

Cleveland State University

**ANALYSIS AND OPTIMIZATION OF
THE CHIP LEVEL PARAMETER IN
GENERALIZED DIRECT SEQUENCE
SPREAD SPECTRUM FOR IMPROVED
WORST-CASE PERFORMANCE**

By

Noah J. Deetz

A Thesis

Submitted to the Faculty
of the Cleveland State University
in partial fulfillment of the requirements for
the degree of Master of Science in Electrical Engineering

Cleveland, Ohio
May 5th, 2010

Abstract

Spread spectrum modulation has long been used to suppress interference. Direct sequence spread spectrum (DSSS) modulates a message signal using two signalling levels, and can be thought of as a simplified random modulator. It has been proven that using five signalling levels improves worst-case error probability in a channel corrupted by thermal noise and an unknown and arbitrary interference signal of bounded power. It has also been proven that an asymptotically optimal random modulator for this channel distributes the message signal uniformly on the surface of an N -dimensional sphere.

In five level DSSS the signalling levels are ± 1 , 0 , and a level in between known as $\pm z$. This thesis examines the effect of altering z upon the worst-case error probability. It also explores which z values result in uniform distribution of the message signal on the surface of an N -dimensional sphere. Optimal values of z were found to be independent of noise or interference level, and only dependent on N . It was also found that the system does not allow exact uniform distribution of the message signal, but a range exists for a given N over which it is very close. The optimal z values fall within this range.

Contents

1	Introduction	9
1.1	Spread Spectrum Overview	9
1.2	DSSS and Anti-Jam Capabilities	11
2	Summary of Previous Research	15
2.1	Introduction	15
2.2	Direct Sequence Spread Spectrum	15
2.3	Generalization of DSSS	17
2.4	Further Generalization of DSSS	21
2.4.1	Overview	21
2.4.2	System Model	22
2.4.3	Measure of Performance	27
2.5	Bounds on Worst-Case Error Probability	28
2.5.1	Upper Bound	29
2.5.2	Lower Bound	30
3	Examining the Effect of Altering z on Worst-Case Error Probability	

Bounds	35
3.1 Introduction	35
3.2 Upper Bound	36
3.3 Lower Bound	40
3.3.1 Examining the Effect of Thermal Noise on the Lower Bound	43
3.3.2 Valid Range for Optimizing the z Parameter	50
3.3.3 Finding the Optimal z Value	54
3.3.4 Optimal z Value for $N=3-20$	68
4 Geometric Approach to Finding the Optimal z Value	75
4.1 Introduction	75
4.2 Demonstration of How Uniform Distribution is Calculated	77
4.3 Results	86
4.3.1 $N=3$	86
4.3.2 $N=4$	87
4.3.3 $N=6$	92
4.3.4 $N=9$	92
4.3.5 Summary of Results	92
5 Results and Conclusion	99
5.1 Comparison of Results	99
5.2 Suggestion for Further Research	103
A Matlab Script for Evaluating Lower Bound	107

B Matlab Script for Geometric Analysis**113**

List of Figures

2.1	Vector Distribution of Generalized DSSS with $N = 3$	19
2.2	Normalized Vector Distribution of Generalized DSSS with $N = 3$. . .	20
2.3	Vector Distribution of Regeneralized DSSS with $N = 3$	23
2.4	Normalized Vector Distribution of Regeneralized DSSS with $N = 3$. .	24
2.5	Comparison of Upper and Lower Bound: $N = 20$, $z = .03$, $E_b/N_0 =$ 10 dB	33
3.1	Effect of Altering z on Upper Bound with $N = 20$	37
3.2	Effect of Altering z on Upper Bound with $N = 8$	38
3.3	Effect of Altering z on Upper Bound with $N = 20$ and $E_b/N_0 = 20$ dB	39
3.4	E_b/E_I Ratio Needed for 10^{-6} Error Rate for Given z with $N = 20$ and $E_b/N_0 = 20$ dB	41
3.5	Effect of Altering z on Lower Bound with $N = 20$	42
3.6	E_b/E_I Ratio Needed for 10^{-3} Error Rate for Given z with $N = 20$ and $E_b/N_0 = 200$ dB	44
3.7	Optimal z vs. Signal to Interference Ratio: No Noise and $N = 5$. . .	46
3.8	Error Probability vs. Signal to Interference Ratio: No Noise and $N = 5$	47

3.9	Optimal z vs. Signal to Interference Ratio: $E_b/N_0 = 10$ dB and $N = 5$	48
3.10	Error Probability vs. Signal to Interference Ratio: $E_b/N_0 = 10$ dB and $N = 5$	49
3.11	Optimal z vs. Signal to Interference Ratio: $E_b/N_0 = 0$ dB and $N = 5$	51
3.12	Error Probability vs. Signal to Interference Ratio: $E_b/N_0 = 0$ dB and $N = 5$	52
3.13	Upper Bound Experiment Range: No AWGN	55
3.14	Upper Bound Experiment Range: $E_b/N_0 = 10$ dB	56
3.15	Upper Bound Experiment Range: $E_b/N_0 = 0$ dB	57
3.16	Optimal Chipping Level for $N = 5$	59
3.17	Worst-Case Error Probability vs. z : $E_b/E_I = -5$ dB	60
3.18	Worst-Case Error Probability vs. z : $E_b/E_I = 0$ dB	61
3.19	Worst-Case Error Probability vs. z : $E_b/E_I = 5$ dB	62
3.20	Worst-Case Error Probability vs. z : $E_b/E_I = -5$ dB, Close Up	63
3.21	Minimal Worst-Case Error Probability: $N = 5$ and $z = .33$	64
3.22	Optimal Chipping Level for $N = 10$	65
3.23	Worst-Case Error Probability vs. z : $E_b/E_I = -5$ dB	66
3.24	Minimal Worst-Case Error Probability: $N = 10$ and $z = .24$	67
3.25	Optimal Chipping Level for $N = 15$	69
3.26	Worst-Case Error Probability vs. z : $E_b/E_I = -5$ dB	70
3.27	Minimal Worst-Case Error Probability: $N = 15$ and $z = .15$	71
3.28	Optimal z Value for Various N	72
3.29	Relationship Between Optimal z and N	73

3.30	Probability vs. E_b/E_I for Various z Values: $N = 7, E_b/N_0 = 0$	74
4.1	2-Dimensional Signalling Vectors Prior to Normalization	78
4.2	2-Dimensional Signalling Vectors Before and After Normalization	79
4.3	Distances From Edge and Corner Reference Vectors with $z = .1$	81
4.4	Intersections of Distance Plots for $N = 2$	82
4.5	Close View of Intersections of Distance Plots for $N = 2$	83
4.6	Uniform Vector Distribution When $z = .4142$	84
4.7	Distances From Edge and Corner Reference Vectors with $z = .4142$	85
4.8	Intersections of Distance Plots for $N = 3$	88
4.9	Close View of Intersections of Distance Plots for $N = 3$	89
4.10	Intersections of Distance Plots for $N = 4$	90
4.11	Close View of Intersections of Distance Plots for $N = 4$	91
4.12	Intersections of Distance Plots for $N = 6$	93
4.13	Close View of Intersections of Distance Plots for $N=6$	94
4.14	Intersections of Distance Plots for $N = 9$	95
4.15	Close View of Intersections of Distance Plots for $N = 9$	96
4.16	First and Last Intersection Points of Distance Plots for Various N	97
4.17	Plot of First and Last Intersection Plots vs. N	98
5.1	Comparison of Lower Bound Optimal z with Range of Intersection Points of Distance Plots	101
5.2	Extended Comparison of Bound Based and Geometric Optimal z Results	102

Chapter 1

Introduction

1.1 Spread Spectrum Overview

Spread spectrum is a commonly used robust signal structuring technique. The historical origins of this technique are a bit murky, but the invention of the concept of spread spectrum has frequently been attributed to actress Hedy Lamarr and composer George Antheil [1]. Their device used a piano roll to alternate transmission through 88 different frequencies in order to control radio guided torpedoes without detection. However, secure communications techniques including spread spectrum were heavily researched and tested by both Axis and Ally powers during World War II. Because of the sensitive nature of this military research, its exact origin is unclear [1]. However it came about, spread spectrum theory and technology eventually became available to the public around the late 1970's. In addition to military applications, spread spectrum use can now be commonly found in commercial applications such as bluetooth, wi-fi, and cordless telephones [2].

The basic idea behind spread spectrum communications is to increase the bandwidth of the transmitted signal by a spreading factor which is normally a few orders of magnitude greater than the data rate. This spreading factor is fixed and is not a function of the message signal. This differentiates spread spectrum from other wideband communication systems such as wideband FM [3]. While it may seem counter-intuitive to intentionally increase system bandwidth, doing so has a few important benefits. The benefit that is of most interest to this thesis is improved interference rejection. Spread spectrum systems are adept at overcoming narrowband interference, such as that from hostile jamming. Spread spectrum also has a low probability of detection by an unauthorized user. This is because a unique sequence is required to demodulate the signal which is only known to transmitter and receiver. In fact, without knowledge of this sequence, spread spectrum is difficult to even detect. Through the use of orthogonal spreading codes, it is possible to allow many users to share the same bandwidth with limited interference. This technique is known as code division multiple-access (CDMA). The use of CDMA can compensate for the increase in bandwidth in some systems.

A few different techniques can be used to spread the signal bandwidth. The first is direct sequence spread spectrum (DSSS). DSSS systems achieve spreading by multiplying an already data modulated carrier with a pseudorandom sequence of ± 1 's. This sequence, referred to as a chipping sequence, is known by both the transmitter and receiver. The signal is demodulated twice, once to remove the carrier, and again to remove the spreading sequence. DSSS is the spread spectrum technique that this thesis will focus on and will be discussed later in greater detail. Frequency hopping

(FH) is another commonly used spread spectrum technique. In FH systems the frequency of a message data modulated carrier is changed according to a pseudorandom sequence. This sequence and the multiple transmission frequencies must be known to both the transmitter and receiver. Some systems use a hybrid combination of FH and DSSS [2].

As previously stated, the spreading of a signal is implemented by using a chipping sequence constructed from a pseudorandom or pseudonoise (PN) sequence. This is always done prior to data transmission. The PN sequence has probability distribution resembling that of a random sequence, but is called pseudorandom because the pattern is in fact known to both transmitter and receiver [3]. In the case of DSSS, the PN sequence chops data bits into smaller pieces, sometimes referred to as chips. This turns the transmitted signal into something almost noise-like, allowing it to blend into the background. The receiver correlates the incoming signal with the known PN sequence in order to de-spread the signal and pull it out from the noise. This also allows spread spectrum systems to combat narrowband interference. The next section explain this capability in greater detail.

1.2 DSSS and Anti-Jam Capabilities

This thesis is concerned with the interference rejection properties of DSSS. When the message signal, which is relatively narrowband, is multiplied by the chipping signal, which has a wide bandwidth, the result is a wideband signal. The multiplication of the two signals in the time domain will cause the convolution of the signals in the

frequency domain, resulting in a wideband signal. At the receiver, the first thing done to the received signal is to multiply it again by the chipping sequence. The signal will then be filtered and passed along as the recovered data signal [3].

Now consider the effect of narrowband interference, possibly from hostile jamming, on the transmitted signal. A jammer will possibly attempt to place a high-power, narrowband signal somewhere inside the transmitted bandwidth of the message signal. When this signal is picked up by the receiver, it will be multiplied by the chipping signal. This will result in the bandwidth of the interfering signal being spread out in frequency, while the message signal is simultaneously being de-spread. With the energy of the interference spread out and filtered, its degrading effect on the message signal will be lessened. The basic premise behind DSSS interference rejection is that one multiplication by the chipping sequence will spread the signal in the frequency domain, and two multiplications will despread the signal in the frequency domain [3].

It would be possible to examine a DSSS system in the presence of statistically defined interference, yielding data for one specific type of channel. In [4] however, a more general approach is taken. Here, the interfering source is not statistically defined. The only assumption is that the power of the interfering signal is bound. Using these conditions and a worst-case error criterion, it is possible to find some minimum guaranteed performance level for the system.

In [4] it was found that the optimal performance under these conditions occurs when the message signal is uniformly distributed over the surface of an N -dimensional sphere, with N being the number of chips per symbol. This would be too complex, however, to physically realize. In [5], a 3-level chipping sequence is used to approxi-

mate uniform distribution. This shows some improvement over the 2-level sequence. Continuing this trend, in [6] a 5-level chipping sequence is investigated, showing some improvement over the 3-level sequence. In the 5-level sequence, the 5 levels used were ± 1 , 0, and $\pm z$. z was chosen to be .4143 as this value yields uniform distribution of the message signal for the 2-dimensional case. In these tests N was set to 20, and no thermal noise was added. At an error probability of 10^{-6} , 3-level DSSS showed an improvement of around .7 dB over ordinary DSSS. 5-level DSSS added another .3 dB improvement. The theoretical maximum improvement over ordinary DSSS was found to be around 1.4 dB (double check and reference).

The goal of this thesis is to investigate the value of the chipping level, z , in order to find the optimal value that minimizes the worst-case effects of an interfering signal bound only by power on 5-level DSSS, and to investigate the overall effect of changing z for different values of N . This will be accomplished by analyzing both the worst-case error probability of the system, and the geometry of the system.

Chapter 2 will summarize previous research that is relevant to the topic of this thesis. The most important are the bounds on worst-case error performance derived for the 5-level DSSS. Chapter 3 will explore the effects of changing z by evaluating the bounds on worst-case error performance. Chapter 4 will also explore the effects of changing z , but by examining the geometry of the system. Chapter 5 will summarize and compare the results from chapter 3 and 4, as well as introduce some ideas for future research.

Chapter 2

Summary of Previous Research

2.1 Introduction

The previous section gave a brief background on SS systems, and discussed the interference rejection properties of DSSS. It also introduced the purpose of this thesis. This section will go further into the mathematics behind DSSS. It will also introduce generalizations of DSSS that have been shown in previous research to have beneficial effect towards interference rejection. Finally, we will review ways that have been developed to measure the worst-case error probability in such systems.

2.2 Direct Sequence Spread Spectrum

DSSS is best demonstrated by examining uncoded direct sequence binary phase shift keying (DS-BPSK). An equation describing a BPSK with power S and radian fre-

quency ω_0 is as follows

$$s(t) = d_n \sqrt{2S} \cos(\omega_0 t) \quad (2.1)$$

where the phase modulating data

$$d_n = \begin{cases} 1, & \text{with probability of } .5 \\ -1, & \text{with probability of } .5 \end{cases} \quad (2.2)$$

and

$$nT_b \leq t < (n+1)T_b, \text{ where } n=\text{integer}. \quad (2.3)$$

This gives us a waveform whose phase shifts by 180 degrees with a period of T_b based on a sequence of data bits (d_n).

This waveform can be transformed into DS-BPSK by adding a spreading signal.

In this case, the spreading signal will be a PN sequence, c_k , where

$$c_k = \begin{cases} 1, & \text{with probability of } .5 \\ -1, & \text{with probability of } .5 \end{cases} \quad (2.4)$$

c_k will be N times faster than the data rate, with N generally being several orders of magnitude larger. N is also referred to as the chipping rate. The duration of each chip pulse is then

$$T_c = \frac{T_b}{N} \quad (2.5)$$

The addition of the spreading sequence gives us a DS-BPSK waveform in the following form

$$x(t) = d_n c_{nN+k} \sqrt{2S} \cos \omega_0(t) \quad (2.6)$$

with

$$nT_b + kT_c \leq t < (n+1)T_b + (k+1)T_c \quad (2.7)$$

$$k = 0, 1, 2, \dots, N-1 \quad (2.8)$$

$$n = \text{integer} \quad (2.9)$$

In the time domain, the resulting signal will appear to be a BPSK signal with a bit rate of N . In the frequency domain, the spectrum of the initial BPSK will have been spread out.

The signal will reach the receiver with some added thermal noise, and possibly other interference, such as that from a hostile jammer. The signal is then demodulated. A replica of the PN sequence is then applied to the demodulated signal. This despreads the signal and removes some narrowband interference.

2.3 Generalization of DSSS

It has been shown in [4] that for random modulators, as blocklength is increased the worst-case error performance asymptotically approaches optimality if the message signal is uniformly distributed onto the surface of an N -dimensional sphere. Here optimality means that the signal to interference ratio required to guarantee a certain

worst-case performance is minimized. A well designed random modulator should distribute the message signal uniformly onto the surface of an N -dimensional sphere. In this case, N is the number of chips per symbol. DSSS can be thought of as a special case of random modulation [7].

DSSS can also be visualized as mapping a message signal onto the surface of an N -dimensional sphere. Consider the overly simplified case where $N=3$. When the spreading sequence is applied, the message will be mapped onto the vertices of a cube. This is shown in Figure (insert figure). DSSS can be altered in a simplistic manner to better approximate uniform distribution of the message signal, and therefore improve worst-case error performance. This altered DSSS scheme will be referred to as generalized DSSS.

The purpose of generalized DSSS is to better approximate uniform distribution of the transmitted message vector. Taking the DSSS 3-D case a step further, imagine adding vectors to the center of the faces and the midpoint of the edges of the above figure. In order to have equal energy, these vectors would need to be projected onto the face of a sphere. The resulting vectors can be expressed as follows. Vertices: $\frac{1}{\sqrt{3a}}(\pm a, \pm a, \pm a)$. Edges: $\frac{1}{\sqrt{2a}}(0, \pm a, \pm a)$, $\frac{1}{\sqrt{2a}}(\pm a, 0, \pm a)$ and $\frac{1}{\sqrt{2a}}(\pm a, \pm a, 0)$. Faces: $\frac{1}{\sqrt{a}}(0, 0, \pm a)$, $\frac{1}{\sqrt{a}}(\pm a, 0, 0)$ and $\frac{1}{\sqrt{a}}(0, \pm a, 0)$. These vectors are shown in Figure 2.2. It is easy to see that this model better approximates uniform distribution than ordinary DSSS. For this simplified case, 26 different vectors can be transmitted. In general, $3^N - 1$ unique vectors will be included in the signal set. The -1 is due to the fact that the all zero case cannot be used.

When analyzing the performance of generalized DSSS, bounds must be used. This

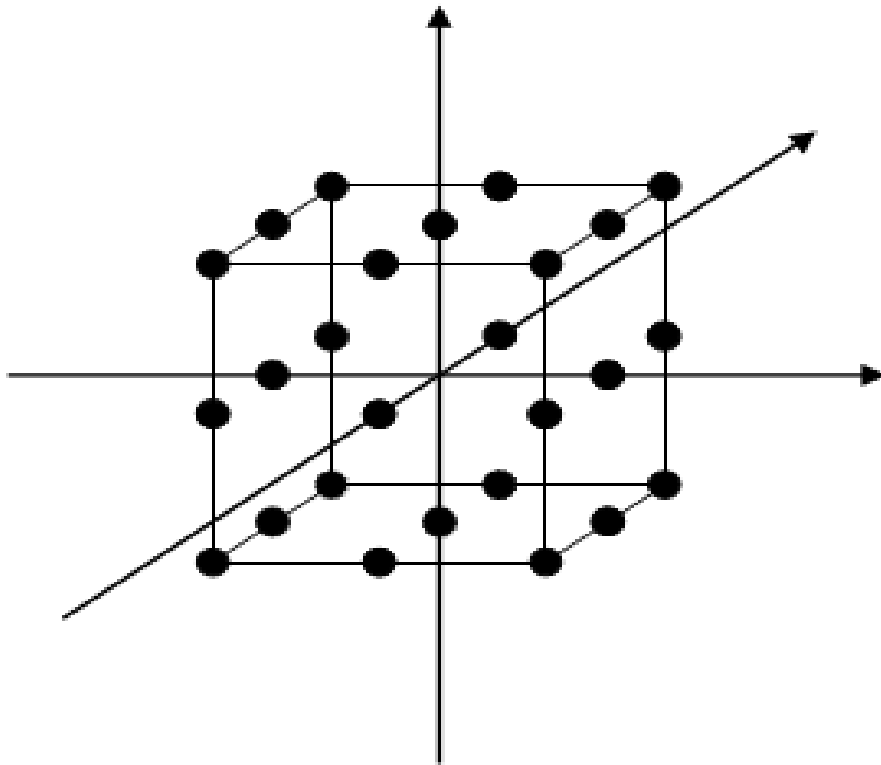


Figure 2.1: Vector Distribution of Generalized DSSS with $N = 3$

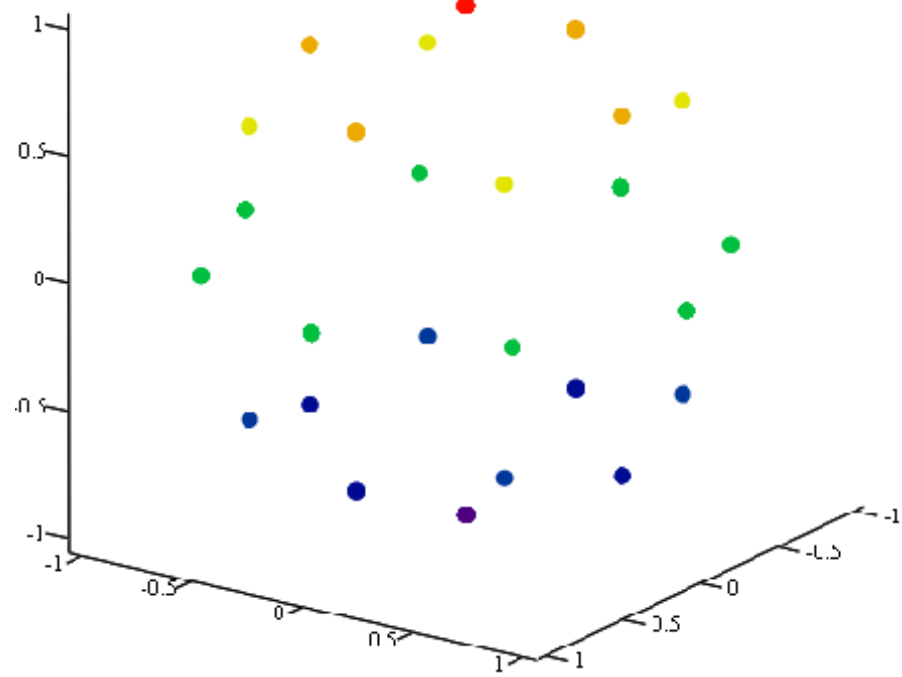


Figure 2.2: Normalized Vector Distribution of Generalized DSSS with $N = 3$

is due to the fact that a direct solution is prohibitively complex. The generation of these bounds will be discussed in the following chapter. The resulting improvement in worst-case error performance has been examined by [5]. It was found that generalized DSSS yielded a .7 dB improvement over ordinary DSSS, in the upper and lower bounds on worst-case error performance. This improvement occurred with $N=20$, and no AWGN added to the channel. Generalized DSSS can be realized by simply adding a zero to the alphabet of the chipping sequence. The available elements of the chipping sequence would then be ± 1 and 0. This is a fairly simple change to make, and improves the systems worst-case error performance.

