COMMUNICATION SYSTEMS DESIGN FOR DOWNHOLE ACOUSTIC TELEMETRY

A Dissertation

by

AHMED REDISSI

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Scott Miller
Committee Members,	Tie Liu
	Raffaella Righetti
	Ding Zhu
Head of Department,	Miroslav Begovic

May 2019

Major Subject: Electrical Engineering

ABSTRACT

The goal of this dissertation is to design a reliable and efficient communication system for downhole acoustic communication. This system is expected to operate in two different modes. A broadband high data rate mode in case of transmission of an image or a video file and a narrowband low data rate mode in case of transmission of sensor readings. This communication system functions by acoustic vibration of the pipes and uses them as the channel instead of installing long cables in areas that are hard to reach. However, this channel has unique characteristics where it exhibits several passbands and stopbands across the frequency spectrum. The communication system is expected to get around those challenges in both modes of operation. In the broadband case, the system uses Orthogonal Frequency Division Multiplexing to transmit data across multiple orthogonal frequencies spanning multiple passbands combined with an error-correction code to recover some of the losses caused by the channel. In the narrowband case, a short packet is transmitted at a low data rate where the signal spectrum can fit inside one passband. However, transmitting short packets induces a new synchronization problem. This dissertation investigates and explores in detail the problem of synchronization on short packets where each synchronization stage is examined. A simple algorithm that exploits the presence of error-correction codes is proposed for the frame synchronization stage and demonstrated to approach the optimal solution. Then, all synchronization stages are combined in order to study the effect of propagated errors caused by imperfect synchronization from one stage to the next and what can be done in the design of the

packet and the receiver structure to mitigate those losses. The resulting synchronization procedure is applied to the pipe strings and demonstrated to achieve desirable levels of performance with the assistance of equalization at the receiver.

DEDICATION

To Mom and Dad, thanks for the endless love and support.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supervised by a dissertation committee consisting of Professors Scott Miller, Tie Liu, and Rafaella Righetti of the Department of Electrical and Computer Engineering and Professor Ding Zhu of the Department of Petroleum Engineering.

All other work conducted for the dissertation was completed by the student independently.

Funding Sources

Graduate study was supported by an assistantship from Texas A&M University.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

CHAPTER I

INTRODUCTION

Motivation and Background

Downhole acoustic telemetry is a process in which acoustic waves are used as means to carry information and measurements taken at inaccessible or remote areas such as wells. The process of telemetry in general is used in multiple fields and industries such as oil and gas, medicine, agriculture, water management, energy monitoring, transportation, mining, retail, and so on. When it comes to the oil and gas industry in particular, acoustic telemetry is extremely beneficial in three ways. First, it allows for the measurement of important parameters such as depth, pressure, temperature, flow, porosity, and viscosity. Obtaining these measurements is crucial in determining how much oil and natural gas can be extracted from a well. Acoustic telemetry offers access to the well by extracting all of this information from an area that is generally inaccessible or potentially hazardous for humans. Second, it allows for around the clock monitoring and inspection of pipes of equipment. For example, acoustic waves can be used to check for leaks and corrosion in a pipe by vibrating the pipe and observing the resulting modes. Third, acoustic telemetry offers a cheap and reliable alternative to long, expensive, and high maintenance cables that would have to be installed in these remote

areas. Instead, the acoustic wave will be carrying the information needed while the pipes act as a wireless channel.

At this point, there needs to be a reliable and energy efficient communication system that carries the required information from a sensor placed at the bottom of the well to the technicians and engineers at the surface so they can analyze the data and make decisions. The type of information that needs to be transmitted can be either a sensor measurement or a captured image or video. Each one requires a different communication system because of the channel characteristics and the different metrics for evaluating energy efficiency. The sensor reading consists of a real number converted into a short sequence of bits while the image or the video are converted into a much longer sequence of bits. The sensor readings can be transmitted using a low data rate single carrier system while the image or video is better suited for a high data rate system with a wide bandwidth such as Orthogonal Frequency Division Multiplexing (OFDM) [1]. As for energy efficiency, the metric is the word error rate for short packets and bit error rate for long packets. The reasoning behind using two different metrics is explained as follows. When a sensor reading is converted into a bit sequence, a single error in this sequence will result in an entirely different number. Hence, the entire packet must be transmitted and received correctly. However, when an image is converted into a much longer bit sequence, an error in a few bits will not result in an entirely different image. Each of the proposed communication systems has a unique challenge. For the low data rate case, synchronization is a major concern and is extremely crucial for the correct transmission of the packet. For the high data rate case, the challenge is overcoming the

presence of the stopbands by either avoiding them or compensating for the losses they cause.

This dissertation is organized as follows. First, a survey of energy efficient communication systems and synchronization algorithms is conducted in the literature. Second, the channel response is fully characterized in chapter II. Then in chapter III, an OFDM communication system for the long packet case is proposed. Chapters IV, V, VI, and VII treat the design of signaling format and the synchronization problem for the short packet case. Finally, all synchronization stages are combined in chapter VIII to study the effect of propagating errors and how to limit them.

Literature Review

There has been a significant amount or works in the literature dealing with each of the topics of this dissertation. We will focus on the works that deal with energy efficiency and synchronization for short packets. When it comes to the topic of energy efficient communication combined with error correction codes [2] that can be applied to short packets, the following works are noteworthy. The authors of [3] attempt to find balance between energy efficiency and bandwidth efficiency for short packet communications. The authors of [4] combine erasure-correction and error-correction decoding for a more reliable transmission and studies the tradeoff between relying on one over the other. The works of [5,6] find bounds on coding rates for short packet transmission over Rayleigh fading channels in addition to multiple antennas transmission schemes. A review of short packet communication techniques is provided in [7]. The parameters that affect efficiency of a communication system are identified in

[8]. The author of [9] finds bounds on the best possible probability of error for coded systems in a Gaussian channel while the authors of [10] finds the best possible probability of error as a function of the block size. The work of [11] models a channel with synchronization errors as a duplication, deletion, and substitution channel. Finally, [12] finds the block error probability of convolutional codes treated as block codes.

When trying to look for answers on synchronization for coded short packets, some works in the literature mentioned an entirely unique class of codes designed for synchronization known as comma-free codes. The author of [13] talks about error correcting codes where any code formed by the concatenation of the tail of one code with the beginning of the next code is not a valid code. The author of [14] provides a guideline for creating comma-free codes with the maximal number of words. Then, [15] constructed high rate comma-free codes by combinatorial design. However, comma-free codes are only useful when trying to distinguish between two consecutive codes. Our problem consists of performing the different levels of synchronization on a single transmitted code surrounded by noise. Therefore, we needed to find more answers.

For coarse and symbol synchronization on short packets, no sufficient answers were found in the literature while the standard synchronization techniques such as Gardner [16], Early-Late [17], and Mueller-Muller [18] did not offer a satisfying performance over an energy-based detection method. For phase synchronization, [19] proposes a phase estimation technique that uses a pilot symbol as an initial guess and improves on it by iterative soft decision decoding. The work of [20] uses Monte Carlo methods to estimate the distribution of the unknown phase and proposes several

approximations to simplify the algorithm. Carrier and phase synchronization of short packet turbo coded signals is proposed in [21] by maximum likelihood iterative soft decision or maximizing the Mean Squared Soft Output cost function. Then, [22] resolves the phase ambiguity using Hypothesis Testing on different frame synchronization algorithms.

Meanwhile at the frame synchronization stage, there has been a significant number of works that can be divided in two categories. The first category is for works that utilize a known pilot sequence or a sync word to perform the synchronization operation. For example, [23] performs frame synchronization by correlating the received signal with the pilot sequence and choosing the time delay that maximizes this correlation. A synchronization technique is proposed in [24] and provides several most likely frame starting positions and narrows it down to one. This is known as the List Synchronizer. Then, [25] achieves Maximum a Posteriori (MAP) frame synchronization using packet energy and the sync word to find the packet starting location. The work of [26] derives the Maximum Likelihood (ML), correlation, and high Signal-to-Noise Ratio (SNR) rules for frame synchronization in Additive White Gaussian Noise (AWGN) channels while [27] repeats the same exercise for flat fading channels. Then, [28] provides lower bounds for ML, correlation, and high SNR frame synchronization decision rules in AWGN channels. The author of [29] proposes improving frame synchronization by terminating the trellis at the all zero state when using convolutional codes. In [30], frame synchronization is performed by treating the received symbols as a Markov chain in AWGN.

The second category of works exploits the presence of error correction codes to aid the synchronization. For instance, [31] proposes an algorithm that uses the code structure and combining two previous algorithms. This algorithm uses mode separation to estimate the frame boundary where a Log-Likelihood Ratio (LLR) is computed and its distribution is plotted. The correct frame starting position is the one with the most bimodal distribution. A frame synchronization method that inserts the pilots as a mid-amble instead of the preamble is proposed in [32] and uses the decoder to determine the frame location as the mid-amble will be recognized later at the decoding stage. The work of [33] performs blind frame synchronization by checking if the observed frame is a valid code word and [34] uses the same idea on cyclic codes. A synchronization algorithm is developed in [35] and uses the code structure to eliminate the need for overhead through Factor Graphs and the Sum-Product Algorithm [36]. The authors of [37] implement code-aided frame synchronization using decoder decisions as MAP probabilities for Low Density Parity Check (LDPC) and Turbo codes and the authors of [38] use the structure of Turbo codes to perform frame synchronization without preamble. An iterative receiver without preamble using Expectation Maximization and the Sum-Product Algorithm is introduced in [39]. A comparison of different algorithms that use the code structure for frame synchronization and phase ambiguity resolution can be found in [40] while [41] proves that frame synchronization techniques using the code structure with no overhead coincide with the MAP frame synchronization under certain conditions. Finally, [42] proposes a blind MAP frame synchronization method for codes having a sparse parity check matrix and analyzes it further in [43].

CHAPTER II

CHANNEL CHARACTERIZATION

Before we design a communication system, we need to find out what the channel response looks like. The channel in this case is a sequence of steel pipes that were interconnected by joints. These pipes will be acoustically vibrated by a transmitter. Then, several receivers and relays will be placed along the way to pick up the signal and relay it further until its final destination. It was proposed in [44] and experimentally verified in [45] that these pipes have frequency selective properties with certain fixed frequency passbands and stopbands. If a signal is transmitted at a frequency that falls in a stopband, it will be effectively lost. While if it is transmitted at a frequency that falls in a passband, the message will go through but might suffer from a magnitude loss whose strength depends on which passband was chosen in addition to a phase shift.

The experimental results of [45] showed that the acoustic vibration of the pipe strings excites three vibrational modes. The first mode is the longitudinal L(0,1) mode where vibrations take place in the radial and azimuthal directions of the pipes. The second is the torsional T(0,1) mode where vibrations take place in the azimuthal direction. The last is the flexural F(1,1) mode where vibrations take place in all three radial, longitudinal, and azimuthal directions. It was also shown in these experiments that the L(0,1) mode is the most dominant in terms of amplitude clearly overwhelming the others two.

Assuming longitudinal excitation only, we ran a simulation in order to visualize the channel response. In this simulation, an OFDM signal with Binary Phase Shift Keying (BPSK) subcarriers composed of a known sequence of +1s and -1s was transmitted. This way, we set up the received signal to easily extract the channel's frequency and phase response. If x is the transmitted signal and y is the received signal in a noise-free environment, then the channel gain $h(f)$ and the phase offset $\theta(f)$ for each frequency can extracted in the following manner:

$$y(t) = h(f)x(t)e^{j(\theta(f)+angle(x(t))} \tag{2.1}$$

Since x is a known sequence of +1s and -1s and the channel gain and phase offset are constant in time, then:

$$y(t) = \begin{cases} h(f)e^{j\theta(f)}, if\ x(t) = 1 \\ -h(f)e^{j(\theta(f)+\pi)}, if\ x(t) = -1 \end{cases} \tag{2.2}$$

$$h(f) = |y(t)| \tag{2.3}$$

$$\theta(f) = \begin{cases} angle\big(y(t)\big), if\ x(t) = 1 \\ angle\big(y(t)\big) - \pi, if\ x(t) = -1 \end{cases} \tag{2.4}$$

The bandwidth of this signal spans the frequencies between 500Hz and 3kHz with a frequency spacing of 2Hz for each OFDM subcarrier. This signal was allowed to propagate through a sequence of 100 steel pipe segments interconnected by joints. The parameters of these pipes and joints are summarized in table 2.1.

Table 2.1: Dimensions of the pipe segments and joints

	Outer Diameter (mm)	Thickness (mm)	Length (m)
Pipe Segments	73.66	3.81	9.86665
Joints	95.25	15.87	0.13335

The signal was then received at the beginning of the 10^{th},20^{th},30^{th},40^{th}, and 50^{th} pipe segments out of the total 100 segments used.

We will look at the channel's frequency response, phase response, and demonstrate how the different reflections from the beginning and end of the pipes and all the joints along the way interfere with the desired signal and cause Inter-Symbol-Interference (ISI). The magnitude of the received symbols was considered to be the channel frequency response while their phase was taken as the channel phase response after subtracting 180° in case the transmitted symbol had a negative sign.

Frequency Response

Figure 2.1 shows what the channel frequency response looks like when the signal is measured at the beginning of the 10^{th} pipe segment. It clearly shows the presence of passbands and stopbands as they were expected to occur.

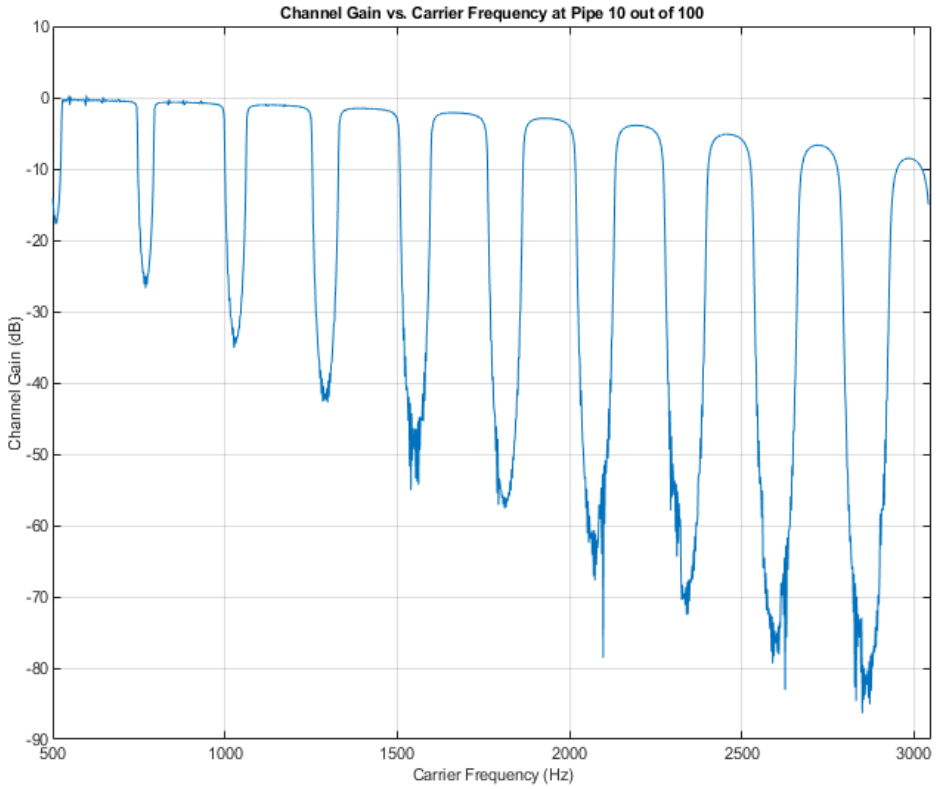

Figure 2.1: Channel response measured at pipe no. 10 out of 100 pipe segments

We can clearly observe from figure 2.1 that as the carrier frequency increases the passbands get narrower and weaker and that the stopbands get wider and deeper. First, let's examine how the width of the passbands and stopbands change. Tables 2.3 and 2.4 and figure 2.2 fully describe this behavior. The largest passband has a width of 214Hz and the next passbands shrink in an almost linear fashion. The exact opposite happens to the stopbands where the first stopband is 54Hz wide and the next ones linearly increase in width.

Table 2.2: Approximate width of the passbands measured at pipe no. 10 out of 100

Passband No.	1	2	3	4	5	6	7	8	9	10
Beginning Frequency (Hz)	528	796	1064	1332	1600	1872	2138	2406	2678	2940
Ending Frequency (Hz)	742	994	1248	1504	1758	2006	2266	2522	2774	3030
Bandwidth (Hz)	214	198	184	172	158	134	128	116	96	84

Table 2.3: Approximate width of the stopbands measured at pipe no. 10 out of 100

Stopband No.	1	2	3	4	5	6	7	8	9
Beginning Frequency (Hz)	742	994	1248	1504	1758	2006	2266	2522	2774
Ending Frequency (Hz)	796	1064	1332	1600	1872	2138	2406	2678	2946
Bandwidth (Hz)	54	70	84	96	114	132	140	156	172

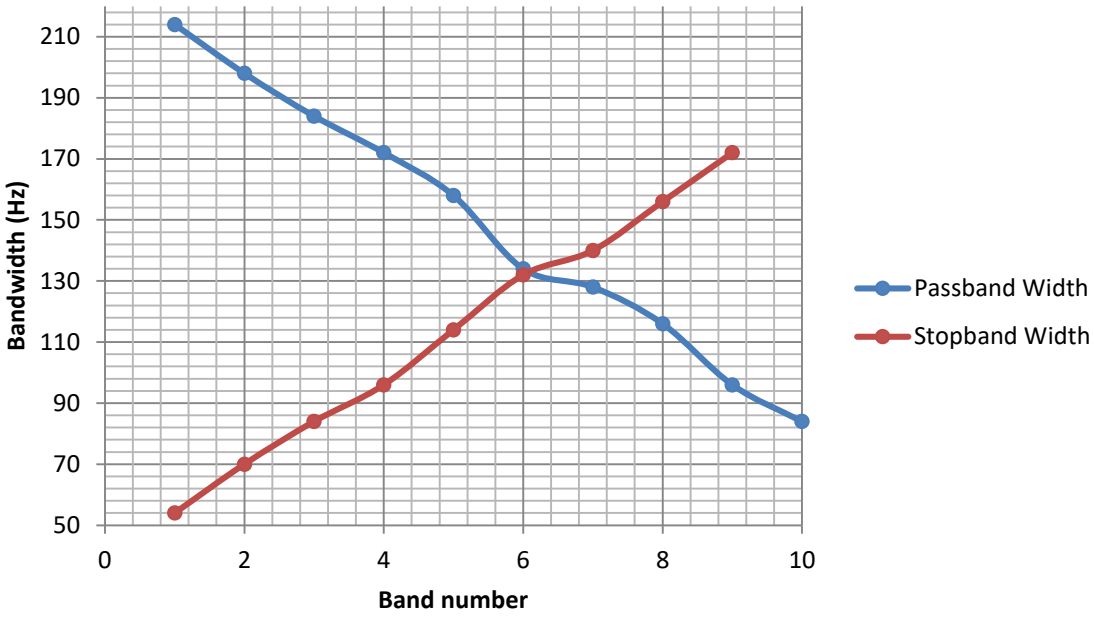

Figure 2.2: Passband and stopband width vs. carrier frequency

The strength of the passbands and the depth of the stopbands also change both in frequency and distance as we move further down the sequence of pipes from the 10th pipe to the 50th pipe. Tables 2.4 and 2.5 and figures 2.3-2.6 describe this behavior. In terms of distance, the same passband experiences an exponential (linear in dB) decay in strength as the measurement location goes farther down the pipe strings. In a similar fashion, stopbands will linearly gain more depth. In terms of frequency and at the same distance, the passbands will exponentially lose strength as the frequency increases. On the other hand, the stopbands will linearly get deeper for higher frequencies.

Table 2.4: Passband strength (dB) vs. distance

Location\Passband No.	1	2	3	4	5	6	7	8	9	10
10	-0.5	-0.75	-1	-1.5	-2.15	-3	-4	-5.15	-6.7	-8.5
20	-0.85	-1.4	-2.1	-3	-4.3	-5.8	-7.8	-10.25	-13.3	-17
30	-1	-2	-3.2	-4.5	-6.4	-8.7	-11.7	-15.4	-20	-25.4
40	-1.4	-2.6	-4.1	-6	-8.5	-11.2	-15.55	-20.4	-26.6	-34
50	-2.5	-4	-6	-8	-10.8	-14.6	-19.4	-25.5	-33.2	-42.35

Table 2.5: Stopband depth (dB) vs. distance

Location\Stopband No.	1	2	3	4	5	6	7	8	9
10	-26.5	-35	-42.8	-51	-57.5	-68	-72	-79	-85
20	-43	-54	-66	-63.6	-84	-85	-97.25	-102	-110.5
30	-45.5	-54.4	-75.5	-74.75	-85.4	-89.45	-105.5	-112.2	-124.1
40	-57.65	-55.1	-72.8	-74	-99.15	-98	-115.8	-123.2	-138.6
50	-52.2	-61	-86.35	-89.55	-102.55	-109.7	-127.5	-135.3	-153.75

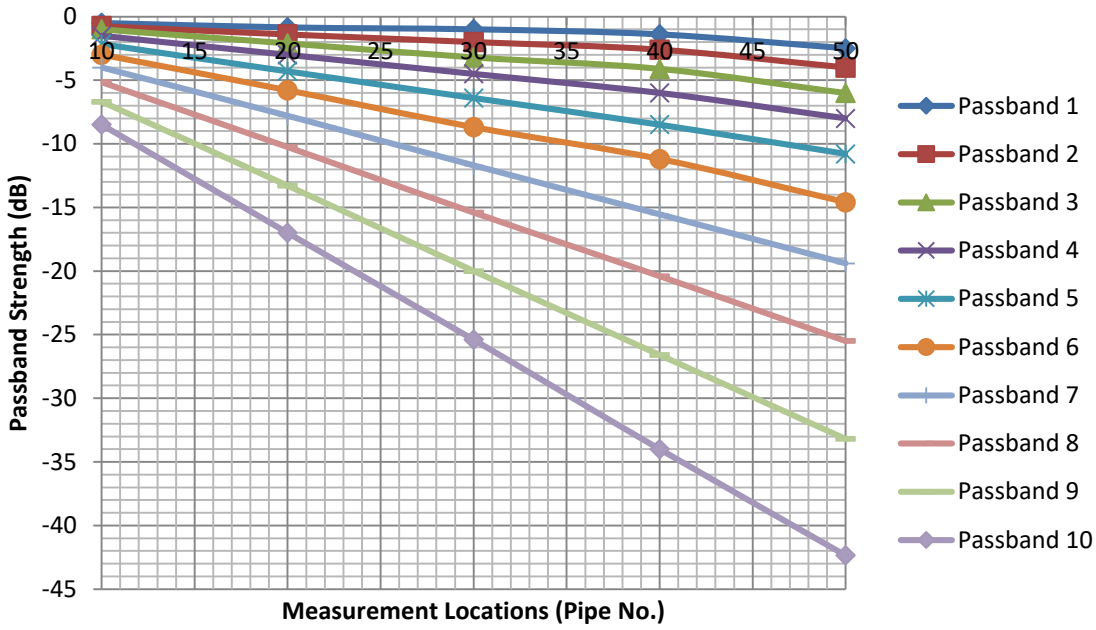

Figure 2.3: Passband strength vs. distance

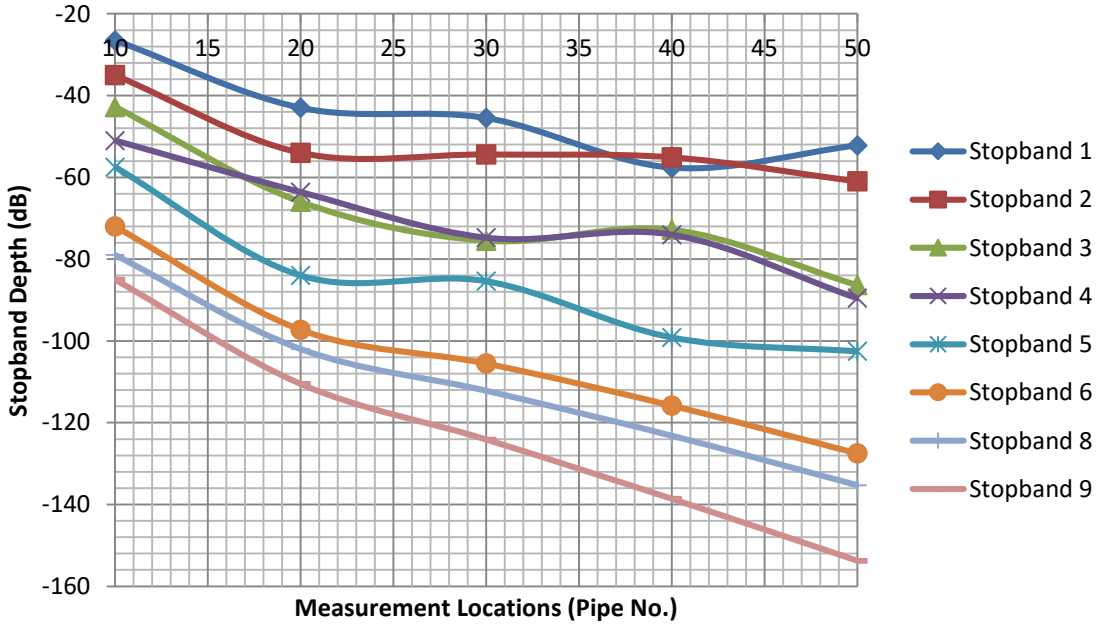

Figure 2.4: Stopband depth vs. distance

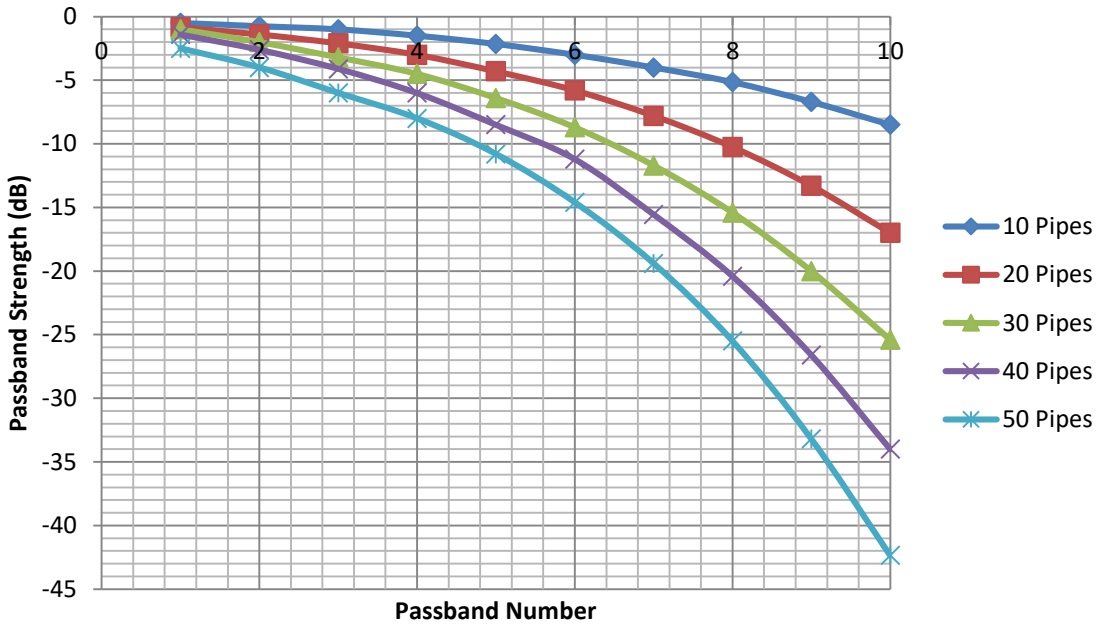

Figure 2.5: Passband strength vs. frequency

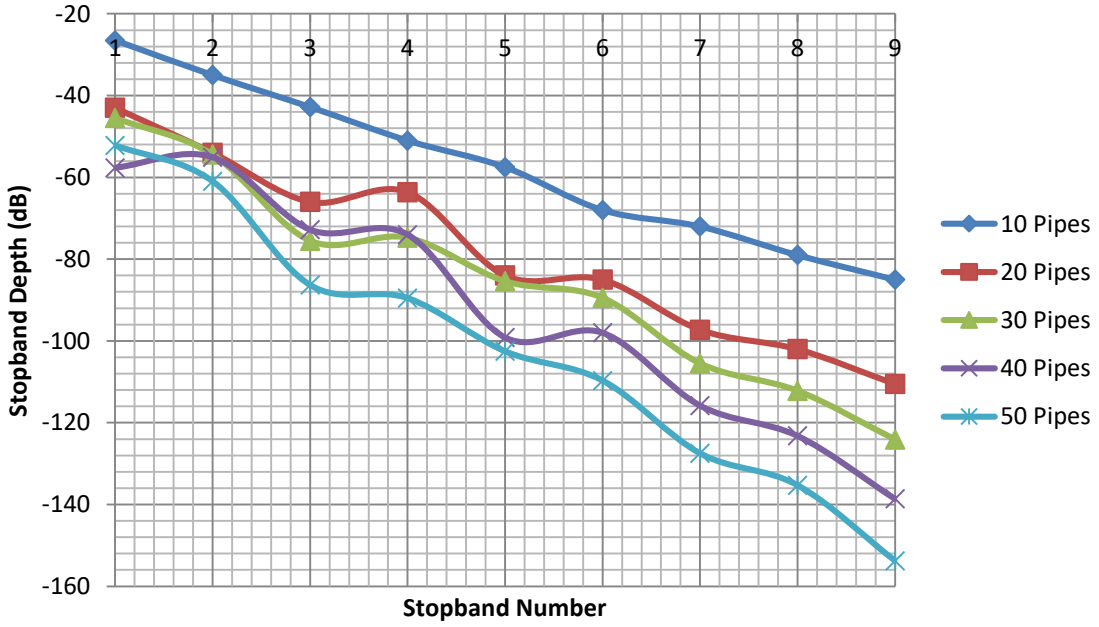

Figure 2.6: Stopband depth vs. frequency

Phase Response

In addition to the frequency response, the channel also introduces an unknown phase offset to all symbols transmitted at the same frequency. Unlike the frequency response, the phase response of the channel is not fixed and very difficult to predict. Any inaccuracy in the estimation of the time delay, can result in a different phase offset. Figure 2.7 shows an example of this phase offset where the signal constellation was either rotated by about 45˚ counter clockwise or by about 135˚ clockwise. In this figure, the dots represent the noisy received symbols for a BPSK signal.

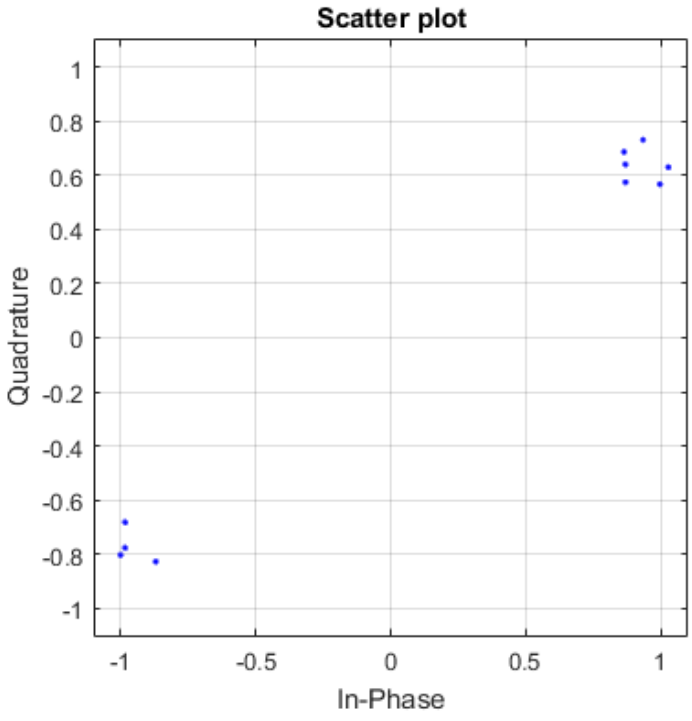

Figure 2.7: Phase offset induced by the channel

Effect of Measurement Location

We also test the effect of the measurement location within the same pipe segment on the channel gain and phase offset. We used 50 steel pipes whose dimensions are the same as in table 2.1 and transmitted the same signal as before. The measurements were performed at the end and the middle of the 25^{th} pipe segment and the channel gain is shown in figures 2.8 and 2.9 for the end and middle measurements respectively. While the locations of the passbands and stopbands in figures 2.8 and 2.9 are the same, the shape and strength of the passbands are different. For the measurements at the end of the pipe segment, the passbands are somewhat flat. Meanwhile for the measurements at the middle of the pipe segment, the passbands are slanted and offer a positive (in dB) channel gain that can amplify the signal if those frequencies were used for transmission.

