Error Rates vs. SNR using 30 iterations of sum-product decoding for code rates: 1/2, 2/3A, 2/3B, 3/4A, 3/4B, 5/6 (n = 576)

References

[Ga63] Gallager, R. G., *Low Density Parity Check Codes*, Monograph, M.I.T. Press, 1963.

[Go05] Goldsmith, Andrea. *Wireless Communications*. New York, NY: Cambridge UP, 2005.

[IE06] "IEEE Standard for Local and Metropolitan Area Networks Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems." 802.16e-2005 and IEEE Std 802.16-2004/Cor1-2005 (2006): 626-630, 819-822.

[La98] Lathi, B P. *Modern Digital and Analog Communication Systems*, Oxford University Press, 1998.

[La07] Laudon, Kenneth C., and Jane P. Laudon. *Management Information Systems*. Prentice Hall, 2007.

[Le03] Lee, Dr. Lin-Nan. LDPC Codes, Application to Next Generation Communication Systems, Hughes Network Systems, 2003.

[Le00] Leon-Garcia, Alberto, and Indra Widjaja. *Communication Networks*. Boston, MA: McGraw Hill, 2000.

[Co04] Lin, Shu, and Daniel J. Costello, Jr. *Error Control Coding Second Edition*. Upper Saddle River, NJ: Pearson Prentice Hall, 2004.

[Pr06] Prakash Agrawal, Dharma, and Qing-An Zeng. *Introduction to Wireless and Mobile Systems*. Toronto: Thomson, 2006.

[Ri01] Thomas J. Richardson and Rüdiger L. Urbanke, “Efficient Encoding of Low-Density Parity-Check Codes”, *IEEE Transactions in Information Theory*, 47(2), February 2001

[Sh48] C. E. Shannon, ``A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp. 379-423 and 623-656, July and October, 1948

[Zi01] Ziemer, Rodger E., and Roger L. Peterson. *Introduction to Digital Communication*. Upper Saddle River, NJ: Prentice Hall, 2001.