2.4 Further Generalization of DSSS

2.4.1 Overview

Generalized DSSS demonstrated that worst-case error performance can be improved by adding a third level to the chipping sequence, thereby creating a closer approximation to the ideal uniform distribution of the message signal over an N -dimensional sphere. The logical next step would be to add more levels to the chipping sequence to see if this trend of improvement continues. This is exactly what was done in [6].

In [6] DSSS with a 5-level chipping sequence, referred to as regeneralized DSSS, is examined. The 5 levels used are $0, \pm 1$, and a level in between 0 and ± 1 , denoted as $\pm z$. When visualizing these points in the three dimensional case, we would be adding midpoints to the generalized DSSS vectors. In order for the vectors to lie on

the surface of an N -dimensional cube, at least one component must be a ± 1 . These leaves a total of 98 vectors in the $N=3$ case. This is shown in Figure 2.3. The small grey points represent the vectors that have been added from the three-level case. In general, $5^N - 3^N - 1$ unique vectors will be included in the signal set. Projecting these points onto the surface of a sphere, to ensure equal energy, yields the following vectors: Vertices: $\frac{1}{\sqrt{3a}}(\pm a, \pm a, \pm a)$. Edges: $\frac{1}{\sqrt{2a}}(0, \pm a, \pm a)$, $\frac{1}{\sqrt{2a}}(\pm a, 0, \pm a)$ and $\frac{1}{\sqrt{2a}}(\pm a, \pm a, 0)$. Faces: $\frac{1}{\sqrt{a}}(0, 0, \pm a)$, $\frac{1}{\sqrt{a}}(\pm a, 0, 0)$ and $\frac{1}{\sqrt{a}}(0, \pm a, 0)$. Added points: $\frac{1}{\sqrt{1+2z^2}}(\pm 1, \pm z, \pm z)$, $\frac{1}{\sqrt{1+2z^2}}(\pm z, \pm 1, \pm z)$, $\frac{1}{\sqrt{1+2z^2}}(\pm z, \pm z, \pm 1)$, $\frac{1}{\sqrt{1+z^2}}(\pm 1, \pm z, 0)$, $\frac{1}{\sqrt{1+z^2}}(\pm z, \pm 1, 0)$, $\frac{1}{\sqrt{1+z^2}}(\pm 1, 0, \pm z)$, $\frac{1}{\sqrt{1+z^2}}(\pm z, 0, \pm 1)$, $\frac{1}{\sqrt{1+z^2}}(0, \pm 1, \pm z)$, $\frac{1}{\sqrt{1+z^2}}(0, \pm z, \pm 1)$, $\frac{1}{\sqrt{2+z^2}}(\pm 1, \pm z, \pm 1)$, $\frac{1}{\sqrt{2+z^2}}(\pm z, \pm 1, \pm 1)$, and $\frac{1}{\sqrt{2+z^2}}(\pm 1, \pm 1, \pm z)$. This is shown in Figure 2.4. Since regenerated DSSS will be the focus of this thesis, a system model and measure of performance will be presented in the following sections.

2.4.2 System Model

A mathematical model has been developed for the analysis of regenerated DSSS in [6]. The purpose of developing this model was to analyze the worst-case error performance of the system. In order to do this, no specific interference signal is modelled. Instead, a more generalized approach is used where the signal is corrupted by AWGN, and an arbitrary interfering signal with the only stipulation being a bound on average power. This allows the model to take under consideration many different forms of interference.

Regenerated DSSS can be expressed as follows

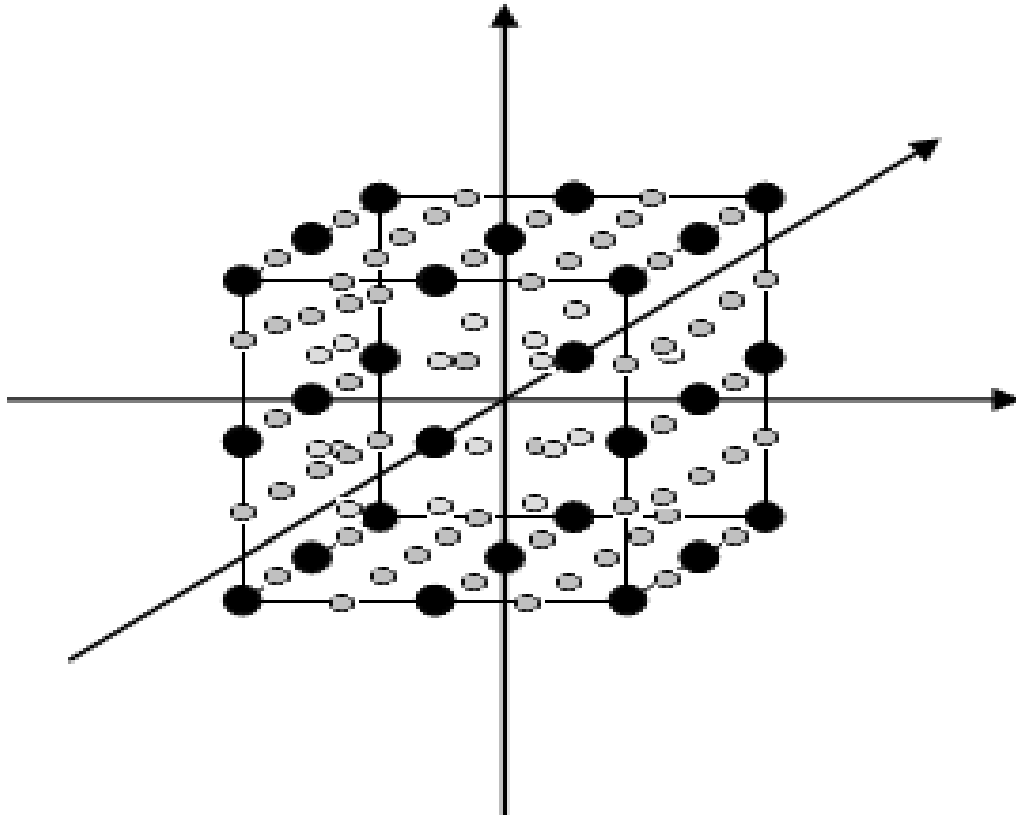


Figure 2.3: Vector Distribution of Regeneralized DSSS with $N = 3$

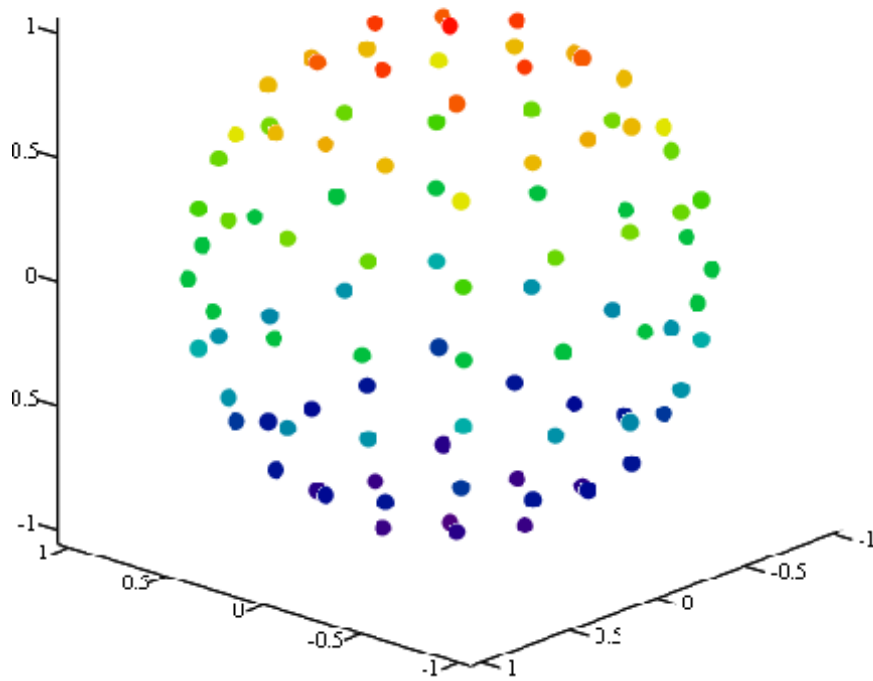


Figure 2.4: Normalized Vector Distribution of Regeneralized DSSS with $N = 3$

$$x(t) = \sqrt{\frac{2E_b}{T}} bca(t) \cos(\omega_0 t) \quad (2.10)$$

where

$$0 \leq t \leq T \quad (2.11)$$

In equation 2.10, b is a binary message sequence taking a value of ± 1 with equal probability for a duration of T seconds. The energy normalization constant c has been found to be

$$c = \sqrt{\frac{N}{\sum_{j=0}^{N-1} a_j^2}} \quad (2.12)$$

where N is the number of chips per symbol. $a(t)$ is the chip waveform with the following equation

$$a(t) = \sum_{j=0}^{N-1} a_j \psi(t - jT_c) \quad (2.13)$$

where a_j is the chip (in this case, a random process taking on a value of ± 1 , $\pm z$, or 0), ψ is a unit-power chip pulse shape, and T_c is the duration of the chip pulse, in seconds.

As the signal travels through the channel, it will be corrupted by AWGN, $n(t)$, and an unknown interfering signal, $s(t)$. $n(t)$ is modelled as a Gaussian process with two-sided power spectral density $\frac{N_0}{2}$. $s(t)$ will be used to model an interfering source with unknown statistics. This could be from a variety of sources such as multiple

access interference or hostile jamming. The only requirement on $s(t)$ is that it is independent of $x(t)$ and $n(t)$, and its average power is bound, such that

$$\int_0^T s^2(t)dt \leq E_I \quad (2.14)$$

The received signal is then

$$r(t) = x(t) + s(t) + n(t). \quad (2.15)$$

In order to simplify the model, $r(t)$, $x(t)$, $s(t)$ and $n(t)$ can be expressed in an N -dimensional signal space. This is done by projecting the model onto N orthonormal basis vectors. This changes our equations as follows;

$$\mathbf{x} = \sqrt{\frac{E_b}{N}} b c \mathbf{a} \quad (2.16)$$

where

$$\mathbf{a} \triangleq (a_0, a_1, \dots, a_{N-1}) \quad (2.17)$$

AWGN can now be expressed as

$$\mathbf{n} \triangleq (n_0, n_1, \dots, n_{N-1}) \quad (2.18)$$

and the interfering signal as

$$\mathbf{s} \triangleq (s_0, s_1, \dots, s_{N-1}) \quad (2.19)$$

The received signal is now

$$\mathbf{r} = \mathbf{x} + \mathbf{s} + \mathbf{n} \quad (2.20)$$

and the optimal receiver (found to be the standard correlation receiver in [4]) estimates the transmitted bit as follows

$$\hat{b} = \text{sgn}[\mathbf{r} \cdot \mathbf{a}] \quad (2.21)$$

2.4.3 Measure of Performance

The measure of performance used for the system will be worst-case error probability. This has been derived in [6] as follows: the probability of an error in regeneralized DSSS is the probability that the estimate made by the receiver does not match what was transmitted. Mathematically, this can be expressed as

$$\varepsilon \triangleq \Pr\{\hat{b} \neq b\} \quad (2.22)$$

Specifically for this system the equation can be written as

$$\varepsilon = \Pr\{\mathbf{r} \cdot \mathbf{a} > 0 | b = -1\} \quad (2.23)$$

This can be rewritten as

$$\varepsilon = \Pr \left\{ \left(-\sqrt{\frac{E_b}{N}} ca + \mathbf{s} + \mathbf{n} \right) \cdot \mathbf{a} > 0 \right\} \quad (2.24)$$

$$= \Pr \left\{ \mathbf{s} \cdot \mathbf{a} + \mathbf{n} \cdot \mathbf{a} > \sqrt{\frac{E_b}{N}} ca \cdot \mathbf{a} \right\} \quad (2.25)$$

With some substitutions and manipulations, this becomes

$$= \Pr \left\{ \frac{1}{\sqrt{k'}} s \cdot \mathbf{a} + \eta > \sqrt{E_b} \right\} \quad (2.26)$$

where k' is the hamming weight of \mathbf{a} , or

$$k' \triangleq \sum_{j=0}^{N-1} a_j^2 \quad (2.27)$$

and η is a zero mean random variable and has variance of $\frac{N_0}{2}$.

This derivation is for the error probability. However we wish to analyze the worst-case error probability. To do this we need to consider all interference, s , that satisfies the bounds on average energy. This yields a definition of worst-case error probability as follows;

$$\varepsilon_{wc} \triangleq \max_s \Pr \left\{ \frac{1}{\sqrt{k'}} s \cdot \mathbf{a} + \eta > \sqrt{E_b} \right\} \quad (2.28)$$

This equation is too complex to be solved directly, therefore bounds are used to evaluate it. The derivation of these bounds is presented in the next section.

2.5 Bounds on Worst-Case Error Probability

In this section, the work of [6] in deriving upper and lower bounds on the worst-case error probability of regeneralized DSSS will be summarized.

2.5.1 Upper Bound

To derive the upper bound on worst-case error probability it was decided to use the Chernoff bound. When applied to equation 2.28 the following is obtained:

$$\varepsilon_{wc} \leq \max_s E_{a,\eta} \left\{ e^{\lambda \left(\frac{1}{\sqrt{k'}} s \cdot a + \eta \right)} \right\} e^{-\lambda \sqrt{E_b}} \quad (2.29)$$

$$= \max_s E_a \left\{ e^{\frac{\lambda}{\sqrt{k'}} s \cdot a} \right\} E_\eta \left\{ e^{\lambda \eta} \right\} e^{-\lambda \sqrt{E_b}} \quad (2.30)$$

which can be simplified to

$$\varepsilon_{wc} \leq \max_s E_a \left\{ e^{\frac{\lambda}{\sqrt{k'}} s \cdot a} \right\} e^{\lambda^2 \frac{N_0}{4} - \lambda \sqrt{E_b}} \quad (2.31)$$

when the expectation of η is computed.

The next step is evaluating $E_a \left\{ e^{\frac{\lambda}{\sqrt{k'}} s \cdot a} \right\}$. The solution depends on the distribution of $\{-1, -z, 0, z, 1\}$ in the chipping sequence. The expectation was found to be

$$E_a \left\{ e^{\frac{\lambda}{\sqrt{k'}} s \cdot a} \right\} = E_k \left\{ \sum_{l=0}^{N-k} \binom{N-k}{l} 2^l E_a \left\{ \prod_i \exp \left(\frac{\lambda s_i a_i}{\sqrt{k+lz^2}} \right) \right\} \right\} \quad (2.32)$$

When this is maximized over all s that satisfy the constraint on average power, the result is

$$\begin{aligned} \max_s E_a \left\{ e^{\frac{\lambda}{\sqrt{k'}} s \cdot a} \right\} &= \frac{1}{5^{N-3N}} \sum_{k=1}^N \binom{N}{k} 2^k \sum_{l=0}^{N-k} \binom{N-k}{l} 2^l \cosh^k \left(\lambda \sqrt{\frac{E_I}{N(k+lz^2)}} \right) \\ &\quad \times \cosh^l \left(\lambda z \sqrt{\frac{E_I}{N(k+lz^2)}} \right) \end{aligned} \quad (2.34)$$

When this is combined with equation 2.29 the result is the upper bound on ε_{wc} ;

$$\varepsilon_{wc} \leq \min_{\lambda \geq 0} \left\{ \frac{1}{5^N - 3^N} \sum_{k=1}^N \binom{N}{k} 2^k \sum_{l=0}^{N-k} \binom{N-k}{l} 2^l \cosh^k \left(\lambda \sqrt{\frac{E_I}{N(k+lz^2)}} \right) \right. \\ \left. \times \cosh^l \left(\lambda z \sqrt{\frac{E_I}{N(k+lz^2)}} \right) e^{\lambda^2 \frac{N_0}{4} - \lambda \sqrt{E_b}} \right\} \quad (2.35)$$

To simplify the above equation, the following are defined

$$\sigma_I \triangleq \frac{E_b}{E_I} \quad (2.36)$$

$$\sigma_n \triangleq \frac{E_b}{N_0} \quad (2.37)$$

$$\lambda' \triangleq \lambda \sqrt{E_b} \quad (2.38)$$

σ_I and σ_n are the signal to interference ratio and the signal to noise ratio, respectively.

Substituting these into equation 2.35 yields

$$\varepsilon_{wc} \leq \min_{\lambda' \geq 0} \left\{ \frac{1}{5^N - 3^N} \sum_{k=1}^N \binom{N}{k} 2^k \sum_{l=0}^{N-k} \binom{N-k}{l} 2^l \cosh^k \left(\frac{\lambda'}{\sqrt{N(k+lz^2)\sigma_I}} \right) \right. \\ \left. \times \cosh^l \left(\frac{\lambda' z}{\sqrt{N(k+lz^2)\sigma_I}} \right) e^{\frac{\lambda'^2}{4\sigma_n} - \lambda'} \right\} \quad (2.39)$$

This equation can be used to evaluate the upper bound of ε_{wc} .

2.5.2 Lower Bound

The lower bound to ε_{wc} has been found by considering the equation

$$\varepsilon = \Pr \left\{ \frac{1}{\sqrt{k'}} s \cdot a + \eta > \sqrt{E_b} \right\} \quad (2.40)$$

while considering our system model requirement on the average power of the interfering signal for which the following is defined

$$s_i = \begin{cases} \sqrt{\frac{E_I}{r}}, 0 \leq i \leq r-1 \\ 0, r \leq i \leq N-1 \end{cases} \quad (2.41)$$

The tightest lower bound can then be found by evaluating equation 2.40 with respect to this definition, and maximized over r . Doing so yields

$$\varepsilon = \Pr \left\{ \sqrt{\frac{1}{k+lz^2}} s \cdot a + \eta > \sqrt{E_b} \right\} \quad (2.42)$$

Which has been found in [6] to be

$$\varepsilon = E_{k,l,m} \left\{ Q \left(\sqrt{2\sigma_n} - \sqrt{\frac{2\sigma_n}{(k+lz^2)\sigma_I}} m \right) \right\} \quad (2.43)$$

where

$$m = \sum_{i=0}^{r-1} a_i \quad (2.44)$$

The expectation over the number of ± 1 's, and $\pm z$'s is then found and combined with equation 2.43. This yields the following as the lower bound on ε_{wc} for regenerated DSSS

$$\varepsilon_{wc} = \left\{ \begin{aligned} & \max_{1 \leq r \leq N} \sum_{k=1}^N \sum_{l=0}^{N-k} \sum_{a=0}^{\min(r,k)} \sum_{b=\max(0,k+r-N-a)}^{\min(r,k)-a} \\ & \times \sum_{c=0}^{\min(r-a-b,l)} \sum_{d=\max(0,l+k+r-N-a-b-c)}^{\min(r-a-b,l)-c} \binom{r}{a+b} \binom{N-r}{k-a-b} \binom{a+b}{a} \\ & \quad \times \binom{r-a-b}{c+d} \binom{N-r-k-a-b}{l-c-d} \binom{c+d}{c} \\ & \quad \times \frac{2^{k+l-a-b-c-d}}{5^N - 3^N} \times Q \left(\sqrt{\sigma_n} \left(1 - \frac{a-b+q(c-d)}{\sqrt{(k+lz^2)\sigma_I r}} \right) \right) \end{aligned} \right\} \quad (2.45)$$

where a, b, c and d are the number of 1's, -1's, z 's and $-z$'s in r , respectively.

Figure 2.5 shows plots of the upper and lower bound together. This was done by evaluating the bound equations using Matlab, with $N=20$, $z=.03$, and $\sigma_n=10$ dB.

The next section will use Matlab to examine the effects of changing z on the bounds.