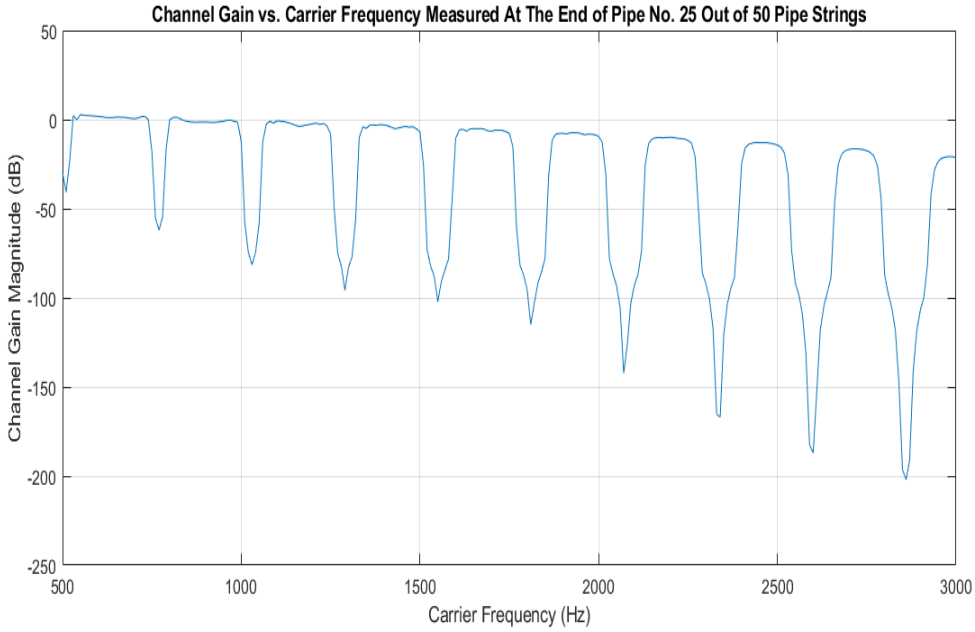

Figure 2.8: Channel gain vs. carrier frequency measured at the end of pipe no. 25 out of 50 pipe segments

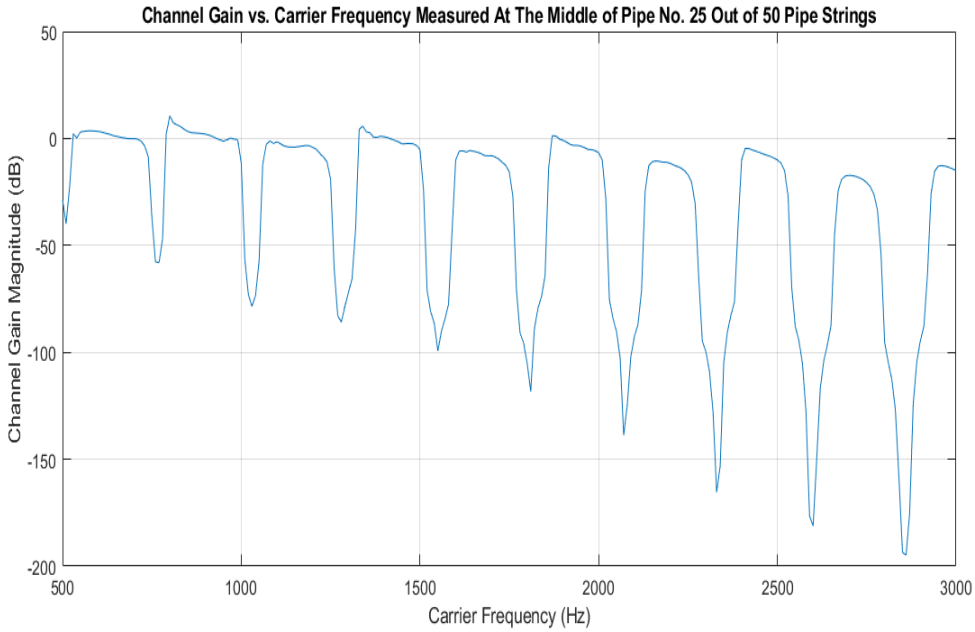

Figure 2.9: Channel gain vs. carrier frequency measured in the middle of pipe no .25 out of 50 pipe segments

If we repeat the same experiment and make the measurements at the 10th pipe segment instead of the 25th segment, we get a similar shape but we start to see some fluctuations show up in the first passbands for both frequency and phase responses as shown in figure 2.10 and 2.11. This behavior was predicted and explained in [44]. For n pipe segments, we expect to see n fluctuations because each of the pipe segments requires a frequency change to fit an extra half of a wavelength causing a resonance at those frequencies. Since the strength of the passbands decays as a function of distance and frequency, these fluctuations will in turn lose strength until they disappear.

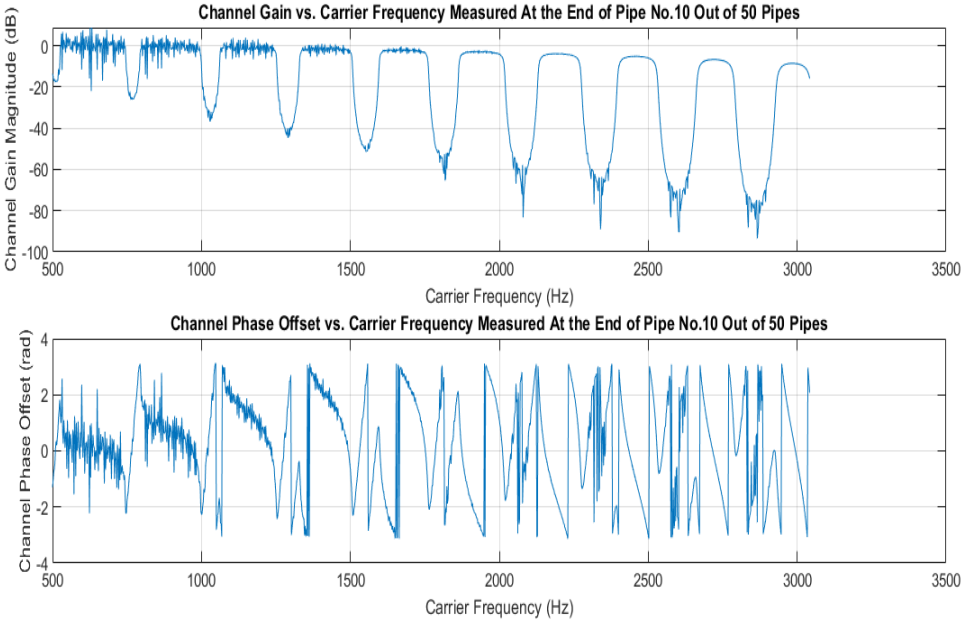

Figure 2.10: Channel gain and phase offset vs. carrier frequency measured at the end of pipe no. 10 out of 50 pipe segments

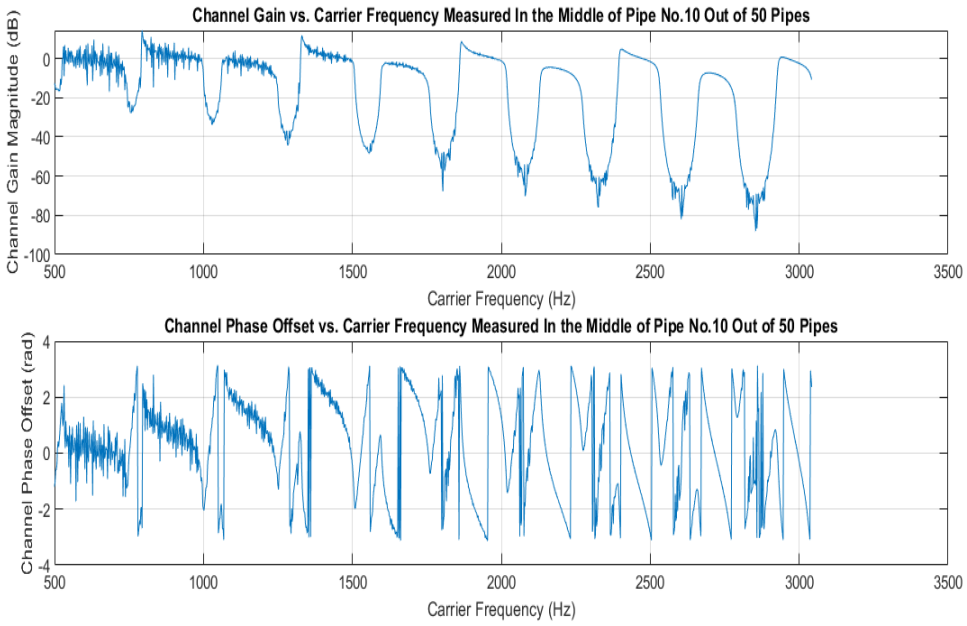

Figure 2.11: Channel gain and phase offset vs. carrier frequency measured in the middle of pipe no. 10 out of 50 pipe segments

Inter-Symbol-Interference

Since the acoustic wave is used to vibrate the pipes and carry the information, we expect to see several reflections at the receiver that can be placed in two categories. The first category is the reflections coming from the joints and those arrive much sooner but are weaker. The second category is the reflections coming from the end and the beginning of the pipe strings and they arrive later but are stronger than the joints reflections. Depending on the measurement location, these reflections can arrive before the end of the desired signal and cause an unwanted interference. The reflected waveform representing a transmitted symbol will interfere with the desired waveform representing another symbol causing a distortion. This is known as Inter-Symbol-

Interference. Figure 2.12 shows this behavior and the presence of reflections for the transmitted signal in a noise-free environment.

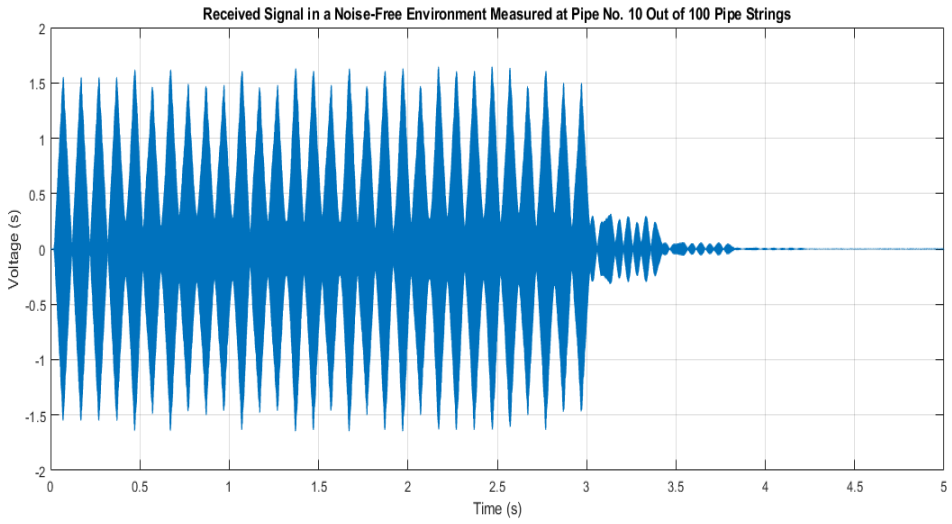

Figure 2.12: Received signal in a noise-free environment

Given the channel's frequency selectivity, we propose two different solutions for the two communication modes of our application. For the broadband case where an image or a video is transmitted, we will design an OFDM signal whose bandwidth spans multiple passbands and avoids all the stopbands in-between. For the narrowband case where sensor readings consisting of very short packets are periodically transmitted, we will design a single carrier system whose bandwidth fits inside one passband and find the synchronization algorithms that give us the best possible performance with minimal complexity.

CHAPTER III

HIGH DATA RATE COMMUNICATION SYSTEM

Application Requirements

Before we can design this communication system, we need to know the required data rate to support this application. We would like to be able to transmit either a short video or small image. The design constraint is that the signal should not last more than one minute in order to quickly receive the video or the image. However, the spectrum that we need to use is limited to 2.5-3kHz and half of which is unusable stopbands. We also noticed that the passbands quickly decay in amplitude as the bandwidth increases. This means that a significant portion of the signal will suffer from a major loss and will require a lot of equalization and signal processing to recover the losses. In order to avoid the high receiver complexity, we propose using an OFDM system that allows for high data rates while minimizing receiver complexity. This system eliminates the need for equalization because it splits the broadband frequency selective channel into much narrower sub-channels where the bandwidth of each sub-channel is sufficiently small that each sub-channel behaves in a non-frequency-selective manner.

The total bandwidth of the passbands is approximately 1484Hz. In an OFDM system, this translates to about 1484 bits/sec if we intend to use all of them. In table 3.1, we summarize the application specifications and the required data rate for the signal to not last more than one minute.

Table 3.1: Required data rate for each proposed application

Application	Compression Ratio	Required Data Rate (bits/sec)
5 Seconds Video (240p, 256 colors, 24 frames/sec)	30:1	44800
Image (480x480 pixels, 256 colors)	10:1	3072
Image (480x480 pixels, 16 colors)	10:1	1536

We can immediately see that the video application is not achievable for this channel as it requires a signal constellation with more than 2 billion points if we intend to use all passbands in the spectrum. However, the colored images are achievable and we will go through the design process for the communication system for each one of them in the next sections.

Signaling Format

In an OFDM system, the transmitted symbols are split across both time and frequency such that those frequencies, often referred to as subcarriers, are orthogonal. The spacing between those frequencies is chosen such that the channel response in that very narrow bandwidth is flat which eliminates the need for equalization at the receiver. The orthogonality of those frequencies allows for their corresponding waveforms to be

transmitted at the same time without any self-interference. For a given bit rate $R_b = \frac{1}{T_b}$ where T_b is the bit duration and a number of subcarriers N_c, the frequency spacing is proportional to the bit rate:

$$\Delta f \propto \frac{1}{N_c T_b} = \frac{R_b}{N_c} \tag{3.1}$$

Then, the transmitted symbols can be split between time and frequency as shown in figure 3.1.

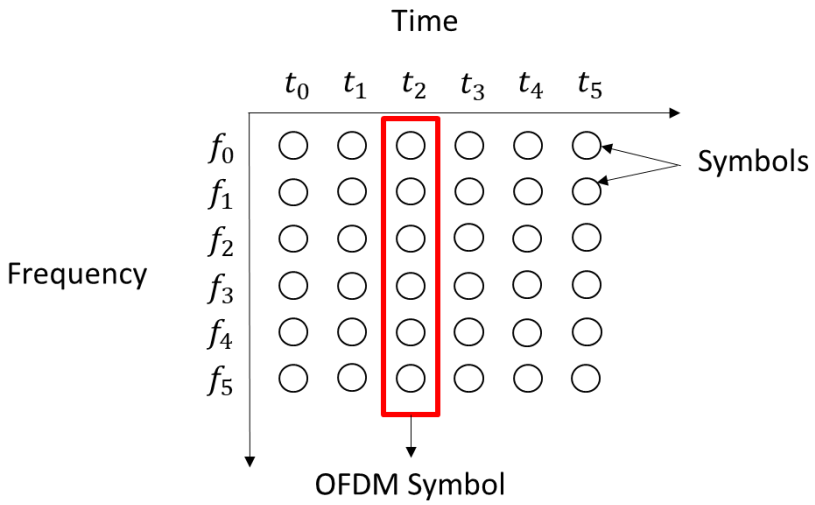

Figure 3.1: Splitting of symbols in an OFDM system

We need carefully choose the carrier frequency and the data rate to match the width and location of the used passbands. This can be done by placing the carrier frequency at the start of the first used passband and choosing the data rate such that the signal bandwidth ends at the edge of the last used passband. Since the strength of the

passbands decays as a function of frequency, it will be more energy-efficient to start with the first passband in the frequency range of interest and extend the signal bandwidth to the next set of passbands that will be used. In order to avoid data falling in a stopband, we will place zeros in the frequencies that fall in the stopband. Since the passbands and stopbands locations are fixed in time, the receiver can easily pick off the transmitted data from the desired frequencies.

The OFDM transmission can be achieved by following the procedure outlined in figure 3.2. The inverse Fast Fourier Transform is used to place the transmitted symbols at the desired frequency spacing. Once that is achieved, a portion is copied from the end of each OFDM symbol and placed before the same OFDM symbol in time. This procedure is known as inserting a cyclic prefix and it is done to combat ISI where a waveform representing one symbol interferes with and distorts the waveform representing another symbol. Finally, the signal is mounted on a carrier frequency to create the waveforms with the desired orthogonal frequencies and the OFDM symbols are transmitted sequentially in time. This process is then reversed at the receiver as shown in figure 3.2.

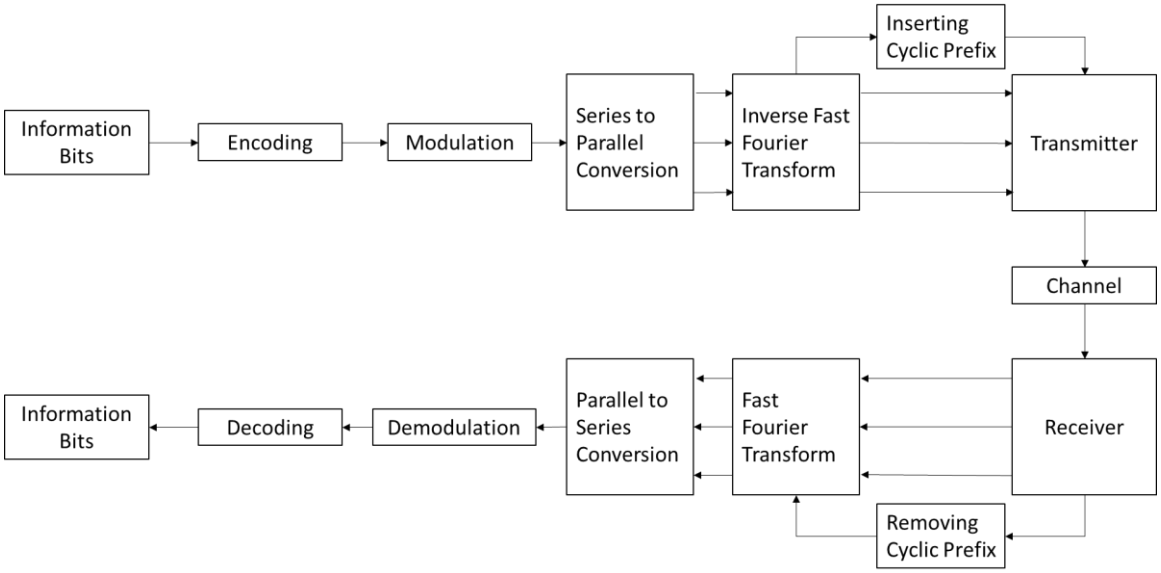

Figure 3.2: Procedure for transmitting and receiving an OFMD signal

This process will result in a transmitted signal whose time and frequency characteristics are shown in figure 3.3. It can clearly be seen that the signal spectrum mirrors the frequency response of the channel where all the meaningful data was placed in the passbands while zeros were placed in the stopbands.

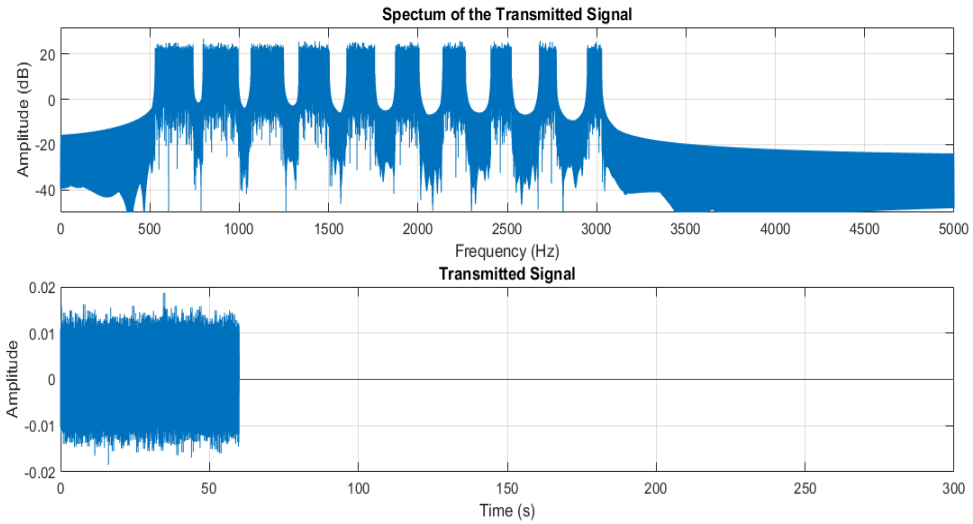

Figure 3.3: OFDM signal in time and frequency domains

Synchronization

The first OFDM symbol is used as a pilot symbol that is known at the receiver. This is done to perform both time and phase synchronization. For time synchronization, the pilot OFDM symbol is reconstructed at the receiver and correlated with the received signal and the time delay with the highest correlation is considered to be the correct time delay. For phase synchronization, figures 2.10 and 2.11 showed that the channel's phase response changes with each carrier frequency. Hence, each subcarrier requires its own phase synchronization procedure. For each subcarrier, the phase of the pilot symbol is used to estimate the offset, resolve the phase ambiguity, and correct the phase offset. Since the pilot phase is known at the receiver, it can be used to resolve the 180˚ phase ambiguity of whether the signal constellation was rotated by $\hat{\theta}_k$ or $\hat{\theta}_k + \pi$ in the counter clockwise direction where $\hat{\theta}_k$ is the phase offset estimate for the kth subcarrier. Once the

phase offset is estimated and the ambiguity is resolved, the signal constellation can be rotated by the appropriate amount to undo this offset.

Effect of Different Parameters on Performance

Effect of Frequency Spacing

As mentioned earlier, the frequency spacing (subcarrier bandwidth) Δf for an OFDM system has a big effect on performance when the passband used has a lot of fluctuations as shown in figures 2.10 and 2.11. Smaller frequency spacing will lead to better performance as the channel response within the subcarrier bandwidth becomes more flat. We ran a simulation to test the performance for a frequency spacing of 1Hz, 2Hz, 4Hz, 6Hz, and 12Hz. In this simulation, the OFDM signal had a narrowband and its bandwidth was able to fit inside the first passband at a carrier frequency of 1.1kHz and the bit rate was fixed to 48 bits/sec. The measurements were recorded at the end of the 10^{th} pipe string out of a total of 50 pipe strings. The results are shown in figure 3.4 and indicate that the smallest frequency spacing offered a better performance over the others. The 2Hz frequency spacing was used for the rest of the simulations since there was minimal difference between the 1Hz and 2Hz case.

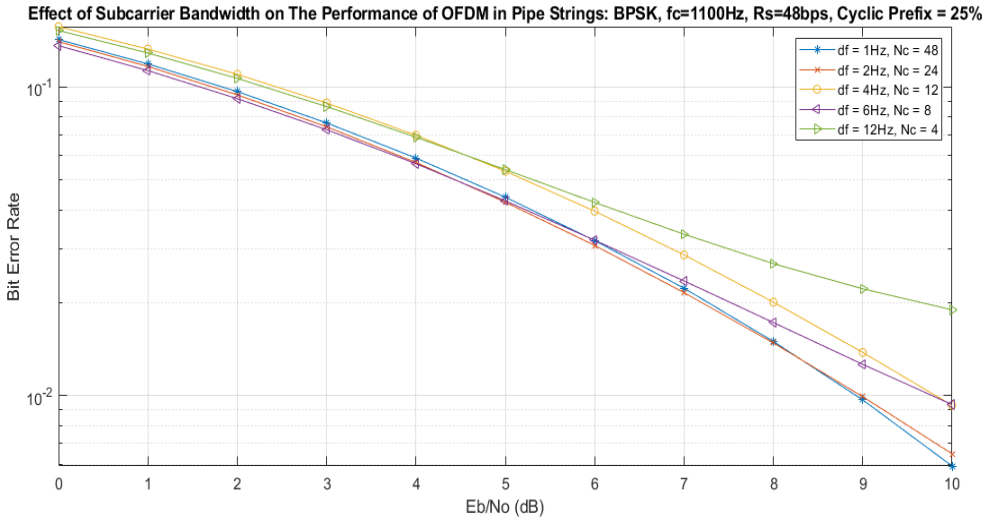

Figure 3.4: Effect of frequency spacing on the performance of OFDM in pipe strings

Effect of Cyclic Prefix

Since an OFDM system requires inserting a cyclic prefix to combat ISI and simplify the channel estimation process, we need to know how long does the cyclic prefix need to be in order to guarantee the best possible performance. Usually, longer cyclic prefix leads to better performance as the guard interval gets bigger and giving more time for the reflections of each OFDM symbol to arrive and then get discarded. However, longer cyclic prefix requires adding more overhead to the signal and thus increasing the cost of transmission. Thus, we need to choose the length of the cyclic prefix that offers the best overall performance. We tested different scenarios where the length of the cyclic prefix was 50%, 25%, 12.5%, and 6.25% of each OFDM symbol. The same signal for the frequency spacing test was used in this test and the results are

illustrated in figure 3.5. The results show that for low SNR, the smaller percentages of cyclic prefix were slightly better because most of the errors were caused by noise rather than ISI. Thus, the extra cost of overhead for longer cyclic prefix made the performance slightly worse. But for larger SNR, most of the errors are caused by ISI so the longer the cyclic prefix, the better the performance gets. The field testing done in [45] showed that the signal strength was between 0.2V to 1V while the noise was 1mV. This means that we expect to operate with an SNR for 45-60dB. Therefore, we will choose the 50% cyclic prefix for the rest of the simulations in this chapter.

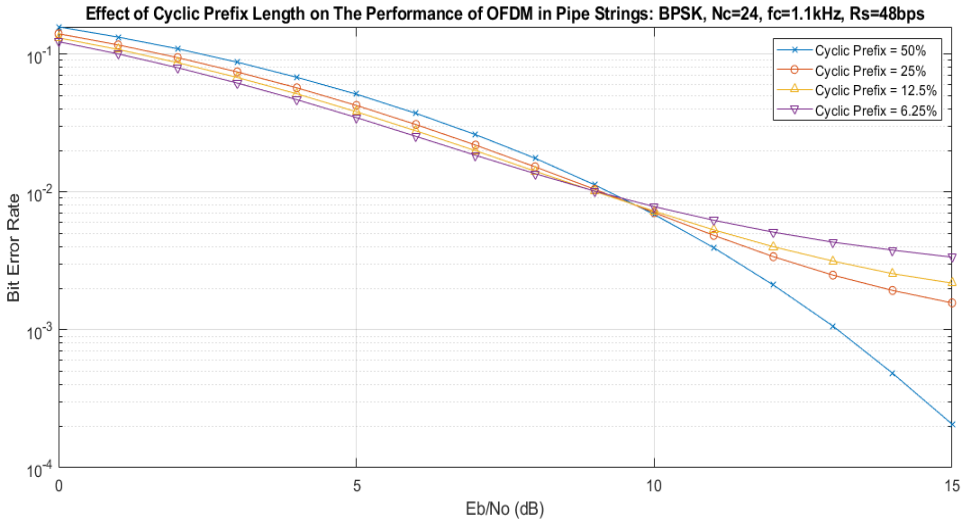

Figure 3.5: Effect of cyclic prefix length on the performance of OFDM in pipe strings

Effect of Bandwidth

We would like to test the effect of the bandwidth (number of passbands) on the overall performance of the proposed OFDM communication system. From what we know about the channel's frequency response, the passbands get weaker and narrower for higher frequencies. This means that from the point of view of power efficiency, it is more efficient for the signal bandwidth to be accommodated inside one passband and the more passbands are added, the worse the performance will get. However, since the application requires the bandwidth to span multiple passbands, we need to know many passbands we can use before performance is significantly degraded. We ran a simulation where the number of passbands included in the signal bandwidth was incrementally increased and used the proposed convolutional code with QPSK modulation. Figure 3.6 illustrates the results and confirms that as the number of passbands increases, the performance gets worse. In this test, the signal was measured at the 10^{th} pipe string out of 50.

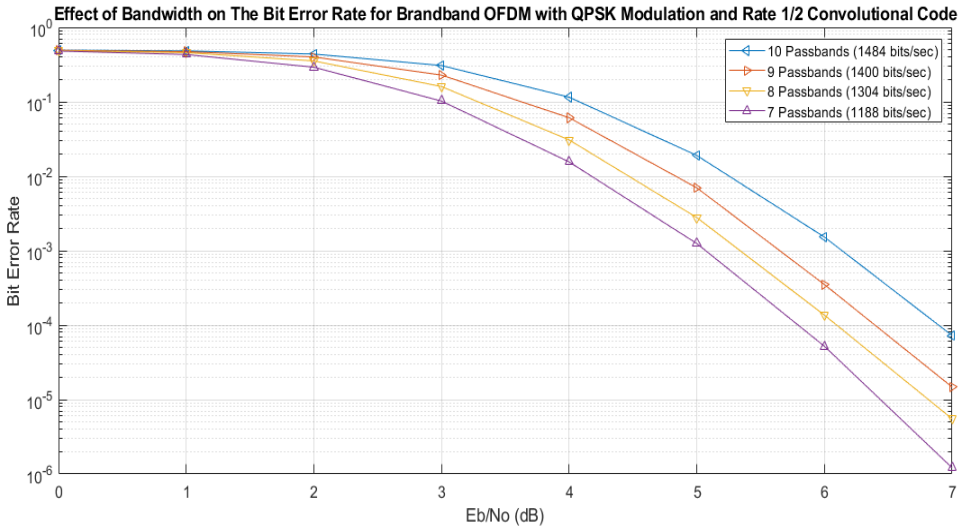

Figure 3.6: Effect of bandwidth on the performance of OFDM in pipe strings

Effect of Channel Coding and Modulation Order

In order to accommodate the application requirements, we need to find the right combination of modulation order and code rate. During the design process, we may have to compare a system with a lower modulation order and a higher code rate against another communication system with a higher modulation order and a lower code rate. Generally, the number of errors caused by increasing the modulation order is bigger than the increase in error-correction capability caused by lowering the code rate. To illustrate this with an example, a QPSK system with a rate 1/2 code is better than an 8-PSK system with a rate 1/3 code which is in turn better than a 16-PSK system with a rate 1/4 code as shown in figure 3.7.

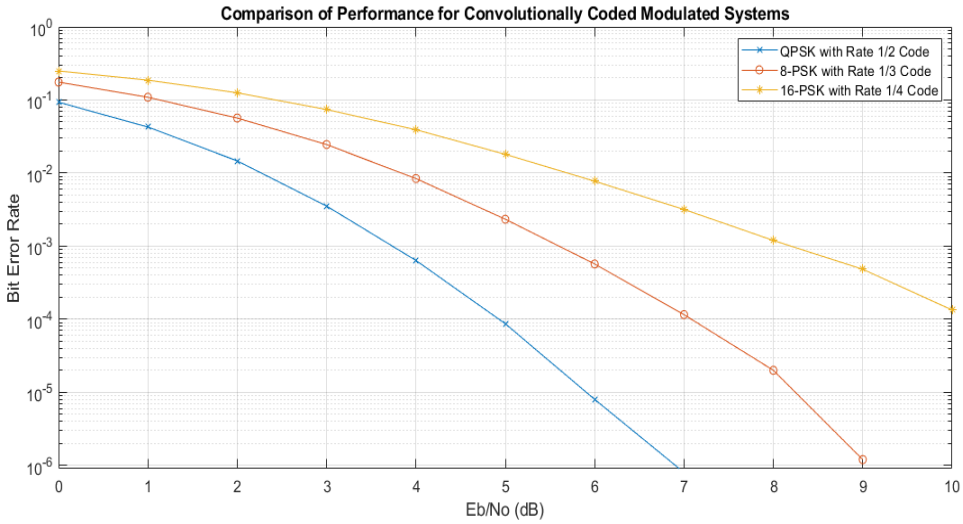

Figure 3.7: Comparison of performance for modulated systems with convolutional codes

When channel coding is used, it generally offers a great advantage over an uncoded system in terms of bit error rate. However, this is only true when the SNR is high enough because at low SNR, error-correction codes will perform worse than the uncoded systems as the number of errors exceeds the error-correction capability of the code. In this scenario, the closest codeword to the received codeword will be different from the transmitted codeword. Hence, the decoding algorithm will return the wrong codeword and end up creating more errors. For each combination of modulation and error-correction code, there is an SNR threshold where the coded system overtakes the uncoded system and starts to perform better from that point forward. This threshold will get higher as the modulation order gets higher. For example, the coded system will quickly outperform the uncoded system when BPSK or QPSK are used. But as the

modulation order increases to 8, 16, 32, and so on, the SNR threshold will quickly get higher as shown in figures 3.8-3.10 in an AWGN channel.

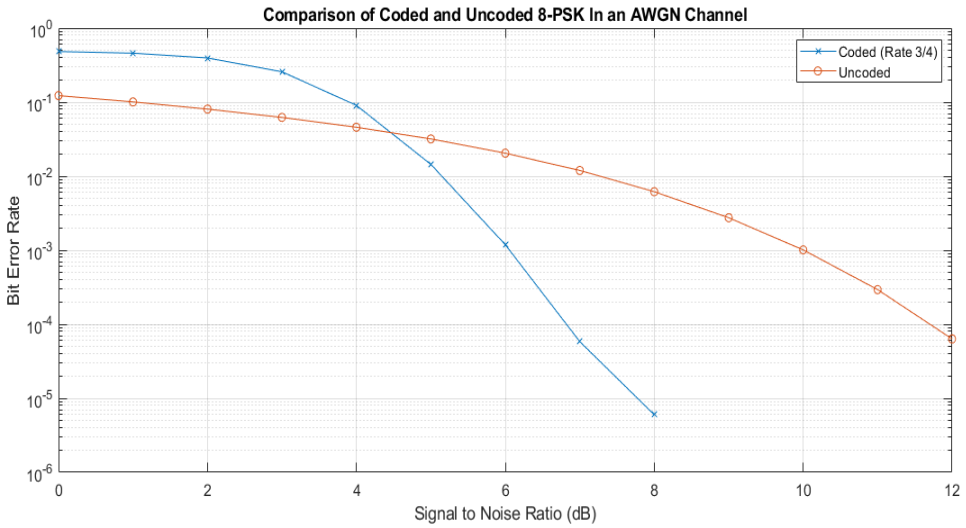

Figure 3.8: Comparison of coded and uncoded 8-PSK in an AWGN channel

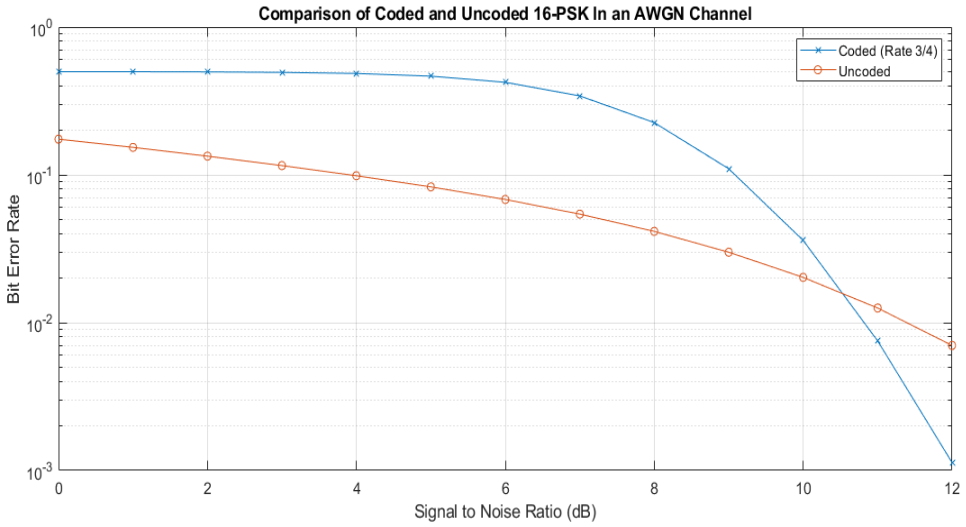

Figure 3.9: Comparison of coded and uncoded 16-PSK in an AWGN channel

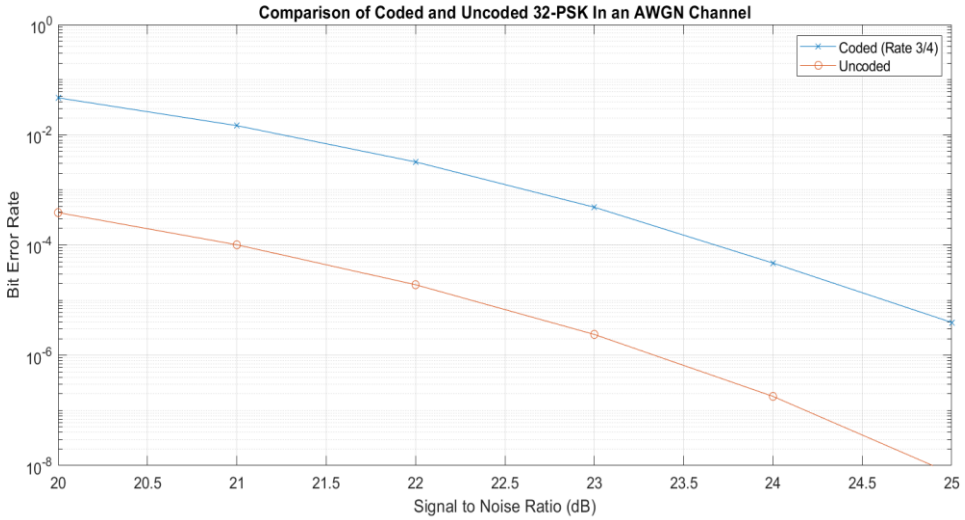

Figure 3.10: Comparison of coded and uncoded 32-PSK in an AWGN channel

In addition, the presence of a channel gain can positively or negatively affect this SNR threshold. If the channel gain is positive (in dB) and boosts the signal, the SNR threshold will get lower. If the channel gain is negative (in dB) and weakens the signal, the SNR threshold will get higher. This phenomenon is shown in figures 3.11-3.13 for a 16-QAM system. When the modulation order is further increased to 64-QAM in figure 3.14, the combination of the negative channel gain and the high modulation order will push the SNR threshold much higher.

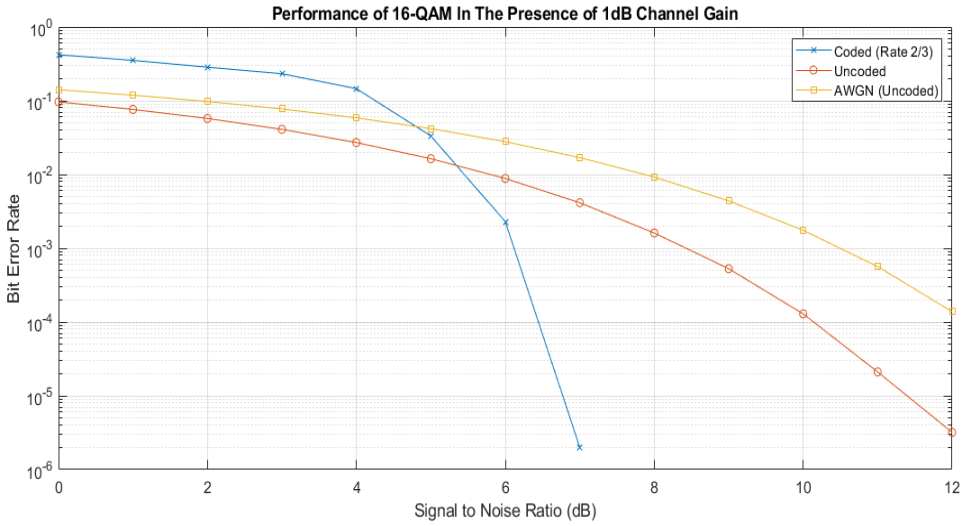

Figure 3.11: Performance of 16-QAM in the presence of 1dB channel gain

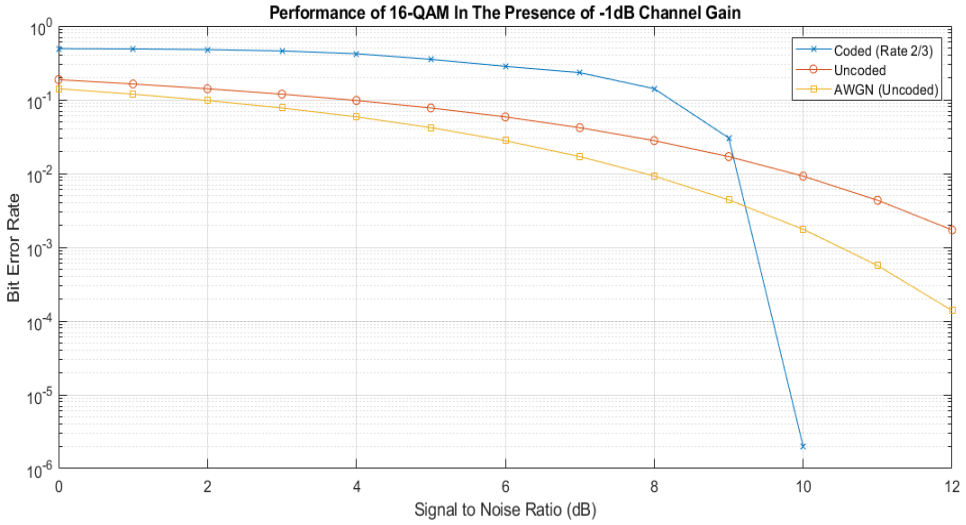

Figure 3.12: Performance of 16-QAM in the presence of -1dB channel gain

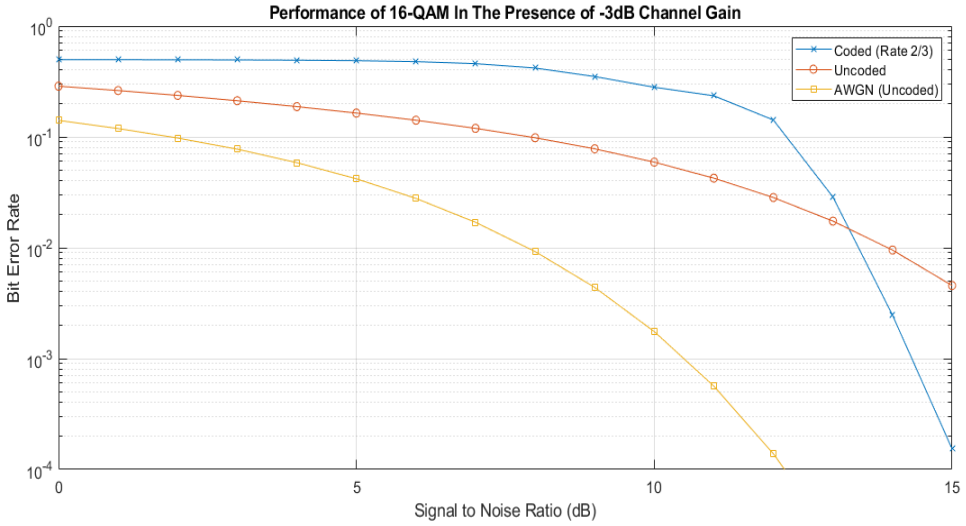

Figure 3.13: Performance of 16-QAM in the presence of -3dB channel gain

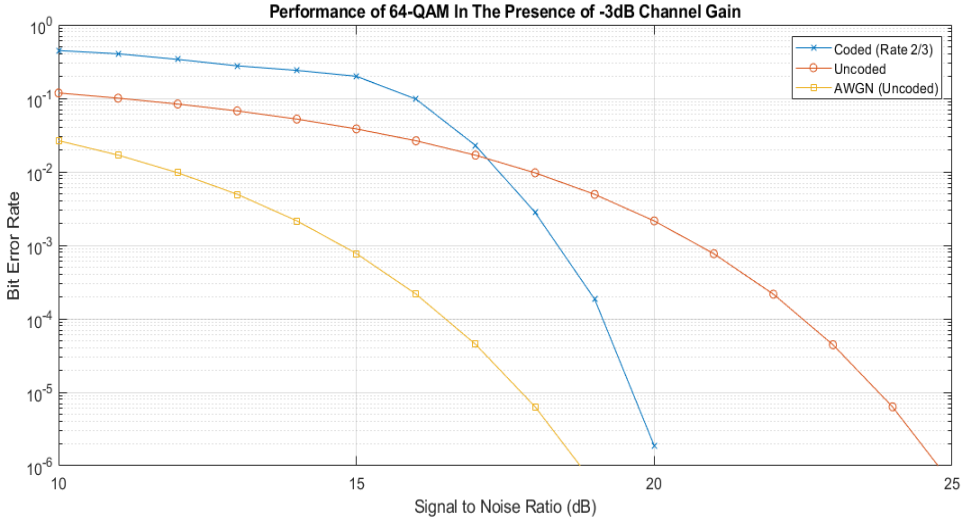

Figure 3.14: Performance of 64-QAM in the presence of -3dB channel gain

Design Process and Results

Finding the best possible communication system requires the optimization of five different parameters: data rate (or application), propagation distance, modulation order, code rate, and the bit error rate. This will lead to an almost infinite number of possible designs. If we fix a subset of these parameters and modify the others to find the best system, the result will be different if we fix another subset of these parameters and modify the remaining ones. For example, if we want to design a system to operate over a fixed short distance, a system with larger bandwidth (more passbands) may perform better than a system with a narrower bandwidth (less passbands). However, if we fix the data rate, a system with less passbands will perform better than a system with more passbands over longer distances.

In order to limit the scope of the design process, we will fix the data rates for each application, as in table 3.1, and attempt to find the system that can travel the longest distance before the bit error rate falls below 0.1%. We came up with a design procedure and an elimination process to limit the number of possible designs for each application. This procedure may not lead to the best possible design but will try to get as close as possible to it. There are three guidelines that we will use during this process to eliminate some of the possible designs. These guidelines are:

- If two communication systems use the same modulation order and different numbers of passbands. The one with more passbands will be eliminated in favor of the one with less passbands. This is justified by the results of figure 3.5.

- A communication system with a higher modulation order and a lower code rate will be eliminated in favor of a communication system with a lower modulation order and a higher code rate. This is justified by the results of figure 3.6.
- A communication system with a very high modulation order (greater than 4096) will be eliminated to reduce receiver complexity and avoid very poor performance.

The design process will follow the next steps for each application:

- Step 1: Given the required data rate, find the required number of bits per symbol and the modulation order M for each number of passbands by:

$$M = 2^{\left\lceil \frac{Required\ Data\ Rate}{Bandwidth} \right\rceil} \tag{3.2}$$

- Step 2: Eliminate any communication system if there is another one with the same modulation order and less passbands or if $M > 4096$.
- Step 3: For each passband from the remaining options, calculate the required code rate by:

$$Code\ Rate = \frac{Required\ Data\ Rate}{Bandwidth \times \log_2 M} \tag{3.3}$$

- Step 4: Choose a new code rate from a selection of available rates such as: 1/3, 1/2, 2/3, 3/4, and 1 (no coding). Calculate the new modulation order by:

$$M_{new} = 2^{\left\lceil \frac{Required\ Data\ Rate}{Bandwidth \times New\ Code\ Rate} \right\rceil} \tag{3.4}$$

- Step 5: Choose the lowest code rate from the selection such that $M_{new} \leq 2M$.
- Step 6: Calculate the resulting data rate and verify that it is greater than or equal to the data rate required by the application.

- Step 7: Look up the required SNR to reach a bit error rate of 0.1% for M-ary modulation and the coding gain for the chosen code if applicable.
- Step 8: Calculate the SNR margin by:

$$SNR\ Margin = Starting\ SNR - Required\ SNR\ for\ Modulation + Coding\ Gain \tag{3.5}$$

- Step 9: Approximate the achievable distance by finding how many pipe segments the signal can travel before the channel gain is below $-SNR\ Margin$. We will use the results of figure 2.3 and perform interpolation if necessary. For a system with multiple passbands, we will only look at the channel gain of the last passband since losing the symbols transmitted at that passband will result in a bit error rate that exceeds the chosen 0.1% limit.
- Step 10: Choose the system that can achieve the longest distance.

Tables 3.2 and 3.3 show the results of step 1 for both image applications.

Table 3.2: Required modulation order for a colored image (16 colors)

Bandwidth	Required Number of Bits/Symbol	Modulation Order
10 Passbands (1484 Hz)	2	4
9 Passbands (1400 Hz)	2	4
8 Passbands (1304 Hz)	2	4
7 Passbands (1188 Hz)	2	4
6 Passbands (1060 Hz)	2	4
5 Passbands (926 Hz)	2	4
4 Passbands (768 Hz)	2	4
3 Passbands (596 Hz)	3	8
2 Passbands (412 Hz)	4	16
1 Passband (214 Hz)	8	256

Table 3.3: Required modulation order for a colored image (256 colors)

Bandwidth	Required Number of Bits/Symbol	Modulation Order
10 Passbands (1484 Hz)	3	8
9 Passbands (1400 Hz)	3	8
8 Passbands (1304 Hz)	3	8
7 Passbands (1188 Hz)	3	8
6 Passbands (1060 Hz)	3	8
5 Passbands (926 Hz)	4	16
4 Passbands (768 Hz)	4	16
3 Passbands (596 Hz)	6	64
2 Passbands (412 Hz)	8	256
1 Passband (214 Hz)	15	32768

After the elimination process in step 2, the remaining steps were performed on the surviving systems and the results are summarized in tables 3.4 and 3.5 where the initial SNR was 60dB.

Table 3.4: Approximate achievable distance for a colored image (16 colors)

Bandwidth	4 Passbands	3 Passbands	2 Passbands	1 Passband
Required Code Rate	1	0.86	0.93	0.9
Modulation Order	4	8	16	256
New Code Rate	2/3	3/4	3/4	1
New Modulation Order	8	16	32	256
New Data Rate	1544	1597	1545	1712
Required Modulation SNR (dB)	10	11	14	22
SNR Margin (dB)	56	55	52	38
Approximate Achievable Distance (No. Pipes)	378	490	463	412

Table 3.5: Approximate achievable distance for a colored image (256 colors)

Bandwidth	6 Passbands	4 Passbands	3 Passbands	2 Passbands
Required Code Rate	0.96	1	0.86	0.93
Modulation Order	8	16	64	256
New Code Rate	3/4	1	3/4	1
New Modulation Order	16	16	128	256
New Data Rate	3180	3072	3129	3296
Required Modulation SNR (dB)	11	11	18	22
SNR Margin (dB)	55	49	48	38
Approximate Achievable Distance (No. Pipes)	202	336	434	355

According to the results of table 3.4 and 3.5, we need to use 3 passbands for both applications. The 16 colors image requires 16-QAM modulation with a rate 3/4 code while the 256 colors image requires 128-QAM modulation with a rate 3/4 code. Both systems were implemented and signals for the 16 and 256 colors image can respectively travel slightly more than 6km (600 pipe segments) and between 5km and 6km (500-600 pipe segments) before falling below the bit error rate threshold of 0.1%. At 6km, the bit error rate for the 16 colors image was 0.085%. For the 256 colors image, the bit error rate was 0.035% at 5km and 0.65% at 6km. Both systems outperformed the expectations in tables 3.4 and 3.5.

CHAPTER IV

LOW DATA RATE COMMUNICATION SYSTEM

Signaling Format

We now proceed to designing a single carrier narrowband communication system for the first mode of transmission in our application where sensor readings are periodically transmitted through acoustic vibration of the pipes. In this application, the required data rate (1-50 bits/sec) is small enough for the signal bandwidth to fit inside one passband. The sensor readings consist of a few dozen bits transmitted at low power and very low data rates. We expect a packet of 10-30 information bits to be encoded using an error-correction code and modulated using a BPSK scheme.

BPSK is chosen in this scenario because it offers the best power efficiency among all M-ary PSK, M-ary PAM, and M-ary QAM modulation schemes. Meanwhile, M-ary FSK offers better power efficiency for large M and is equivalent to an orthogonal BPSK signal encoded with a Hadamard code. However, orthogonal codes have a very low code rate of $\frac{k}{2^k}$ where k is the number of information bits. It will be demonstrated later than low code rates do not offer a better performance for short packets when synchronization is taken into account. Hence, BPSK remains the best option for modulation.

As for channel coding, we propose using a convolutional code to protect the information bits from errors. The reasoning behind this decision is that Turbo and LDPC codes are not suitable for short packets while block codes that are suitable for short

packets do not have a good enough error-correction capability. Hence, convolutional codes can operate on short packets while offering a decent error-correction capability. When it comes to pulse shaping, it will be demonstrated later that the choice of the pulse shape affects synchronization and the choice of the symbol rate affects the level of ISI in the received signal. Both parameters will have to be optimized to get the best possible performance under those circumstances. The carrier frequency will have to be chosen at the center of the passband if we intend to occupy the entire passband. If we want to take advantage of the positive channel gain when the measurements are taken in the middle of the pipe string and the passband is slanted, we may want to skew the carrier frequency toward the start of the passband.

In general, a communication system works in the following manner: information bits get encoded with an error-correction code and modulated. Then, they get up-sampled from one sample per symbol to match the sampling rate of the pulse shaping filter. Next, the up-sampled signal goes through the pulse shaping filter to receive its final form before transmission. The signal is then mounted on a carrier frequency and transmitted through the channel to reach the receiver where the mixers will bring it down to its baseband frequency. The received signal that has been corrupted by the channel is then put through the matched filter which is matched to the pulse shaping filter. From here, the synchronization stages begin.

Synchronization for Short Packets

Synchronization is necessary because the receiver does not know exactly when a signal is transmitted. There is also an unknown propagation delay where the signal

traveled for a duration of time between the transmitter and the receiver. The goal of synchronization is to let the receiver know exactly when the signal starts and to undo any phase offset caused by the channel so that the signal can be processed and the meaningful data can be successfully extracted. First at the coarse frame synchronization stage, the receiver tries to get a rough estimate of where the signal begins. Then, at the symbol synchronization stage, the receiver attempts to find the offset at which to down-sample the output of the matched filter. Once down-sampling has been performed, the receiver will find the exact starting position of the signal and correct any phase offset. From there, the signal is demodulated and decoded to extract the desired information. The structure of this communication system is illustrated in figure 4.1.

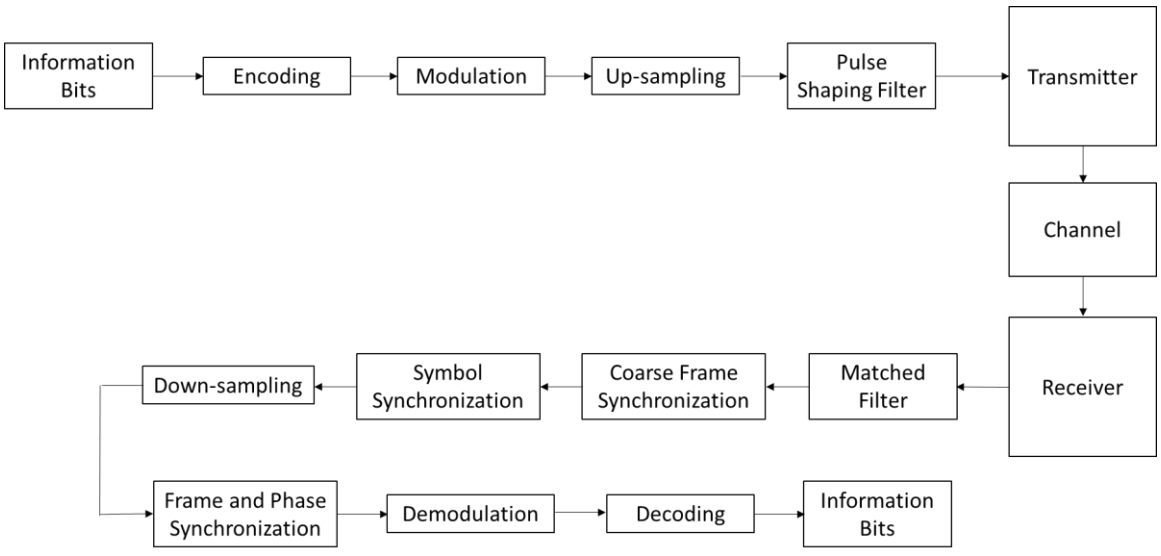

Figure 4.1: General structure of a communication system

In this dissertation, we will measure the performance of our communication system by the probability of packets loss. It is the probability that after all synchronization stages are done and after decoding the signal, the resulting decoded information bits are different from the transmitted information bits in at least one location. Our goal is to minimize this probability and the complexity of the synchronization process as much as possible by controlling different parameters and by choosing what we believe is the appropriate order for the different synchronization stages. We will perform all of our analysis on an AWGN channel for simplicity and then we will apply our results on the pipe strings.

At the frame synchronization stage, we will be using a decoder-assisted method where the decoding algorithm for an error-correction code will determine the correct time delay. Our signal is very short in time and is placed inside a much larger uncertainty window at an unknown time. Using the decoder-assisted frame synchronizer will create an issue of complexity if each time delay is assumed to be equally likely to be the correct one. Therefore, we propose starting with a simple coarse and symbol synchronization algorithm that will narrow down the initial uncertainty window to a much smaller uncertainty window where complexity will be greatly reduced. We will illustrate this with an example. Let us suppose that 10 information bits were encoded to produce 20 coded bits. These coded bits went through a pulse shaping function to produce 200 samples where each coded bit is represented by 10 samples. Let us suppose that they are placed within an uncertainty window of size 2000 samples at a randomly chosen location. This means that there are 1800 possible time delays and our decoder

will to have to treat each one of them as a likely time delay and will have to run the decoding algorithm 1800 times.

Now let us suppose that, through coarse synchronization, we managed to shrink this window to twice the size of the transmitted signal where it contains 400 samples. Inside this smaller uncertainty window, there are only 200 possible time delays so the decoder will have to run the decoding algorithm 200 times instead of 1800 times. We can reduce the complexity further through symbol synchronization techniques. Symbol synchronization helps the receiver find the correct time to down-sample the contents of the received window to retrieve the coded bits for the decoding procedure. If done correctly, we can further reduce the number of possible time delays to 20 times only. The decoder will now have to run the decoding algorithm 20 times instead of 1800 times without coarse and symbol synchronization.

If we proceed to perform frame synchronization without symbol synchronization first, our chances of decoding the packet correctly will be greatly diminished as shown by the simulation results in figure 4.2 where the probability of packet loss was simulated for a packet of 10 information bits encoded using a rate 1/2 convolutional code in a window 20 times the size of the packet. In figure 4.2, we down-sampled the signal at on offset of $0.1T_s$ and $0.2T_s$, where T_s is the symbol duration, to indicate how much loss is induced by not performing symbol synchronization before frame synchronization.

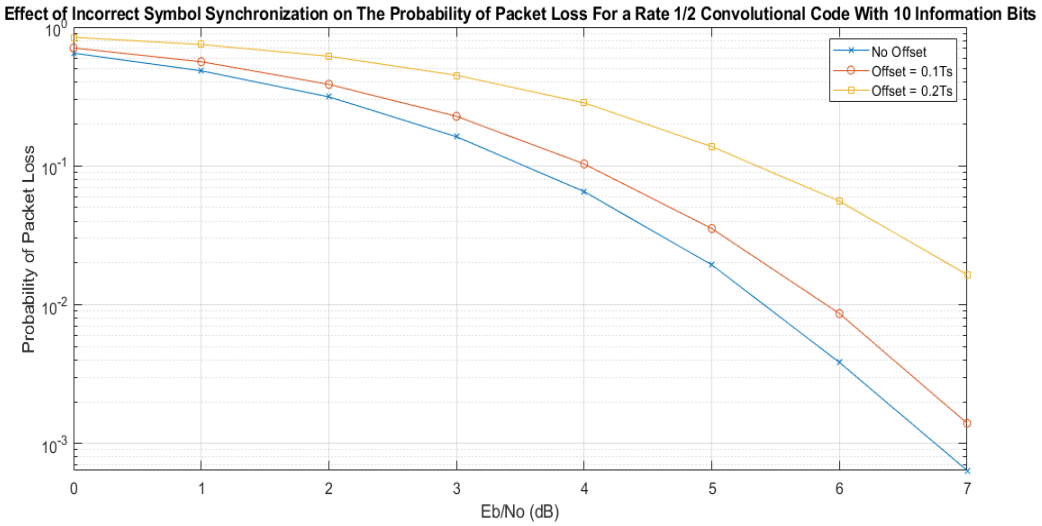

Figure 4.2: Effect of incorrect symbol synchronization on the probability of packet loss for a rate 1/2 convolutional code with 10 information bits

Let us also suppose that the channel caused a phase offset where the received samples were rotated by an unknown amount in either direction. Given that the frame synchronization procedure relies on the coded bits modulated using BPSK instead of the overall energy of the packet, it must be done coherently. Therefore, we find it necessary to perform phase synchronization and undo this offset before we proceed to frame synchronization. In figure 4.3, we show the effect of performing frame synchronization without compensating the phase offset first. Coarse and symbol synchronization were assumed to be perfect in this simulation.

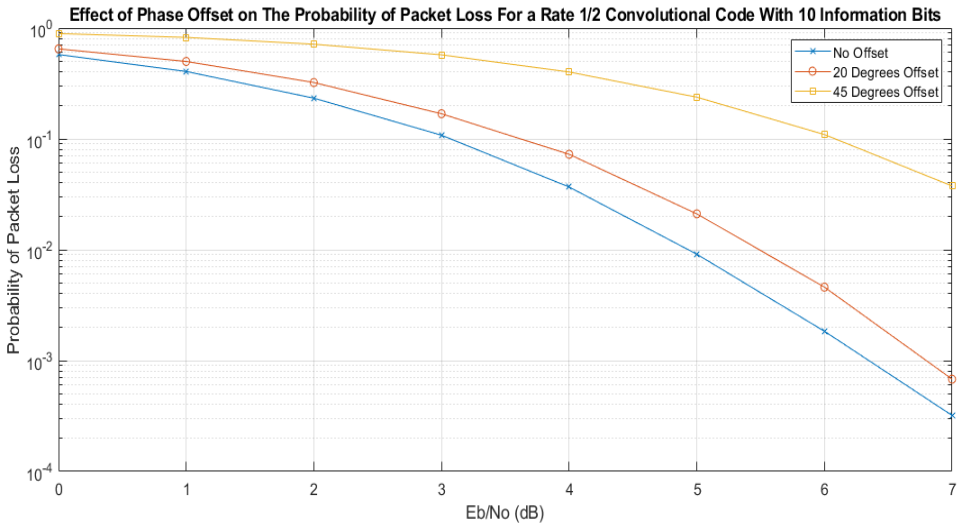

Figure 4.3: Effect of phase offset on the probability of packet loss for a rate 1/2 convolutional code with 10 information bits

CHAPTER V

COARSE AND SYMBOL SYNCHRONIZATION FOR VERY SHORT PACKETS

Problem Setup

At the beginning of the coarse synchronization stage, the signal is somewhere inside a large uncertainty window surrounded by Gaussian noise and the goal is to shrink it to a much smaller window where the signal is sure to be found. At this early stage, there are not many options available. We can either use a pilot sequence to crudely identify the beginning of the signal or try to detect an energy peak as the signal is likely to have more energy than the noise. We cannot use error-correction coding to try to identify a valid code word inside the window as we have to down-sample the signal and correct the phase offset first. The use of a pilot sequence on a short packet is not energy-efficient as will be demonstrated later. Thus, our only choice is to detect an energy peak.

Let us assume a BPSK modulated signal is transmitted in an Additive White Gaussian Noise channel. Let n be the total number of coded bits in the transmitted packet and k be the number of information bits. Let N_{sps} be the number of samples per symbol so the transmitted symbol has $N = nN_{sps}$ samples of which $K = kN_{sps}$ samples correspond to the information portion of the signal. Figure 5.1 illustrates the synchronization problem as it shows the output of the matched filter as a function of time.

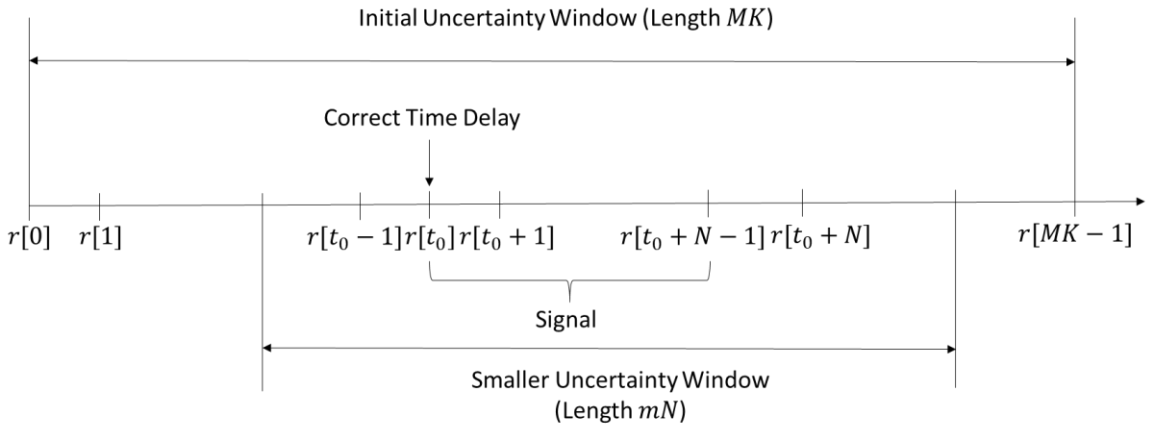

Figure 5.1: Output of the matched filter

In figure 5.1, M is the coefficient of the large uncertainty window such that this window is M times the length of the information portion of the signal (excluding any coded bits or overhead), m is a rational coefficient indicating the size of the smaller uncertainty window such that mN is an integer, t_0 is the signal starting time, and $r[t]$ is a sample of the received signal at the output of the matched filter:

$$r[t] = \begin{cases} s[t-t_0] + W[t], if\ t_0 \le t \le t_0 + N - 1 \\ W[t], otherwise \end{cases} \quad (5.1)$$

where $s[t]$ is the transmitted signal and $W[t]$ is a Gaussian noise sample from $\mathcal{N}(0, \sigma^2)$ such that $\sigma^2 = \frac{1}{\frac{k}{n} \times \frac{E_b}{N_0}}$. The notation $\mathcal{N}(\mu, \sigma^2)$ indicates a Gaussian distribution with mean μ and variance σ^2.