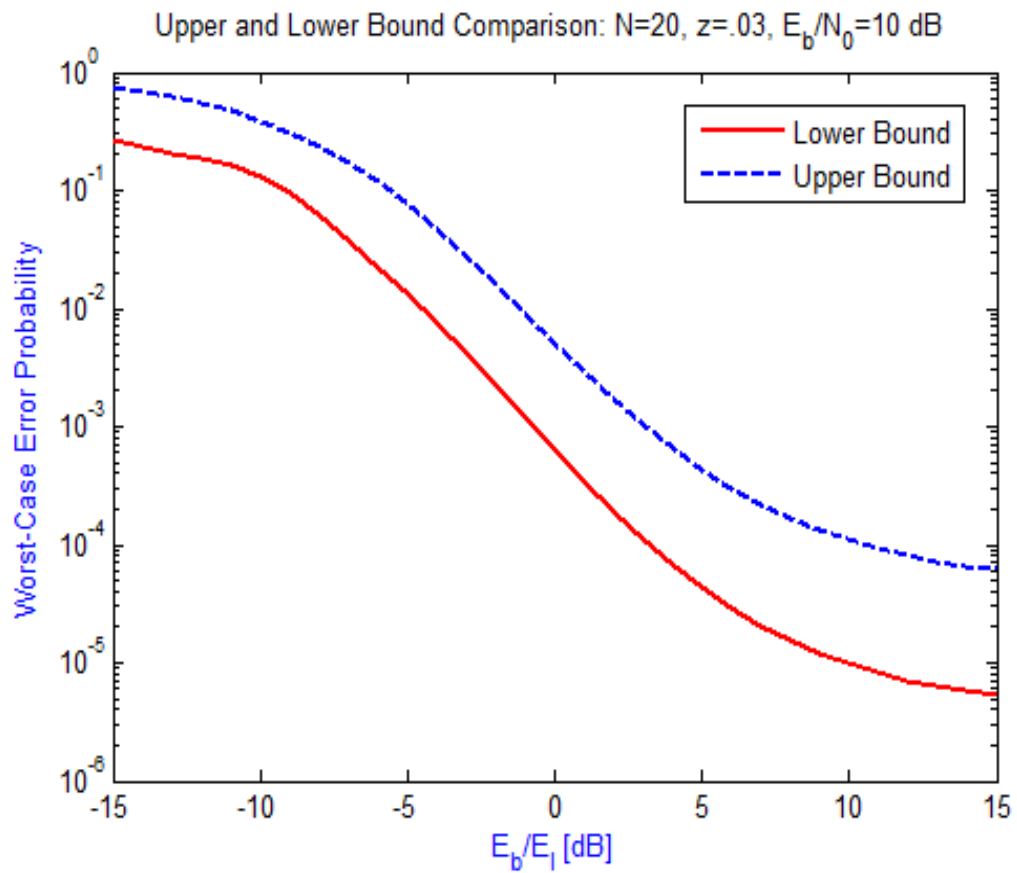


Figure 2.5: Comparison of Upper and Lower Bound: $N = 20$, $z = .03$, $E_b/N_0 = 10$ dB

Chapter 3

Examining the Effect of Altering z on Worst-Case Error Probability Bounds

3.1 Introduction

The previous research on regeneralized DSSS was performed with a static value for the z parameter in the chipping sequence. The goal of this chapter is to investigate how changes in the z parameter affect the worst-case error probability of the system, with the hopes of finding optimal values for z . The obvious way to go about this is to directly evaluate the equation for worst-case error with various z values. Since this equation has no direct solution, the bounds, as derived in [6], are evaluated instead. The equations will be evaluated and the results plotted using Matlab scripts.

Keep in mind that the goal of this research is to find z values that result in the

best worst-case error performance, however, there are a few other factors to consider. The first is N , or the number of chips per symbol. This can also be thought of as the number of dimensions in the signal space. Since the optimal z value will uniformly distribute the message signal among these N dimensions, it is thought that optimal z will change with N .

Other factors to consider are the signal to interference ratio, $\frac{E_b}{E_I}$, and the signal to thermal noise ratio, $\frac{E_b}{N_0}$. It is thought that the optimal z value should stay consistent for the range of $\frac{E_b}{E_I}$ that the system can tolerate. The same is thought to be true for $\frac{E_b}{N_0}$, due to the fact that the interference rejection capabilities of DSSS do not have any effect on AWGN.

3.2 Upper Bound

Since the upper bound is a simpler equation to evaluate, it was examined first. Figure 3.1 shows the effect of changing z on the upper bound to worst-case error performance.

In Figure 3.1 the upper bound on worst-case error performance is plotted against signal to interference ratio. No AWGN was added, and N was taken to be 20 chips per symbol. From this plot, it would seem that increasing z degrades performance. If an error probability of 10^{-6} is used as a figure of merit, changing z from .1 to .9 degrades performance by around 1dB. This trend of performance degrading as z is increased appears with various N values, and when noise is added. This is shown in Figure 3.2 and Figure 3.3.

This trend can be seen more clearly if we examine what $\frac{E_b}{E_I}$ ratio is necessary to

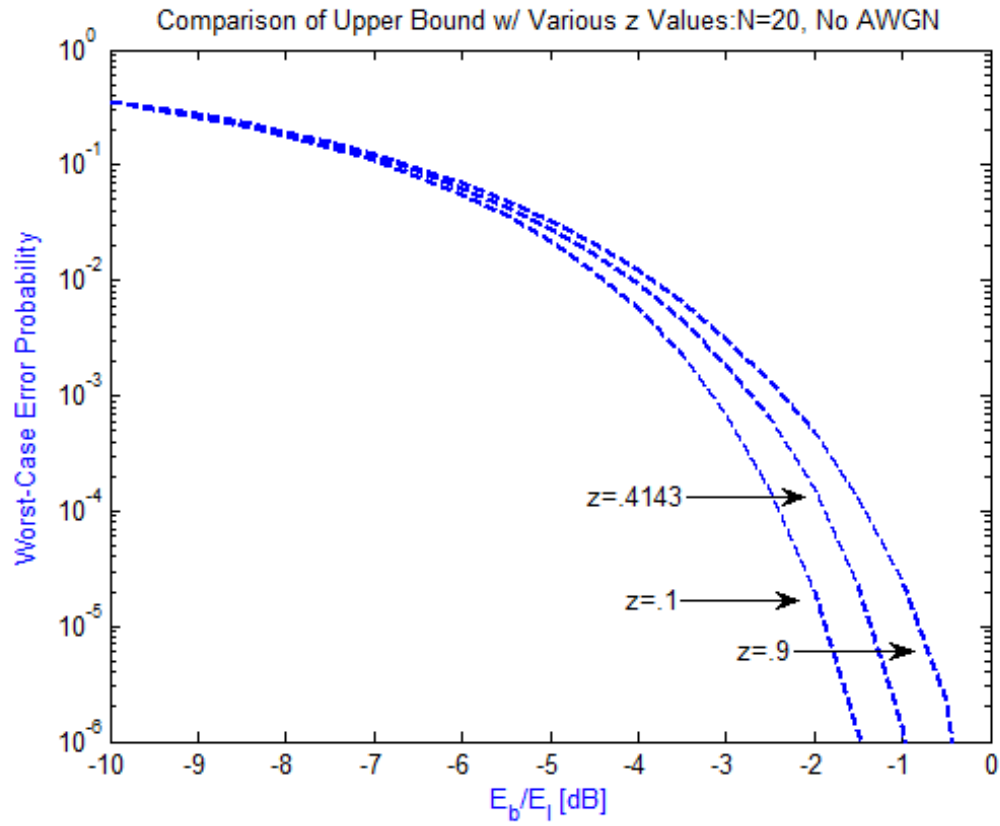


Figure 3.1: Effect of Altering z on Upper Bound with $N = 20$

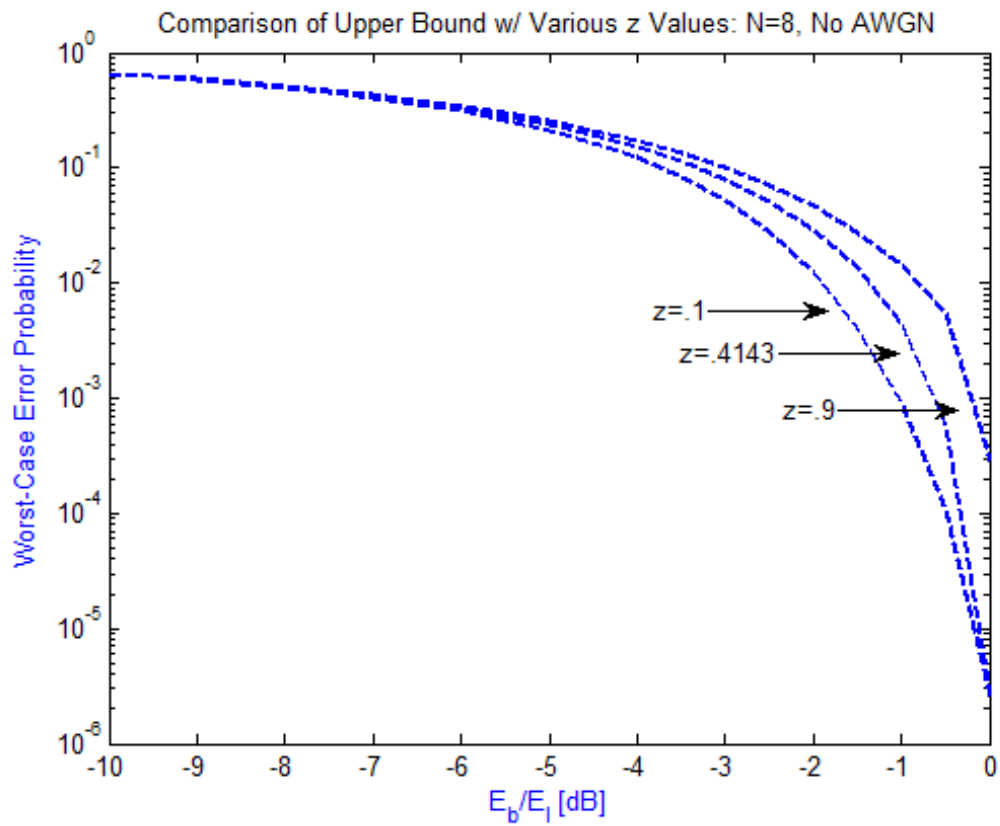


Figure 3.2: Effect of Altering z on Upper Bound with $N = 8$

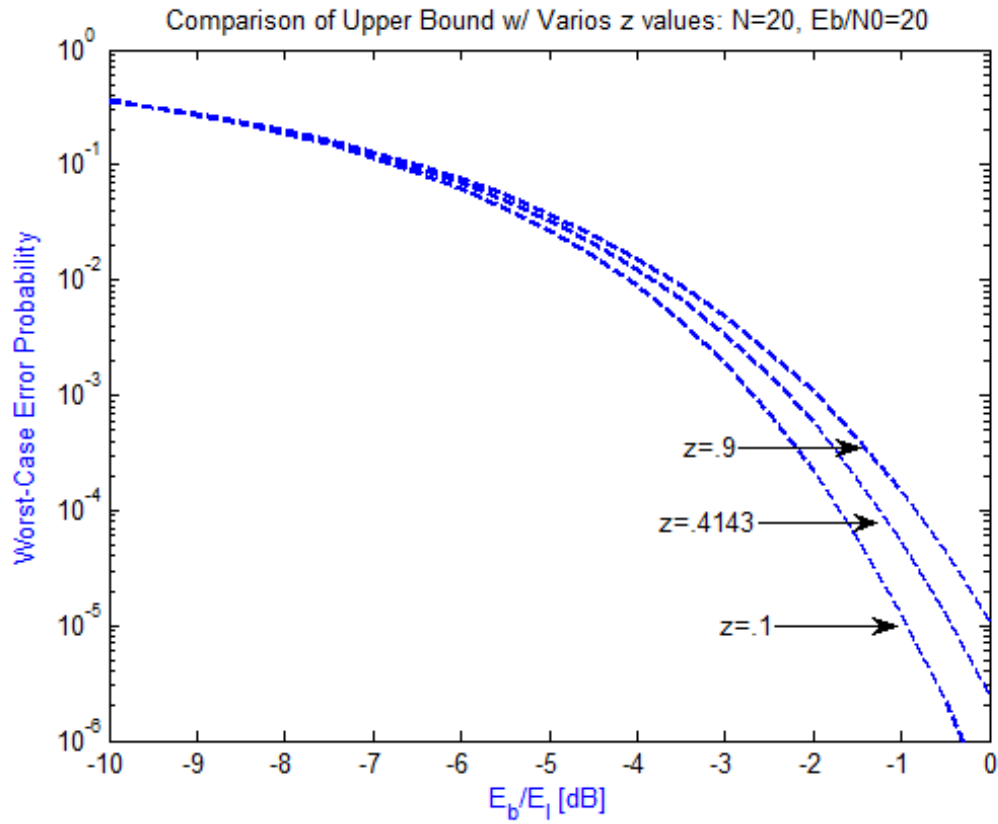


Figure 3.3: Effect of Altering z on Upper Bound with $N = 20$ and $E_b/N_0 = 20$ dB

achieve 10^{-6} error probability, for a range of z values. This is shown in Figure 3.4.

Here it is apparent that as z is increased, more energy is needed to guarantee a given worst-case error performance. The energy needed increases with z , and seems to be nearly linear over some areas of the plot. This data contradicts the initial idea that when the z value is found that uniformly distributes the message signal, the system will be optimized. If this were the case, the lowest point in the plot would occur somewhere within the range of z , and not at an extreme. Keep in mind however that the equation being evaluated is the upper bound, not an exact solution. It is possible that the upper bound is simply not sensitive to changes in z in a way that correlates with its optimization. It would seem that no useful information can be obtained from further calculations involving the upper bound, so we will shift focus to examining the lower bound.

3.3 Lower Bound

The effect of changing z on the lower bound on worst-case error performance will now be examined. To begin, Matlab was used to evaluate the lower bound for various values of z . This is shown in Figure 3.5. The Matlab program used can be found in Appendix A. In Figure 3.5 we can see that the lower bound behaves differently from the upper bound as z is changed. Error probability improves when z is changed from .1 to .2, and then seems to degrade as z is further increased. This would indicate the presence of some optimal z value somewhere around .2. This tells us that some information about optimal z values can be extracted by further examination of the

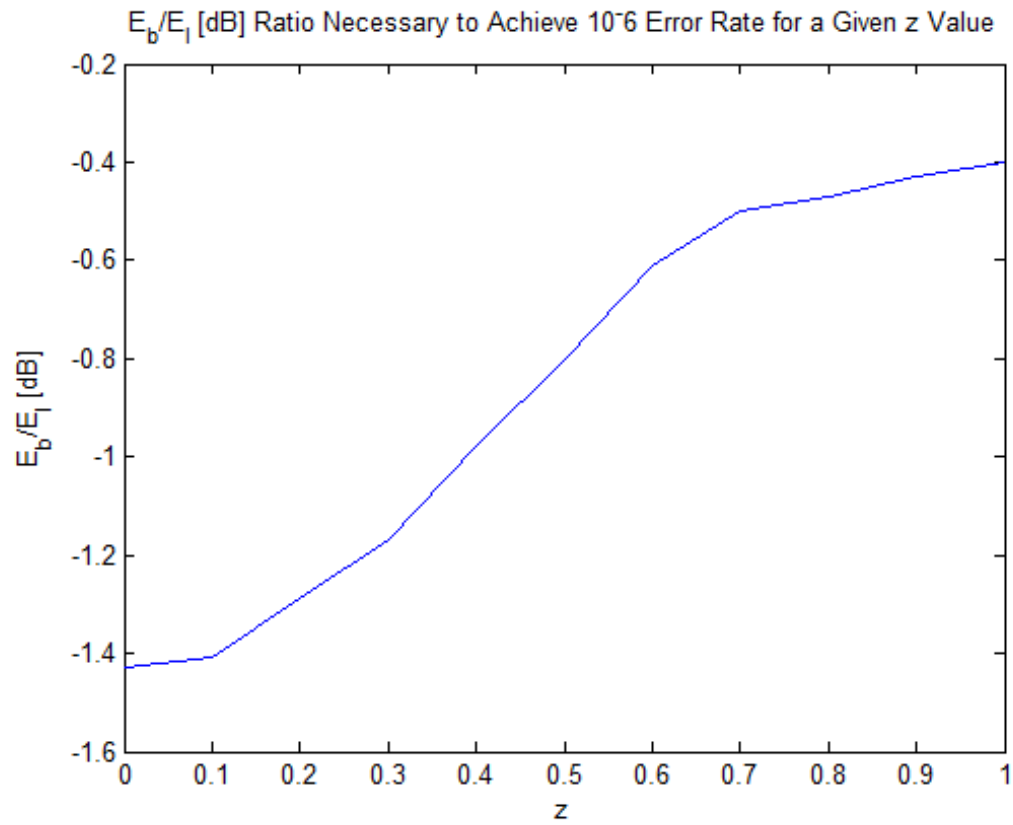


Figure 3.4: E_b/E_I Ratio Needed for 10^{-6} Error Rate for Given z with $N = 20$ and $E_b/N_0 = 20$ dB

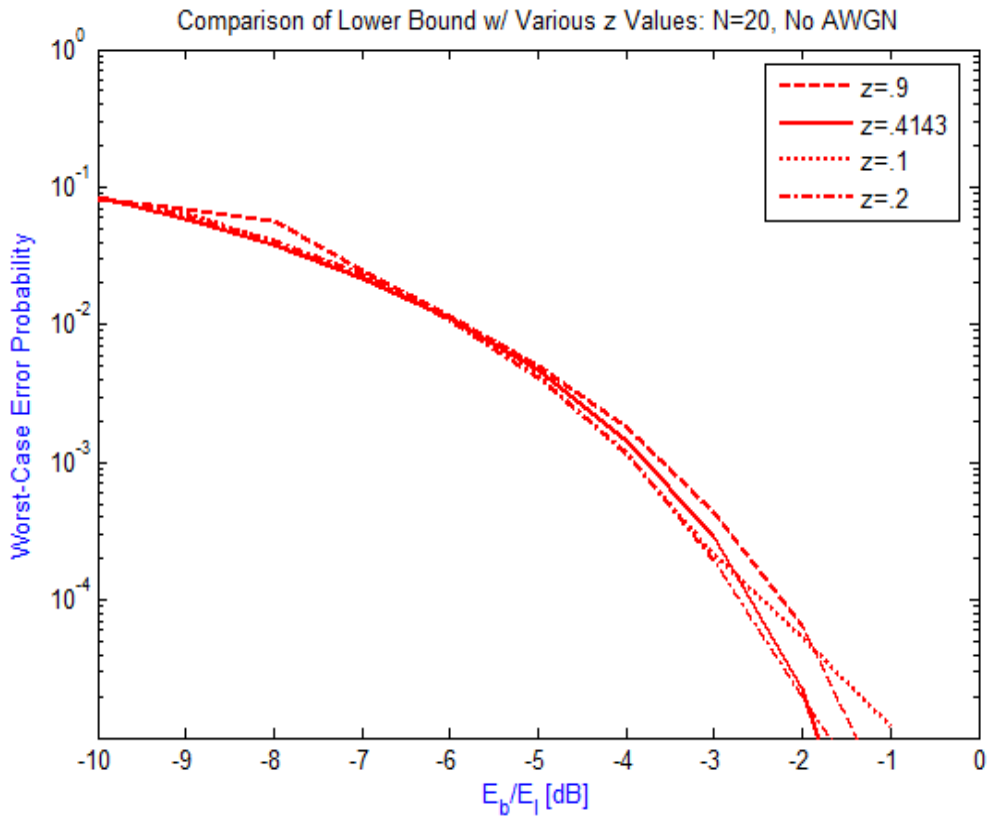


Figure 3.5: Effect of Altering z on Lower Bound with $N = 20$

lower bound. The difference in behavior between the upper and lower bound when z is changed is apparent in Figure 3.6. This figure shows the $\frac{E_b}{E_I}$ ratio necessary to achieve 10^{-3} error probability, over a range of z values, for both the upper and lower bound. N was set to 20, and no thermal noise was added. We can see that the upper bound plot increases monotonically, while the lower bound plot seems to have a minima at around $z=.3$.

In order to focus in on the what effect z has on the lower bound, and which z value minimizes worst-case error probability, a program was constructed using Matlab scripts. This program computes the lower bound for z values ranging from 0 to 1 at set intervals. It then looks at which z value yields the best performance for different signal to interference ratios. Other variables that can be altered are N , the number of chips per symbol, and $\frac{E_b}{N_0}$, the signal to thermal noise ratio. One thing to note is the limitations of the program with regards to N . Raising N exponentially increases the number of calculations needed to evaluate the lower bound, and the lower bound will need to be evaluated many times, depending on the number of z values tested. Therefore N will be limited to a maximum of 20.

3.3.1 Examining the Effect of Thermal Noise on the Lower Bound

No Thermal Noise Added

To begin we will look at the results of running the Matlab script with $N=5$ and a signal to thermal noise ratio of 10000, which essentially means that thermal noise has

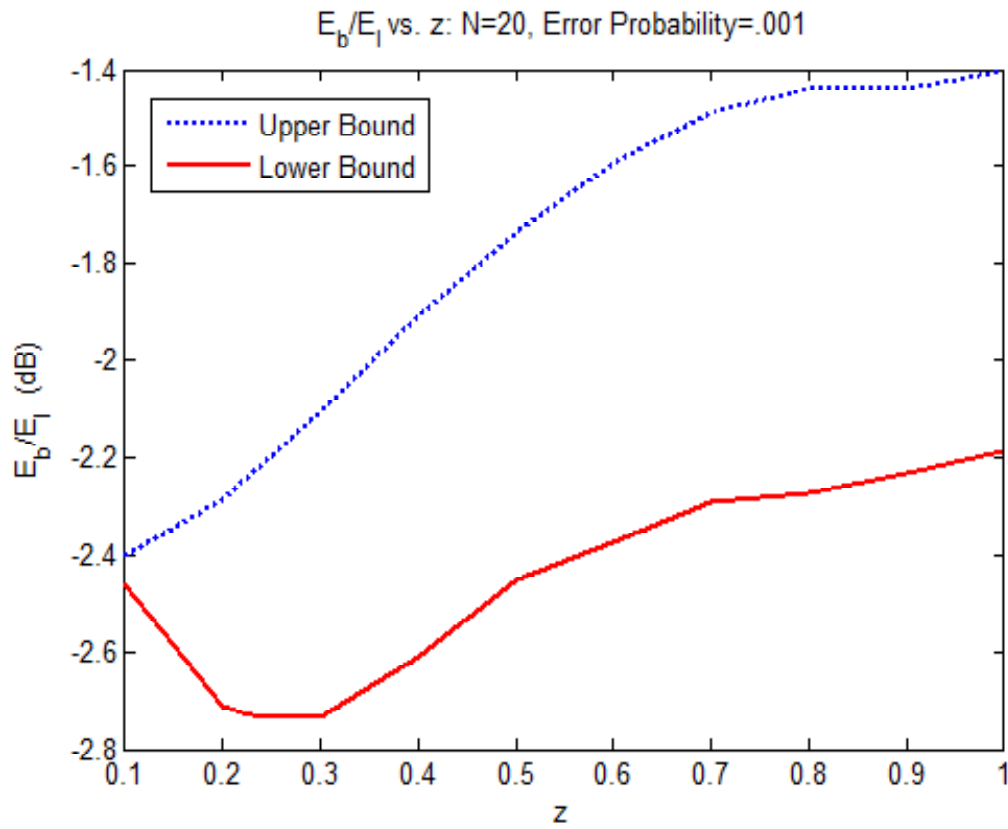


Figure 3.6: E_b/E_I Ratio Needed for 10^{-3} Error Rate for Given z with $N = 20$ and $E_b/N_0 = 200$ dB

no effect on the system. The optimal z value for a range of signal to interference levels is shown in Figure 3.7. Figure 3.8 shows the resulting worst-case error probability when these z values are used. From Figure 3.7 it is evident that for this case, optimal z is not a static value over signal to interference ratio, but instead varies from .1 to 1 before settling at 0 when the signal to interference ratio becomes 0dB. Looking at the corresponding worst-case error probability when these z values are used (Figure 3.8) we see a waterfall curve that is somewhat jagged. Where optimal z becomes 0, the probability curve indicates that the probability has dropped too far to be calculated.

$$E_b/N_0 = 10\text{dB}$$

We will now look at the effect of adding some thermal noise to the system. Running the same Matlab script with $N=5$ and a signal to noise ratio of 10 yields the optimal z values shown in Figure 3.9. The corresponding worst-case error probability is shown in Figure 3.10.