At the coarse and symbol synchronization stage, a sliding window having a size of N samples is swept across the samples of the matched filter output. In each case, the window is delayed by one sample. Then, its content gets down-sampled by a factor of

N_{sps} and the energy for each set of samples is calculated each time. We define S_t to be the energy of each sliding window:

$$S_t = \sum_{i=0}^{n-1} \left| r[t + iN_{sps}] \right|^2 \; for \; t = 0,1, \dots, (M-1)K - 1 \tag{5.2}$$

Then we will choose the sliding window that returned the highest energy. Let $\widehat{t_0}$ be a coarse estimate of t_0 that is found by:

$$\widehat{t_0} = \underset{t=0,\dots,(M-1)K-1}{\operatorname{argmax}} S_t \tag{5.3}$$

Once we found $\widehat{t_0}$, we step back by $\frac{N(m-1)}{2}$ samples to center the signal inside the smaller uncertainty window and select the next mN samples. The content of the smaller window is then down-sampled by N_{sps} producing the received samples that will be forwarded to the phase and frame synchronizer. If we choose to accept $\widehat{t_0}$ as the correct time delay and proceed to decoding immediately, we will see a substantial loss in performance as indicated by the results in figure 5.2. Therefore, we need a much finer synchronization procedure to receive the packet correctly.

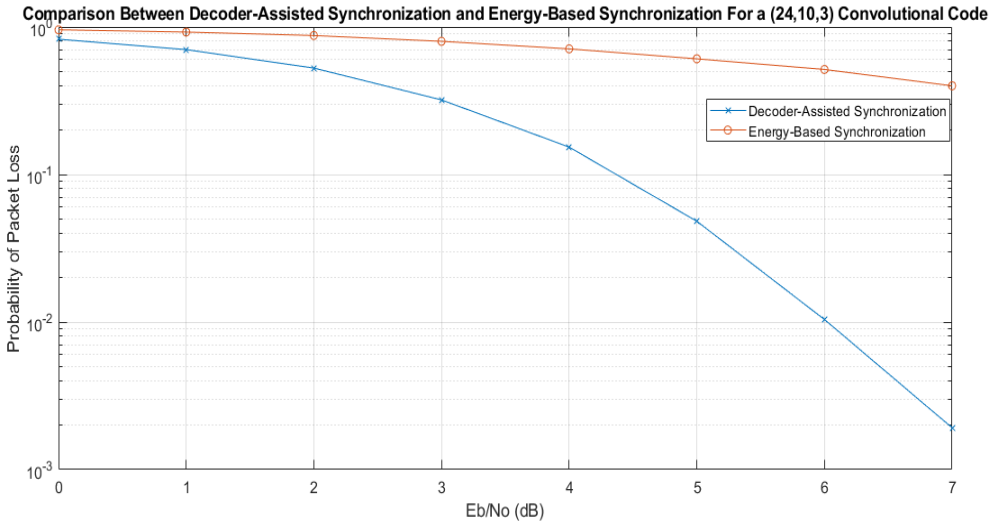

Figure 5.2: Comparison between decoder-assisted synchronization and energy-based synchronization for a (24,10,3) convolutional code

For convolutional codes, the notation (n, k, K_L) will be used across this dissertation to respectively indicate the total number of coded bits n, the number of information bits k, and the constraint length of this code K_L. When convolutional codes are used on short packets, it is important to distinguish between the designed code rate and the effective code rate. We typically terminate the message with $K_L - 1$ zeros to reset the encoder back to the all zero state, those extra zeros count as overhead when calculating the code rate. Both desired and effective code rates are calculated as follows:

$$designed\ code\ rate = \frac{k + K_L - 1}{n} \tag{5.4}$$

$$effective\ code\ rate = \frac{k}{n} \tag{5.5}$$

Effect of Uncertainty Window Size

In a typical monitoring sensor application where short packets are used, a battery-powered sensor is sleeping most of the time to conserve power but wakes up in a periodic fashion to make a reading of some parameter, convert that reading into a short sequence of bits, encode it, and then transmit it at the lowest possible SNR that guarantees an acceptable probability of error. On the other side, the receiver has been recording data for hours if not days expecting a transmission at any time. When this transmission takes place, the receiver has to sort through hours of noisy recordings to find the signal of interest. Since the transmitted packet is short, the signal will also be short in the time domain and is not likely to exceed 1 second. This less than 1 second signal is now concealed within hours of random noise. Let's illustrate this with an example: if a signal consisting of 10 bits is transmitted every hour at a low bit rate of 10 bits/sec, the signal lasts exactly 1 second and the receiver has an uncertainty window of 3600 seconds where the 1 second signal is. Trying to find this signal without a long pilot sequence and at a relatively weak SNR is like trying to find a needle in haystack. The chances of finding this signal successfully can be very low and unsatisfactory.

To quantify this problem, we need to run some simulations that can measure our chances of a successful transmission. We will measure the performance of the coarse and symbol synchronization algorithms by the probability of signal capture error (PSCE). The PSCE is the probability that a signal does not fall in its entirety inside the smaller uncertainty window after coarse and symbol synchronization. The main parameter that keeps us from getting an acceptable PSCE is the size of the large initial

uncertainty window described here by the coefficient M. In these simulation, we uniformly generated 10 information bits, encoded them using a convolutional code, modulated them using a BPSK scheme, and then placed them inside a window that is M times the size of the signal with additive noise simulating the AWGN channel. We ran the coarse and symbol synchronization algorithm and calculated the PSCE by using the Monte Carlo method. We count the number of times that the entire signal does not fall inside the smaller window and divide it by the total number of trials. The results of these simulations are shown in figure 5.3. In this particular simulation, a (24,10,3) convolutional code was used.

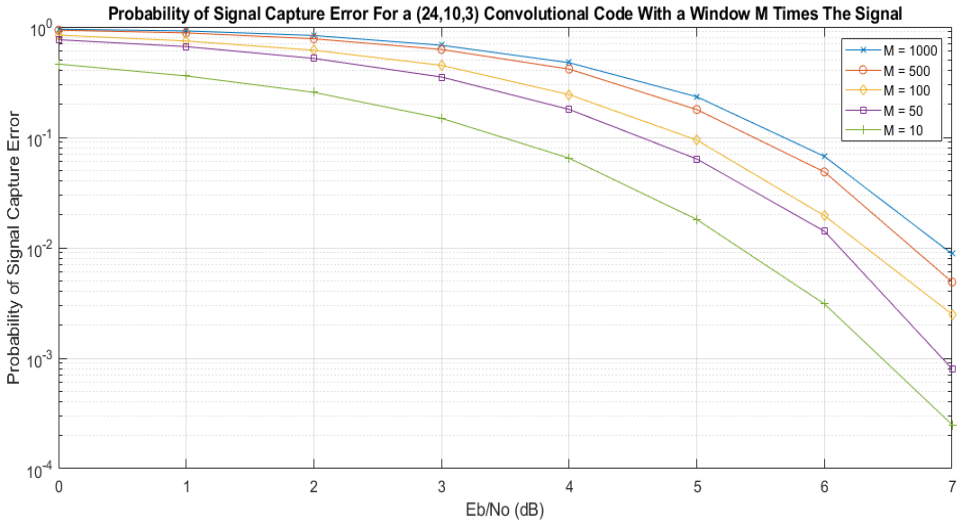

Figure 5.3: Probability of signal capture error for a (24,10,3) convolutional code with a window M times the signal

These results indicate that the probability of capturing the entire signal in the smaller window rapidly increases as the window gets larger. If the entire signal is not captured in the smaller window, the chances of correctly decoding the packet are almost zero. At this point, we need alternative solutions. In our original problem of transmitting sensor readings used to monitor some parameter, one possible solution is transmission scheduling. In this setting, we program both the sensor and the receiver to wake up at a known time to record the value of the desired parameter and go back to sleep. For example, at the top of every hour, the sensor will make the reading, send it through, and go back to sleep. The receiver will wake up around the same time expecting an incoming signal, record it, process it, and then go back to sleep. This can result in a much smaller uncertainty window where M is relatively small enough for us to be able to reliably find the signal. However, if an emergency transmission is to be made, we will have to make a decision depending on the urgency of this transmission. For example, if the message has a lower urgency and just a warning signal, we may have to wait until it is time for the next scheduled transmission and give a delayed response. However, if the urgency is high, we may have to accept a higher cost of transmission such as boosting the SNR or adding a long pilot sequence. Since emergency transmission do not occur very often and may be even rare, it may be feasible depending on the application to choose the latter option. Another option is to have a limited number of messages to transmit in case of emergency and pre-program those messages into the receiver so that the receiver can scan the observation window for these specific signals and be able to decode them. For example, in the context of oil and gas industry, if a leak is detected, the sensor will just

transmit the message “leak” or “leak detected” and the receiver knowing exactly what the binary sequence for this message is can search for it and decode it.

Effect of Pulse Shaping

It is important to note that the choice of the pulse shape will affect the performance of this coarse and symbol synchronization technique. In order to maximize the chances of $\widehat{t_0}$ being as close as possible to t_0, we believe the pulse shape needs to satisfy the following criteria:

- For each symbol period, the autocorrelation of the pulse shaping function needs to produce only one clear peak. This makes the correct sampling time stand out above all the other possible sampling times.
- The pulse shaping function needs to cross 0 between two consecutive symbols. This is done to force the output of the match filter to change value between two identical consecutive symbols making it more likely to detect the peak at the correct time delay.
- The autocorrelation of the pulse shaping function needs to avoid immediate transition from its peak to zero and should have a relatively slow transition. This is done so that if the estimated time delay is off from the correct one by a few samples, we do not lose too much signal strength.

To test the validity of these criteria, we ran several Monte Carlo simulations where each time, a different pulse shaping filter was used. Figure 5.4 shows the results of these simulations on the (24,10,3) convolutional code where a square pulse is compared to a triangular pulse, a delta function, and a half-square pulse where the square

function occupies half the duration of the pulse. The square pulse does not satisfy the second criterion while the delta function does not satisfy the third criterion. The triangle and half-square pulses satisfy all the criteria.

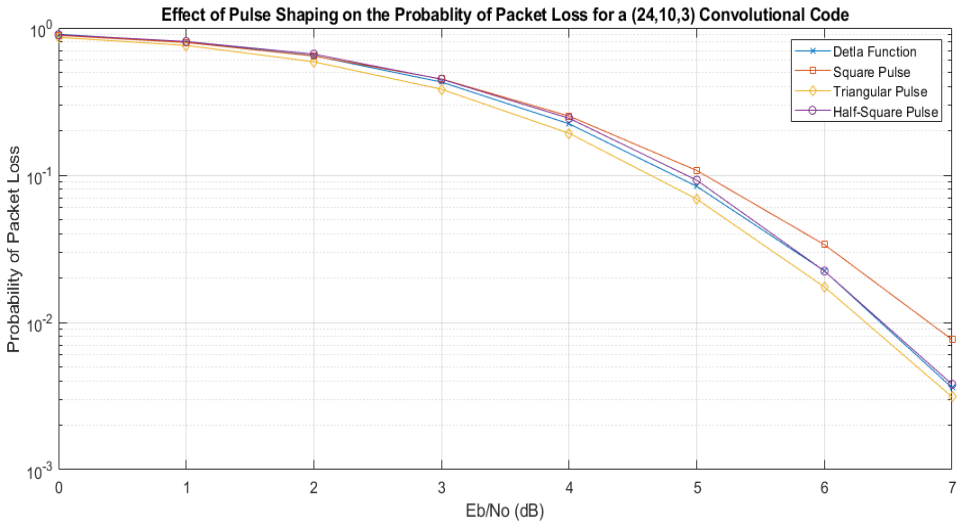

Figure 5.4: Effect of pulse shaping on the probability of packet loss for a (24,10,3) convolutional code

According to figure 5.4, a rectangular pulse will perform worse than all the others for short packets. This is due to the occurrence of consecutive 1s or 0s in the transmitted sequence. For a square pulse, identical consecutive symbols will result in a straight line at the output of the matched filter as shown in figure 5.5. When corrupted by noise, it is possible for incorrect sampling times to get additive constructive noise that will result in a higher energy than the correct sampling time. However, identical

consecutive symbols on a triangular pulse will have a dip between them as shown in figure 5.6 thus minimizing the chances of an incorrect sampling time rising above the correct one. On the other hand, this problem does not occur for a long packet because the event described above will not affect the overall energy of the much longer packet. However, a triangular pulse or a delta function will have a wider bandwidth compared to the square pulse. In the remaining sections of this paper, a triangular pulse was used in all simulations since the delta function has an infinite bandwidth that will not fit inside a passband.

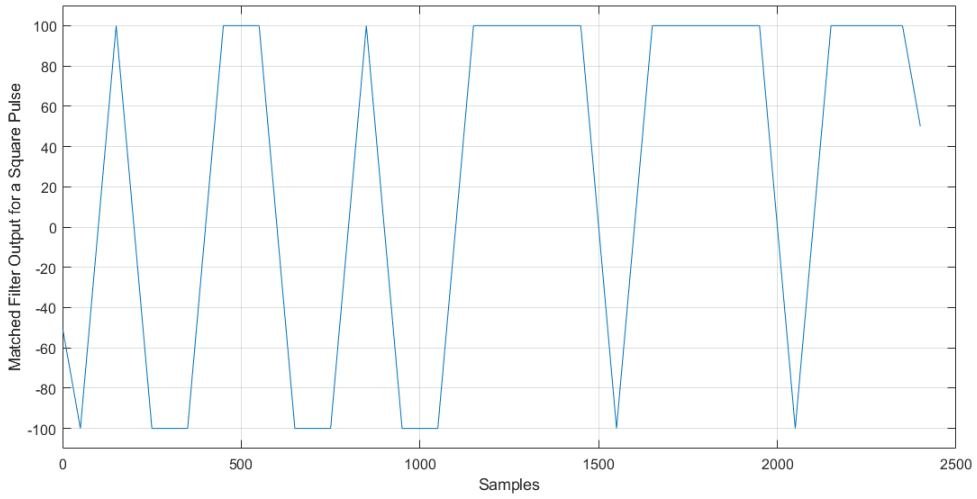

Figure 5.5: Matched filter output for a square pulse

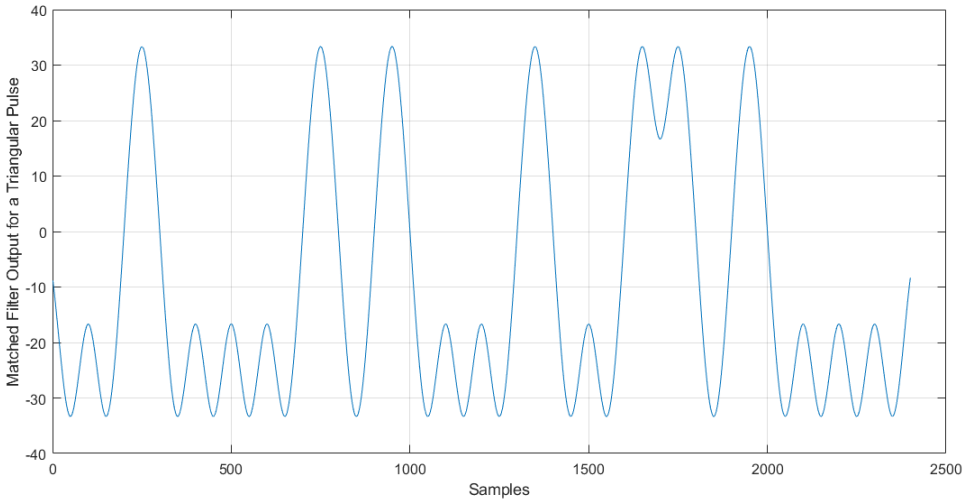

Figure 5.6: Matched filter output for a triangular pulse

Theoretical Approximation

In order to verify these simulation results, we would like to set up a theoretical model that can match and predict the PSCE as much as possible. Finding an exact expression for the PSCE can be difficult due to the dependence between the sliding windows that share multiple common samples. Thus, we will settle for a close approximation. There are two key assumptions here. First, let us assume that we are working at one sample per symbol. Second, when we find the coarse estimate of the time delay $\widehat{t_0}$, the smaller window of size mn is large enough to encompass all the samples of $r[t]$ that are found in window that share common samples with S_{t_0}. Therefore, we assume that the different random variables representing the energy S_t are independent and share no common samples. This will result in M different S_t that are far and independent from each other. In simple terms, instead of using $(M-1)K$ random

variables to compute the PSCE, we will only use M independent random variables. Let ${Z_i}^{(t)}$ be the ith sample in the sliding window at time t out of M possible windows, then:

$$ {Z_i}^{(t)} = r\left[t + iN_{sps}\right] for\ t = 0,1, \dots, M-1 \tag{5.6} $$

$$ S_t = \sum_{i=0}^{n-1} \left|{Z_i}^{(t)}\right|^2 \ for\ t = 0,1, \dots, M-1 \tag{5.7} $$

In order to calculate the PSCE, we need to know the distribution of S_t which can be determined by finding the distribution of $\left|{Z_i}^{(t)}\right|^2$. We know that $\left|{Z_i}^{(t)}\right|^2$ follows a Chi-Squared distribution but since S_t is a sum of n independent random variables, we can use the Central Limit Theorem to approximate it to a Gaussian random variable. In Appendix A, we show that for a BPSK signal:

$$ E\left[\left|{Z_i}^{(t)}\right|^2\right] = \begin{cases} \sigma^2 + 1, if\ t_0 \le t \le t_0 + n - 1 \\ \sigma^2, otherwise \end{cases} \tag{5.8} $$

$$ Var\left(\left|{Z_i}^{(t)}\right|^2\right) = \begin{cases} \sigma^4 + 2\sigma^2, if\ t_0 \le t \le t_0 + n - 1 \\ \sigma^4, otherwise \end{cases} \tag{5.9} $$

Therefore, $S_{t_0} \sim \mathcal{N}(n(\sigma^2 + 1), n(\sigma^4 + \sigma^2))$ and $S_t \sim \mathcal{N}(n\sigma^2, n\sigma^4)$ otherwise. From there, the probability of capturing the entire signal is approximated by:

$$ P(capture) \approx P\left(\bigcup_{t=0, t\neq t_0}^{M-1} S_{t_0} \ge S_t\right) = P\left(\bigcup_{t=0, t\neq t_0}^{M-1} S_t \le s \,|S_{t_0} = s\right) \tag{5.10} $$

$$ P(capture) \approx \int_{-\infty}^{\infty} \left(1 - Q\left(\frac{s - n\sigma^2}{\sqrt{n}\sigma^2}\right)\right)^{M-1} \frac{1}{\sqrt{2\pi Var(S_{t_0})}} e^{\frac{-(s - E[S_{t_0}])^2}{2Var(S_{t_0})}} ds \tag{5.11} $$

The PSCE can then be approximated by:

$$P(capture\ error) \approx 1 - P(capture) \tag{5.12}$$

We tested this approximation of the PSCE and we found it to be very close the exact PSCE obtained from Monte Carlo simulations in the scenarios of interest. Figures 5.7 and 5.8 show the comparison between the two for a rate 1/2 convolutional code. The larger and initial observation window was set to respectively 10 and 5 times the size of the signal while the smaller window was three times the size of the signal. Both cases show that the difference between the approximated PSCE and the actual PSCE is less than 0.5dB. This indicates that our theoretical model can sufficiently approximate the PSCE for relatively small uncertainty windows. However, when the uncertainty window is much larger, such as for $M = 100$ or $M = 1000$, the approximation errors become too large to neglect and the proposed model is no longer accurate.

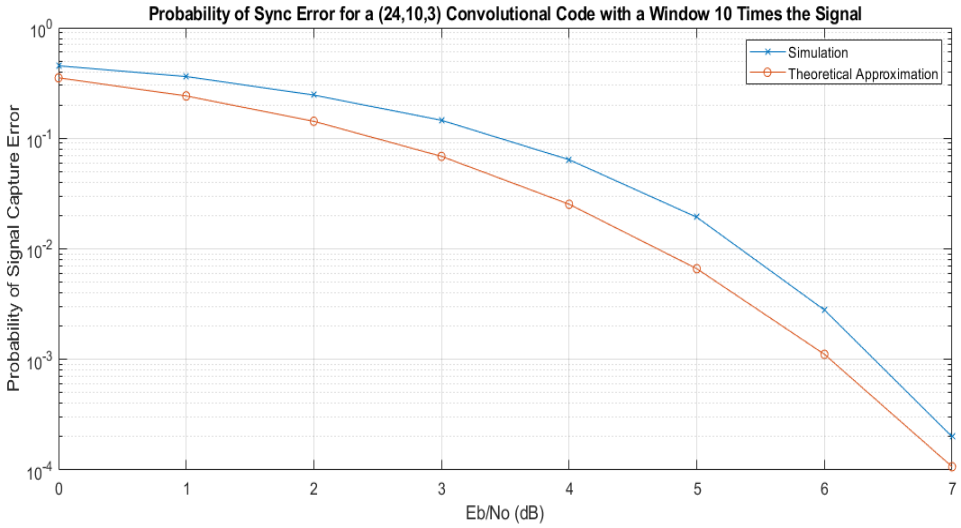

Figure 5.7: Probability of signal capture error with a window 10 times the signal

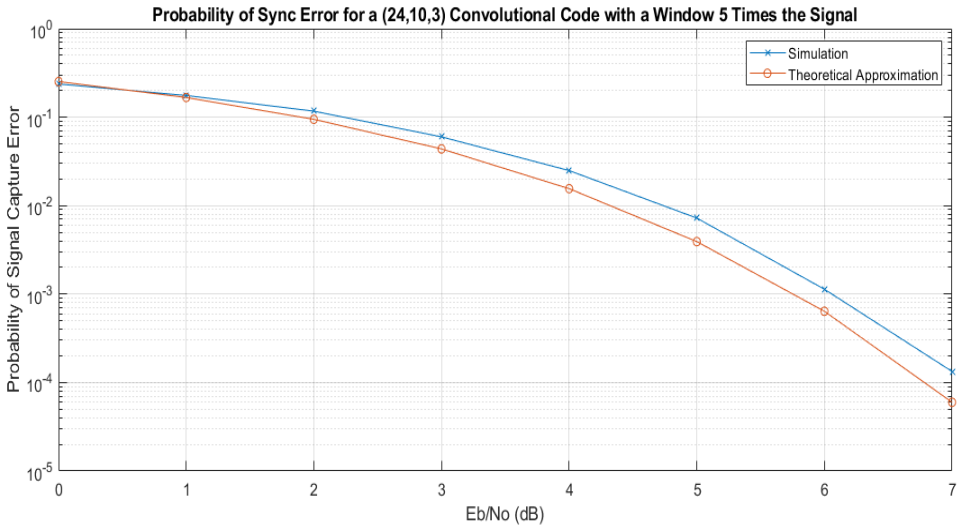

Figure 5.8: Probability of signal capture error with a window 5 times the signal

CHAPTER VI

FRAME SYNCHRONIZATION FOR VERY SHORT PACKETS

Problem Setup

While coarse synchronization tries to get a rough estimate of where the signal begins, frame synchronization aims at finding the exact time delay. This makes frame synchronization more complicated than coarse synchronization as there is no room for error especially when very short packets are involved. If the estimated time delay is off from the correct time delay by even 1 sample, the entire decoded packet will be different from the transmitted packet and all the information will be lost.

After both coarse and symbol synchronization stages are completed, we expect to have a small observation window, likely twice the size of the signal, that has been down-sampled to where each sample represents only 1 symbol. The desired signal is somewhere inside this window at an unknown location and surrounded by random noise. Let $\boldsymbol{s}$ be the vector containing the transmitted symbols, $\boldsymbol{w}$ is a vector containing the noise samples, n is the length of the signal in symbols, m is the coefficient of the smaller observation window as indicated in the previous chapter, and t_0 is the time delay that we are looking for. Figure 6.1 illustrates the described frame synchronization problem.

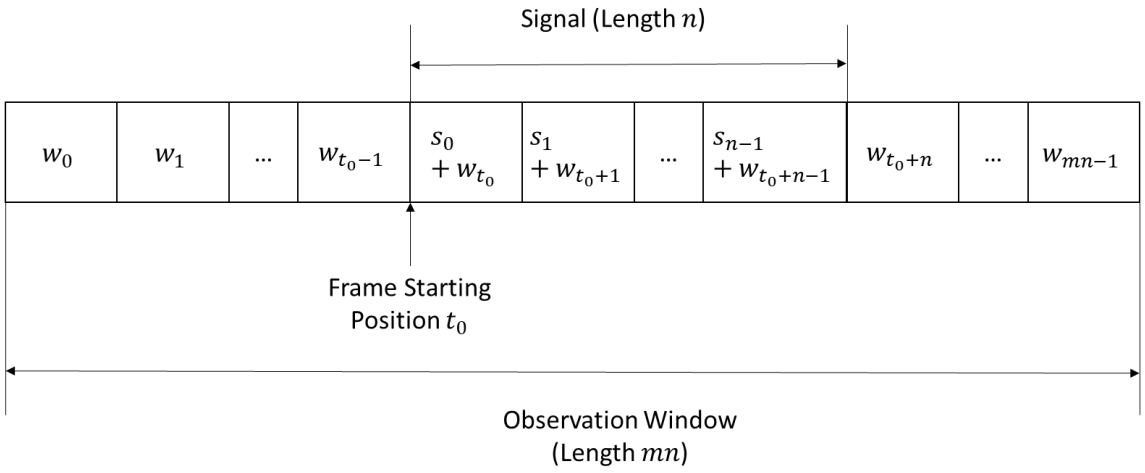

Figure 6.1: General received frame structure

All the samples in figure 6.1 can be summarized in the vector $\boldsymbol{r}$ containing all the received samples:

$$\boldsymbol{r} = [0,0,\dots,0,s_0,s_1,\dots,s_{n-1},0,\dots,0] + [w_0,w_1,\dots,w_{mn-1}] \quad (6.1)$$

Assuming the entire signal was captured during coarse and symbol synchronization stages, the correct time delay t_0 is equally likely to be somewhere between 0 and $(m-1)n-1$. We need to come up with a procedure to reliably find t_0. If the information bits were encoded using an error-correction code at the transmitter, we propose a simple and straight forward synchronization algorithm that can achieve optimal if not perfect synchronization in multiple scenarios.

Decoder-Assisted Frame Synchronization

Let us assume that k information bits were encoded at the transmitter with an (n, k) error-correction code where n is then number of coded bits. This (n, k) code may not necessarily be a standard block code but can be a convolutional or LDPC code. Such codes typically do not have a size limit and usually operate on an infinitely long stream of bits but since we are working with very short packets, we will treat them as block codes. The proposed frame synchronization algorithm consists of sliding a window $\boldsymbol{C}$ that has the same length n as the transmitted code that will test the possibility of each time delay being the correct one. The window $\boldsymbol{C}$ starts at r_0 selecting the next n samples and slides all the way to $r_{(m-1)n-1}$ selecting the last n samples in the observation window. Each time, we will evaluate a correlation metric and choose the time delay that maximizes this metric. This is done by following these steps for each possible time delay:

- Step 1: De-interleave the n coded bits in case interleaving was used.
- Step 2: Decode the n coded bits by following the decoding algorithm for the chosen (n, k) code.
- Step 3: Re-encode the resulting k information bits according to the encoding algorithm for the (n, k) code.
- Step 4: Re-interleave the n coded bits in case interleaving was used.
- Step 5: Re-modulate the resulting n coded bits according to the modulation scheme that was used at the transmitter. This will create a new window $\boldsymbol{C}'$.

- Step 6: Calculate the correlation ρ_j between the contents of $\boldsymbol{C}$ and the contents of $\boldsymbol{C}'$ according to:

$$\rho_j = \sum_{i=0}^{n-1} C_i \, C'_i \; for \; j = 0,1, \dots, (m-1)n - 1 \tag{6.2}$$

- Step 7: Choose the time delay that maximizes the correlation according to:

$$t_0 = \underset{t=0,\dots,(m-1)n-1}{\operatorname{argmax}} \rho_t \tag{6.3}$$

The main idea behind this method is the when we are observing an incorrect time delay, if we decode and then re-encode the contents of this window, we will end up with an entirely different window that looks nothing like what we observed. However, if we are looking at the correct time delay, repeating this procedure will result in a window that is very similar to the one we are observing.

Maximum Likelihood Frame Synchronization

In order to test the reliability of the proposed frame synchronization algorithm, we need to compare it to an optimal Maximum a Posteriori (MAP) frame synchronization algorithm. But first we need to derive it. A MAP frame synchronization algorithm requires choosing the time delay that maximizes the posterior probability density function (pdf) $f(t|\boldsymbol{r})$ for time delay t given the received samples $\boldsymbol{r}$ in equation (6.1). We start by applying the Bayes rule [46]:

$$f(t|\boldsymbol{r}) = \frac{f(\boldsymbol{r}|t)P(t \; is \; the \; chosen \; time \; delay)}{f(\boldsymbol{r})} \tag{6.4}$$

Since we are assuming that each time delay is equally likely to be the correct one, maximizing $f(\boldsymbol{r}|t)$ is equivalent to maximizing $f(t|\boldsymbol{r})$. The problem is then transformed from MAP to ML and $f(\boldsymbol{r}|t)$ can be expressed by:

$$f(\boldsymbol{r}|t) = \sum_{all\ possible\ \boldsymbol{s}} f(\boldsymbol{r}|t, \boldsymbol{s}) P(\boldsymbol{s}\ was\ transmitted) \tag{6.5}$$

where $\boldsymbol{s}$ is one of the all possible modulated codewords and $P(\boldsymbol{s}\ was\ transmitted) = \frac{1}{2^k}$ is the probability of uniformly choosing one codeword out of all possible 2^k codewords to transmit. Then, $f(\boldsymbol{r}|t, \boldsymbol{s})$ is expressed by:

$$f(\boldsymbol{r}|t, \boldsymbol{s}) = \frac{1}{\left(\sqrt{2\pi\sigma^2}\right)^{mn}} e^{\frac{-\sum_{i=0}^{mn-1} {r'}_i^2}{2\sigma^2}} \tag{6.6}$$

where:

$${r'}_i = \begin{cases} r_i - s_{i-t}, & for\ i = t, t+1, \dots, t+n-1 \\ r_i, & otherwise \end{cases} \tag{6.7}$$

This means that for each possible time delay t, we first evaluate the likelihood of $\boldsymbol{s}$ being the transmitted signal, we choose the most likely transmitted signal, and finally evaluate the likelihood of the contents of the observed window being a transmitted signal and the rest of the window being random noise. The correct time delay t_0 is the one that maximizes this likelihood and is found by:

$$t_0 = \underset{t=0,\dots,(m-1)n-1}{\operatorname{argmax}} \max_{all\ possible\ \boldsymbol{s}} f(\boldsymbol{r}|t, \boldsymbol{s}) \tag{6.8}$$

Simulation

Now we are ready to test our proposed algorithm and compare it with the optimal ML frame synchronization algorithm in addition to some of the existing methods in literature. In particular, the method described in [41] involves computing a log-likelihood ratio (LLR) of the syndrome obtained from the parity check matrix of the code used during transmission. We tested this LLR frame synchronization algorithm on LDPC codes since it was designed for codes having a sparse parity check matrix and we believe it will be most effective in those scenarios. The parity check matrices for these codes were generated using the Progressive Edge Growth algorithm in [47]. Both our proposed algorithm and the ML algorithm derived above were tested on convolutional codes.

In these simulations, k information bits were randomly generated following a uniform distribution and then encoded to obtain n coded bits that were modulated using a BPSK scheme. A random time delay t_0 was uniformly generated between 0 and $(m-1)n-1$ where $m=2$. The coded and now modulated symbols where then placed inside a window of size mn at position t_0 and Gaussian noise was added to simulate an AWGN channel. Then we run each one of the algorithms describe above to estimate t_0. Once t_0 is found, we extract the received signal and decode it. We then compare the resulting k bits with the transmitted information bits and count the probability of error. If the decoded bits are different from the transmitted bits in at least 1 location, we declare that the entire packet is lost. We then count the number of times this event occurs and divide it by the number of times the packet was transmitted to estimate the probability of

packet loss using the Monte Carlo principle. We also experimented with different packet sizes to see how much do we have to increase the packet size by to reach perfect synchronization where the exact time delay is known. Figures 6.2-6.4 show the simulation for 10, 20, and 100 information bits respectively.