These figures clearly show different results than when the test was performed without noise. At around -2 dB $\frac{E_b}{E_I}$ the optimal z value begins to stabilize. From this point on, it fluctuates between .33 and .35. Observing the corresponding error probability, it can be seen that at around -2 dB $\frac{E_b}{E_I}$, the worst-case error probability begins its steepest descent. Note that in this case the error probability is much higher than the previous case for a given $\frac{E_b}{E_I}$ ratio. This is because some thermal noise has been added. If the plot were extended, error probability would eventually reach some error floor caused by the thermal noise.

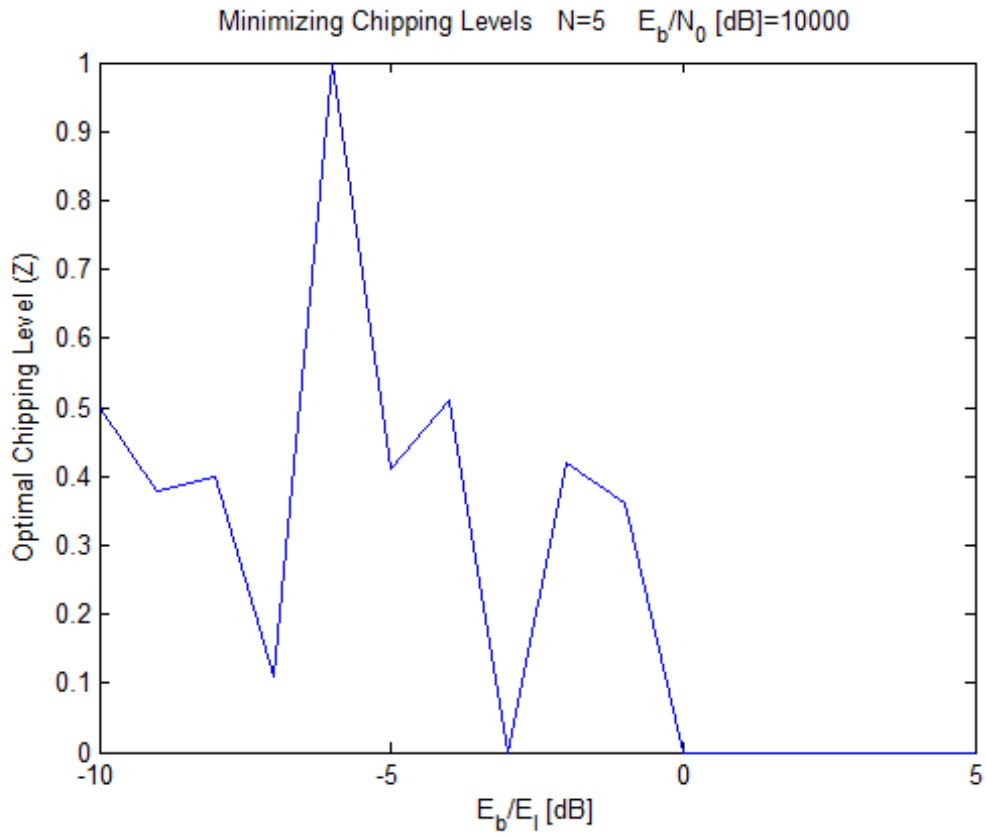


Figure 3.7: Optimal z vs. Signal to Interference Ratio: No Noise and $N = 5$

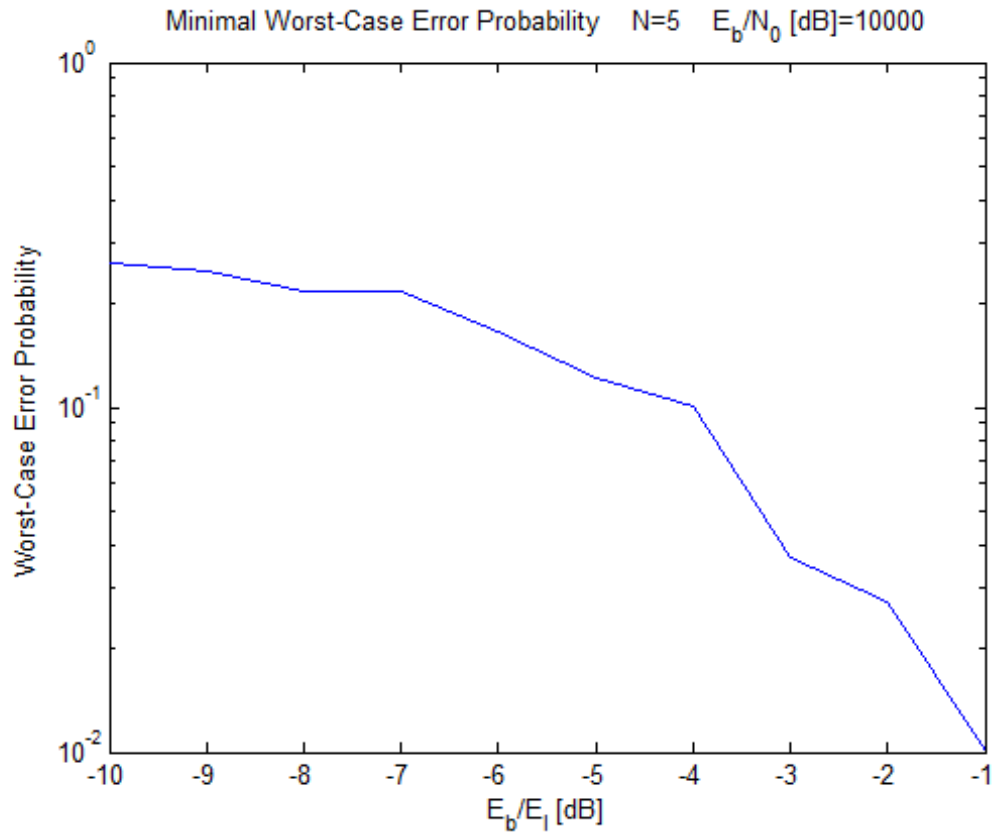


Figure 3.8: Error Probability vs. Signal to Interference Ratio: No Noise and $N = 5$

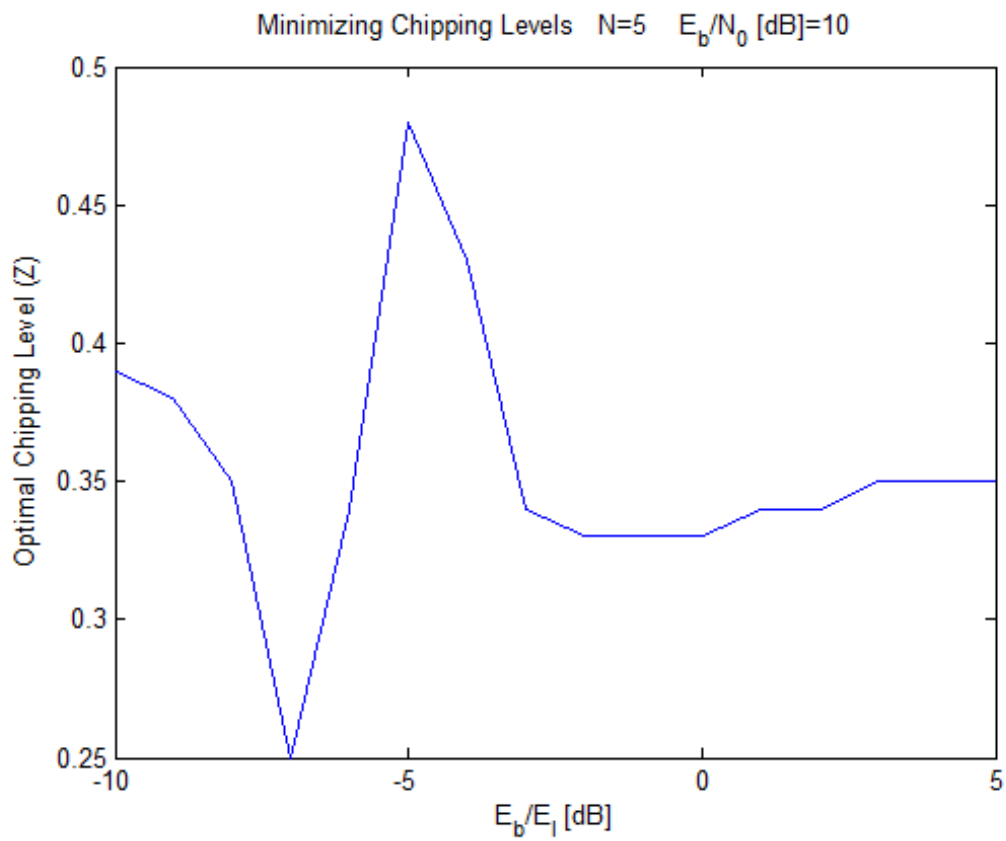


Figure 3.9: Optimal z vs. Signal to Interference Ratio: $E_b/N_0 = 10$ dB and $N = 5$

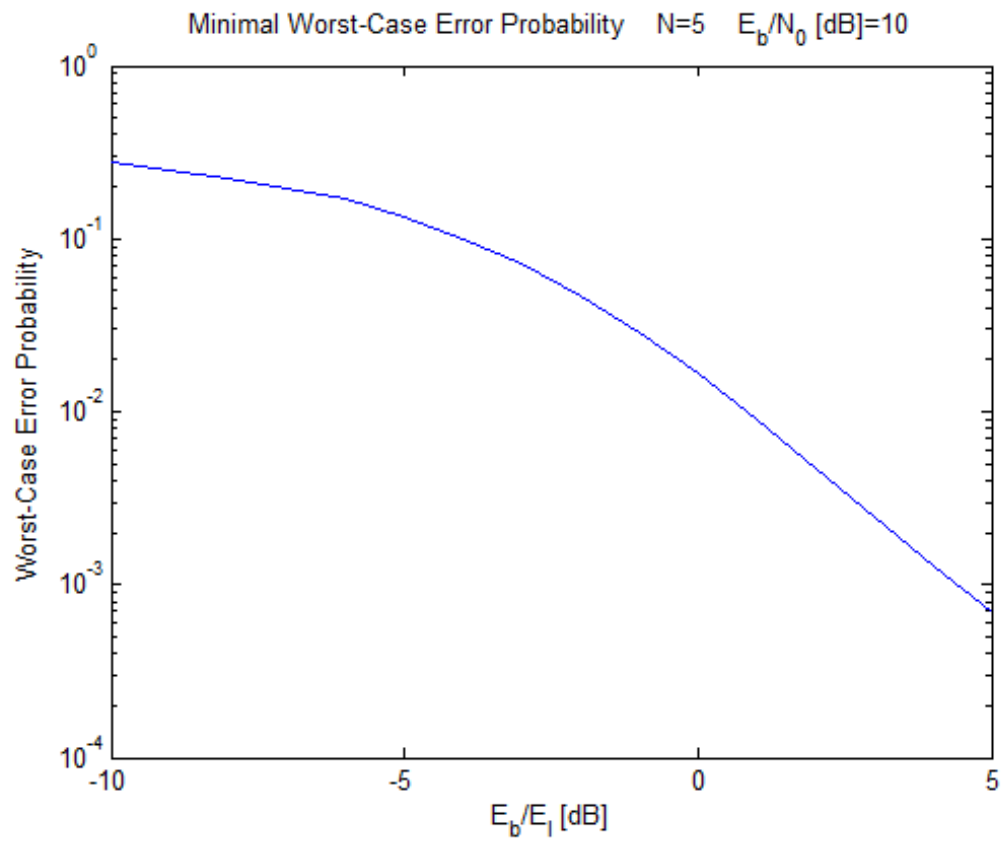


Figure 3.10: Error Probability vs. Signal to Interference Ratio: $E_b/N_0 = 10\text{dB}$ and

$$N = 5$$

$$E_b/N_0 = \mathbf{0dB}$$

To see if this trend continues, we will run the same Matlab script with $N=5$, and a signal to noise ratio of 0. The resulting optimal z values are shown in Figure 3.11, with the corresponding worst-case error probability shown in Figure 3.12.

These plots show that with more thermal noise added, the optimal z value becomes completely stable at a low $\frac{E_b}{E_I}$ ratio. In this case, it becomes completely stable at around -7 dB $\frac{E_b}{E_I}$. However, with 0 dB $\frac{E_b}{N_0}$, the overall system performance is not very good, as the thermal noise has moved the error probability floor up to around 10^{-1} .

3.3.2 Valid Range for Optimizing the z Parameter

The previous section demonstrated the effect that thermal noise has on the optimization of the z parameter. Since spread-spectrum does not provide benefits in terms of error probability for the all AWGN channel, the amount of thermal noise should not have any significant effect on the outcome of the experiment. However, since there are obvious effects when noise is added, this must be examined, and valid ranges for running the Matlab script in terms of $\frac{E_b}{E_I}$ and $\frac{E_b}{N_0}$ must be discussed.

When considering the effect of thermal noise we must remember that we are not performing a direct simulation of the system, or even a direct evaluation of the worst-case error probability. We are evaluating the lower bound on worst-case error probability maximized over the interfering signal. This maximization occurs over discrete points. The addition of AWGN smooths out the lines connecting these points and yields more accurate results. With no AWGN added, the plot becomes jagged,

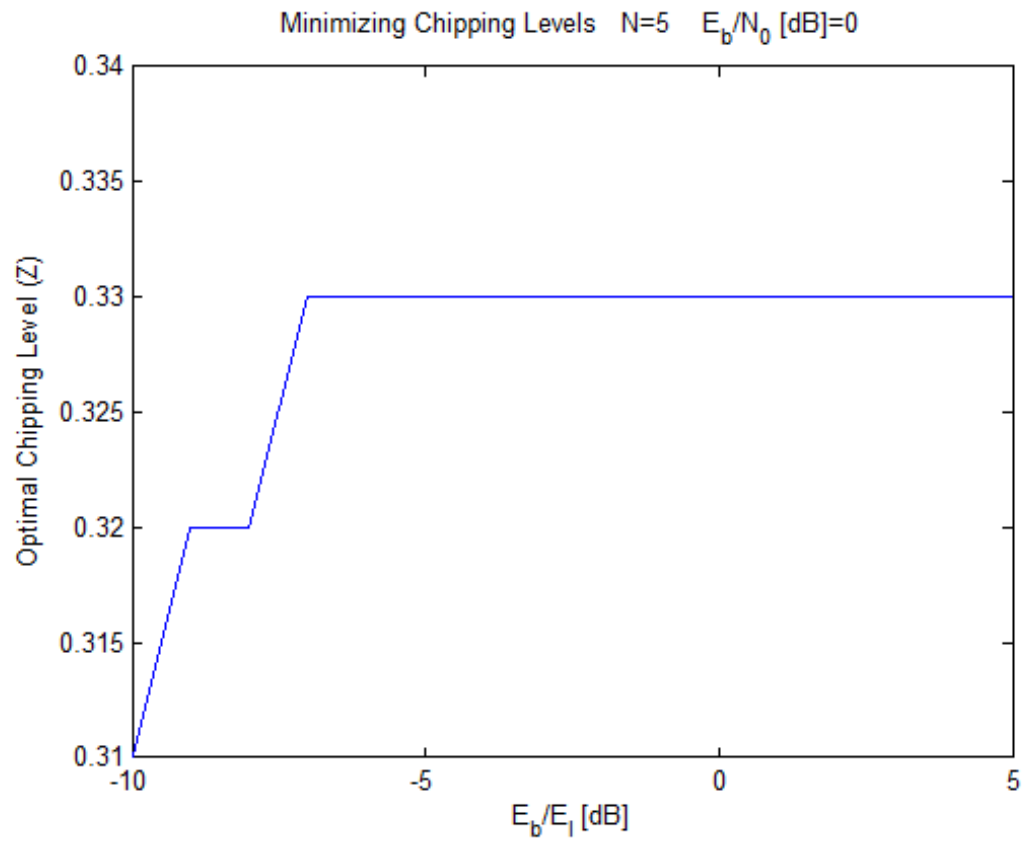


Figure 3.11: Optimal z vs. Signal to Interference Ratio: $E_b/N_0 = 0$ dB and $N = 5$

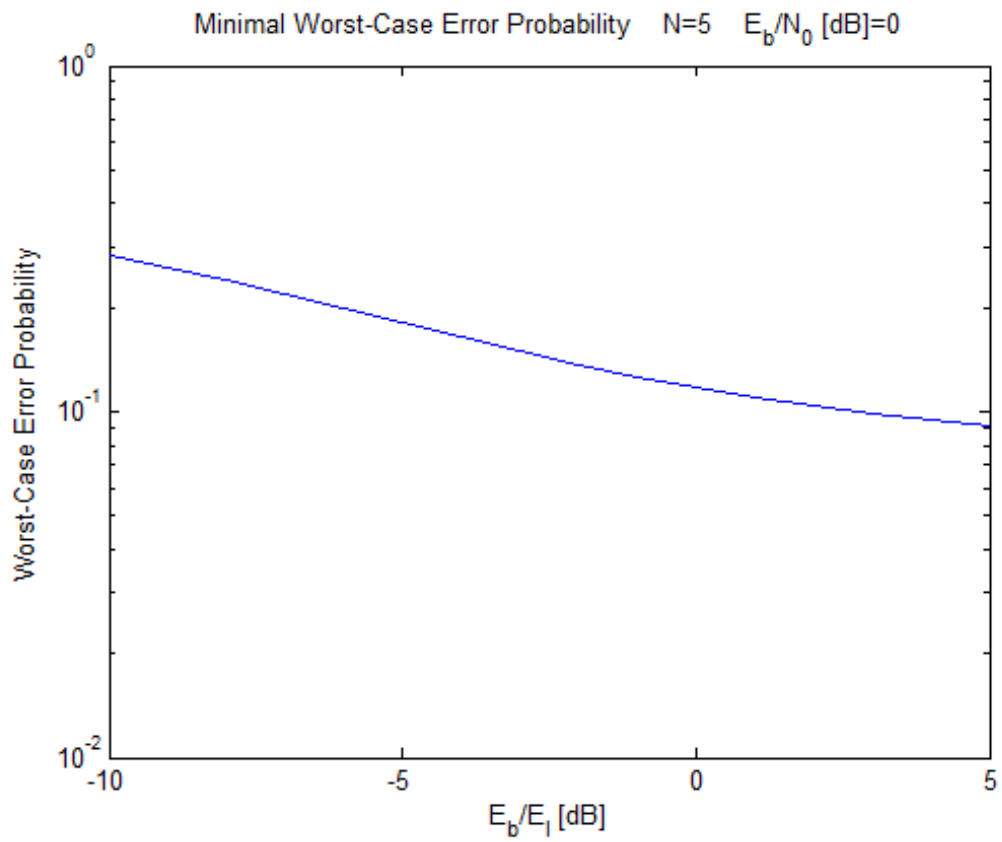


Figure 3.12: Error Probability vs. Signal to Interference Ratio: $E_b/N_0 = 0$ dB and

$N = 5$

as seen in Figure 3.9 and Figure 3.7. The discontinuity in these plots are caused by the discrete optimization process, and are not reliable data points. Ideally, since we are optimizing the system for performance under an arbitrary interference source, we should be examining results with no AWGN. However, since adding AWGN yields more accurate results, finding the optimal z value will be done with some AWGN added. These results are still valid considering, theoretically, that DSSS does not improve performance in the AWGN channel. Also, when we look at the results, we will see that once the optimal z value becomes stable, it will remain the same value for increasing levels of AWGN.

To graphically look at the valid ranges of the simulation, consider Figure 3.13, Figure 3.14, and Figure 3.15. These figures display the optimal z value for $N=3$, over a range of $\frac{E_b}{E_I}$ ratios from -20 dB to 20 dB. Figure 3.13 shows this simulation with no noise added. It is apparent that under these conditions, z fluctuates wildly, and is 0 at for much of the plot. Again, these discrepancies are a result of the jagged nature of the discrete maximization of the lower bound when no AWGN is present. Figure 3.14 shows the same range with a $\frac{E_b}{N_0}$ ratio of 10 dB. Here we can see how z is beginning to smooth out as the $\frac{E_b}{E_I}$ ratio is improved. There are still some fluctuations, but they are minor compared to the no AWGN case. The portion of the graph where z is at 0 is a result of the interfering signal being too strong and completely wiping out the message signal. Finally, Figure 3.15 shows the same range with a $\frac{E_b}{N_0}$ ratio of 0 dB. Here, z becomes completely stable after the initial period where interference overpowers the message signal. In this case it can be seen that optimal z settles at a value of .38. The case where $\frac{E_b}{N_0} = 10$ dB was fluctuating around this value as well.

For any level of $\frac{E_b}{N_0}$ ratio lower than 0, optimal z will settle at exactly .38, for $N=3$. This trend remains the same when N is increased, the only differences being the value on which optimal z settles decreases as N increases, and the level of $\frac{E_b}{E_I}$ ratio at which optimal z becomes stable decreases as N increases.