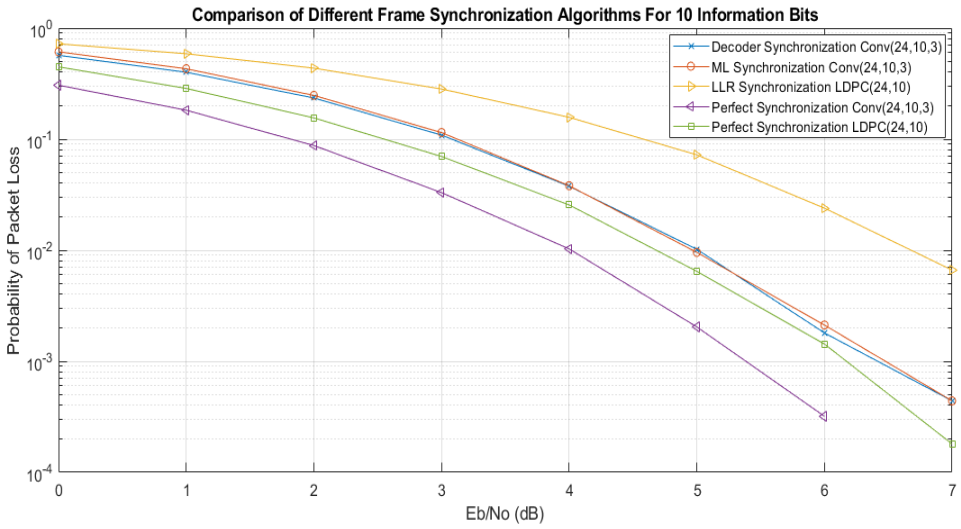

Figure 6.2: Comparison of different frame synchronization algorithms for 10 information bits

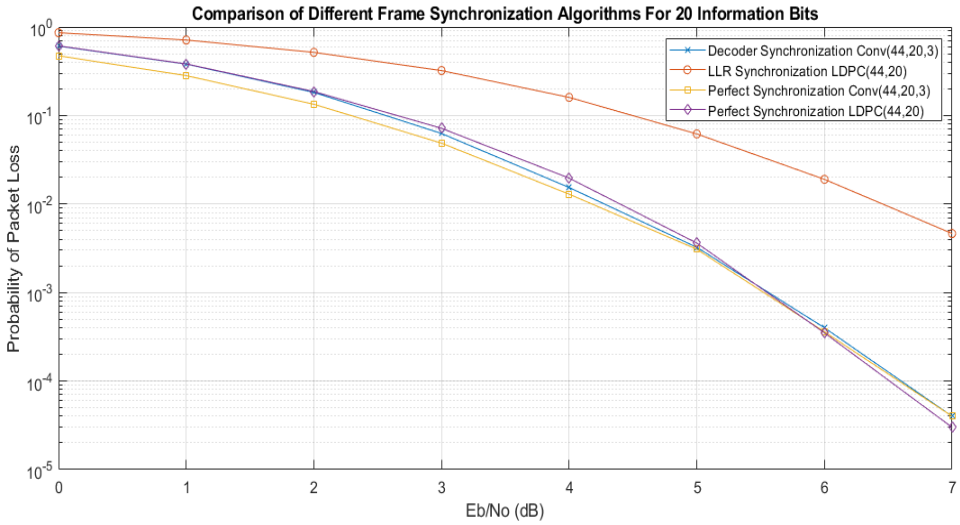

Figure 6.3: Comparison of different frame synchronization algorithms for 20 information bits

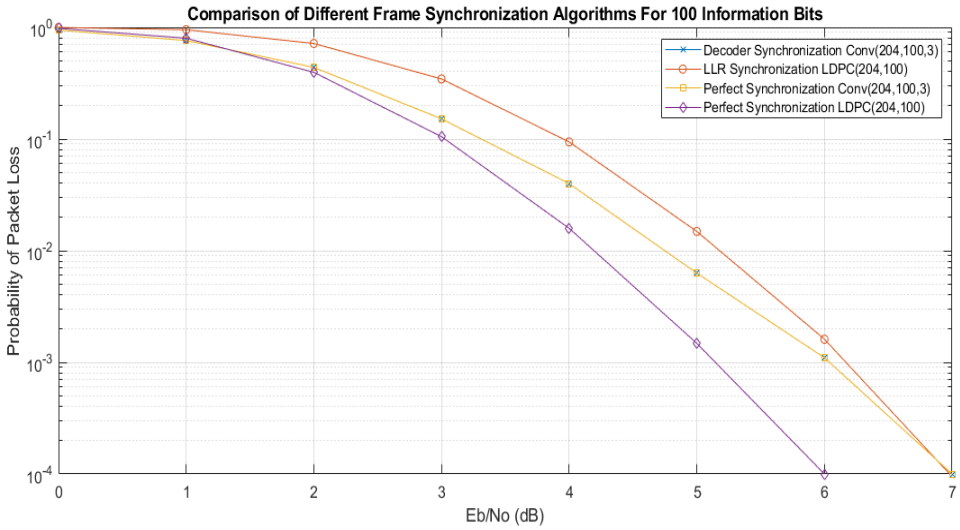

Figure 6.4: Comparison of different frame synchronization algorithms for 100 information bits

As shown in figure 6.2, when the very short packet consisting of 10 information bits was transmitted, the LLR synchronization algorithm was found to be 2.75dB behind perfect synchronization. This significant loss is caused by the fact that when LDPC codes are used on very short packets, the parity check matrix will not be sparse. Since having a sparse parity check matrix is a requirement for the LLR synchronization algorithm, it is to be expected that this method is not effective on very short packets. Meanwhile, both the proposed and ML algorithm were only at a 1dB disadvantage from perfect synchronization. This indicates that the proposed algorithm is equivalent to the optimal ML synchronization algorithm in this scenario. In fact, whenever soft-decision decoding is used, the proposed algorithm will always be equivalent to the ML algorithm. When block codes are used, soft decision decoding requires searching through all possible 2^k transmitted codewords to find the closest codeword in terms of Euclidean Distance much like the ML synchronization algorithm. When convolutional codes are used, the soft-decision Viterbi algorithm [48] finds the closest codeword without having to do an exhaustive search as described above. Hence, the proposed frame synchronization algorithm accompanied with soft-decision decoding produces the optimal probability of packet loss.

When the packet size is increased to 20 information bits, the ML synchronization algorithm can no longer be applied as there is more than 1 million possible codewords. But the proposed decoder-assisted algorithm reached the performance under perfect synchronization while the LLR synchronization algorithm is still 2.25dB behind its targeted performance. When the packet size reaches 100 information bits, the LLR

synchronization algorithm begins to close in on its corresponding performance at perfect synchronization and is only 1 dB behind while the proposed algorithm matches the performance of its targeted perfect synchronization almost perfectly. Another interesting conclusion that can be drawn from figures 6.2-6.4 is that for very short packets, convolutional codes are superior to LDPC codes but when the packet size grows, LDPC code will catch up and overtake convolutional codes. This is demonstrated by the performance under perfect synchronization for both coding schemes. Finally, a direct comparison between the proposed algorithm and the LLR synchronization algorithm on an (20,10) LDPC code is shown in figure 6.5 and indicates that the decoder-assisted synchronization algorithm is still closer to the performance under perfect synchronization than the algorithm in [41].

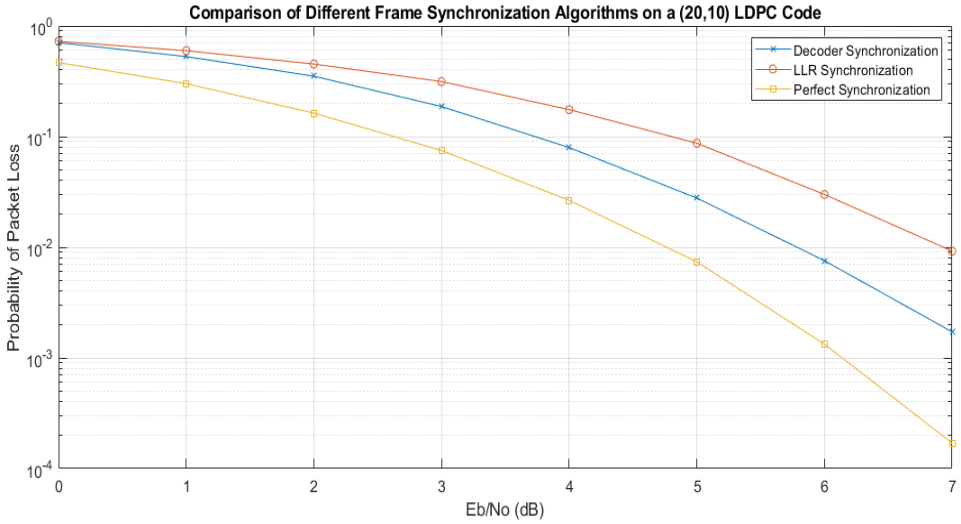

Figure 6.5: Comparison of different frame synchronization algorithms on a (20,10) LDPC code

Effect of Coding Scheme

Since we are taking advantage of the presence of error-correction codes to perform synchronization, our choice of coding scheme and decoding algorithm will affect the overall performance. Therefore, we need to carefully decide how are we going to encode and decode the transmitted packet. We demonstrated earlier that when soft-decision decoding is used for synchronization, it is equivalent to the optimal ML frame synchronizer and we see no point in settling for suboptimal performance using hard decision decoding. Now we just need to find the best coding scheme to reliably transmit and decode a very small and fixed number of bits.

For such a short packet, the candidates for best coding schemes are block codes, convolutional codes, LDPC codes, and turbo codes. The latter coding scheme requires very large packets in order for its interleaver to take full effect and it is hence eliminated. Block codes such as Hamming, Golay, BCH, and Reed-Solomon can work well on short packets but they do not have an efficient soft-decision decoding algorithm. Their hard-decision decoding algorithms are suboptimal and their soft-decision decoding algorithms require an exhaustive search over 2^k possible transmitted codewords. This search will have to be repeated $2(m-1)n$ times to account for each possible time delay and phase ambiguity resolution, as will be shown later. Therefore, it is not desirable to use block codes since they require the decoding algorithm to run $2^{k+1}(m-1)n$ times. As for LDPC codes, they were designed to have a low density parity check matrix where 0s massively outnumber 1s in the matrix. This will not be the case for short packets in order to satisfy the parity check requirements. By elimination, we believe that convolutional

codes are the best to use in this scenario as they check all the boxes when it comes to having an efficient soft decision decoding algorithm, not having any requirements over the packet size or the parity check matrix.

We ran a few simulations to test the validity of this claim. We generated 12 information bits and encoded them using a (24,12) Golay code, a (24,12) LDPC code, and a (28,12,3) convolutional code. As illustrated in figure 6.6, the convolutional code clearly outperforms both the Golay and LDPC code.

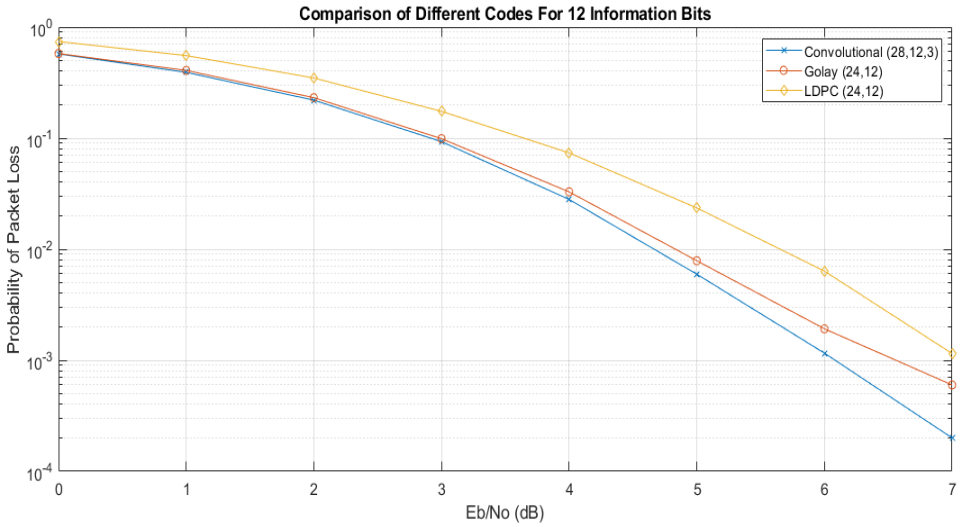

Figure 6.6: Comparison of different codes for 12 information bits

One of the major advantages of convolutional codes is that, unlike block codes, we do not have restrictions on the block size or the error-correction capability. So we can tune convolutional codes to achieve a desired probability of packet loss while still

fixing the number of information bits. If our ultimate goal was to transmit k information bits at a given SNR and a probability of error that does not exceed a given upper limit, we can experiment with the code rate and the constraint length of the convolutional code to find the best option as will be shown later.

Phase Synchronization and Ambiguity Resolution

The proposed frame synchronization algorithm can also be used to help with phase synchronization and ambiguity resolution because a codeword that had it bits flipped is likely not going to be valid codeword and hence the frame synchronization algorithm can tell us whether our phase estimate is correct.

Let us assume that the channel induced an unknown phase offset denoted by θ. After both coarse and symbol synchronization, we will end up with a received vector $\boldsymbol{r}$ whose sampled can be expressed by:

$$r_t = \begin{cases} s_{t-t_0} e^{i\theta} + w_t, if\ t_0 \le t \le t_0 + n - 1 \\ w_t, \qquad otherwise \end{cases} \tag{6.9}$$

where w_t is a Gaussian noise sample and $i = \sqrt{-1}$. The phase offset θ can be first estimated by:

$$\hat{\theta} = angle\left(\sqrt{\frac{1}{mn}\sum_{j=0}^{mn-1} r_j{}^2}\right) \tag{6.10}$$

However, there is still 180˚ ambiguity since the signal constellation was either rotated by $\hat{\theta}$ in the counter clockwise direction or by $\hat{\theta} + \pi$. This is where Hypothesis

Testing can be used to solve this ambiguity. Let $\mathcal{H}_0$ be the hypothesis that signal was rotated by $\hat{\theta}$ and $\mathcal{H}_1$ be the hypothesis that the signal was rotated by $\hat{\theta} + \pi$.

We can test each hypothesis by rotating the received samples by $\hat{\theta}$ and running the frame synchronization algorithm to find the maximum correlation $\rho_{t_0|\mathcal{H}_0}$, then we can rotate the samples by $\hat{\theta} + \pi$ and run the frame synchronization algorithm again to find the maximum correlation $\rho_{t_0|\mathcal{H}_1}$. The phase offset will be decided by:

$$\theta = \begin{cases} \hat{\theta}, if\ \rho_{t_0|\mathcal{H}_0} > \rho_{t_0|\mathcal{H}_1} \\ \hat{\theta} + \pi, if\ \rho_{t_0|\mathcal{H}_0} < \rho_{t_0|\mathcal{H}_1} \end{cases} \tag{6.11}$$

CHAPTER VII

THEORETICAL APPROXIMATION FOR THE PROBABILITY OF SYNCHRONIZATION ERROR

In this chapter, we propose a theoretical model to approximate the probability of synchronization error (PSE) for the decoder-assisted frame synchronization algorithm on very short packets. This model is primarily used to support the simulation results shown previously and provide a guideline to finding the PSE for high SNR situations that require extremely long Monte Carlo simulations. Deriving the exact expression of the PSE for a convolutional, block, and LDPC code can be difficult so we will primarily use a random code in this model since the coded bits are independent and we will rely on some approximations and estimations to find a tight upper bound on the PSE.

Expressing the Probability of Synchronization Error

The decoder-assisted frame synchronization for a signal encoded using a random code follows these steps:

- Step 1: A sliding window having the same size as the codeword is moved across the received window to test every possible time delay.
- Step 2: For every time delay, the content of the window is correlated with every codeword in the codebook.
- Step 3: The codeword with the highest correlation with the content of the window is recorded.

- Step 4: The time delay with the highest correlation calculated from the previous step is chosen as the correct time delay.

The PSE is the probability that the chosen time delay is different from the correct time delay and is expressed by:

$$Pr(sync\ error) = Pr(\bigcup_{i=1, i\neq t_0}^{M} Z_{t_0} < Z_i) \leq \sum_{i=1, i\neq t_0}^{M} Pr(Z_{t_0} < Z_i) \qquad (7.1)$$

where Z_{t_0} represents the maximum correlation at the correct time delay, Z_i is the maximum correlation at the ith incorrect time delay, and M is the total number of possible time delays.

The Union Bound is used in equation (7.1) to provide an upper bound to the exact PSE because the exact expression is difficult to obtain since the different Z_i are not independent random variables. In order to evaluate this probability, we need to find the distributions of Z_{t_0} and all the different Z_i.

Let n be the number of coded bits in the (n, k) random code and σ^2 be the noise variance that is inversely proportional to the SNR. We can safely assume that Z_{t_0} is a Gaussian random variable with parameters $\mathcal{N}(n, n\sigma^2)$. This assumption is accurate because at the correct time delay, the correct transmitted codeword in the random codebook will achieve the highest correlation with the contents of the observed window almost all the time. The top histogram representing the samples of Z_{t_0} in figure 7.1 verifies this assumption. Meanwhile, the distribution of Z_i is unknown because it represents the maximum of correlated Gaussian random variables whose covariance

matrix is not full rank. A mathematical model for such a distribution does not exist but we can find an empirical one. The bottom two histograms in figure 7.1 show this distribution when the observation window contains a partial and incomplete signal and when it contains noise samples only. A direct observation of these two histograms indicate that the distribution of Z_i resembles a Gamma distribution [49] whose parameters need to determined or at least approximated.

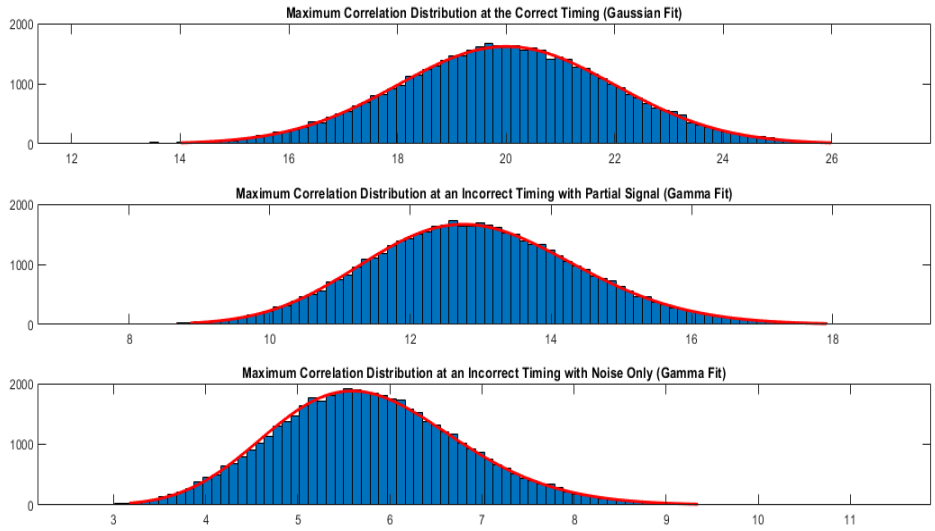

Figure 7.1: Metric distribution at various time delays for a (20,10) random code

Now that we have an idea about the distributions of the maximum correlation at the correct an incorrect time delays. We can further express the probability that the

maximum correlation at ith incorrect time delay is greater than the maximum correlation at the correct time delay.

$$Pr\left(Z_{t_0} < Z_i\right) = \int_{-\infty}^{\infty} Pr\left(Z_i > z | Z_{t_0} = z\right) f_{Z_{t_0}}(z) dz \tag{7.2}$$

$$Pr\left(Z_{t_0} < Z_i\right) = \int_{0}^{\infty} Pr\left(Z_i > z | Z_{t_0} = z\right) f_{Z_{t_0}}(z) dz + \int_{-\infty}^{0} f_{Z_{t_0}}(z) dz \tag{7.3}$$

where $f_{Z_{t_0}}(z)$ is the pdf of Z_{t_0}. Since Z_i is assumed to have a Gamma distribution, the first integral in (7.3) starts from 0 because a Gamma random variable is always positive. However, when z is negative, we get $Pr\left(Z_i > z | Z_{t_0} = z\right) = 1$. The next step is to approximate the shape α and rate β for a Gamma distribution whose pdf is expressed by:

$$f(x) = \frac{\beta^{\alpha}}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x} \tag{7.4}$$

where $\Gamma(x)$ is the Gamma function:

$$\Gamma(x) = \int_{0}^{\infty} t^{x-1} e^{-t} dt \tag{7.5}$$

In the probability $Pr\left(Z_i > z | Z_{t_0} = z\right)$, a Gamma random variable is being conditioned on a Gaussian random which means we need to find the shape and rate of a conditional Gamma random variable. To make things simpler, we will approximate Z_i and Z_{t_0} as jointly Gaussian. Based on that approximation, we can find the conditional mean $\mu_{Z_i|Z_{t_0}}$ and variance ${\sigma_{Z_i|Z_{t_0}}}^2$ of Z_i conditioned on Z_{t_0} and deduce the shape and rate of Z_i. The next set of equations will demonstrate this procedure:

$$\mu_{Z_i|Z_{t_0}} = \mu_{Z_i} + \frac{\sigma_{Z_i}}{\sigma_{Z_{t_0}}} . \rho_{Z_i, Z_{t_0}} . (z - \mu_{Z_{t_0}}) \tag{7.6}$$

$$\sigma_{Z_i|Z_{t_0}}{}^2 = \left(1 - \rho_{Z_i,Z_{t_0}}{}^2\right).\sigma_{Z_i}{}^2 \tag{7.7}$$

$$\beta = \frac{\mu_{Z_i|Z_{t_0}}}{\sigma_{Z_i|Z_{t_0}}{}^2} \tag{7.8}$$

$$\alpha = \beta.\mu_{Z_i|Z_{t_0}} \tag{7.9}$$

where $\rho_{Z_i,Z_{t_0}}$ is the correlation coefficient between Z_i and Z_{t_0}.

As a precaution for the rare event when $\mu_{Z_i|Z_{t_0}}$ is not positive, we will set it to a very small number close to 0 because α and β must be strictly positive. Now we can proceed to express $Pr\left(Z_i > z|Z_{t_0} = z\right)$ using the Gamma Function $\Gamma(x)$ and the Upper Incomplete Gamma Function $\Gamma(x, y)$:

$$P\left(Z_i > z|Z_{t_0} = z\right) = \frac{\Gamma(\alpha, \beta z)}{\Gamma(\alpha)} \tag{7.10}$$

$$\Gamma(x, y) = \int_y^\infty t^{x-1}e^{-t}dt \tag{7.11}$$

The only missing ingredients to evaluate the PSE are μ_{Z_i}, $\sigma_{Z_i}{}^2$, and $\rho_{Z_i,Z_{t_0}}$ that are respectively the mean of Z_i, the variance of Z_i, and the correlation coefficient between Z_i and Z_{t_0}. The next section will demonstrate how they can be found if possible or approximated.

Approximating the Unknown Parameters

At the ith incorrect time delay, the maximum correlation is $Z_i = \max(X_1, \dots, X_N)$ where $N = 2^k$ and k is the number of information bits in the (n, k) random code and X_j is the correlation between the observed window and the jth codeword in the random

codebook. We need to separate between the cases where the observed window contains noise sample only and the case where a partial signal is observed.

Noise Only Windows

If we are observing a window that contains noise samples and correlate it with every codeword in the codebook to obtain X_j. We will find that X_j are correlated and identically distributed Gaussian random variables with zero mean and variance $n\sigma^2$. The correlation coefficient between X_i and X_j is:

$$\rho_{X_i,X_j} = \frac{1}{n}\sum_{p=1}^{n} c_p^{(i)}.c_p^{(j)} \tag{7.12}$$

where $c_p^{(i)}$ is the pth symbol in the ith codeword in the random codebook. The proof of (7.12) is found in Appendix B. All values of ρ_{X_i,X_j} need to be computed and stored in a $2^k \times 2^k$ matrix of correlation coefficients $\boldsymbol{R}$.

Upon careful observation of the contents $\boldsymbol{R}$, its seems to have repeating values. The smaller correlation coefficients are more frequent than the larger ones. Therefore, one possible approximation to find μ_{Z_i} and ${\sigma_{Z_i}}^2$ is to assume that X_j are equally correlated with the same correlation coefficient ρ. This coefficient can be found by taking the average of the absolute value of all non-diagonal entries of $\boldsymbol{R}$. A very similar result can be obtained by the median or the L_2 average instead of the L_1 average. It was demonstrated in [50] that the mean and variance of the maximum of equally correlated Gaussian random variables are related to the mean and variance of the maximum of independent and identically distributed (IID) Gaussian random variables.

For the IID case, the mean $\mu_{Z_i,iid}$ and variance ${\sigma_{Z_i,iid}}^2$ of the maximum are:

$$\mu_{Z_i,iid} = \int_{-\infty}^{\infty} \frac{N}{\sqrt{2\pi}} \sqrt{n\sigma^2} t e^{\frac{-t^2}{2}} (1 - Q(t))^{N-1} dt \tag{7.13}$$

$${\sigma_{Z_i,idd}}^2 = \int_{-\infty}^{\infty} \frac{N}{\sqrt{2\pi}} \sqrt{n\sigma^2} t^2 e^{\frac{-t^2}{2}} (1 - Q(t))^{N-1} dt - {\mu_{Z_i,iid}}^2 \tag{7.14}$$

where $Q(x)$ is the Q-function defined by:

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^{\infty} e^{\frac{-t^2}{2}} dt \tag{7.15}$$

From there μ_{Z_i} and ${\sigma_{Z_i}}^2$ are found as in [50]:

$$\mu_{Z_i} \approx \sqrt{1-\rho}\mu_{Z_i,idd} \tag{7.16}$$

$${\sigma_{Z_i}}^2 \approx \rho + (1-\rho){\sigma_{Z_i,iid}}^2 \tag{7.17}$$

Since there are no common samples between the noise only window and the window at the correct time delay, the correlation coefficient is easily found by:

$$\rho_{Z_i,Z_{t_0}} = 0 \tag{7.18}$$

Windows with Partial Signal

If the observed window now contains some samples of the transmitted signal, the correlations X_j between the window and every codeword in the codebook are correlated and differently distributed Gaussian random variables with different means and an equal variance $n\sigma^2$. Let us define n_b to be the number of symbols present in the observed window and i_c to be the index of the chosen codeword that was transmitted from within the random codebook. Let $d = n - n_b + 1$. Then, the mean of X_j is:

$$\mu_{X_j} = \begin{cases} \sum_{i=1}^{n_b} c_i^{(i_c)} c_{d+i-1}^{(j)}, \text{if the beginning of the signal is observed} \\ \sum_{i=d}^{n} c_i^{(i_c)} c_{i-d+1}^{(j)}, \text{if the end of the signal is observed} \end{cases} \tag{7.19}$$

The correlation between the different X_j are calculated using the same expression for the noise only case in (7.12). Meanwhile the correlation coefficient between Z_{t_0} and X_j is calculated by:

$$\rho_{Z_{t_0},X_j} = \frac{1}{n}\mu_{X_j} \tag{7.20}$$

The correlation coefficients in (7.20) will be useful when calculating $\rho_{Z_i,Z_{t_0}}$. The proof of (7.20) can be found in Appendix B.

In order to estimate μ_{Z_i}, ${\sigma_{Z_i}}^2$, and $\rho_{Z_i,Z_{t_0}}$ for windows with partial signal, it is necessary to distinguish between cases where the code rate is low and where the code rate is high. When the code rate is low, X_j can be treated as independent and differently distributed Gaussian random variables. When the code rate is high, X_j are correlated and differently distributed Gaussian random variables. We will treat any code rate less than 1/5 as low and any code rate greater than 1/5 as high.

Low Code Rate

Since X_j are weakly correlated and almost independent for low code rates, we can simply obtain the mean and variance of Z_i by integrating the pdf of Z_i. Let $f_{X_j}(x)$ and $F_{X_j}(x)$ be the respective pdf and CDF of X_j:

$$f_{X_j}(x) = \frac{1}{\sqrt{2\pi n\sigma^2}} e^{-\frac{(x-\mu_{X_j})^2}{2n\sigma^2}} \tag{7.21}$$

$$F_{X_j}(x) = 1 - Q\left(\frac{x-\mu_{X_j}}{\sqrt{n\sigma^2}}\right) \tag{7.22}$$

Then for $Z_i = \max(X_1, \dots, X_N)$:

$$f_{Z_i}(z) = \sum_{j=1}^{N} f_{X_j}(z) \prod_{q=1, q\neq j}^{N} F_{X_q}(z) \tag{7.23}$$

$$\mu_{Z_i} = \int_{-\infty}^{\infty} z f_{Z_i}(z) dz \tag{7.24}$$

$$\sigma_{Z_i}{}^2 = \int_{-\infty}^{\infty} z^2 f_{Z_i}(z) dz - \mu_{Z_i}{}^2 \tag{7.25}$$

In order to find $\rho_{Z_i, Z_{t_0}}$, we propose using a sequential maximization algorithm. This algorithm treats the maximum of two Gaussian random variables as another Gaussian random variable and was initially proposed in [51]. This algorithm works in the following manner and is explained in details in Appendix C:

- Compute the mean and variance of $\max(X_1, X_2)$.
- Approximate the correlation coefficient $\rho_{X_3, \max(X_1, X_2)}$.
- Approximate the mean and variance of $\max(X_3, \max(X_1, X_2))$.
- Approximate the correlation coefficient $\rho_{X_4, \max(X_1, X_2 X_3,)}$.
- Continue the same procedure until approximating the mean and variance of $\max(X_N, \max(X_1, \dots, X_{N-1}))$.
- Add the random variable Z_{t_0} to the set $X_1, \dots, X_N$ and approximate the correlation coefficient $\rho_{\max(X_1, \dots, X_N), Z_{t_0}}$.

High Code Rate

In this case, we will use a pairwise maximization algorithm proposed in [52] to approximate and the mean and variance of Z_i. This algorithm divides the set of random variables into pairs and treats the maximum of two Gaussian random variables as another Gaussian random variable. This algorithm works in the following manner:

- Compute the mean and variance of $\max(X_1, X_2)$, $\max(X_3, X_4)$,…, $\max(X_{N-1}, X_N)$.
- Approximate the correlation coefficients $\rho_{\max(X_1,X_2),\max(X_3,X_4)},\ldots,$ $\rho_{\max(X_{N-3},X_{N-2}),\max(X_{N-1},X_N)}$.
- Approximate the mean and variance of $\max(\max(X_1, X_2), \max(X_3, X_4)), \ldots, \max(\max(X_{N-3}, X_{N-2}), \max(X_{N-1}, X_N))$.
- Continue the same procedure until approximating the mean and variance of $\max(X_1, \ldots, X_N)$.

This algorithm is explained in detail in Appendix D. The correlation coefficient $\rho_{Z_i,Z_{t_0}}$ is approximated using the same procedure as for the low code rate.

Simulation

The entire procedure for approximating the PSE using the Union Bound was implemented and compared with the actual PSE obtained from Monte Carlo simulations in addition to the approximate PSE where the different parameters were obtained from Monte Carlo simulations instead of running the algorithms described above. We

experimented with different packet sizes and different codes rates and it can be seen in figure 7.2-7.10 that for relatively high SNRs of interest, the PSE obtained from the Union Bound is less than 1dB away from the actual PSE.