In order to find the optimal z value for each N , we must stay in the valid range for this experiment. We must avoid the area where the message signal is overpowered by the interference, and enough AWGN should be present to smooth out the discrete optimization of the lower bound. To satisfy these requirements, optimal z will be found with 0 dB $\frac{E_b}{N_0}$ ratio over a range of -5 dB to 5 dB $\frac{E_b}{E_I}$ ratios.

3.3.3 Finding the Optimal z Value

Introduction

This section will examine in detail how optimal z is found for a few values of N . It will then go on to present a chart of optimal z values for $N=3$ through $N=20$. The range for N is 3-20 because the program does not work at $N=2$, and the calculations become prohibitively long for any N greater than 20.

Optimal z for $N=5$

To begin, we will find the optimal z value for $N=5$ using the Matlab script with the range defined in the previous section. For each $\frac{E_b}{E_I}$ ratio within the range, the script increments z by .01 from 0 to 1 and looks at which value results in the best worst-case error probability. Optimal z for this case was found to be .33, as shown in figure 3.16.

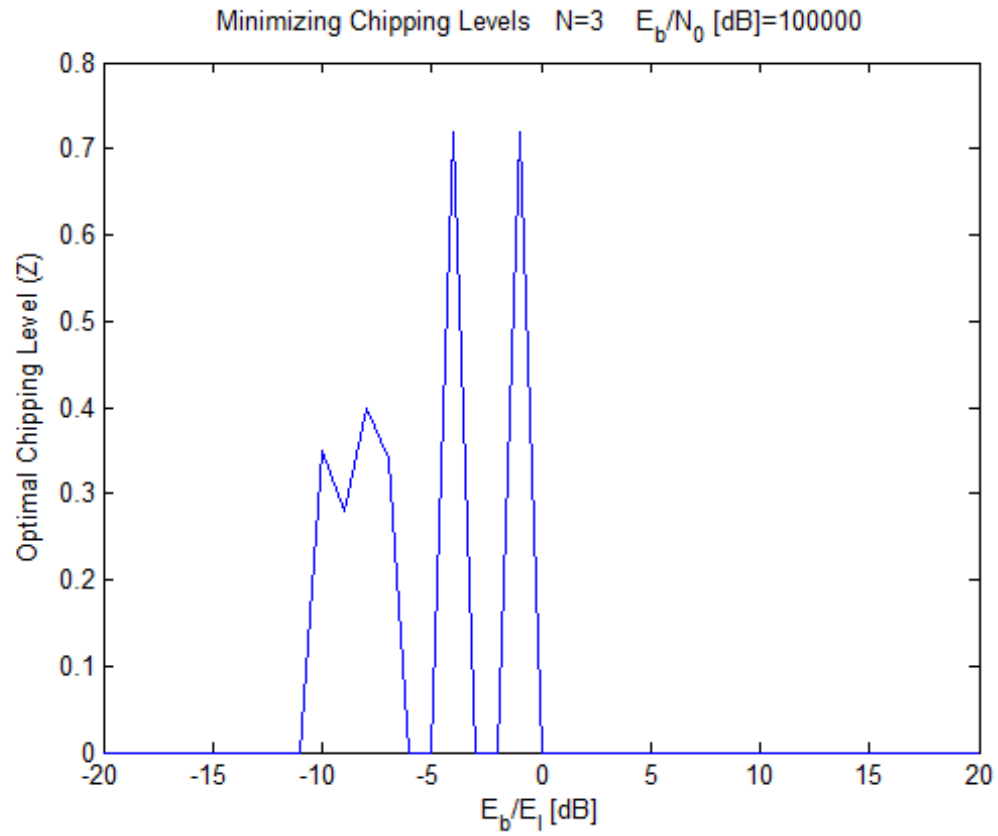


Figure 3.13: Upper Bound Experiment Range: No AWGN

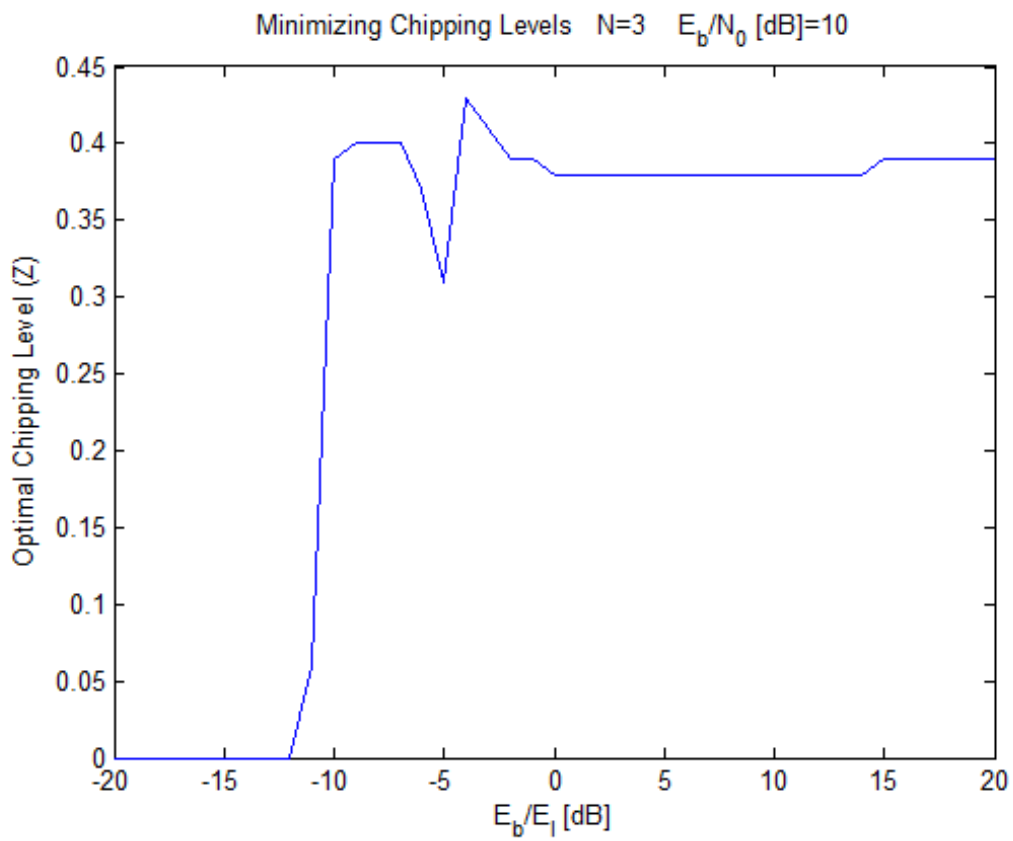


Figure 3.14: Upper Bound Experiment Range: $E_b/N_0 = 10$ dB

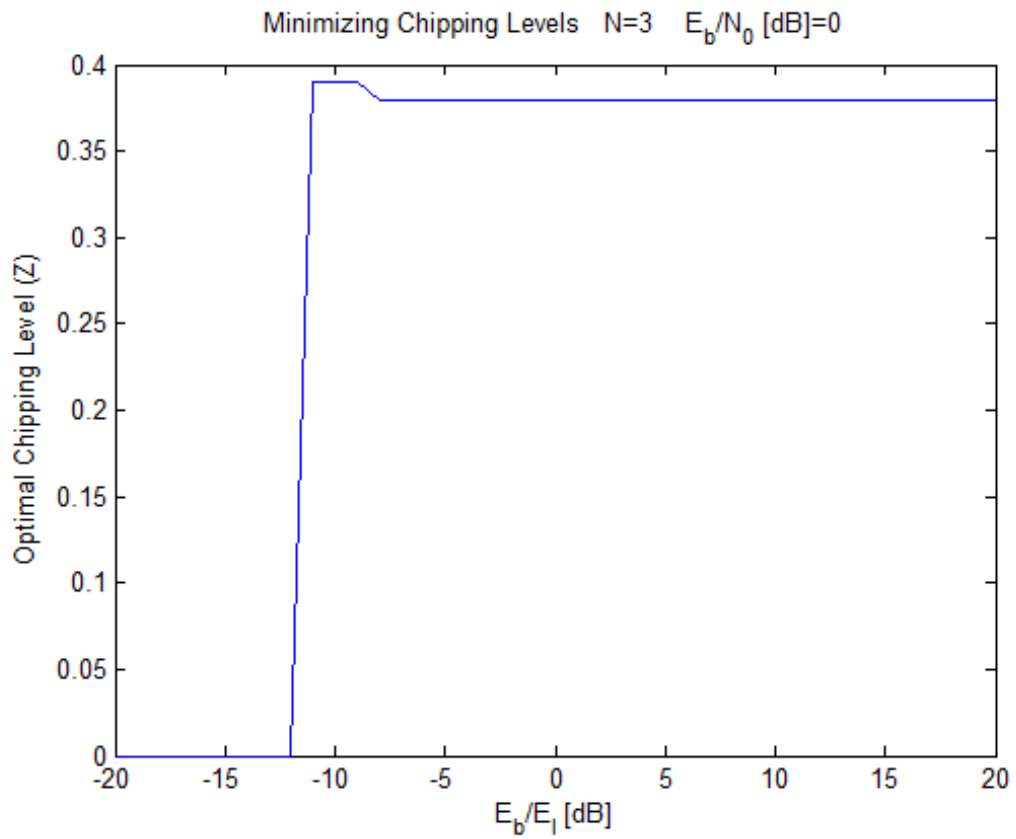


Figure 3.15: Upper Bound Experiment Range: $E_b/N_0 = 0$ dB

It is stable throughout the entire range.

Figure 3.17, Figure 3.18, and Figure 3.19 show the worst-case error probabilities for the range of z , at a fixed $\frac{E_b}{E_I}$ ratio. In these figures, worst-case error probability reaches a distinct minima at $z=.33$ at all three levels of $\frac{E_b}{E_I}$ ratio shown. A close up of this minima is shown in Figure 3.20. Note that the worst-case error probability, while optimized, does not change all that much over the range of z , and is fairly high. This is due to the necessary addition of AWGN. To further illustrate this, the overall worst-case error performance with $z=.33$ over the valid range is shown in Figure 3.21.

Optimal z for $N=10$

We will now perform the same experiment to obtain the optimal z value for $N=10$. In this case optimal z was found to be .24, as shown in Figure 3.22. Figure 3.23 shows the worst-case error probability over the range of z , with a $\frac{E_b}{E_I}$ ratio of -5 dB. This plot, like the similar ones for $N=5$, shows a clear minima at $z=.24$. It appears however, that the initial arc in the graph before the minima is becoming less defined. Figure 3.24 shows the worst-case error probability with $z=.24$. This shows some improvement over the $N=5$ case, but is still quite bad, due to the high levels of AWGN added in.

Optimal z for $N=15$

Next we will examine $N=15$. The optimal z value for $N=15$ was found to be .15. This is shown in Figure 3.25. Again, when we examine the worst-case error probability over

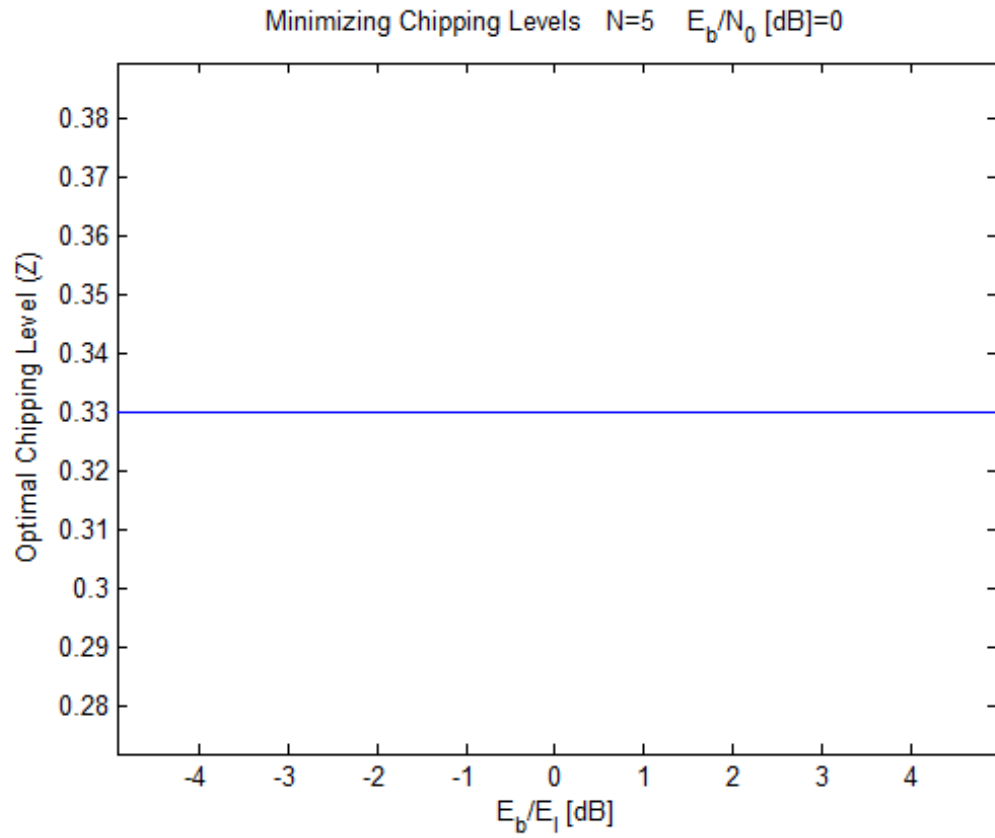


Figure 3.16: Optimal Chipping Level for $N = 5$

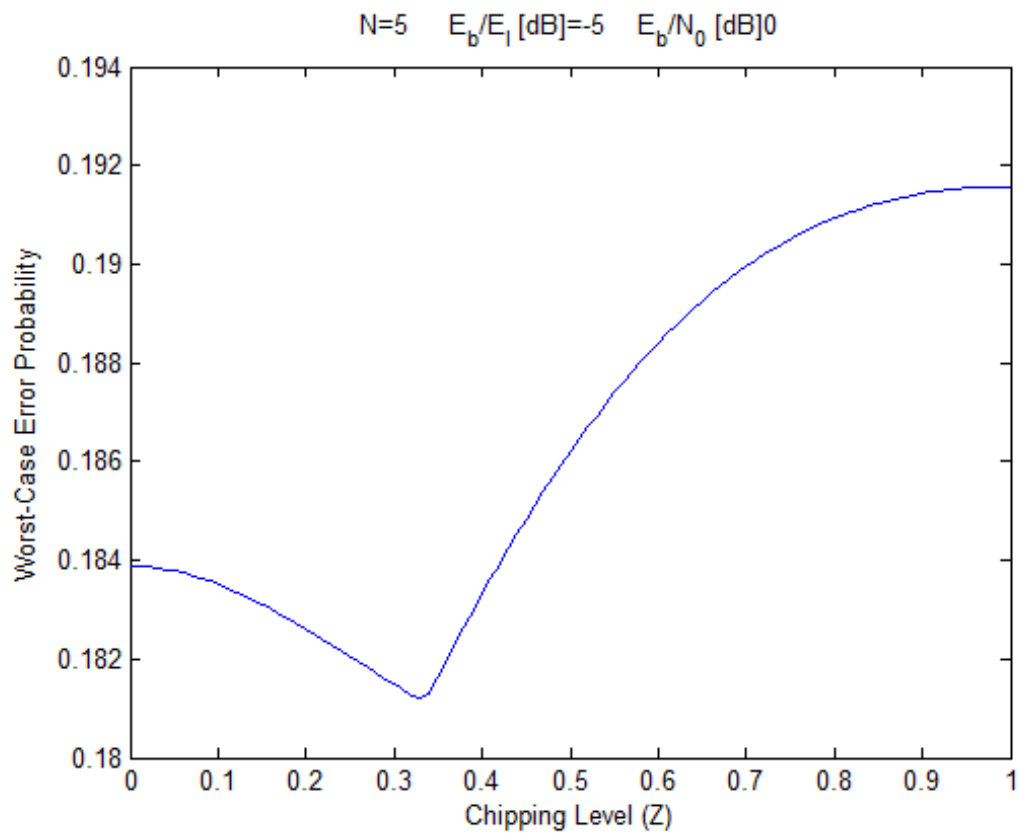


Figure 3.17: Worst-Case Error Probability vs. z : $E_b/E_I = -5$ dB

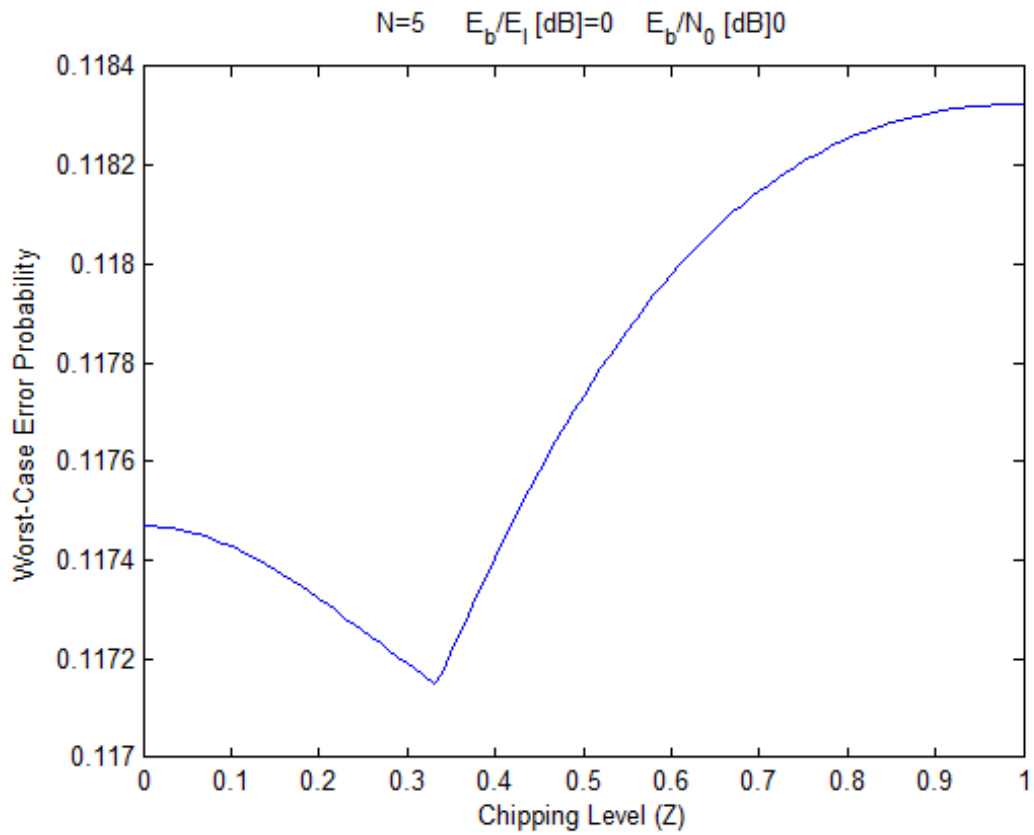


Figure 3.18: Worst-Case Error Probability vs. z : $E_b/E_I = 0$ dB

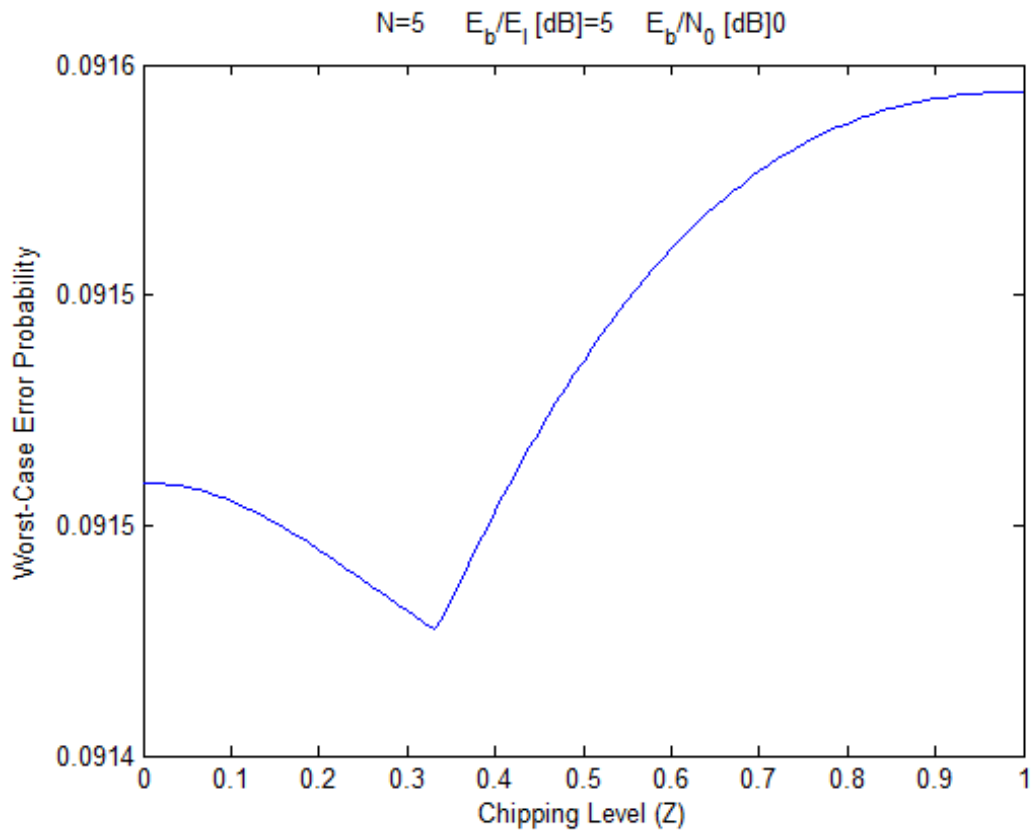


Figure 3.19: Worst-Case Error Probability vs. z : $E_b/E_I = 5$ dB

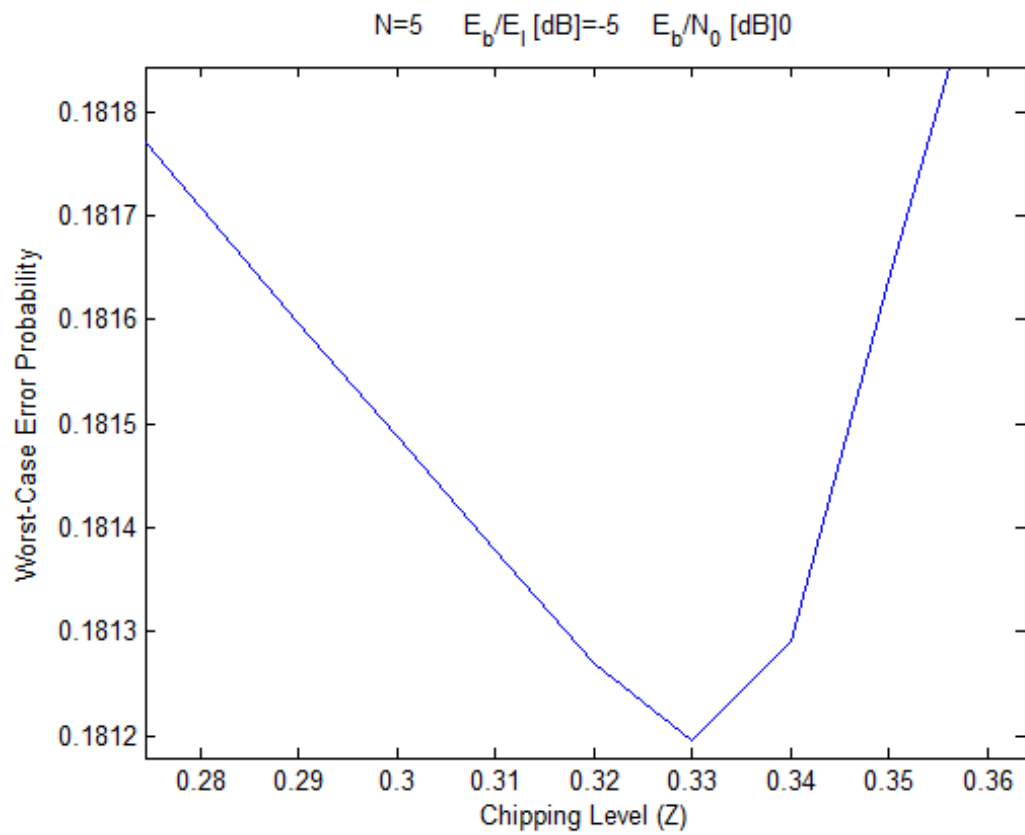


Figure 3.20: Worst-Case Error Probability vs. z : $E_b/E_I = -5$ dB, Close Up

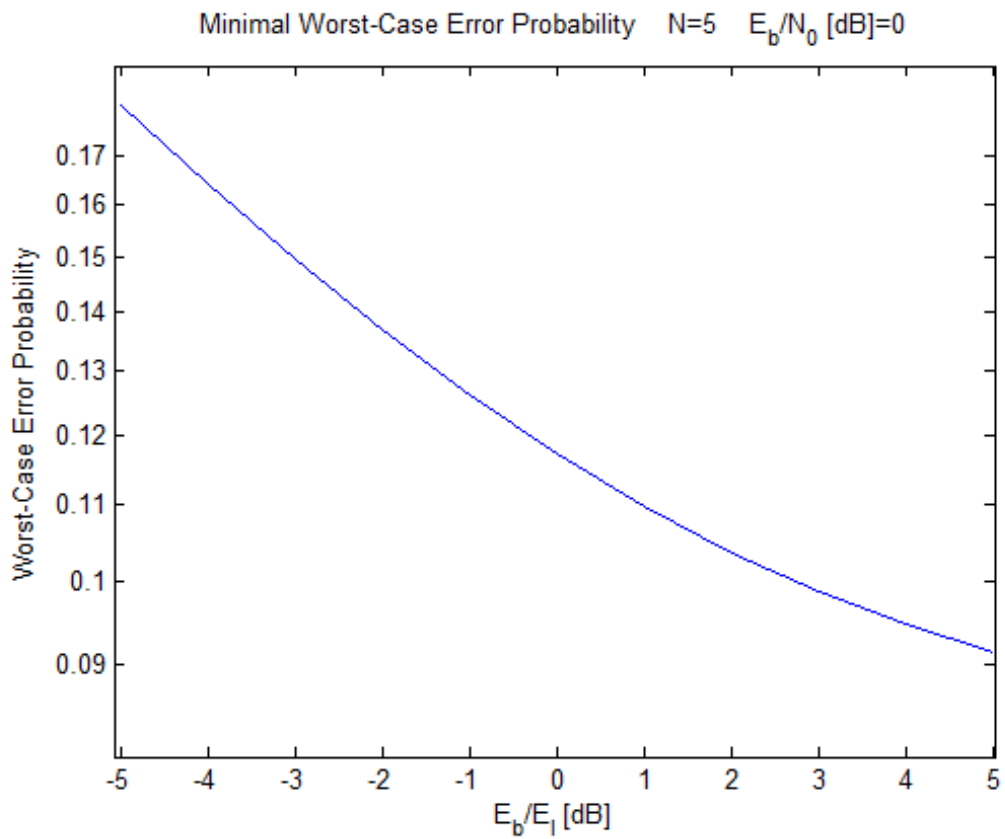


Figure 3.21: Minimal Worst-Case Error Probability: $N = 5$ and $z = .33$

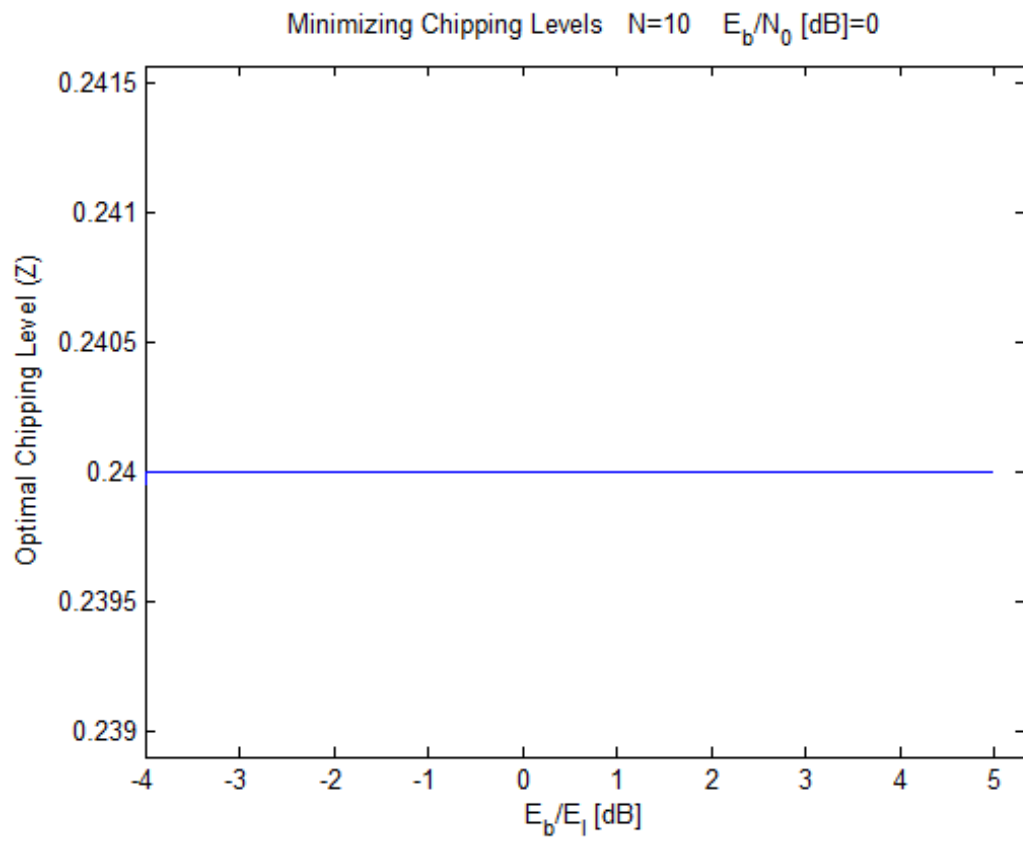


Figure 3.22: Optimal Chipping Level for $N = 10$

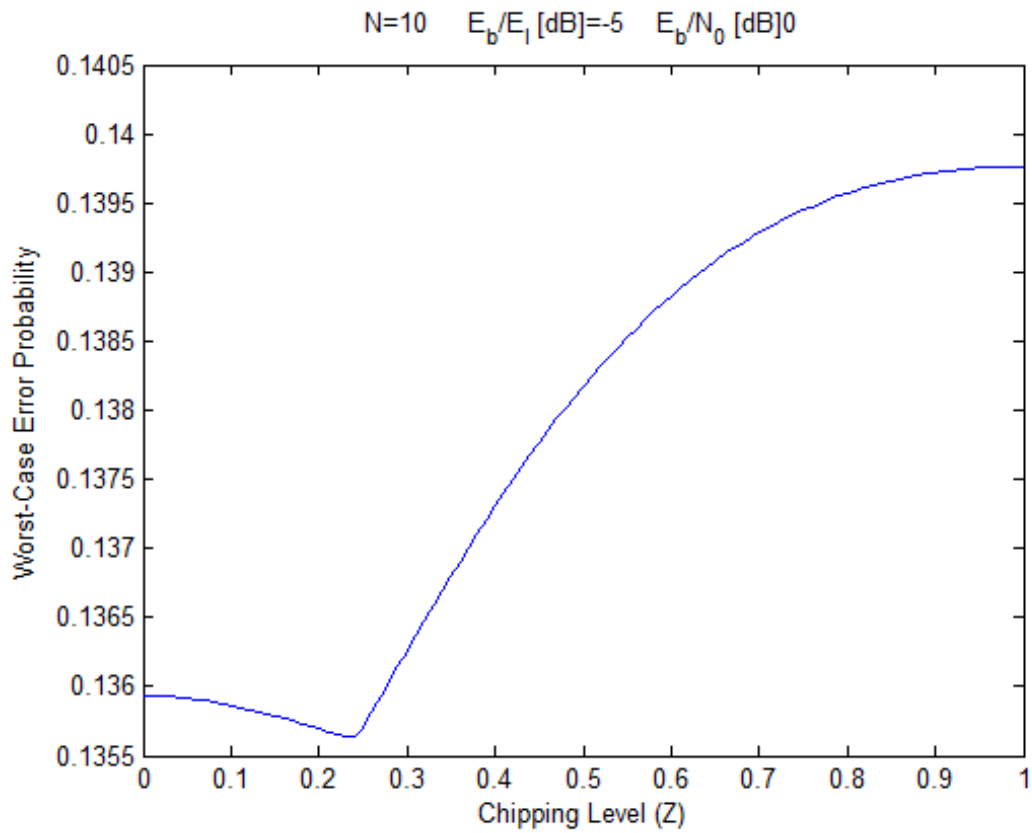


Figure 3.23: Worst-Case Error Probability vs. z : $E_b/E_I = -5$ dB

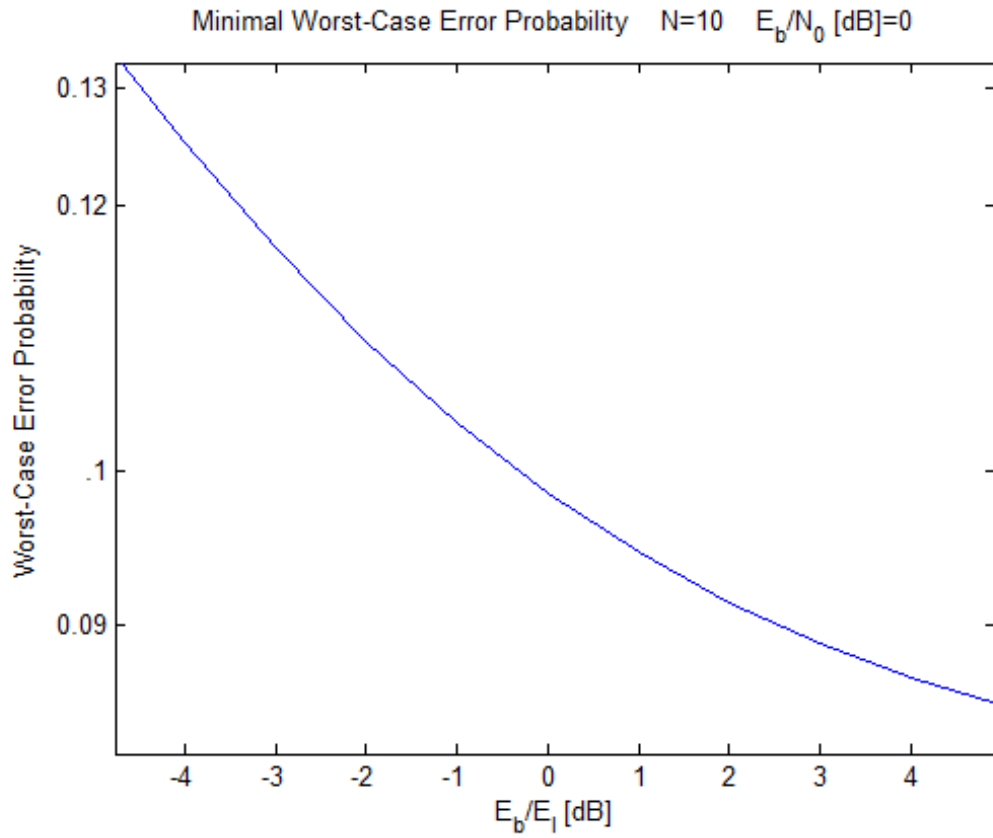


Figure 3.24: Minimal Worst-Case Error Probability: $N = 10$ and $z = .24$

the range of z , the minima appears less defined, as the initial arc has further flattened. This is displayed in Figure 3.26. If we examine the worst-case error probability with $z=.15$ (Figure 3.27) we see that there is a marginal improvement over the $N=10$ case.

3.3.4 Optimal z Value for $N=3-20$

The Matlab script was used to find the optimal chipping level, z , for minimized worst-case error probability, for $N=3$ through $N=20$. The results are summarized in Figure 3.28. The chart shows that the optimal z value decreases as N is increased. This relationship is plotted in Figure 3.29. From this plot it appears as if the relationship between z and N is nearly linear. This trend could not continue though, as N increased, because a negative z value cannot be used.

It should be noted that while we were able to find optimal values for z , it is somewhat difficult to quantify the gain when using the optimal z value in the system. This is because in order to smooth out the discrete optimization of the lower bound, large amounts of thermal noise were added, which result in large error rates. Figure 3.30 shows that the optimal z value does result in the best performance when compared to other z values, but improvement is not large, and the error rate is still high. The improvement however, while small, is still notable because it introduces no system complexity and is essentially free. This is discussed further in the section regarding future research topics.

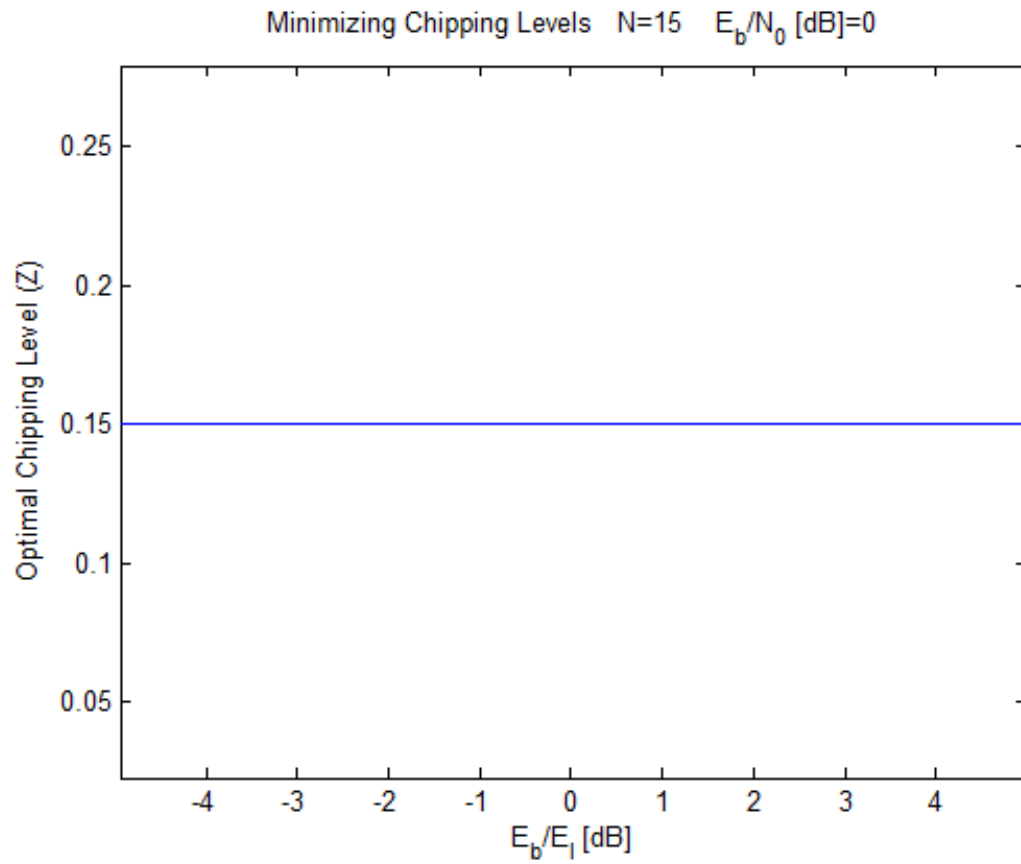


Figure 3.25: Optimal Chipping Level for $N = 15$

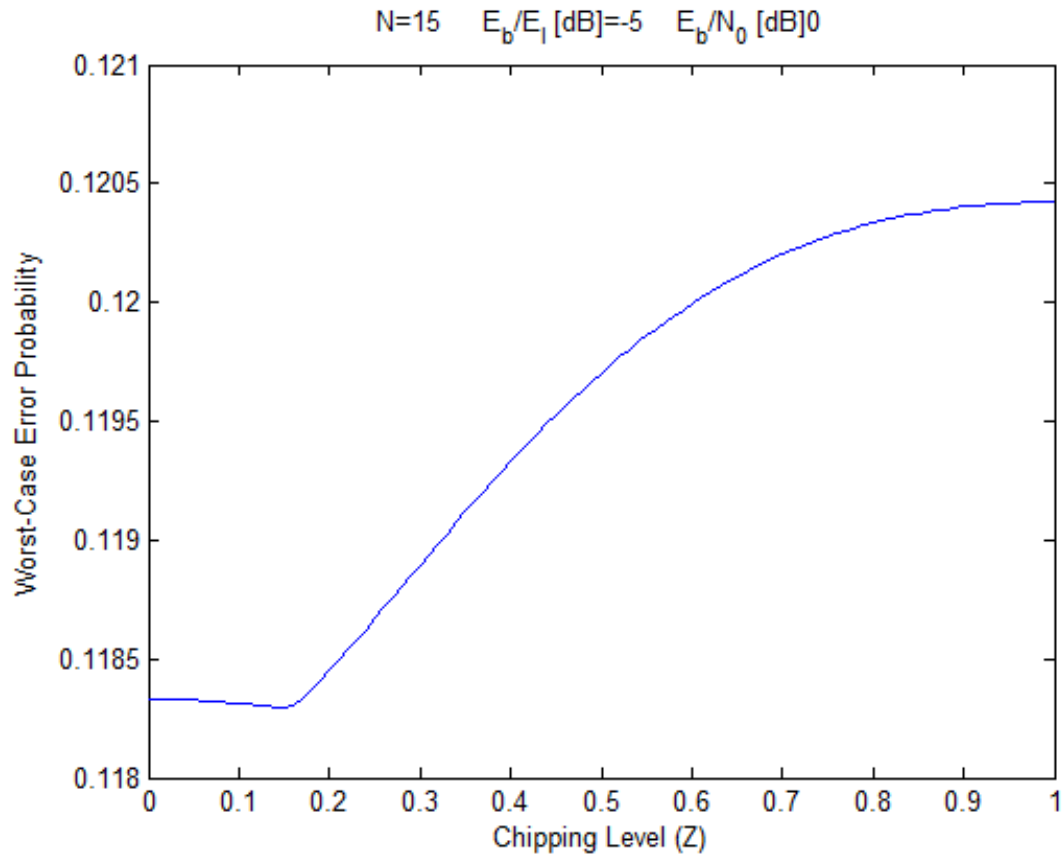


Figure 3.26: Worst-Case Error Probability vs. z : $E_b/E_I = -5$ dB

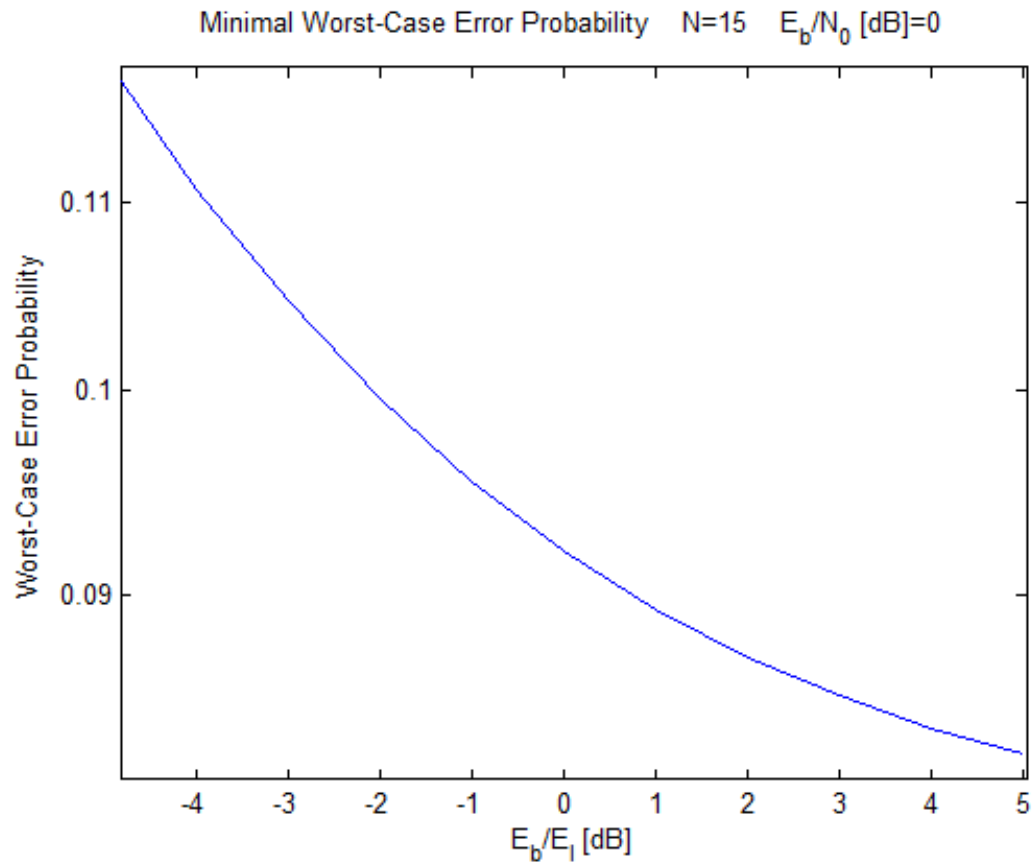


Figure 3.27: Minimal Worst-Case Error Probability: $N = 15$ and $z = .15$

N	Optimal z Value
3	0.38
4	0.35
5	0.33
6	0.31
7	0.29
8	0.27
9	0.25
10	0.24
11	0.22
12	0.02
13	0.18
14	0.17
15	0.15
16	0.13
17	0.11
18	0.09
19	0.07
20	0.03