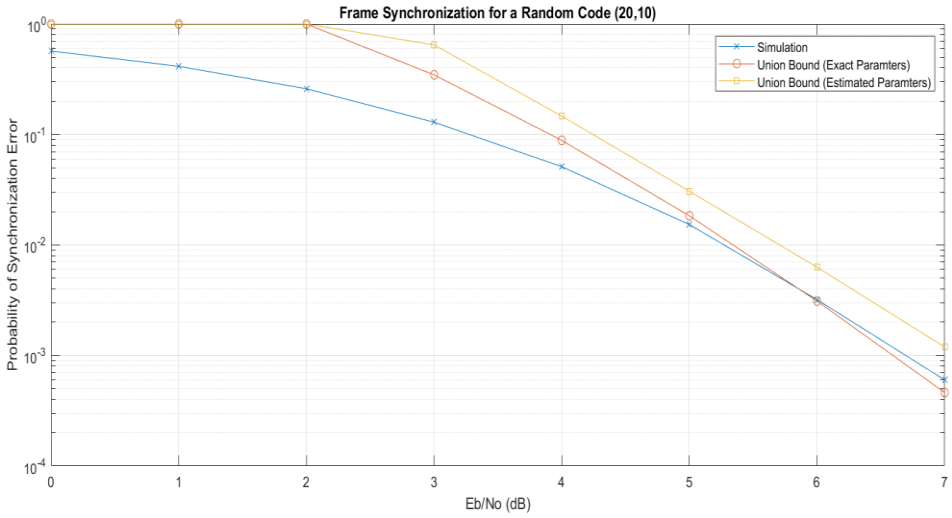

Figure 7.2: Frame synchronization for a random code (20,10)

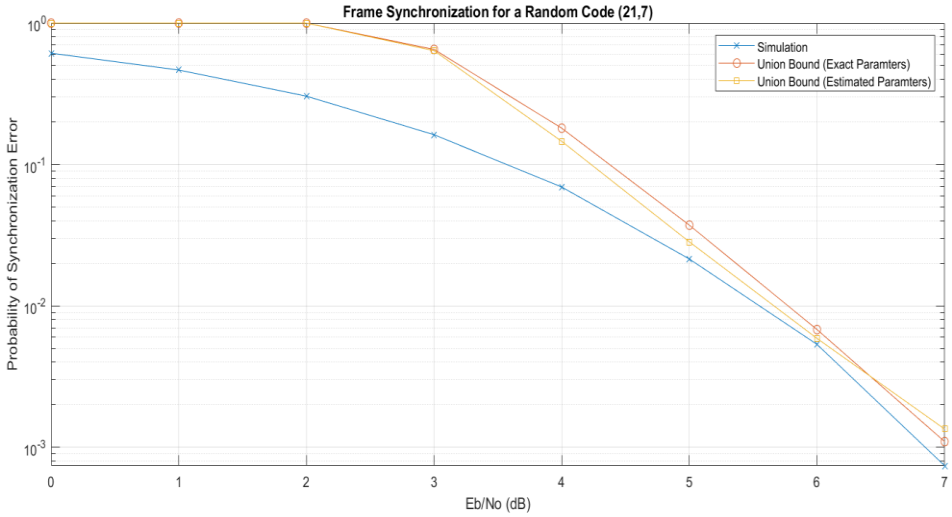

Figure 7.3: Frame synchronization for a random code (21,7)

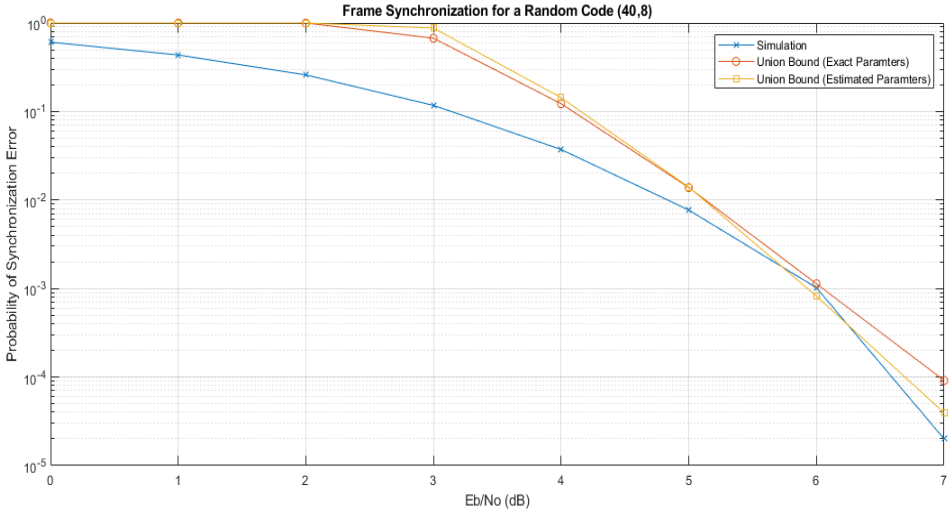

Figure 7.4: Frame synchronization for a random code (40,8)

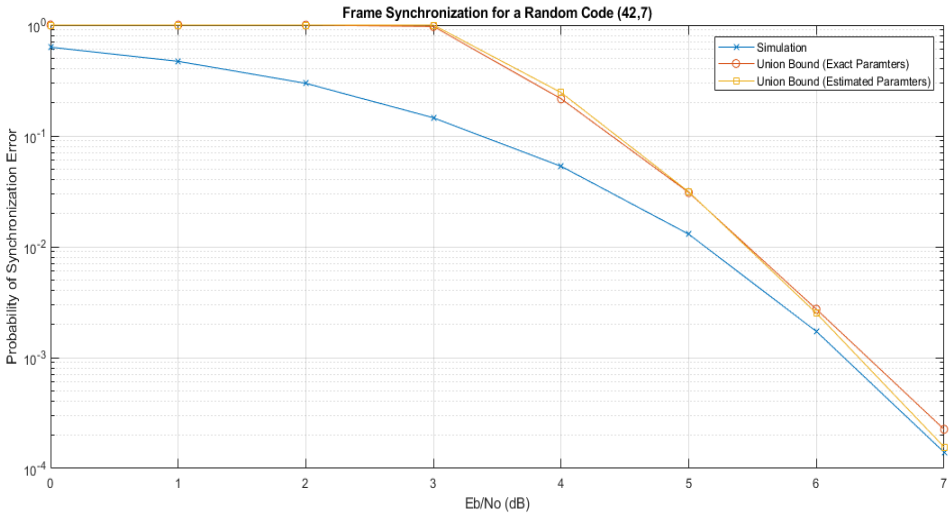

Figure 7.5: Frame synchronization for a random code (42,7)

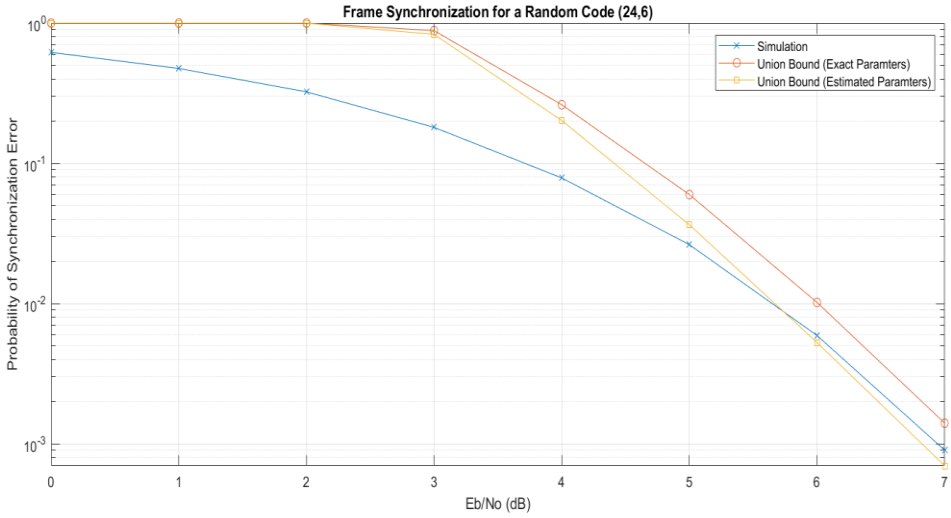

Figure 7.6: Frame synchronization for a random code (24,6)

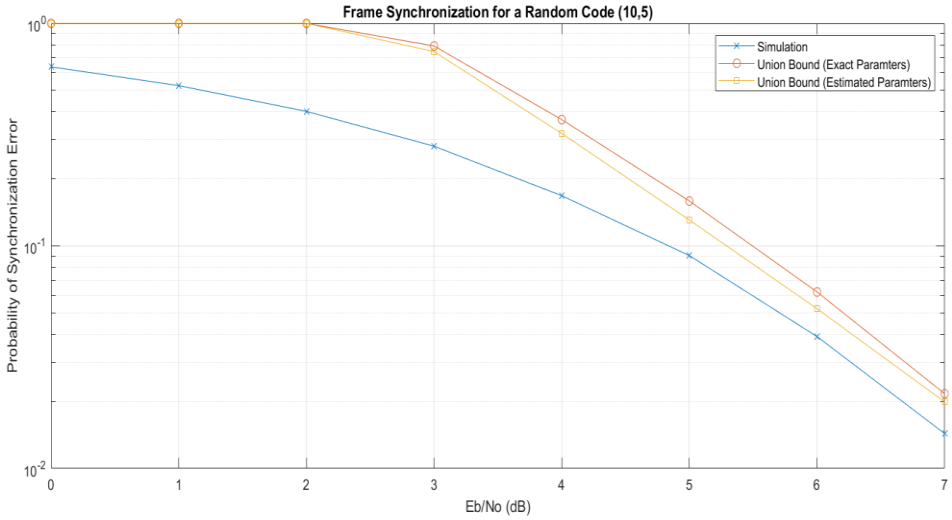

Figure 7.7: Frame synchronization for a random code (10,5)

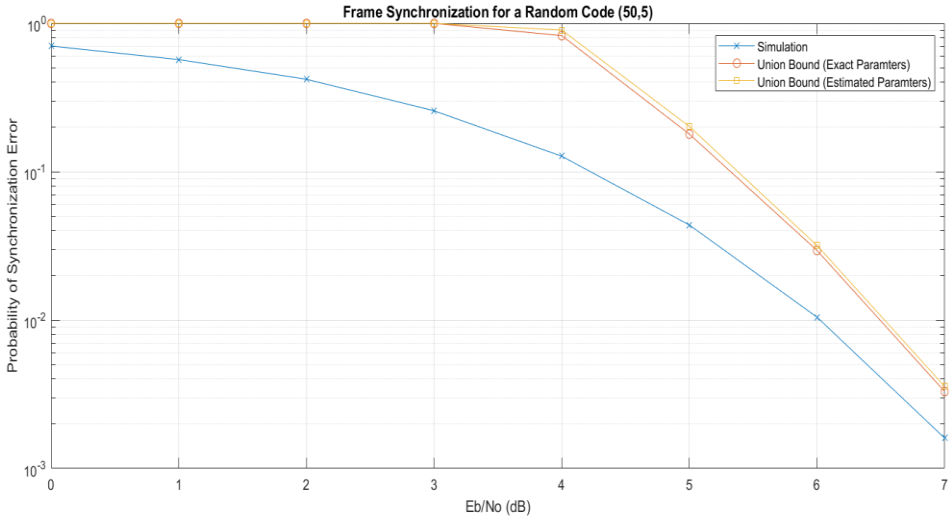

Figure 7.8: Frame synchronization for a random code (50,5)

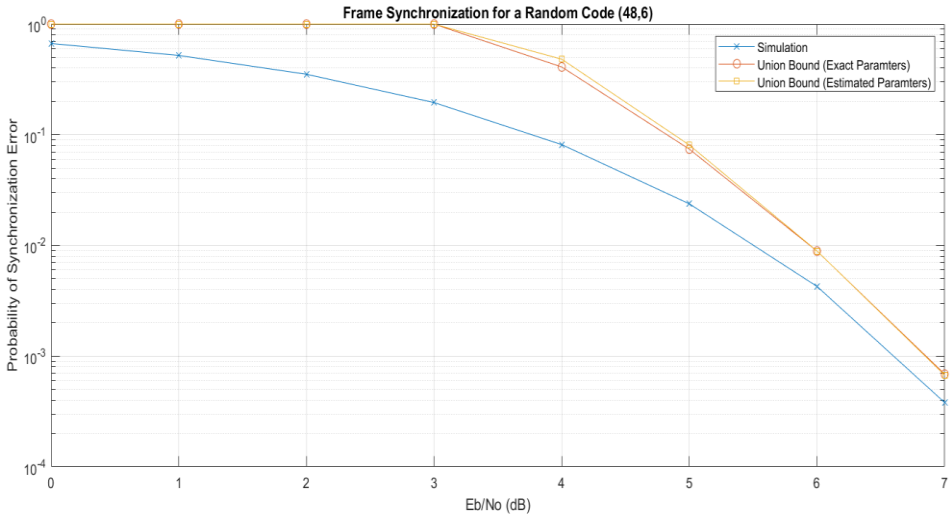

Figure 7.9: Frame synchronization for a random code (48,6)

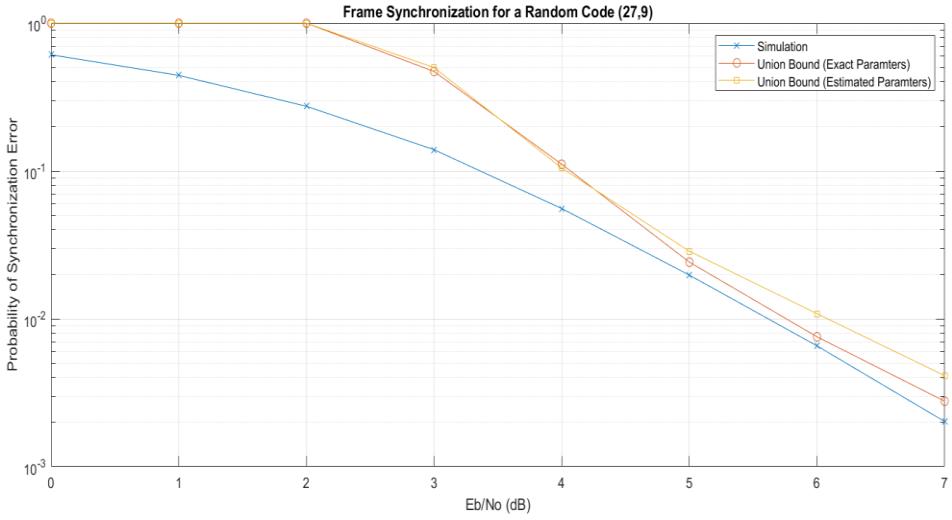

Figure 7.10: Frame synchronization for a random code (27,9)

CHAPTER VIII

EFFECT OF DIFFERENT PARAMETERS ON PERFORMANCE FOR THE LOW DATA RATE COMMUNICATION SYSTEM

Finally, we combined all of the synchronization stages to see the effect of propagating errors on the overall performance of the communication system. The performance is measured in terms of the probability of packet loss where 10 information bits are transmitted in these simulations.

Effect of Code Parameters

We experimented with different convolutional codes by changing the desired code rate and the constraint length each time to find the best possible code to use in this scenario. Figures 8.1-8.3 show the results of these simulations. For each desired code rate there is a minimal difference in performance between the different constraint lengths. This indicates that most of the losses are due to synchronization errors rather than the error-correction capability of the codes. It can also be seen that the codes with a desired rate of 1/2 outperform those with a desired rate of 1/3 or 1/4 by a small margin. This is due to the fact that lowering the code rate reduces the energy per symbol and thus increasing the chances of not capturing the entire signal in the window during coarse synchronization. In the end, the (30,10,6) convolutional code that has a desired rate of 1/2, a constraint length of 6, and an effective rate of 1/3 gave the best performance among these scenarios by a very small margin.

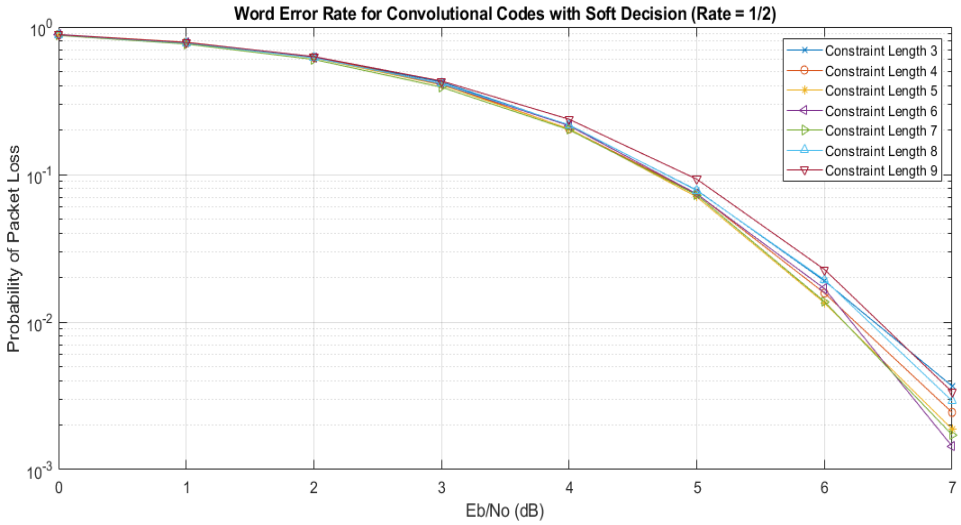

Figure 8.1: Probability of packet loss for rate 1/2 convolutional codes in the presence of synchronization errors (k=10)

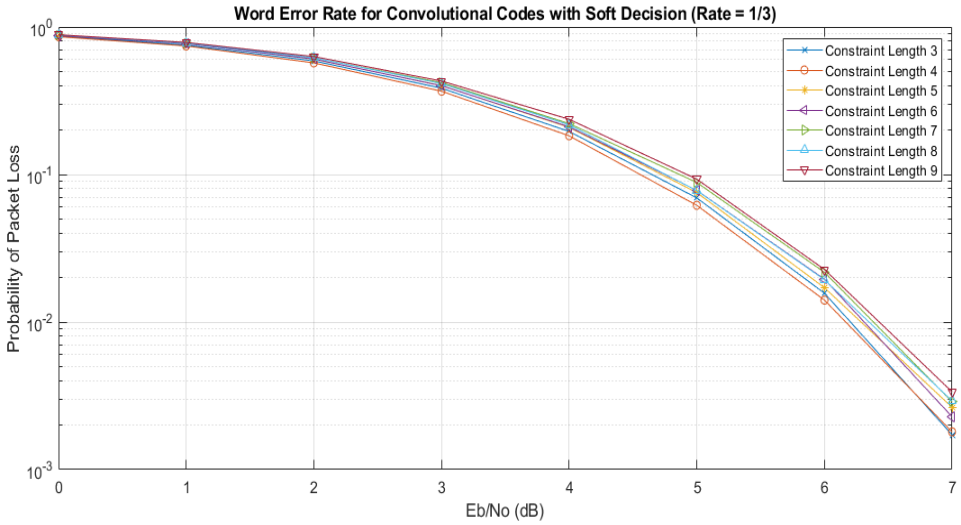

Figure 8.2: Probability of packet loss for rate 1/3 convolutional codes in the presence of synchronization errors (k=10)

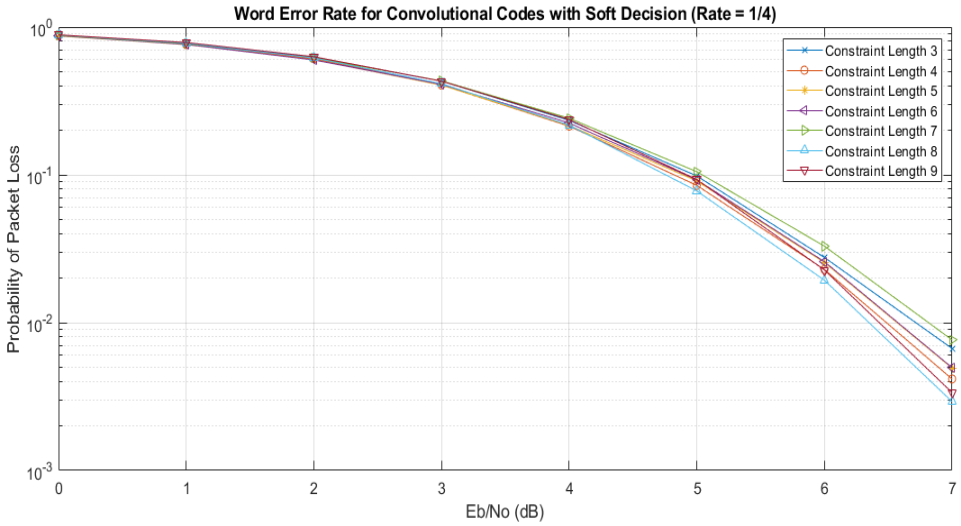

Figure 8.3: Probability of packet loss for rate 1/4 convolutional codes in the presence of synchronization errors (k=10)

However, if we repeat the same exercise with perfect synchronization, we will find that a (42,10,5) convolutional code achieved the best performance in terms of probability of packet loss. The conclusion is, when it comes to short packets, synchronization must be taken into account while designing the communication system and the signal parameters that guarantee the best performance under perfect synchronization are not necessarily the same under synchronization errors. The results for perfect synchronization are shown in figures 8.4-8.6.

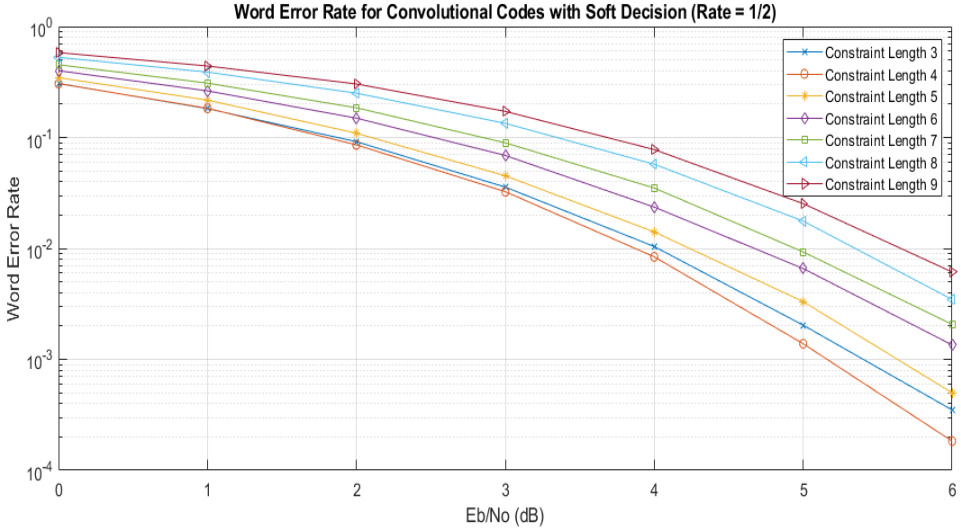

Figure 8.4: Probability of packet loss for rate 1/2 convolutional codes with perfect synchronization (k=10)

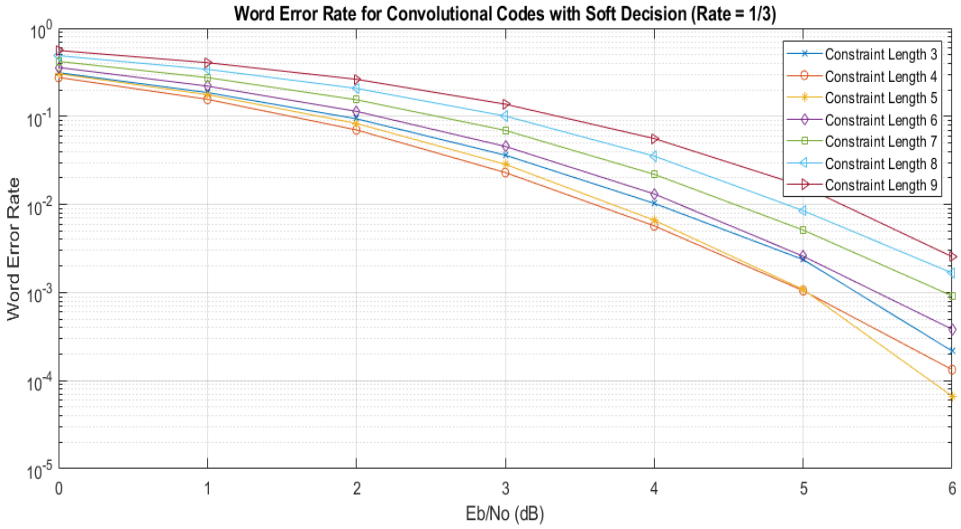

Figure 8.5: Probability of packet loss for rate 1/3 convolutional codes with perfect synchronization (k=10)

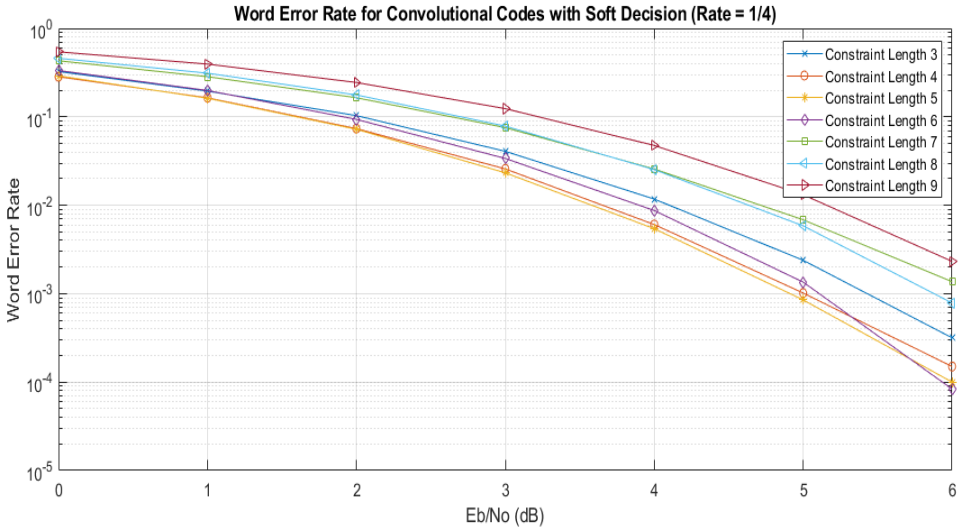

Figure 8.6: Probability of packet loss for rate 1/4 convolutional codes with perfect synchronization (k=10)

Effect of the Smaller Uncertainty Window Size

Next, we wanted to see the effect of the smaller uncertainty window size on the overall performance so we ran the simulations for different window coefficients while the initial window coefficient is $M = 20$. Figure 8.7 shows that a window that is twice the size of the transmitted signal is the best option in this scenario. These results indicate that when the window is too small, we are less likely to capture the entire signal at the coarse and symbol synchronization stage. But in case we do, we have a better chance of performing the phase and frame synchronization correctly as there is less noise in the window and vice versa for when the window is too large. Therefore, the window needs to have a moderate size.

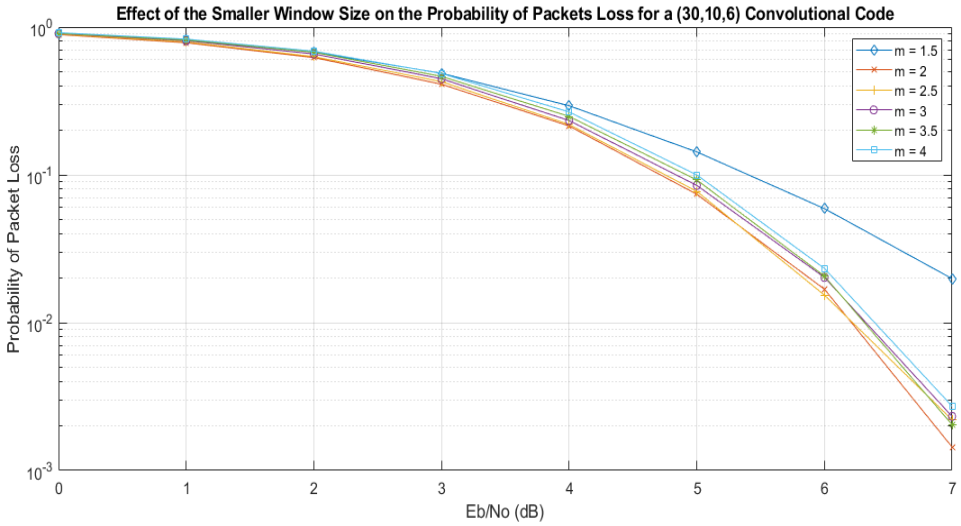

Figure 8.7: Effect of window size on the probability of packet loss for a (30,10,6) convolutional code

Effect of Imperfect Synchronization and Propagating Errors

We wanted to investigate the source of the major losses when all synchronization stages were combined together and whether that is the best possible performance we can get. We ran a few simulations showing how losses increase with the addition of each synchronization stage. We start with perfect synchronization where the exact time delay and phase offset are known, then we implement each synchronization stage one by one as described previously and combine them to see the losses when synchronization is imperfect. The results are illustrated in figure 8.8. In these simulations, $M = 20$, $m = 2$, and the chosen code is a (30,10,6) convolutional code.

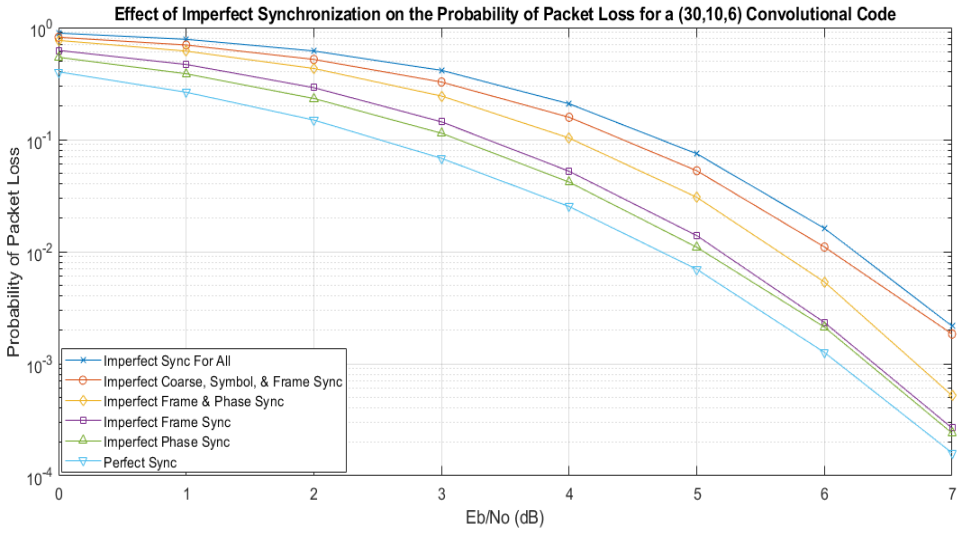

Figure 8.8: Effect of imperfect synchronization on the probability of packet loss for a (30,10,6) convolutional code

Table 8.1 summarizes the results of figure 8.8 and shows the losses from perfect synchronization after the imperfect implementation of each synchronization stage when using a (30,10,6) convolutional code at a 0.2% probability of packet loss.

Table 8.1: Summary of losses due to the imperfect implementation of each synchronization stage

Imperfect implementation of:	Approximate loss from perfect synchronization at 0.2% probability of packet loss (dB)
Phase Synchronization	0.3
Frame Synchronization	0.35
Phase and Frame Synchronization	0.5
Coarse, Symbol, and Frame Synchronization	1.3
Coarse, Symbol, Phase, and Frame Synchronization	1.35

From what we can see in figure 8.8 and table 8.1, frame and phase synchronization have a much smaller impact on the degradation of performance compared to coarse and symbol synchronization after the implementation of the proposed frame synchronization algorithm. The frame synchronization algorithm we developed was proven to reach the best possible performance under soft-decision decoding. Meanwhile, coarse and symbol synchronization can be problematic as they precede frame synchronization and their losses will propagate to the next stages. If the entire signal was not captured during coarse synchronization, the remaining steps will all have to make decisions based on the wrong input. If the correct sampling time was not

chosen during symbol synchronization, the signal that will be forwarded to the frame synchronizer will be weaker than it should be which will diminish our chances of decoding the word correctly.

Effect of Pilot Symbols

We investigated the use of pilot symbols and whether they can help us in closing the gap between the current performance and perfect performance. Adding pilot symbols require making small changes to the synchronization algorithms previously described. At the coarse and symbol synchronization stages, the procedure is the same but S_t is now defined by:

$$S_t = \sum_{i=0}^{n+L-1} \left| r[t + iN_{sps}] \right|^2 + \sum_{i=0}^{L-1} r[t + iN_{sps}] p_i \ \ for\ t = 0,1, \dots, (M-1)K - 1 \tag{8.1}$$

where L is the number of pilot bits and p_i is the ith sample in the pilot sequence. The phase estimation procedure remains the same and the pilot symbols are incorporated in the frame synchronization algorithm to resolve the ambiguity.

At the frame synchronization stage, a sliding window $\boldsymbol{C}$ of length $n + L$ slides across the smaller uncertainty window of size $m(n + L)$ all the way to the last possible time delay and evaluates a correlation metric for each time delay by following these steps:

- Step 1: Strip the first L bits that are supposed to be the pilot symbols.
- Step 2: De-interleave the remaining n bits inside the sliding window in case interleaving was used.

- Step 3: Decode those n bits by following the decoding algorithm for the chosen (n, k) code.
- Step 4: Re-encode the resulting k information bits according to the encoding algorithm for the (n, k) code.
- Step 5: Re-interleave the n coded bits in case interleaving was used.
- Step 6: Attach the known pilot bits ahead of the newly found n coded bits.
- Step 7: Re-modulate the resulting $n + L$ bits according to the modulation scheme that was used at the transmitter. This will create a new window $\boldsymbol{C'}$.
- Step 8: Calculate the correlation ρ_j at the jth time delay between the contents of $\boldsymbol{C}$ and the contents of $\boldsymbol{C'}$ according to:

$$\rho_j = \sum_{i=0}^{n-1} C_i \, C'_i \; for \; j = 0,1, \dots, (m-1)(n+L) - 1 \tag{8.2}$$

- Step 9: Choose the time delay that maximizes the correlation according to:

$$t_0 = \underset{t=0,\dots,(m-1)(n+l)-1}{\operatorname{argmax}} \rho_t \tag{8.3}$$

We experimented with different numbers of pilot bits attached to the beginning of the signal and ran the simulations to see the change in performance. Figure 8.9 shows that pilot symbols did not improve the performance but made it worse. This is because the cost of adding the extra pilot bits outweighs any gain we can achieve in performance. In order for a pilot sequence to be effective in synchronization, it needs to be long and unique that it can be easily detected but also much shorter compared to the information portion of the signal. Since the packet is very short (10 information bits), the pilot sequence might end up being as long as the packet itself which causes a significant waste

of energy. On the contrary, if the packet was 1000 bits long, a pilot sequence that is 100 bits long can be easily detected and has a negligible size compared to the data. Hence, we can reliably perform synchronization at a minimum cost of overhead.