Figure 3.28: Optimal z Value for Various N

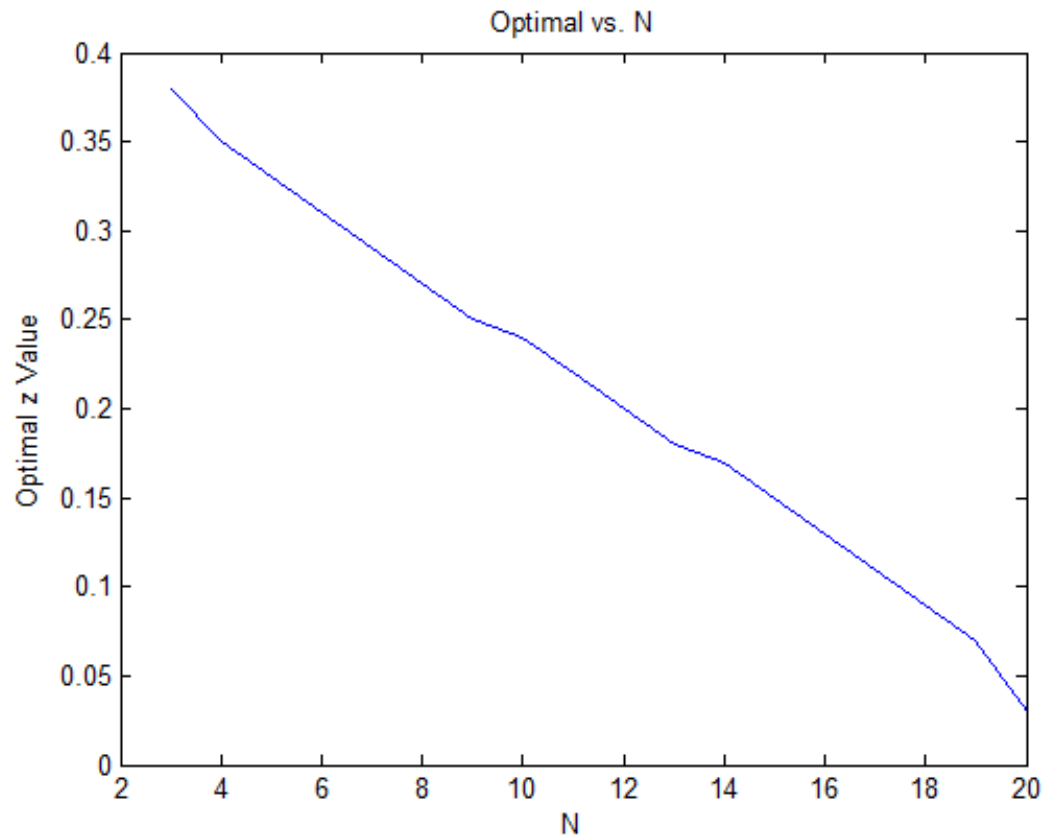


Figure 3.29: Relationship Between Optimal z and N

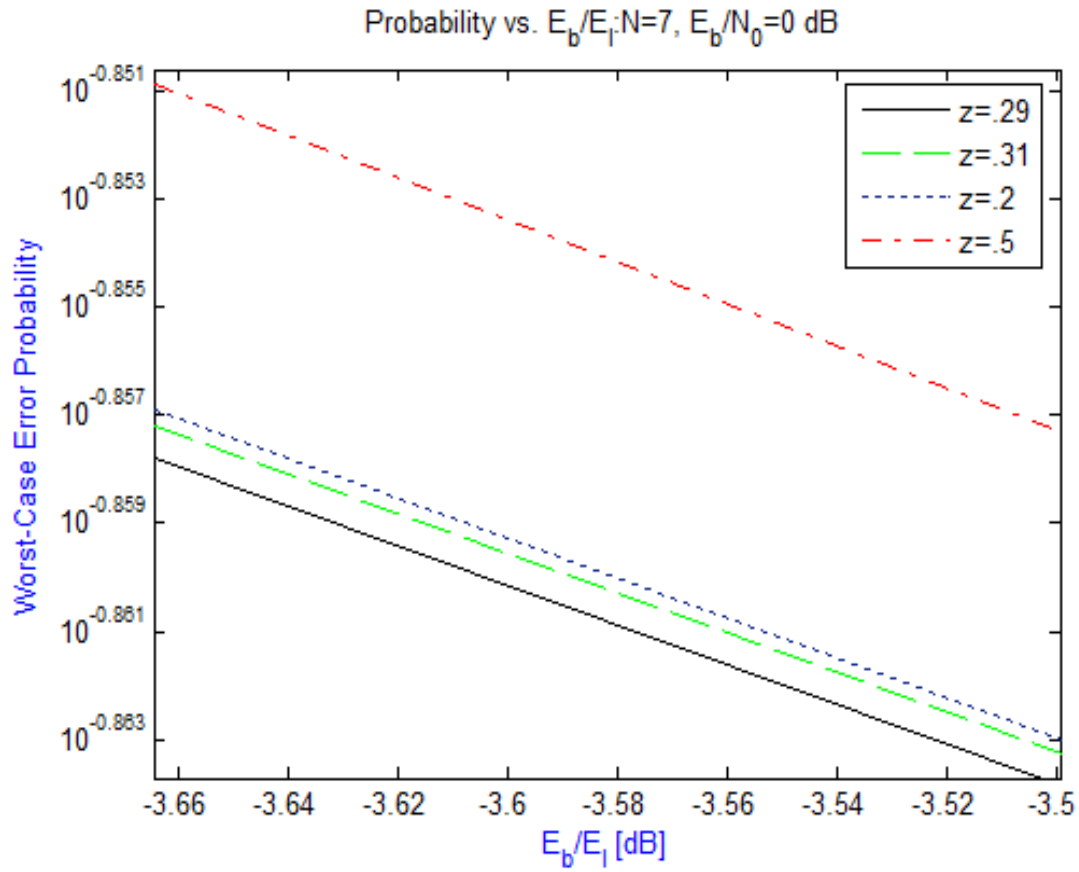


Figure 3.30: Probability vs. E_b/E_I for Various z Values: $N = 7$, $E_b/N_0 = 0$

Chapter 4

Geometric Approach to Finding the Optimal z Value

4.1 Introduction

In addition to investigating the bounds on worst-case error probability to find the optimal chipping level, another approach, independent of the actual regeneralized DSSS system, was taken. This approach is not concerned with AWGN levels, signal to noise ratios, or even error rates. Instead it focuses simply on geometry.

In [4] it has been proven that the best worst-case error performance under the channel conditions described in the previous chapter occur when the message signal is uniformly distributed over the surface of an N -dimensional sphere. Under this assumption, when using regeneralized DSSS which has 5 chipping levels, the best performance should occur when the chipping levels are chosen in such a way that the message vectors are uniformly distributed. The approach taken in this chapter is to

find the value of the chipping level, z , that causes the message vectors to be uniformly distributed over the surface of an N -dimensional sphere.

With regeneralized DSSS, each symbol will be broken up by a chipping sequence. This sequence can be thought of as a vector of length N . The elements of this vector will randomly take on values of 0, ± 1 , and $\pm z$. Each possible combination of N of these elements creates a vector in the N -dimensional signal space. When these vectors are normalized they can be thought of as lying on the surface of an N -dimensional sphere. The best possible worst-case error probability in our channel occurs with uniform distribution over the surface of an N -dimensional sphere. Since it is not possible to achieve completely uniform distribution with only a finite number of elements, 5 in this case, the best we can do is distribute the finite number of vectors as uniformly as possible. In this case, we will define uniform as all vectors being equidistant from their nearest neighbors in terms of Euclidean distance.

In order to determine which z value results in uniform distribution, a program was developed in Matlab. This program can be found in Appendix B. This program works by cycling through every possible vector combination for a given N and computing the total distance from a few different base vectors. This operation is performed for a range of z values. The total distance is then plotted against z . Where the plots for different base vectors intersect is where that z value has created uniform distribution. We will call the z value where this occurs $z_{uniform}$. This will be demonstrated in the next section.

4.2 Demonstration of How Uniform Distribution is Calculated

To further illustrate how $z_{uniform}$ is calculated, we will closely examine the case where $N=2$. Figure 4.1 shows the possible message vectors for $N=2$. Each vector is marked as a circle, and lies somewhere along the surface of a 1x1 square. In this case there are 16 total vectors. We can think of these vectors as being belonging to three distinct categories: corner vectors, edge vectors, and in between vectors. The elements of corner vectors will both be ± 1 's. Edge vectors will contain a 0 and a ± 1 . In between vectors will lie in between the edge and corner vectors and contain a ± 1 , and a $\pm z$. These vectors have been labelled. In the figure z is set to .5. The vectors (0,1), and (1,1) are marked with a dot and will be of greater importance later.

These signalling vectors must be normalized to unit energy. This can be visualized as moving the points onto the surface of a circle. Figure 4.2 shows how the vectors move from the surface of a square to a circle. The vectors marked by dots on the circle will be the vectors used to find uniform distribution.

The method used to find uniform distribution is to increment z by small steps, and to calculate the total Euclidean distance between all of the vectors at each increment. When the total distances from each point are the same, the vectors are uniformly distributed. Of the three vector types we defined earlier, the only one changing is the in between vectors. The corner and edge vectors will stay put. It can be seen from the symmetry of the vectors that the total distance will be the same from each corner vector. The same can be said for the edge vectors. Therefore, for the sake of

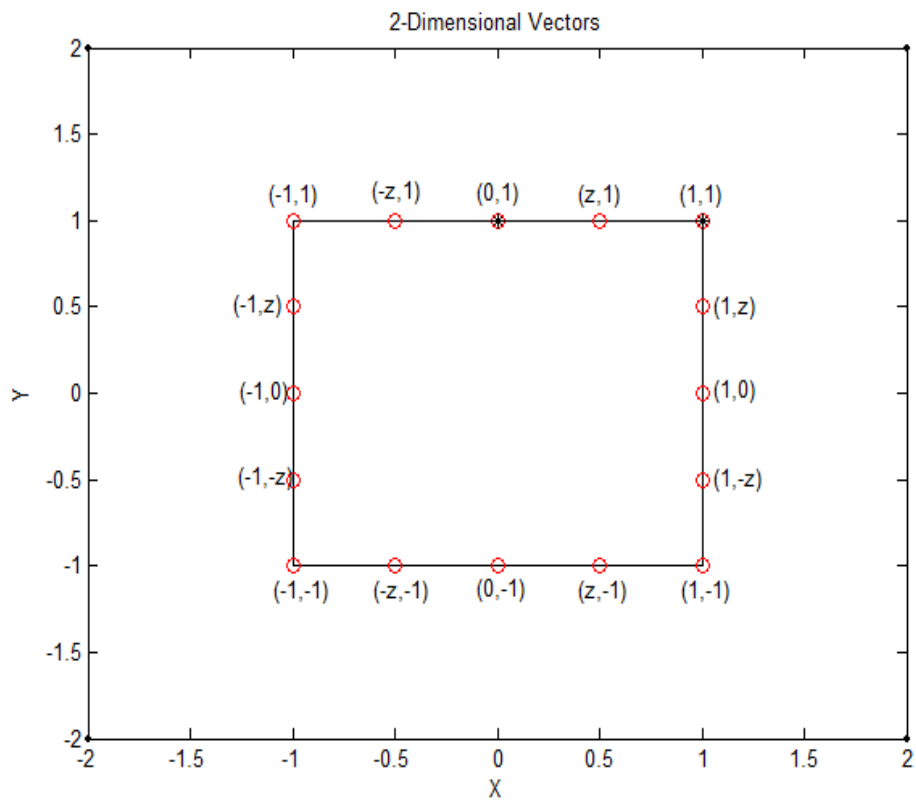


Figure 4.1: 2-Dimensional Signalling Vectors Prior to Normalization

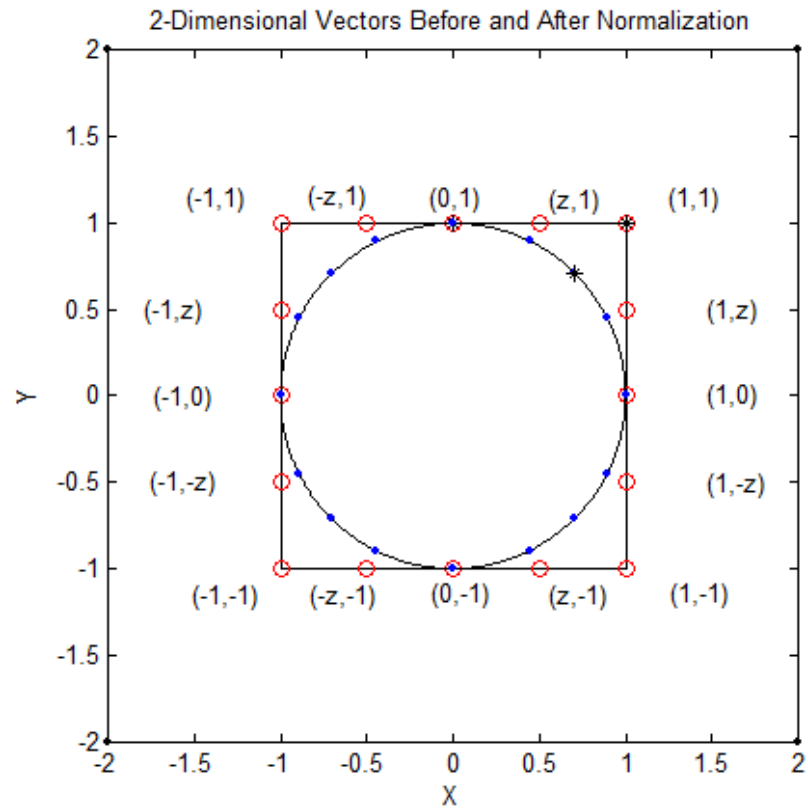


Figure 4.2: 2-Dimensional Signalling Vectors Before and After Normalization

simplifying the calculations, the total distance will only be measured from one edge vector, and one corner vector. These reference vectors will be the ones in the first quadrant. They are marked with a star in Figure 4.2. Figure 4.3 shows an example of the distance calculation with $z=.1$. The dashed lines show the distances from the corner vector, and the solid lines show the distance from the edge vector.

The Matlab program plots the total distance from each reference vector as a function of z . The results from $N=2$ are shown in Figure 4.4. Figure 4.5 shows a close up of the intersection point. The descending line represents the distance from the corner vector. As z is increased the in between vectors move towards the corner vectors, shortening the distance. The ascending line represents the distance from the edge vector. As z is increased, the in between vectors move away from the edge vector. The intersection eventually occurs with $z=.4142$. The total distance at the intersection is not relevant. All that matters is that they are equal.

If we look at the vector distribution when $z=.4142$, it appears to be uniform. Figure 4.6 shows this. This uniform distribution is also apparent when we examine the distance calculations shown in Figure 4.7. Since $N=2$ is a very simple case, it is also possible to confirm that $z=.4142$ leads to uniform distribution through a simpler method. If we think of uniformity in terms of angular separation around the circle, then since there are 16 vectors, 22.5 degrees of separation would be needed to separate each vector. Going back to the vector distribution along the square, it is easy to see that the z value that gives 22.5 degrees separation is $\tan 22.5$, or .4142.

With a few slight modifications, this technique for finding uniform distribution can be applied to higher values of N . In the $N=2$ case we had 2 vector types that did