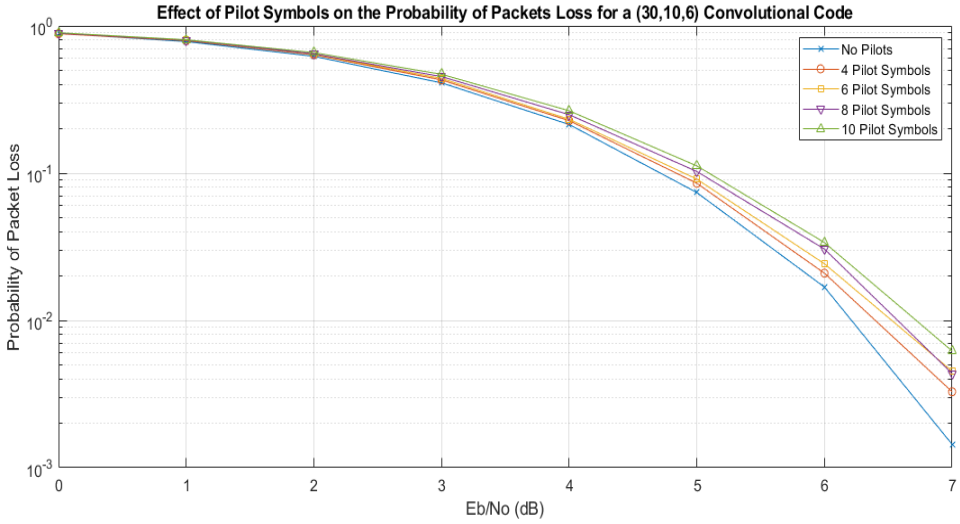

Figure 8.9: Effect of pilot symbols on the probability of packet loss for a (30,10,6) convolutional code

Potential Improvement

Now, we want to know how far are we from the best achievable performance in this scenario. We tested a coarse and symbol synchronization algorithm that we call “one step above optimal” in performance. In this algorithm, we assumed that we know exactly what the transmitted sequence is so at the coarse synchronization stage, we compute a cross-correlation between the transmitted sequence and the received signal and then choose the time delay that returned the highest correlation as a rough estimate of the true

time delay. Then for each sampling time we computed the cross-correlation of the transmitted sequence with the samples of the matched filter output and chose the sampling time that maximizes this correlation.

We ran the simulation for that algorithm and the results are shown in figure 8.10 for the (30,10,6) convolutional code. It can be seen that with a "genie aided" coarse and symbol synchronization algorithm, there is a 0.5dB loss from perfect synchronization where the exact time delay and phase offset are known. There is also less than 0.5dB difference between the current decoder-assisted synchronization and the genie aided coarse and symbol synchronization. The optimal coarse and symbol synchronizer would be somewhere in between those two so any improvement we can make will not gain more than 0.5dB. The optimal coarse and symbol synchronizer will have to compute the cross-correlation between every possible transmitted codeword and the received signal. Then, it will pick the time delay and the sampling time with the highest correlation in both cases.

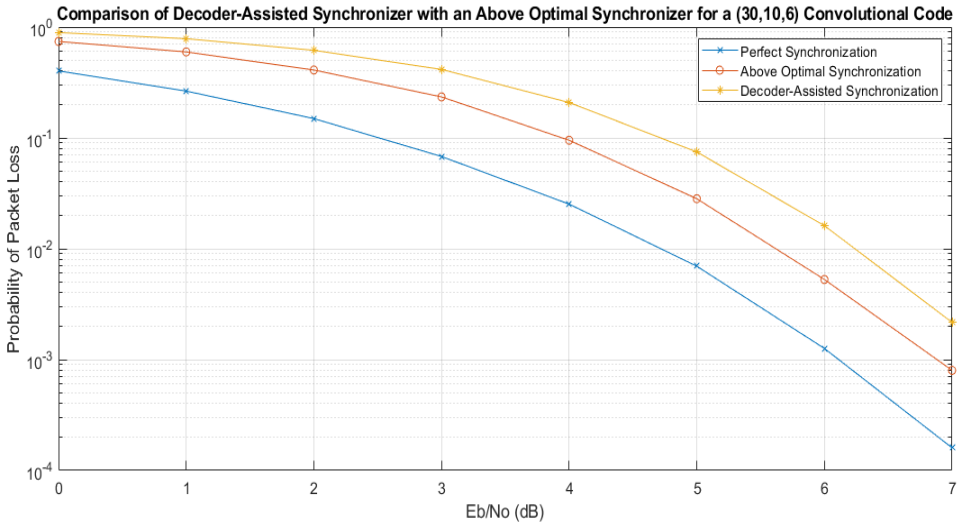

Figure 8.10: Comparison of the decoder-assisted synchronization with the above optimal synchronization

Application to the Pipe Strings Channel

After we optimized the different parameters of the signal to find best performance under an AWGN channel, we put the resulting signaling format through the pipe strings and used the proposed synchronization procedure to see the difference in performance. As we noted previously, the pipe strings have frequency selective properties and can cause ISI. So we need to test the effect of different parameters on performance.

Effect of Measurement Location

As it was indicated previously, when the received signal is measured at the beginning or the end of the pipe segment, the channel response in figure 2.8 is somewhat flat. But if the signal was measured in the middle of the pipe segment, the channel

response in figure 2.9 is slanted and can offer a significant boost to the signal. We put that theory to the test and simulated the transmitted signal that was designed to improve synchronization. The signal, shown in figure 8.11, was recorded at the end and in the middle of the 10^{th} pipe segment in a total of 100 pipe segments.

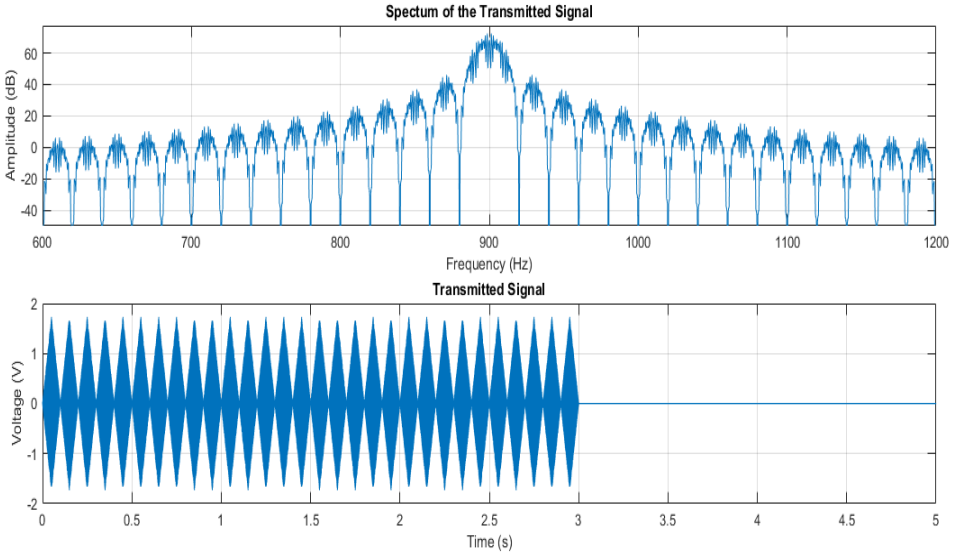

Figure 8.11: Transmitted signal in time and frequency domains

The second passband (796-994 Hz) in figure 2.9 was used for transmission in both cases since it offered the highest channel gain when the signal was recorded in the middle. The carrier frequency was set to 900Hz near the center of the passband. As shown in figure 8.12, recording the signal in the middle of the pipe segment offers a significant improvement and boosts the power of the signal for better synchronization and detection. The channel gain at the chosen carrier frequency was 2dB which is why it

outperforms the AWGN channel. However, the amount of ISI from dispersion and all the reflections will reduce the gain. For the next simulations, all the measurements will be performed in the middle of the pipe segments.

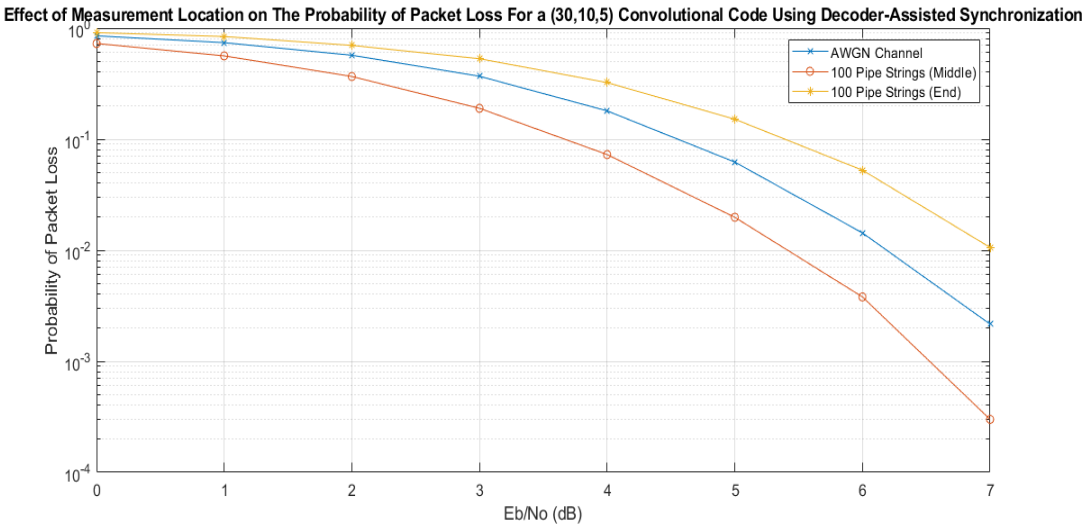

Figure 8.12: Effect of the measurement location on the probability of packet loss for a (30,10,5) convolutional code using decoder-assisted synchronization

Effect of the Bit Rate

The different reflections coming from the joints at the beginning and end of the pipe strings can cause ISI when they arrive at the measurement location before the end of the original copy of the signal. This distortion can be limited by simply increasing the bit rate. Increasing the bit rate makes the signal shorter in the time domain. Since the arrival time of the reflections depends only on the measurement location and the speed of propagation, reducing the bit rate will delay the reflections relative to the end of the

original copy of the signal and consequently reducing the amount of ISI in the channel. However, increasing the bit rate also increases the bandwidth and risks a portion of the signal bandwidth falling in a stopband leading to more losses. Hence, the bit rate needs to be carefully increased such that most of the signal still falls inside the desired passband.

We tested this theory with three different bit rates of 10, 40, and 50 bits/sec. The resulting bandwidth was about 40Hz, 140Hz and 175Hz for the 10, 40, and 50 bits/sec cases respectively. For all cases, the bandwidth was defined as the limiting frequency of the signal at 40dB down from its peak at the center frequency. The results of the simulation are shown in figure 8.13 indicating an improvement for the bit rate increase. For the rest of the simulations, 40 bits/sec was used in all cases since 50 bits/sec did not offer any significant advantage from the 40 bits/sec case.

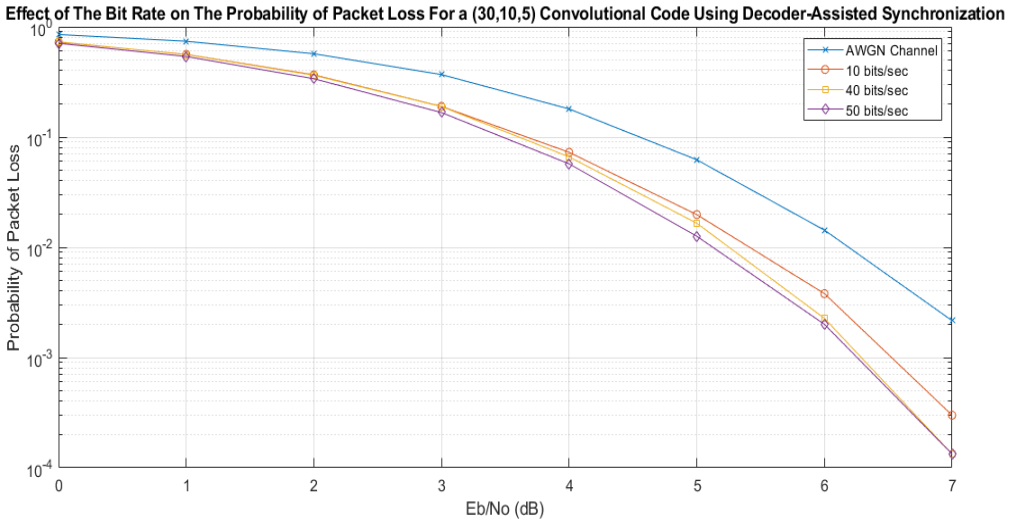

Figure 8.13: Effect of the bit rate on the probability of packet loss for a (30,10,5) convolutional code using decoder-assisted synchronization

Effect of the Number of Pipe Segments

Another parameter that affects the amount of ISI in the received signal is the number of pipe segments in the testbed. When more pipe segments are present, the stronger reflections coming from the end of the pipe strings have to travel longer distances before they get recorded at the receiver. This will also delay the reflections with respect to the original copy of the signal and hence reduce ISI. Increasing the number of pipe segments will also cause more reflections from the joints but those are significantly weaker and will not affect the performance as much. We ran the simulation for 50, 100, and 200 pipe segments where the signal, was recorded in the middle of the 10^{th} pipe segment. The results are shown in figure 8.14 and indicate a major improvement when more pipe segments are present and thus verifying our prediction.

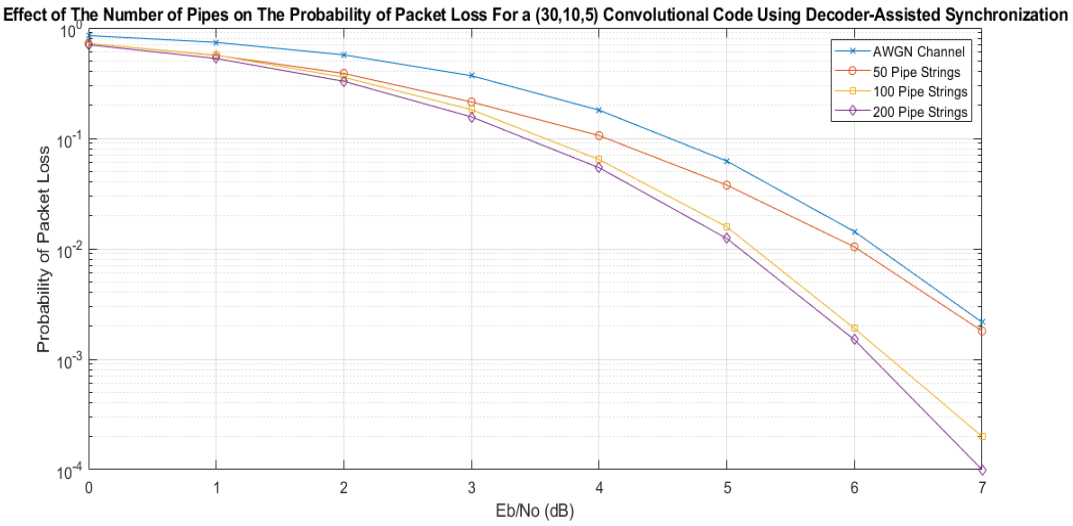

Figure 8.14: Effect of the number of pipes on the probability of packet loss for a (30,10,5) convolutional code using decoder-assisted synchronization

Effect of Propagation Distance

Finally, we tested the effect of the propagation distance on performance to see how much the signal can travel down the pipe strings before performance is significantly degraded. We ran the simulation for 100 pipe segments and the signal was recorded in the middle of the 10^{th}, 20^{th}, 30^{th}, 40^{th}, and 50^{th} pipe segment. The results of figure 8.15 show a significant degradation in performance beyond the 20^{th} pipe segments. This degradation is caused by both the exponential decay of the signal amplitude as it travels for longer distances and the proximity of the measurement location to the end of the pipe strings causing the reflections to arrive sooner and creating more ISI.

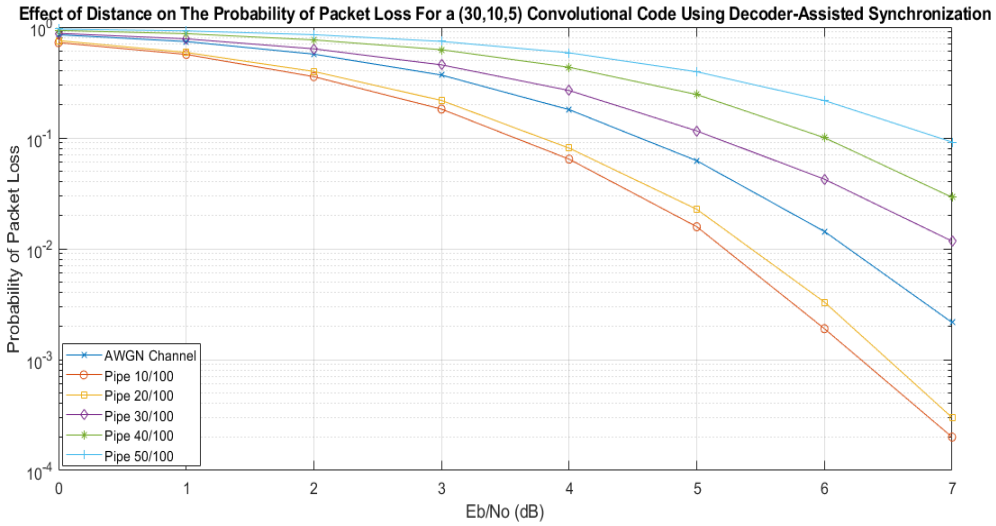

Figure 8.15: Effect of distance on the probability of packet loss for a (30,10,5) convolutional code using decoder-assisted synchronization

In a more realistic scenario, the field testing done in [45] showed that the used transmitter created a 1V signal while the noise at the receiving equipment was 1mV. The resulting SNR in this configuration is 60dB calculated by:

$$SNR = 20\log_{10}\frac{1}{1\times 10^{-3}} = 60dB \tag{8.4}$$

We would like to see how much strength the signal loses are a function of distance so we ran the experiment for a distance between 1 and 6km. Since each pipe segment and joint are 10m long, this means that the signal needs to travel for between 100 and 600 pipe segments. Table 8.2 shows how much strength the signal loses as a function of distance.

Table 8.2: Signal strength as a function of distance

Travelled Distance (km)	1	2	3	4	5	6
Signal Strength Decay (dB)	2	10	17	25	30	37

As we can see from table 8.3 and at a distance of 6km, the signal is still at 23dB above the noise level from its initial 60dB and the probability of packet loss at that distance was less than 0.0004%.

CHAPTER IX

CONCLUSIONS

Summary

In this dissertation, we attempted to design two modes of communication to overcome the challenges imposed by the acoustic vibration of pipe strings for a sensor-based monitoring and measurements of a given parameter. The first mode requires high data rate transmission of either an image or a video in cases of emergency. The second mode requires low data rate transmission of periodic sensor readings consisting of a very short packet of bits. We first described the channel response and studied the behavior of passbands and stopbands in its frequency response. For the high data rate communication system, we designed an OFDM based system whose bandwidth spans multiple passbands and used error-correction coding to recover the lost information. For the low data rate communication system, we designed an appropriate signaling format and focused on the synchronization problem for very short packets and came up with the appropriate synchronization procedure to improve performance.

Main Findings

Channel Characterization

The pipe strings used for transmission of information have a unique frequency response characterized by the presence of several passbands and stopbands that exhibit the following behaviors:

- The passbands get narrower for higher frequencies while the stopbands get wider.
- The passbands get weaker as a function of both increasing frequency and propagation distance.
- The stopbands get deeper as a function of both increasing frequency and propagation distance.
- The passband strength is dependent on the measurement location within each pipe string.
- The passbands exhibit several fluctuations that decay as function of increasing frequency and propagation distance.

This channel also causes a phase offset to the signal and adds Inter-Symbol-Interference to the desired wave due to the accumulation of different reflections coming from both edges of the pipe strings and from the multiple joints along the way.

High Data Rate Communication System

For the high data rate communication system, the data rate offered by the passbands was not able to accommodate the transmission of video but was able to accommodate colored images within the given time constraints.

The proposed OFDM system was able to achieve desirable levels of performance if the different parameters were tuned appropriately to guarantee better energy efficiency. The main conclusions are summarized below:

- Smaller frequency spacing for the OFDM subcarriers offered a better performance since it avoids the fluctuations in the passband by making the

channel response for each subcarrier almost flat. The frequency spacing cannot be more than 2Hz.

- A moderate length cyclic prefix can balance between the extra overhead cost and the elimination of ISI for the SNR region of interest. The cyclic prefix should not be less than 50% of each OFDM symbol. Longer cyclic prefix will give an overall worse performance as the cost of overhead begins to outweigh the benefit in limiting ISI.
- Increasing the signal bandwidth and including more passbands will result in a poorer performance.
- The use of the first 3 passbands for transmission offered the best performance in terms of achievable depth. The signal for the 16 colors image was able to travel slightly more than 6km while the signal for the 256 colors image was able to travel slightly more than 5km.

Low Data Rate Communication System

For the low data rate single carrier communication system, we examined the problem of synchronization for very short packets and studied the effects of synchronization errors on the probability of packet loss. We also proposed a simple and straight forward frame synchronization algorithm for short packet communication that exploits the presence of error-correction codes to perform the synchronization. Finally, we combined all synchronization stages and observed the loss in performance incurred by imperfect coarse, symbol, phase, and frame synchronization. The main conclusions of this section are summarized below:

- The size of the uncertainty window has to be reasonably small for reliable synchronization of very short signals. We proposed scheduled transmissions to help us achieve better synchronization.
- The pulse shape affects synchronization and must be chosen to maximize the performance. We laid out the criteria that need to be satisfied by the pulse shape to improve performance and chose the pulse shape that satisfied all of them.
- The choice of error-correction code affects frame synchronization on short packets and convolutional codes seem to offer the best performance for very short packets.
- The code parameters affect the performance and there needs to be a balance between the error-correction capability and the energy allocation per bit.
- The size of the smaller uncertainty window needs to be optimized to balance between coarse synchronization and the other synchronization stages.
- Coarse and symbol synchronization cause more losses to the overall performance.
- The cost of adding pilot symbols outweighs any gain in performance.
- The achieved performance is very close to the best possible performance given the very short nature of the transmitted packet.
- Recording the signal in the middle of the pipe strings instead of the beginning or end boosts the power of the received signal by benefiting from the channel gain.

- Increasing the bit rate helps reducing ISI by delaying the reflections with respect to the tail of the received signal. The bit rate increase must keep in mind the limits of the chosen passband.
- The addition of more pipe strings to the system improves performance by delaying the reflections and reducing the amount of ISI.
- The system performance rapidly degrades as a function of the propagating distance due to the loss in the signal's amplitude and the amplification of ISI as the signal gets closer to the end.
- The proposed signal was able to travel more than 6km while maintaining a probability of packet loss above the performance threshold.

REFERENCES

[1] M. Alard and R. Lassalle, "Principles of modulation and channel coding for digital broadcasting for mobile receivers," *EBU Technical Review,* no. 224, pp.186-191, August 1987.

[2] T. K. Moon, *Error Correction Coding: Mathematical Methods and Algorithms,* Wiley, 2005.

[3] D. S. Yoo, W. E. Stark, K. P. Yar, and S. J. Oh, "Coding and Modulation for Short Packet Transmission," *IEEE Transactions on Vehicular Technology,* vol. 59, no. 4, pp. 2104-2109, January 2010.

[4] C. Berger, S. Zhou, Y. Wen, P. Willett, and K. Pattipati, "Optimizing Joint Erasure and Error Correction Coding for Wireless Packet Transmissions," *IEEE Transactions on Wireless Communications,* vol. 7, no. 11, pp. 4586-4595, December 2008.

[5] G. Durisi, T. Koch, J. Ostman, Y. Polyanskiy, and W. Yang, "Short-Packet Communications over Multiple-Antenna Rayleigh-Fading Channels," *IEEE Transactions on Communications,* vol. 64, no. 2, pp. 618–629, February 2016.

[6] G. Durisi, T. Koch, J. Ostman, Y. Polyanskiy, and W. Yang, "Short-Packet Communications with Multiple Antennas: Transmit Diversity, Spatial Multiplexing, and Channel Estimation Overhead," *2014 IEEE International Symposium on Wireless Communication Systems,* December 2014.

[7] G. Durisi, T. Koch, P. Popovski, "Towards Massive, Ultra-Reliable, and Low-Latency Wireless Communication with Short Packets," *Proceedings of the IEEE*, vol. 104, no. 9, pp. 1711-1726, August 2016.

[8] D. Slepian, "Bounds on Communication," *The Bell System Technical Journal*, vol. 42, no. 3, pp. 681-707, May 1963.

[9] C. E. Shannon, "Probability of Error for Optimal Codes in a Gaussian Channel," *The Bell System Technical Journal*, vol. 38, no. 3, May 1959.

[10] S. Dolinar, D. Divsalar, and F. Pollara, "Code Performance as a Function of Block Size," *TMO Progress Report* 42-133, May 1998.

[11] Y. Huang, U. Nieren, and P. Gupta, "Energy-Efficient Communication in the Presence of Synchronization Errors," *2013 IEEE International Symposium on Information Theory*, July 2013.

[12] J. K. Wolf and A. J. Viterbi, "On the Weight Distribution of Linear Block Codes Formed from Convolutional Codes," *IEEE Transactions on Communications*, vol. 44, no. 9, pp. 1049-1051, September 1996.

[13] J. Stiffler, "Comma-Free Error-Correcting Codes," *IEEE Transactions on Information Theory*, vol. 11, no. 1, pp. 107-112, January 1965.

[14] W. Eastman, "On the Construction of Comma-Free Codes," *IEEE Transactions on Information Theory*, vol. 11, no. 2, pp. 263-267, April 1965.

[15] Y. Fujiwara and V. D. Tonchev, "High-rate self-synchronizing codes," *IEEE Transactions on Information Theory*, vol. 59, no. 4, pp. 2328-2335, December 2012.

[16] F. Gardner, “A BPSK/QPSK Timing-Error Detector for Sampled Receivers,” *IEEE Transactions on Communications*, vol. COM-34, no. 5, pp. 423-429, 1986.

[17] Bernard Sklar, *Digital Communications: Fundamentals and Applications*, Prentice-Hal, 1988.

[18] K. Mueller and M. Muller, "Timing Recovery in Digital Synchronous Data Receivers," *IEEE Transactions on Communications*, vol. COM-24, pp. 516-531, May 1976.

[19] J. Bhatti and M. Moeneclaey, “Iterative soft-decision-directed phase noise estimation from a DCT basis expansion,” *IEEE 20th International Symposium on Personal, Indoor and Mobile Radio Communications*, September 2009.

[20] F. Simoens, D. Duyck, H. Çırpan, E. Panayirci, and M. Moeneclaey, “Monte Carlo Solutions for Blind Phase Noise Estimation,” *EURASIP Journal on Wireless Communications and Networking*, 2009.

[21] Y. Rahamim, A. Freedman, and A. Richman, “Methods for Carrier Synchronization of Short Packet Turbo Coded Signals,” *2004 IEEE 15th International Symposium on Personal, Indoor and Mobile Radio Communications*, September 2004.

[22] H. Wymeersch, H. Steendam, H. Bruneel, and M. Moeneclaey, “Code-Aided Frame Synchronization and Phase Ambiguity Resolution,” *IEEE Transactions on Signal Processing*, vol. 54, no. 7, July 2006.

[23] J. L. Massey, “Optimum frame synchronization,” *IEEE Transactions on Communications*, vol. 20, no. 4, pp. 115–119, April 1972.

[24] P. Robertson, “A generalized frame synchronizer,” *Proceedings of GLOBECOM*, December 1992, pp. 365–369.

[25] P. Robertson, “Optimum frame synchronization of packets surrounded by noise with coherent and differentially coherent demodulation,” *Proceedings of International Conference on Communications*, pp. 874–879, May 1994.

[26] B. H. Moon and S. S. Soliman, “ML frame synchronization for the Gaussian channel with ISI,” *Proceedings of IEEE International Conference on Communications*, Denver, CO, June 1991, pp. 1698–1702.

[27] P. Robertson, “Maximum likelihood frame synchronization for flat fading channels,” *Proceedings of International Conference on Communications*, pp. 1426–1430, IEEE, 1992.

[28] G. L. Lui and H. H. Tan, “Frame synchronization for Gaussian channels,” *IEEE Transactions on Communications*, vol. 35, no. 8, pp. 818–829, August 1987.

[29] P. Robertson, “Improving frame synchronization when using convolutional codes,” *Proceedings of IEEE GLOBECOM*, pp 1606-1611, Houston, 1993.

[30] H. Huh and J. Krogmeier, “A unified approach to optimum frame synchronization,” *IEEE Transactions on Wireless Communications*, vol. 5, pp. 3700–3711, December 2006.

[31] T. M. Cassaro and C. N. Georghiades, “Frame synchronization for coded systems over AWGN channels,” *IEEE Transactions on Communications*, vol. 32, pp. 484–489, March 2004.

[32] M.K. Howlader and B.D. Woerner, “Decoder-assisted frame synchronization for packet transmission,” *IEEE Journal on Selected Areas in Communications,* vol. 19, no. 12, pp. 2331-2345, December 2001.

[32] S. Houcke and G. Sicot, “Blind frame synchronization for block code,” *Proceedings of EUSIPCO,* European Signal Processing, September 2006.

[33] Z. Jing, H. Zhiping, S. Shaojing, Y. Shaowu, “Blind recognition of binary cyclic codes,” *EURASIP Journal on Wireless Communications and Networking*, December 2013.

[34] C. Herzet, K. Woradit, H. Wymeersch, and L. Vandendorpe, “Code-aided maximum-likelihood ambiguity resolution through free-energy minimization,” *IEEE Transactions on Signal Processing*, vol. 58, pp. 6238–6250, December 2010.

[35] F. R. Kschischang, B. J. Frey, and H. A. Loeliger, “Factor graphs and the sum-product algorithm,” *IEEE Transactions on Information Theory,* vol. 47, no. 2, pp. 498–519, February 2001.

[36] H. Wymeersch and M. Moeneclaey, “ML frame synchronization for turbo and LDPC codes,” *International Symposium on Turbo Codes & Related Topics,* ISTC’, Brest, France, September 2003.

[37] J. Sun and M. Valenti, “Optimum Frame Synchronization for Preamble-less Packet Transmission of Turbo Codes,” *Conference Record of the Thirty-Eighth Asilomar Conference on Signals, Systems and Computers,* November 2004.

[38] D. Jakubisin, C. I. Phelps, and R. M. Buehrer, “Iterative Joint Detection, Decoding, and Synchronization with a Focus on Frame Timing,” *2014 IEEE Wireless Communications and Networking Conference (WCNC)*, November 2014.

[39] H. Wymeersch, H. Steendam, H. Bruneel, and M. Moeneclaey, “Code-aided frame synchronization and phase ambiguity resolution,” *IEEE Transactions on Signal Processing*, vol. 54, pp. 2747–2757, July 2006.

[40] C. Herzet, H. Wymeersch, F. Simoens, M. Moeneclaey, and L. Vandendorpe, “MAP-based code-aided hypothesis testing,” *IEEE Transactions on Wireless Communication*, vol. 7, no. 8, pp. 2856–2860, August 2008.

[41] R. Imad, G. Sicot, and S. Houcke, “Blind frame synchronization for error correcting codes having a sparse parity check matrix,” *IEEE Transactions on Communication*, vol. 57, no. 6, pp. 1574–1577, June 2009.

[43] R. Imad and S. Houcke, “Theoretical analysis of a MAP based blind frame synchronizer,” *IEEE Transactions on Wireless Communication*, vol. 8, no. 11, pp. 5472–5476, November 2009.

[44] D. S. Drumheller, “Acoustical Properties of Drill Strings,” *The Journal of the Acoustical Society of America*, vol. 85, pp. 1048–1064, March 1989.

[45] A. Redissi and S. Miller, “Experimental characterization of the propagation of guided acoustic waves in pipe strings”, *The Journal of the Acoustical Society of America*, vol. 143, pp. 1385-1391, March 2018.

[46] J. Proakis, *Digital Communications*, McGraw-Hill, 1995, p. 21.

[47] X. Yu Hu, E. Eleftheriou, and D. Arbikd, "Regular and irregular progressive edge-growth tanner graphs," *IEEE Transactions on Information Theory,* vol. 51, no. 1, pp. 386-398, January 2005.

[48] A. Viterbi, "A personal history of the Viterbi algorithm," *IEEE Signal Processing Magazine*, vol. 23, no. 4, pp. 120-142, July 2006.

[49] G. Casella and R. Berger, *Statistical Inference*, Second Edition, Pacific Grove, CA: Wadsworth, 2002, p. 99.

[50] D. B. Owen and G. P. Steck, "Moments of Order Statistics from the Equicorrelated Multivariate Normal Distribution," *The Annals of Mathematica Statistics*, vol. 33, pp. 1286-1291, 1962.

[51] C. E. Clark, "The greatest of a finite set of random variables*," Operations Research*, vol. 9, no. 2, pp. 145-162, March/April 1960.

[52] D. Sinha, H. Zhou, and N. V. Shenoy, "Advances in Computation of the Maximum of a Set of Gaussian Random Variables," *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems*, vol. 26, no. 8, August 2007.

APPENDIX A

We will prove equations (5.8) and (5.9). We will split the proof in two parts. The first part will be for the noise only case and the second part will be for the presence of a signal.

Noise Only Case

Let $Z_k = X_k + jY_k$ where $j = \sqrt{-1}$, $Z_k \sim \mathcal{N}(0, \sigma^2)$, and $X_k, Y_k \sim \mathcal{N}(0, \frac{\sigma^2}{2})$ are respectively the real and imaginary parts of Z_k.

$$|Z_k|^2 = {X_k}^2 + {Y_k}^2 \tag{A.1}$$

Define ${Z'}_k$ such that:

$$|{Z'}_k|^2 = \frac{2}{\sigma^2}({X_k}^2 + {Y_k}^2) \tag{A.2}$$

Then $|{Z'}_k|^2 \sim \chi_2^2$ is a Chi-Squared random variable with 2 degrees of freedom. Hence for the mean:

$$E[|{Z'}_k|^2] = \frac{2}{\sigma^2}\left(E\left[{X_k}^2\right] + E\left[{Y_k}^2\right]\right) = 2 \tag{A.3}$$

$$\frac{2}{\sigma^2}E[|Z_k|^2] = 2 \tag{A.4}$$

$$E[|Z_k|^2] = \sigma^2 \tag{A.5}$$

Similarly, for the variance:

$$Var(|{Z'}_k|^2) = \frac{4}{\sigma^4}\left(Var\left({X_k}^2\right) + Var\left({Y_k}^2\right)\right) = 4 \tag{A.6}$$

$$\frac{4}{\sigma^4}Var(|Z_k|^2) = 4 \tag{A.7}$$

$$Var(|Z_k|^2) = \sigma^4 \tag{A.8}$$

Signal Present Case

Let $Z_k = \pm 1 + X_k + jY_k$ where $j = \sqrt{-1}$, $Z_k \sim \mathcal{N}(\pm 1, \sigma^2)$, and $X_k, Y_k \sim \mathcal{N}(0, \frac{\sigma^2}{2})$.

$$|Z_k|^2 = {X_k}^2 \pm 2X_k + E_b + {Y_k}^2 \tag{A.9}$$

The mean is found by:

$$E[|Z_k|^2] = E\left[{X_k}^2\right] \pm 2E[X_k] + 1 + E\left[{Y_k}^2\right] \tag{A.10}$$

$$E[|Z_k|^2] = 2E\left[{X_k}^2\right] + 1 = \sigma^2 + 1 \tag{A.11}$$

The variance is found by:

$$Var(|Z_k|^2) = Var\left({X_k}^2 \pm 2X_k + 1 + {Y_k}^2\right) \tag{A.12}$$

$$Var(|Z_k|^2) = Var\left({X_k}^2 + {Y_k}^2\right) + 4Var(X_k) + 4Cov\left(X_k, {X_k}^2 + {Y_k}^2\right) \tag{A.13}$$

$$Cov\left(X_k, {X_k}^2 + {Y_k}^2\right) = E\left[(X_k - E[X_k])\left({X_k}^2 + {Y_k}^2 - 2E\left[{X_k}^2\right]\right)\right] \tag{A.14}$$

$$Cov\left(X_k, {X_k}^2 + {Y_k}^2\right) = E\left[{X_k}^3 + X_k{Y_k}^2 - \sigma^2 X_k\right] = 0 \tag{A.15}$$

$$Var(|Z_k|^2) = 2Var\left({X_k}^2\right) + 4Var(X_k) = \sigma^4 + 2\sigma^2 \tag{A.16}$$

Equation (A.15) is justified by the fact that X_k and Y_k are Gaussian and uncorrelated hence X_k and ${Y_k}^2$ are also uncorrelated. In addition, since X_k has zero mean, any odd moment of X_k is zero. Equation (A.16) is justified by the fact that $\frac{2{X_k}^2}{\sigma^2}$ is Chi-Squared with 1 degree of freedom and hence:

$$Var\left(\frac{2{X_k}^2}{\sigma^2}\right) = \frac{4}{\sigma^4} Var\left({X_k}^2\right) = 2 \tag{A.17}$$

$$Var\left({X_k}^2\right) = \frac{\sigma^4}{2} \tag{A.18}$$

APPENDIX B

We will prove equations (7.12) for both noise only windows and windows with partial signal.

Noise Only Case

We start with:

$$X_j = \sum_{p=1}^{n} c_p^{(j)} w_p \tag{B.1}$$

This means that $X_j \sim \mathcal{N}(0, n\sigma^2)$. We calculate the covariance between X_i and X_j:

$$Cov(X_i, X_j) = E[X_i X_j] - E[X_i]E[X_i] = E\left[\sum_{p=1}^{n} c_p^{(i)} w_p \,.\, \sum_{q=1}^{n} c_q^{(j)} w_q\right] - 0 \tag{B.2}$$

Since the noise samples are independent from each other:

$$Cov(X_i, X_j) = E\left[\sum_{p=1}^{n} c_p^{(i)} c_p^{(j)} {w_p}^2\right] = \sigma^2 \sum_{p=1}^{n} c_p^{(i)} c_p^{(j)} \tag{B.2}$$

$$\rho_{X_i, X_j} = \frac{Cov(X_i, X_j)}{\sqrt{n\sigma^2}.\sqrt{n\sigma^2}} = \frac{\sigma^2 \sum_{p=1}^{n} c_p^{(i)} c_p^{(j)}}{n\sigma^2} = \frac{1}{n}\sum_{p=1}^{n} c_p^{(i)}.c_p^{(j)} \tag{B.3}$$

Windows with Partial Signal

Let $\boldsymbol{r}$ be the vector of received samples expressed by:

$$\boldsymbol{r} = \left[0, \dots, 0, c_1^{(i_c)}, \dots, c_{n_b}^{(i_c)}\right] + [w_1, \dots, w_n] \tag{B.4}$$

Then X_j is expressed by:

$$X_j = \sum_{p=1}^{n} c_p^{(j)} w_p + \sum_{i=1}^{n_b} c_i^{(i_c)} c_{d+i-1}^{(j)} = \sum_{p=1}^{n} c_p^{(j)} w_p + \mu_{X_j} \tag{B.5}$$

which makes $X_j \sim \mathcal{N}(\mu_{X_j}, n\sigma^2)$. The correlation between X_i and X_j is calculated by:

$$E[X_i X_j] = E\left[\left(\sum_{p=1}^{n} c_p^{(i)} w_p + \mu_{X_i}\right).\left(\sum_{q=1}^{n} c_q^{(j)} w_q + \mu_{X_j}\right)\right] \tag{B.6}$$

$$\begin{aligned} E[X_i X_j] = E\left[\sum_{p=1}^{n} c_p^{(i)} w_p . \sum_{q=1}^{n} c_q^{(j)} w_q\right] + \mu_{X_j} E\left[\sum_{p=1}^{n} c_p^{(i)} w_p\right] \\ + \mu_{X_i} E\left[\sum_{q=1}^{n} c_q^{(j)} w_q\right] + \mu_{X_i}\mu_{X_j} \end{aligned} \tag{B.7}$$

We know that:

$$E\left[\sum_{p=1}^{n} c_p^{(i)} w_p\right] = E\left[\sum_{q=1}^{n} c_q^{(j)} w_q\right] = 0 \tag{B.8}$$

We can simplify the correlation to:

$$E[X_i X_j] = E\left[\sum_{p=1}^{n} c_p^{(i)} c_p^{(j)} {w_p}^2\right] + \mu_{X_i}\mu_{X_j} = \sigma^2 \sum_{p=1}^{n} c_p^{(i)} c_p^{(j)} + \mu_{X_i}\mu_{X_j} \tag{B.9}$$

$$Cov(X_i, X_j) = E[X_i X_j] - E[X_i]E[X_i]$$
$$= \sigma^2 \sum_{p=1}^{n} c_p^{(i)} c_p^{(j)} + \mu_{X_i}\mu_{X_j} - \mu_{X_i}\mu_{X_j} \tag{B.10}$$

$$Cov(X_i, X_j) = \sigma^2 \sum_{p=1}^{n} c_p^{(i)} c_p^{(j)} \tag{B.11}$$

$$\rho_{X_i,X_j} = \frac{Cov(X_i, X_j)}{\sqrt{n\sigma^2}.\sqrt{n\sigma^2}} = \frac{\sigma^2 \sum_{p=1}^{n} c_p^{(i)} c_p^{(j)}}{n\sigma^2} = \frac{1}{n}\sum_{p=1}^{n} c_p^{(i)}.c_p^{(j)} \tag{B.12}$$

The same procedure can be repeated if:

$$\boldsymbol{r} = \left[c_d^{(i_c)}, \dots, c_n^{(i_c)}, 0, \dots, 0\right] + [w_1, \dots, w_n] \tag{B.13}$$

$$X_j = \sum_{p=1}^{n} c_p^{(j)} w_p + \sum_{i=d}^{n} c_i^{(i_c)} c_{i-d+1}^{(j)} = \sum_{p=1}^{n} c_p^{(j)} w_p + \mu_{X_j} \tag{B.14}$$

We will now prove equation (7.20). We start with the maximum correlation at the correct time $Z_{t_0} \sim \mathcal{N}(n, n\sigma^2)$ and assume that $\boldsymbol{r}$ is described by (B.4) and X_j described by (B.5):

$$Z_{t_0} = \sum_{i=1}^{n} (c_i^{(i_c)} + w_{d+i-1})\, c_i^{(i_c)} = n + \sum_{i=1}^{n} c_i^{(i_c)} w_{d+i-1} \tag{B.15}$$

The correlation between Z_{t_0} and X_j is calculated by:

$$E[Z_{t_0} X_j] = E\left[\left(n + \sum_{i=1}^{n} c_i^{(i_c)} w_{d+i-1}\right).\left(\sum_{p=1}^{n} c_p^{(j)} w_p + \mu_{X_j}\right)\right] \tag{B.16}$$

$$E\left[Z_{t_0}X_j\right] = nE\left[\sum_{p=1}^{n} c_p^{(j)} w_p\right] + n\mu_{X_j} + E\left[\sum_{i=1}^{n} c_i^{(i_c)} w_{d+i-1} \cdot \sum_{p=1}^{n} c_p^{(j)} w_p\right] + \mu_{X_j} E\left[\sum_{i=1}^{n} c_i^{(i_c)} w_{d+i-1}\right] \tag{B.17}$$

We know that:

$$E\left[\sum_{p=1}^{n} c_p^{(j)} w_p\right] = E\left[\sum_{i=1}^{n} c_i^{(i_c)} w_i\right] = 0 \tag{B.18}$$

We can simplify the correlation to:

$$E\left[Z_{t_0}X_j\right] = n\mu_{X_j} + E\left[\sum_{i=1}^{n_b} c_i^{(i_c)} c_{d+i-1}^{(j)} {w_i}^2\right] = n\mu_{X_j} + \sigma^2 \sum_{i=1}^{n_b} c_i^{(i_c)} c_{d+i-1}^{(j)} \tag{B.19}$$

$$\begin{aligned} Cov\left(Z_{t_0}, X_j\right) &= E\left[Z_{t_0}X_j\right] - E\left[Z_{t_0}\right]E\left[X_j\right] \\ &= n\mu_{X_j} + \sigma^2 \sum_{i=1}^{n_b} c_i^{(i_c)} c_{d+i-1}^{(j)} - n\mu_{X_j} \end{aligned} \tag{B.20}$$

$$Cov\left(Z_{t_0}, X_j\right) = \sigma^2 \sum_{i=1}^{n_b} c_i^{(i_c)} c_{d+i-1}^{(j)} = \sigma^2 \mu_{X_j} \tag{B.21}$$

$$\rho_{Z_{t_0}, X_j} = \frac{Cov\left(Z_{t_0}, X_j\right)}{\sqrt{n\sigma^2}.\sqrt{n\sigma^2}} = \frac{\sigma^2 \mu_{X_j}}{n\sigma^2} = \frac{1}{n}\mu_{X_j} \tag{B.22}$$

The same procedure can be repeated if $\boldsymbol{r}$ is described by (B.13) and X_j is described by (B.14).

APPENDIX C

SEQUENTIAL MAXIMIZATION ALGORITHM

Let $X_1, \dots, X_5$ be Gaussian random variables that are correlated and differently distributed. Let us define two boxes where one is the maximization box and the other is the correlation box that are illustrated in figure 9.1.

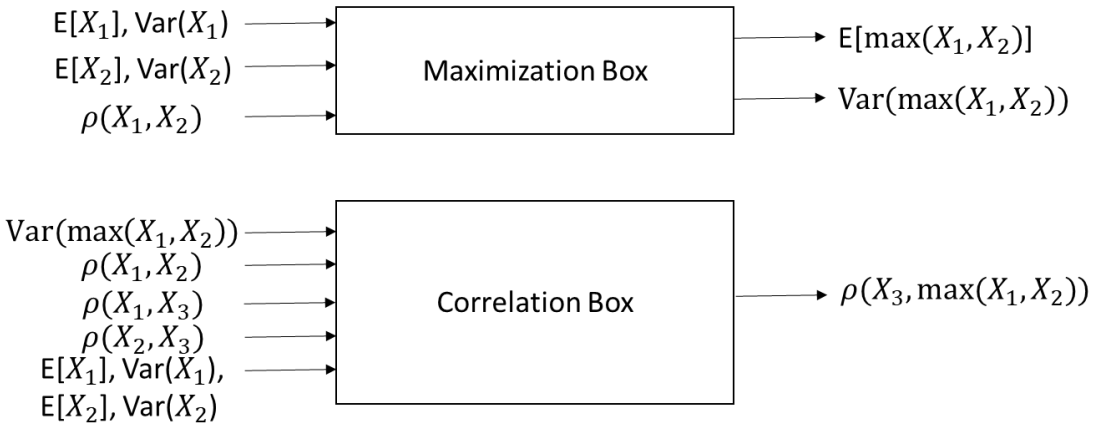

Figure 9.1: Illustration of the Maximization Box and the Correlation Box

Inside the maximization box, the following operations happen as indicated in [51]:

$$a = \sqrt{\left| Var(X_1) + Var(X_2) - 2\rho(X_1, X_2)\sqrt{Var(X_1)Var(X_2)} \right|} \tag{C.1}$$

$$\alpha = \frac{E[X_1] - E[X_2]}{a} \tag{C.2}$$

$$E[\max(X_1, X_2)] = E[X_1].\Phi(\alpha) + E[X_2].\Phi(-\alpha) + a.\phi(\alpha) \tag{C.3}$$

$$\begin{aligned} Var(\max(X_1, X_2)) \\ &= (Var(X_1).(E[X_1])^2).\Phi(\alpha) \\ &+ (Var(X_2).(E[X_2])^2).\Phi(-\alpha) \\ &+ (E[X_1] + E[X_2]).a.\phi(\alpha) - (E[\max(X_1, X_2)])^2 \end{aligned} \tag{C.4}$$

Inside the correlation box, the following operation happens:

$$\begin{aligned} &\rho(X_3, \max(X_1, X_2)) \\ &= \frac{\sqrt{Var(X_1)}.\rho(X_1, X_3).\Phi(\alpha) + \sqrt{Var(X_2)}.\rho(X_2, X_3).\Phi(-\alpha)}{\sqrt{Var(\max(X_1, X_2))}} \end{aligned} \tag{C.5}$$

where $\Phi(x)$ and $\phi(x)$ are the standard normal CDF and pdf respectively.

The algorithm will approximate the mean and variance of $\max(X_1, \dots, X_5)$ in steps 1-4 as illustrated in figures 9.2-9.5.

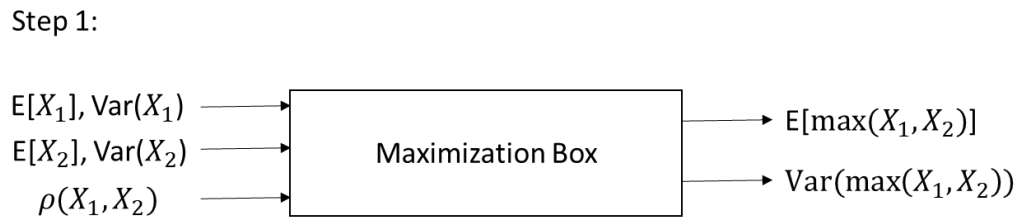

Figure 9.2: Illustration of Step 1

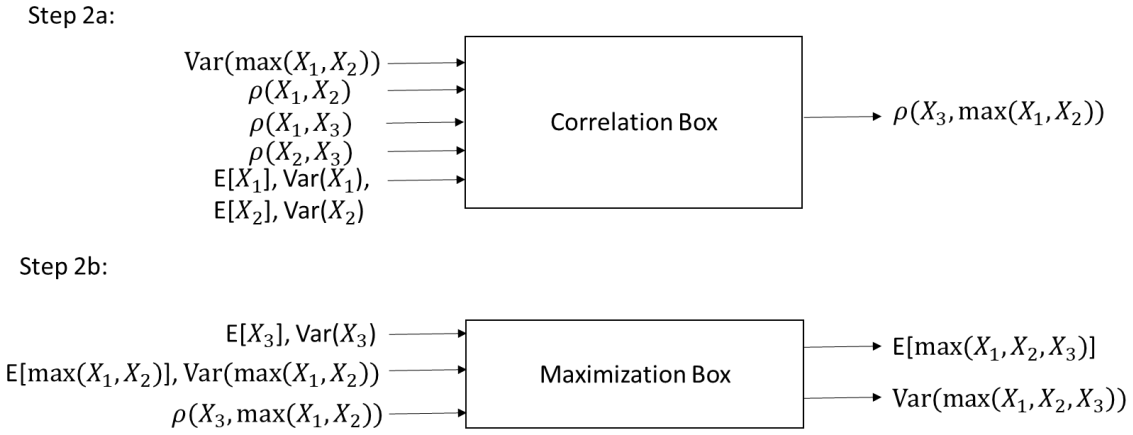

Figure 9.3: Illustration of Step 2

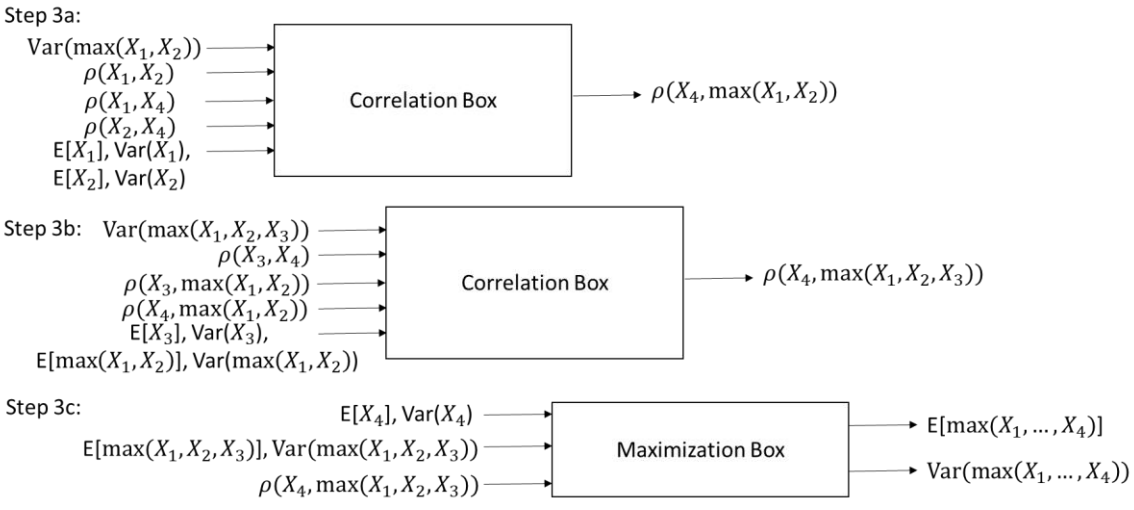

Figure 9.4: Illustration of Step 3

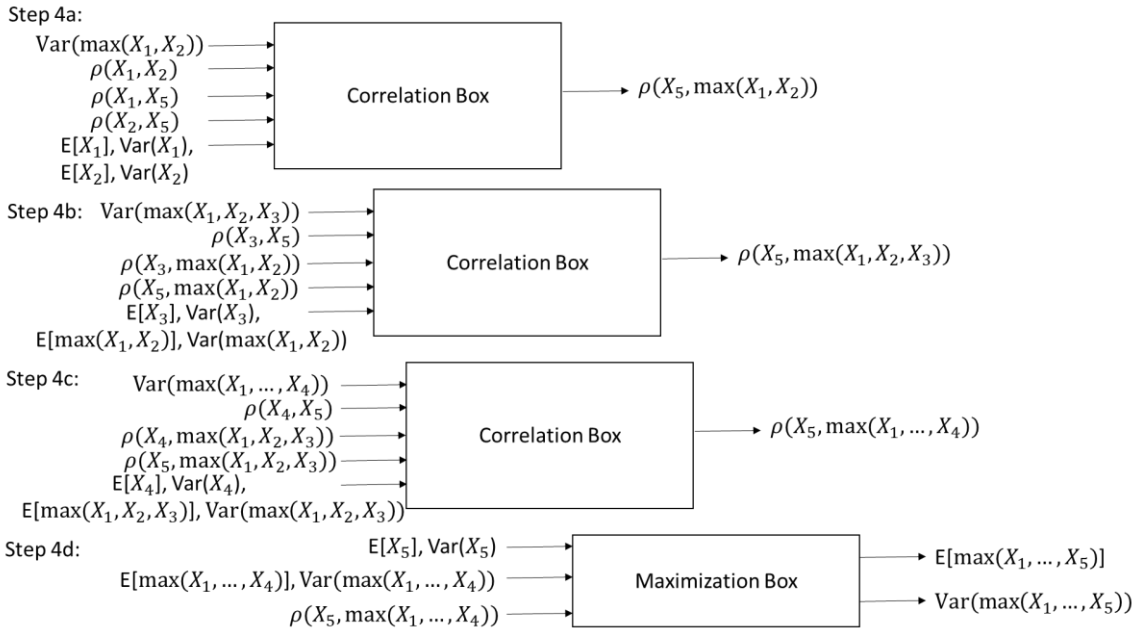

Figure 9.5: Illustration of Step 4

The algorithm can be extended to N random variables but it must be noted that the more random variables are added the more approximation errors will be compounded.

APPENDIX D

PAIRWISE MAXIMIZATION ALGORITHM

Let $X_1, \ldots, X_8$ be Gaussian random variables that are correlated and differently distributed. The same maximization and correlation boxes from Appendix C are used. Let us also introduce the pair correlation box as shown in figure 9.6 and whose contents are illustrated of figure 9.7.

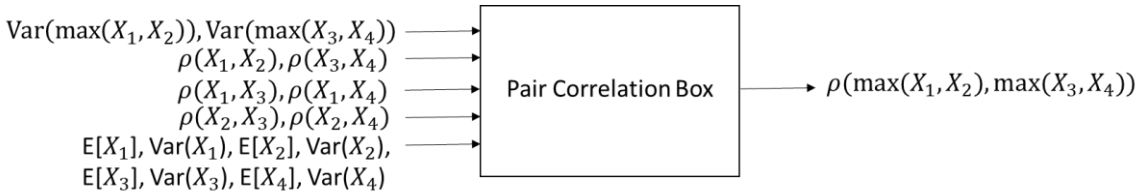

Figure 9.6: Illustration of the Pair Correlation Box

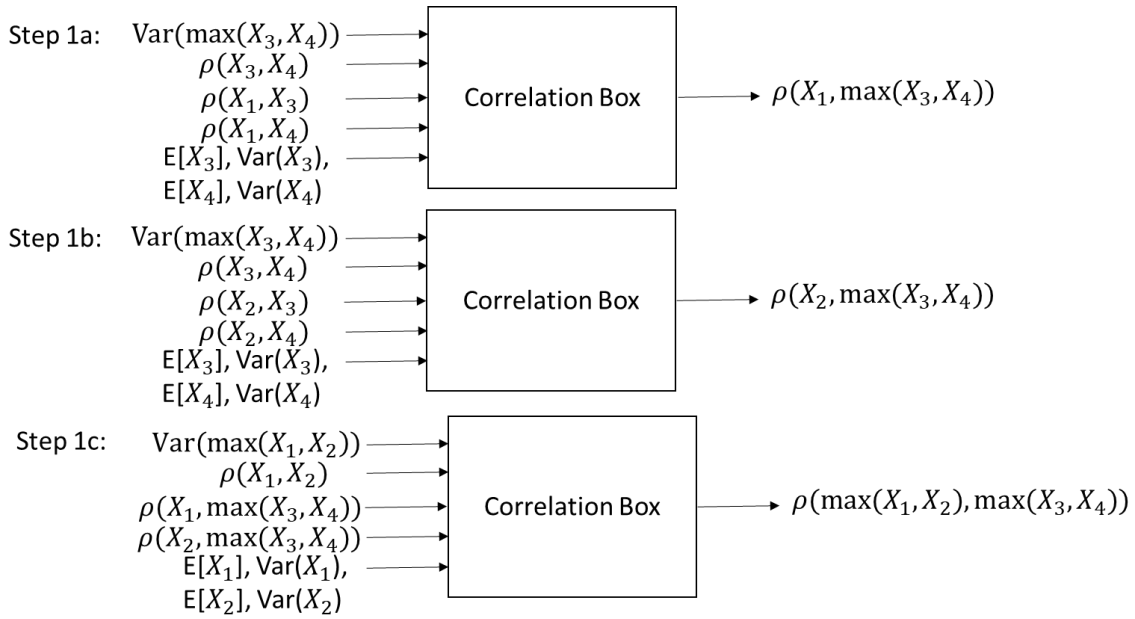

Figure 9.7: Contents of the Pair Correlation Box

The algorithm will approximate the mean and variance of $\max(X_1, \ldots, X_8)$ in steps 1-3 as illustrated in figures 9.8-9.11.

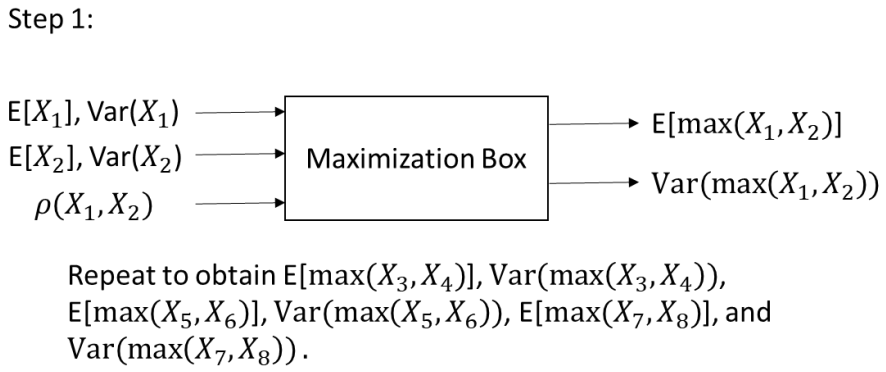

Figure 9.8: Illustration of Step 1

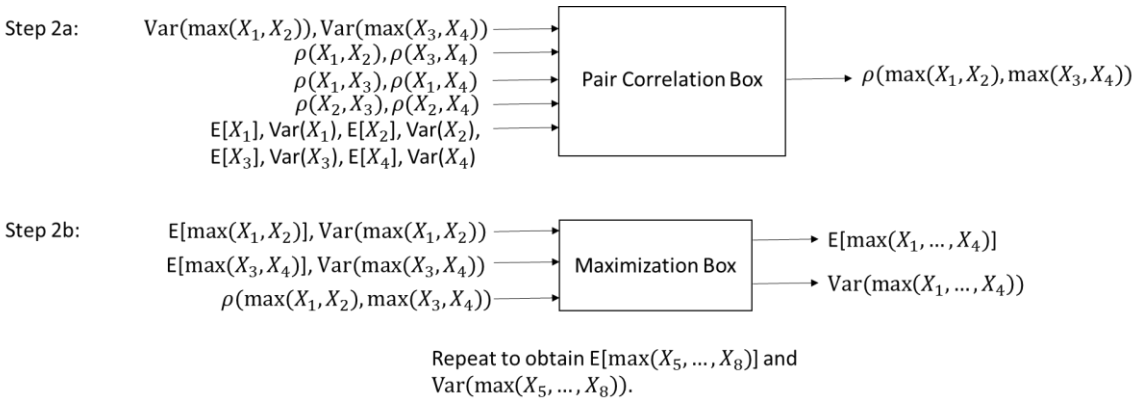

Figure 9.9: Illustration of Step 2

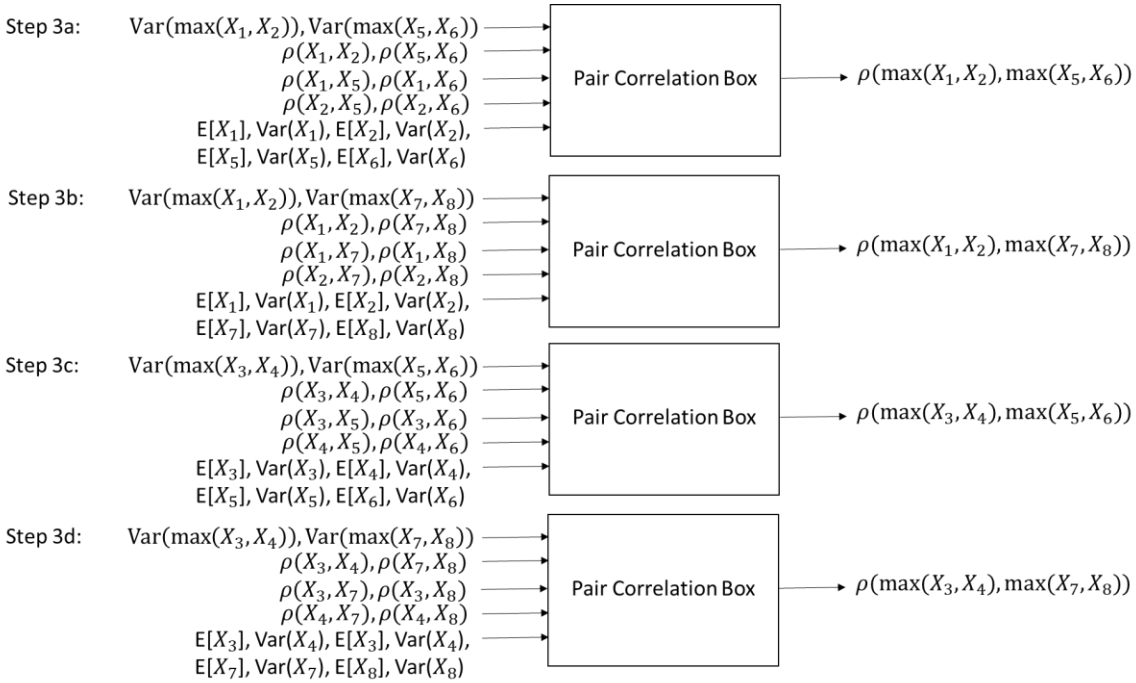

Figure 9.10: Illustration of Steps 3a-3d

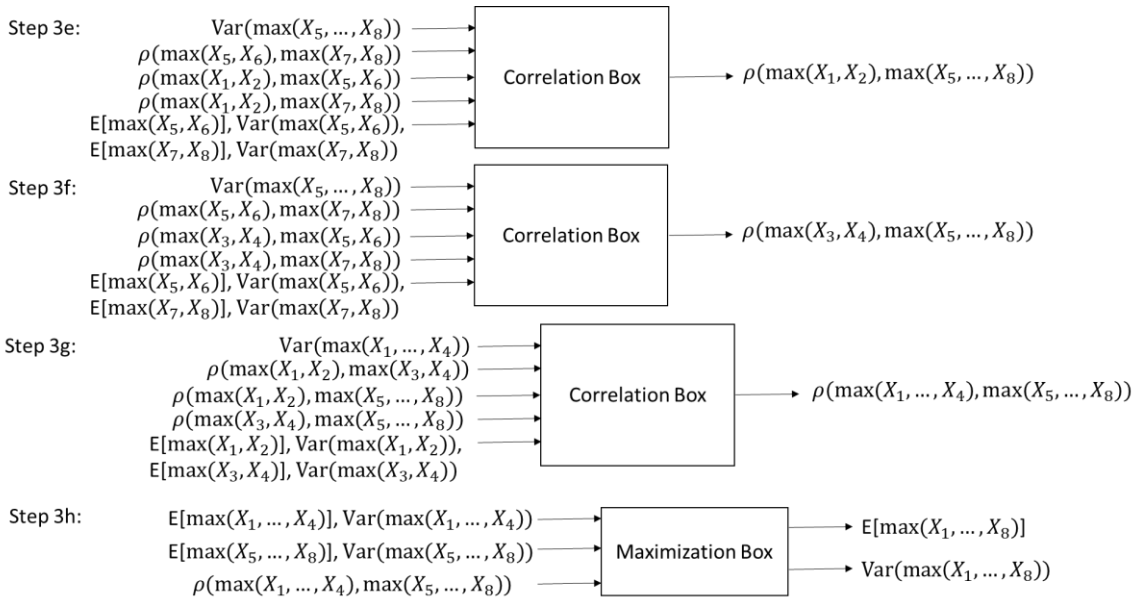

Figure 9.11: Illustration of Steps 3e-3h

The algorithm can be extended to N random variables where N must be a power of 2, but it must be noted that the more random variables are added the more approximation errors will be compounded.