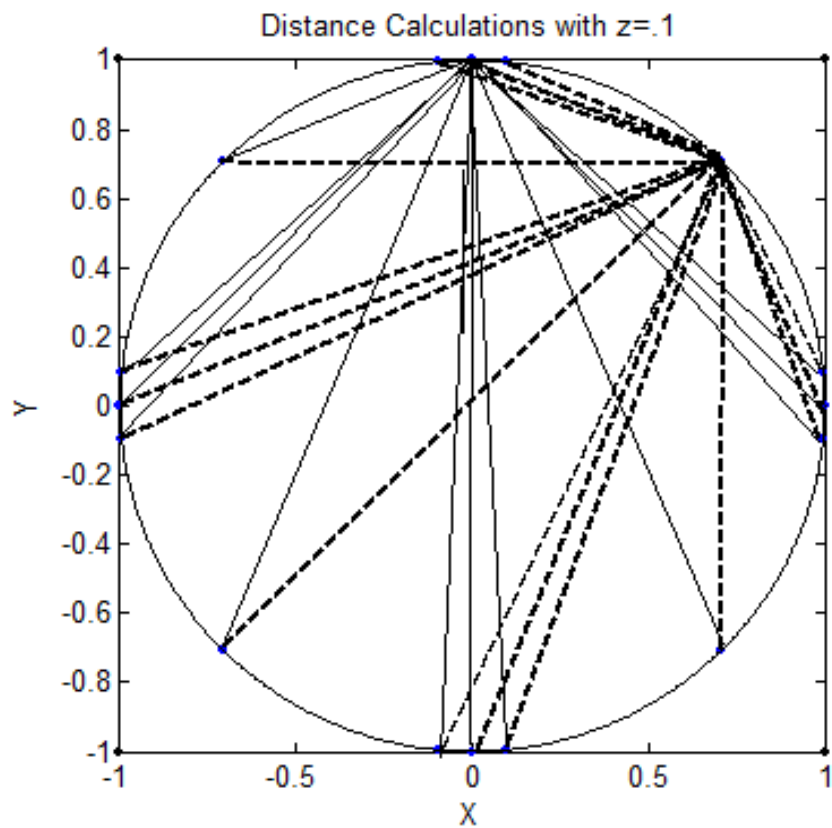


Figure 4.3: Distances From Edge and Corner Reference Vectors with $z = .1$

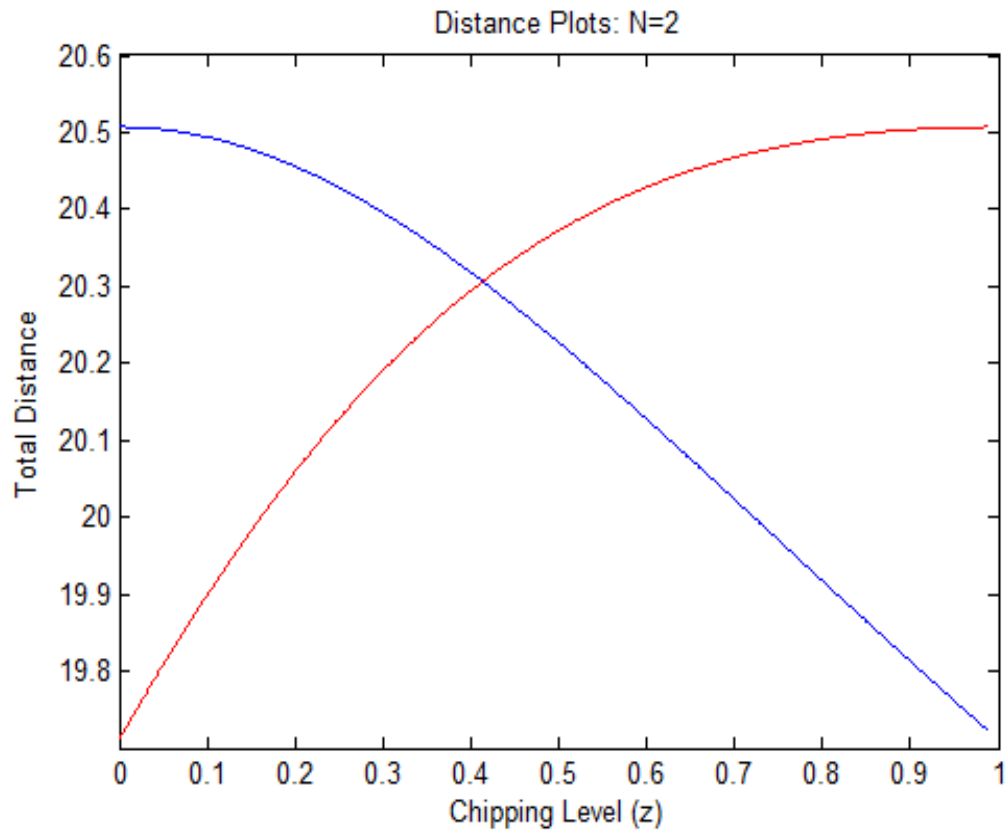


Figure 4.4: Intersections of Distance Plots for $N = 2$

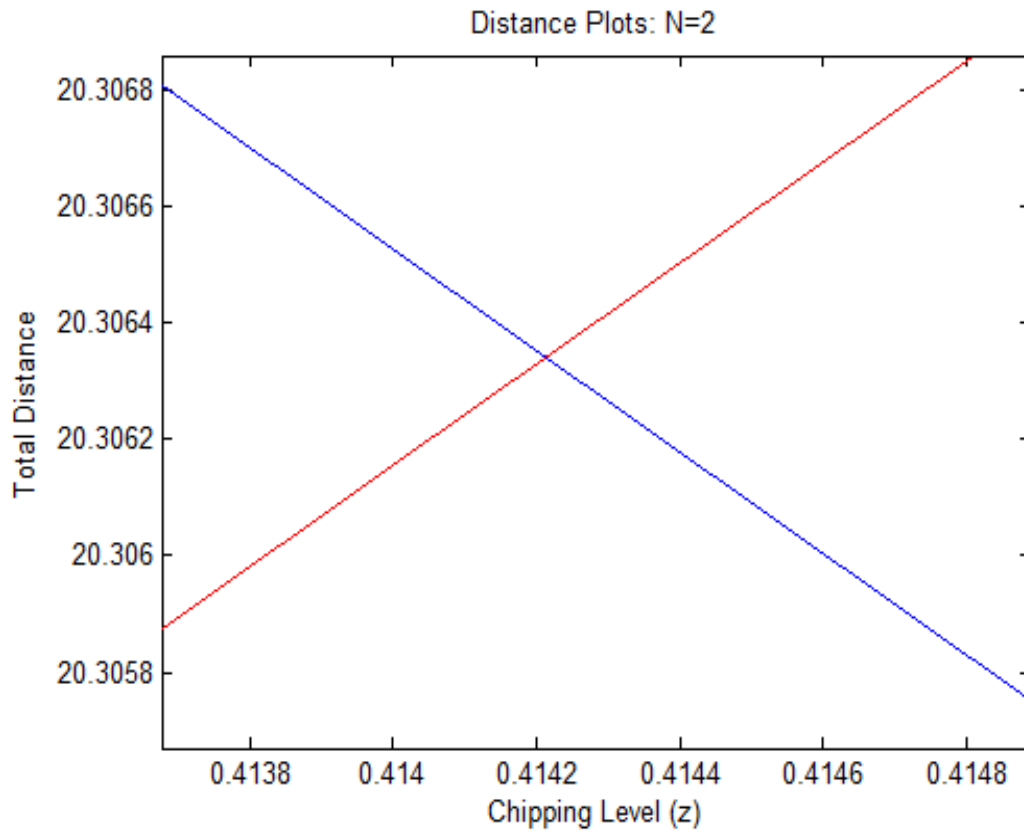


Figure 4.5: Close View of Intersections of Distance Plots for $N = 2$

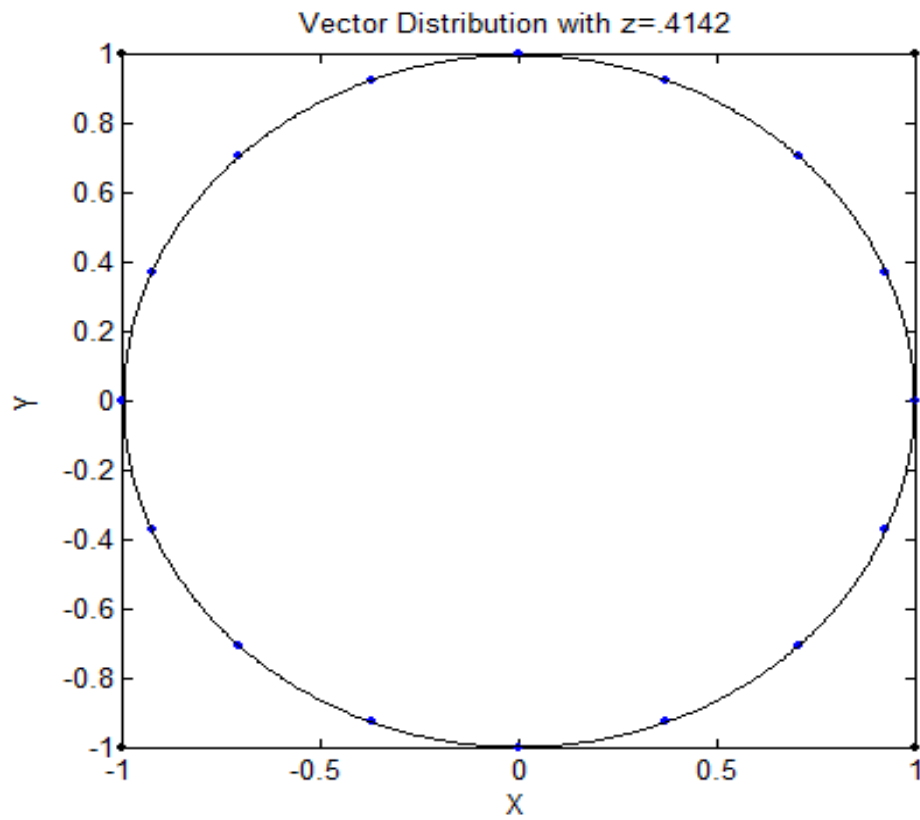


Figure 4.6: Uniform Vector Distribution When $z = .4142$

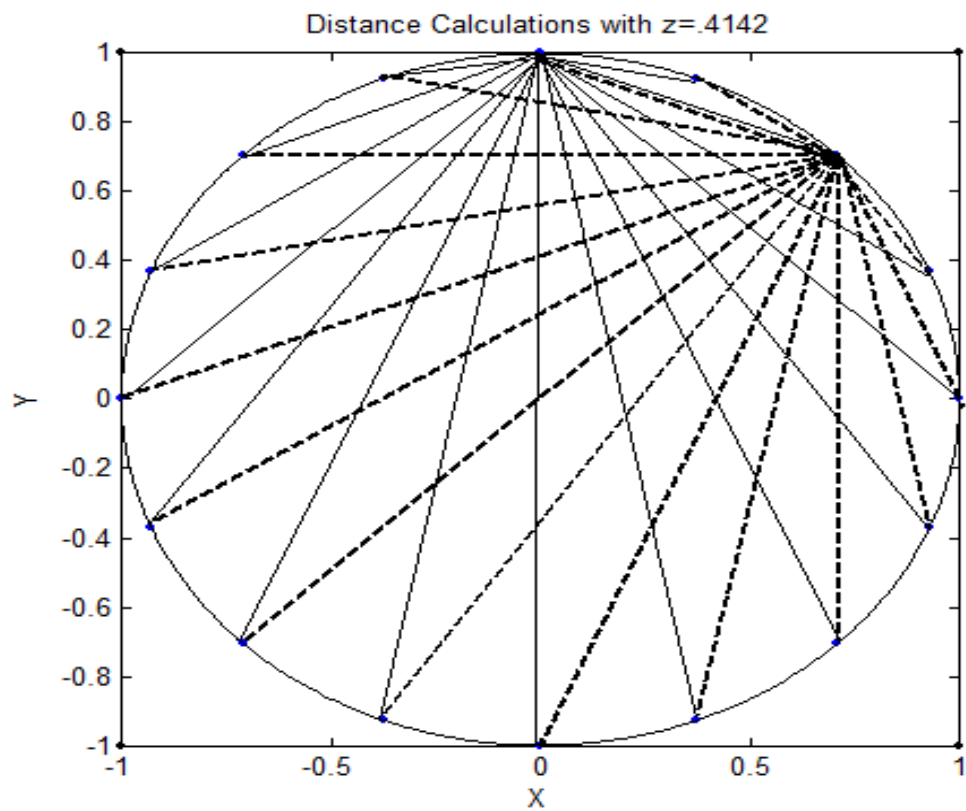


Figure 4.7: Distances From Edge and Corner Reference Vectors with $z = .4142$

not contain z as an element. These were the corner and edge vectors, which were used as reference points. In the $N=3$ case, one more such reference vector is needed. That is, the vector lying on the face of the cube. These reference vectors can be defined by the number of ± 1 's that they contain, also known as the Hamming weight. For instance, in the $N=3$ case, the corner reference vector must have a Hamming weight of 3, the edge vector a weight of 2, and the face vector a weight of 1. To generalize, for N dimensions, there will be N different reference vectors with Hamming weights from N down to 0, resulting in N distance plots.

In the following section we will examine the output of the Matlab program for various values of N . The resulting $z_{uniform}$ values will then be summarized. One thing to note here is the range for which the program can be run. As N increases, the number of possible vectors increases exponentially. There are $5^N - 3^N$ valid vectors for each N . The program must generate and work with all of these vectors for each value of z tested. The time needed to do so becomes increasingly long as N is increased. Therefore we will be limited to a maximum N value of 9.

4.3 Results

4.3.1 $N=3$

The results for $N=2$ were examined in the previous section, so we will move on to look at the results from $N=3$. The output of the Matlab program with $N=3$ is shown in Figure 4.8. This figure shows that the plots of total distance from each

of the three reference vectors do not intersect at exactly the same point, but very close to one another. A closer view of this is shown in Figure 4.9. From this figure it can be seen that the distance plots intersect one another at $z=.373$, $z=.384$, and $z=.4052$. The fact that there is not a single intersection point for all plots indicates that uniform distribution of the vectors is not possible under our conditions. This is because, as explained in the previous section, the program works by increasing z by small increments and measuring the total distance from the reference vectors. By doing this, we are moving all vectors containing z simultaneously, while the vectors not containing a z level remain put. Each vector only moves along a one dimensional path. Exact uniform distribution of the vectors is not possible along this path, and may not be possible at all. However, the close proximity of the distance plot intersections indicates that the vectors are very close to being uniformly distributed at that point. An exact $z_{uniform}$ does not exist, but there is a small range of z over which the vector distribution is very close to being uniform.

4.3.2 $N=4$

The output was examined next for $N=4$. This is shown in Figure 4.10. A close view of the intersections is shown in Figure 4.11.

Here we can see again that there is not a common intersection point for the four lines, but all of the intersections are close together. The intersections also occur at a smaller value of z than with lower N values. The range of intersection points stretches from around $z=.34$ to $z=.39$.

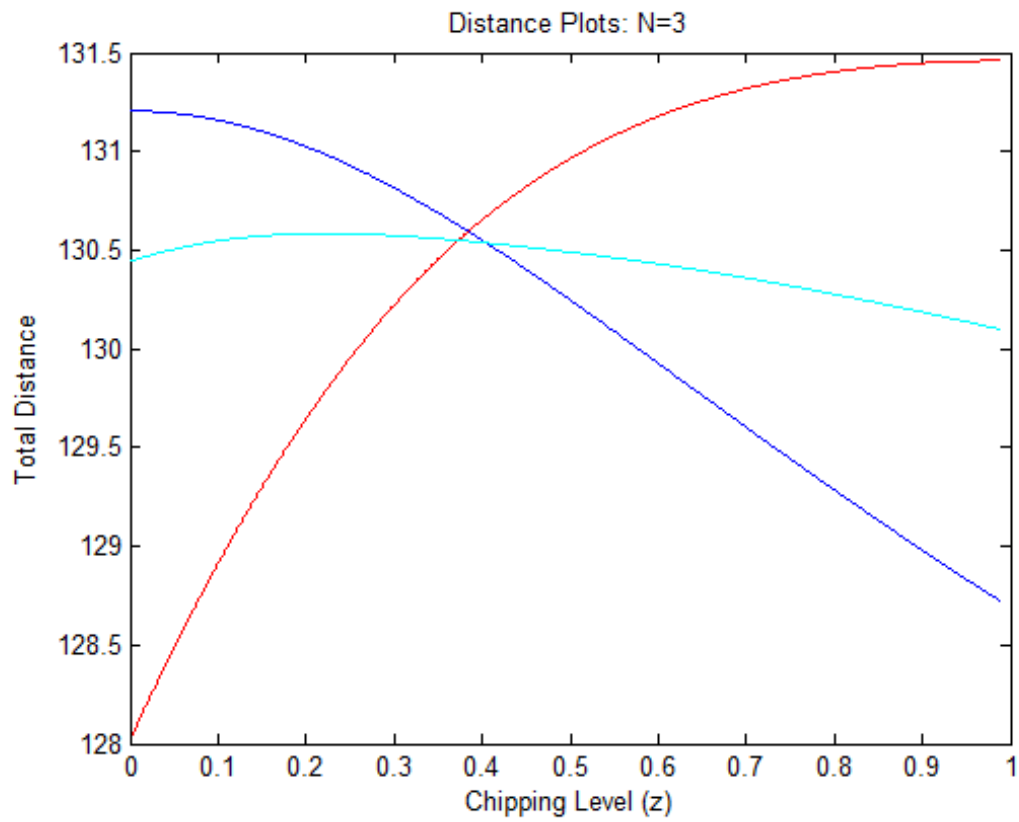


Figure 4.8: Intersections of Distance Plots for $N = 3$

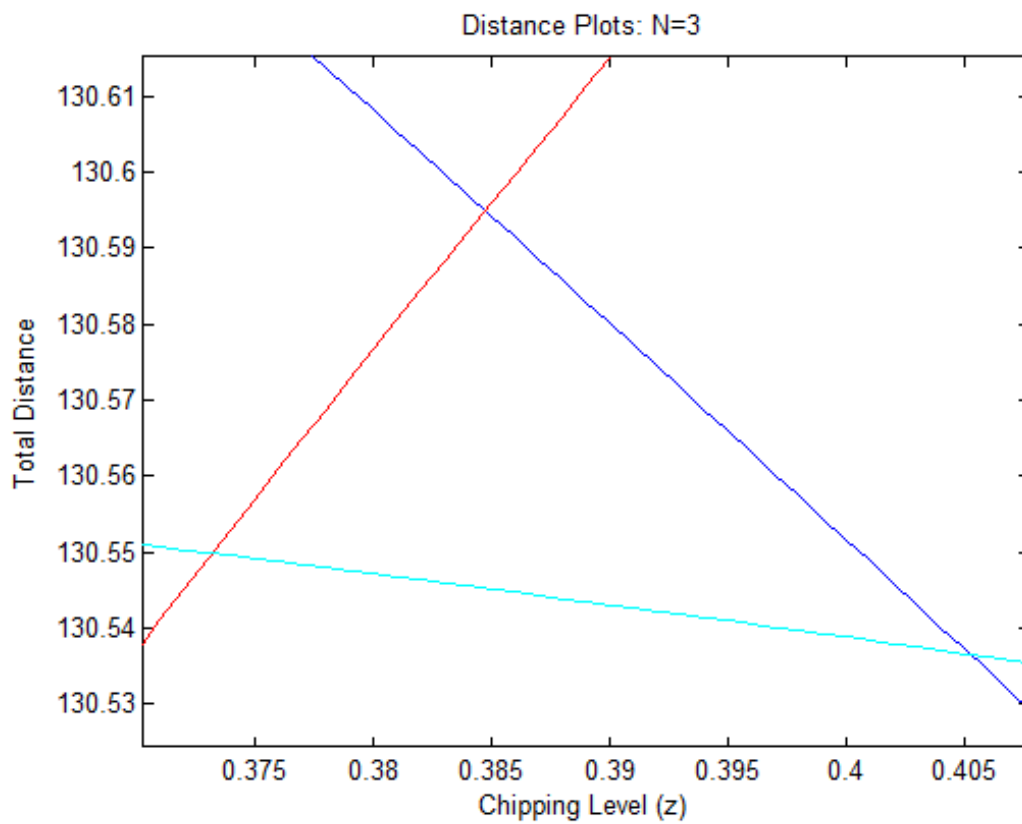


Figure 4.9: Close View of Intersections of Distance Plots for $N = 3$

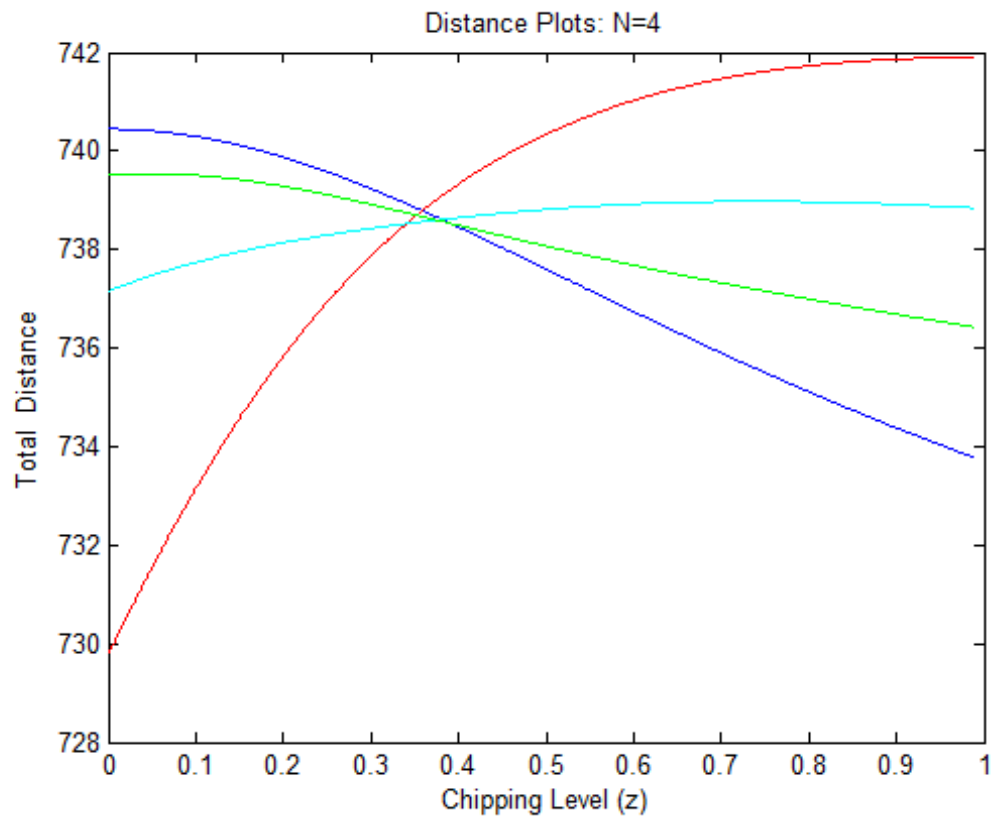


Figure 4.10: Intersections of Distance Plots for $N = 4$

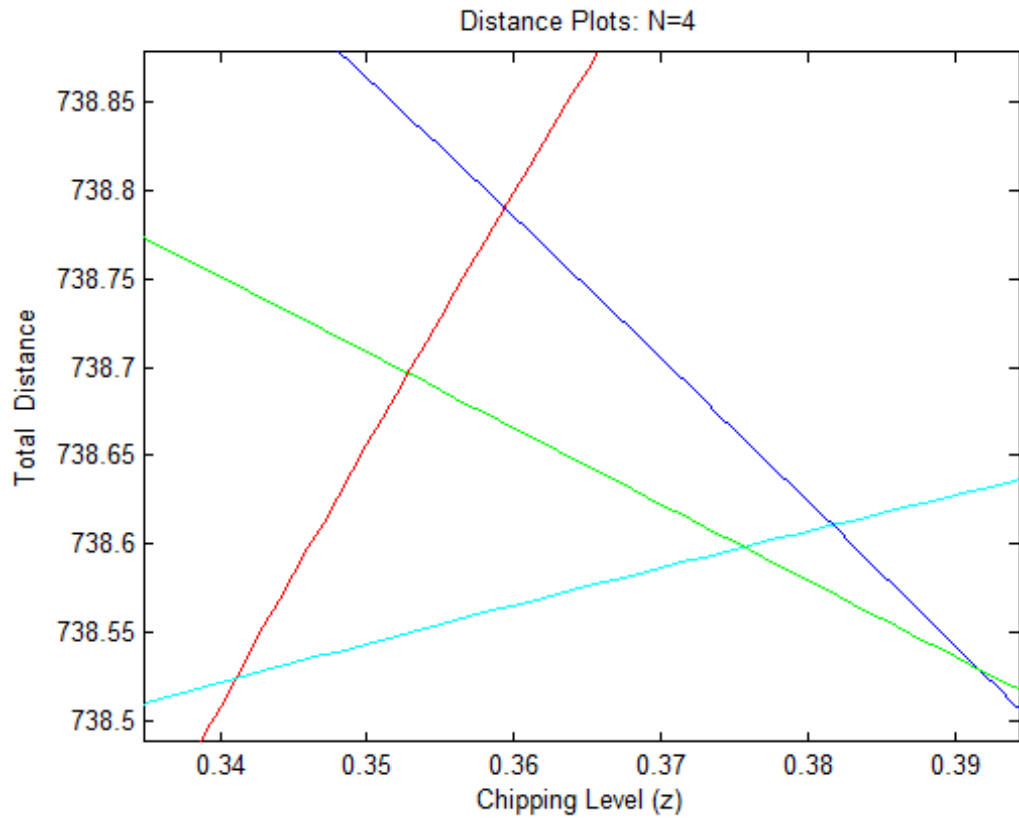


Figure 4.11: Close View of Intersections of Distance Plots for $N = 4$

4.3.3 $N=6$

The results will now be examined for $N=6$. Figure 4.12 shows the distance plots for $N=6$. Figure 4.13 shows a close up of the intersection points.

The trend of multiple intersection points, and a lower z value where the intersections occur, continues with $N=6$. The intersections take place over a range from $z=.29$ to around $z=.355$.

4.3.4 $N=9$

The highest value for N that can be processed by the program in a reasonable amount of time is 9. The results from running the program with $N=9$ are shown in Figure 4.14. Figure 4.15 shows a close up of the intersection points.

Here, again, we see that the intersection points are occurring at lower values of z . In this case they range from around $z=.23$, to $z=.3$. The intersection points are again extremely clustered at the higher values of z .

4.3.5 Summary of Results

A chart summarizing the results of the geometric analysis is shown in Figure 4.16. It is evident that from the chart that as N increases, the intersection points occur at smaller z values. The range over which these intersection points occur becomes slightly larger. Figure 4.17 gives a graphical view of these trends.

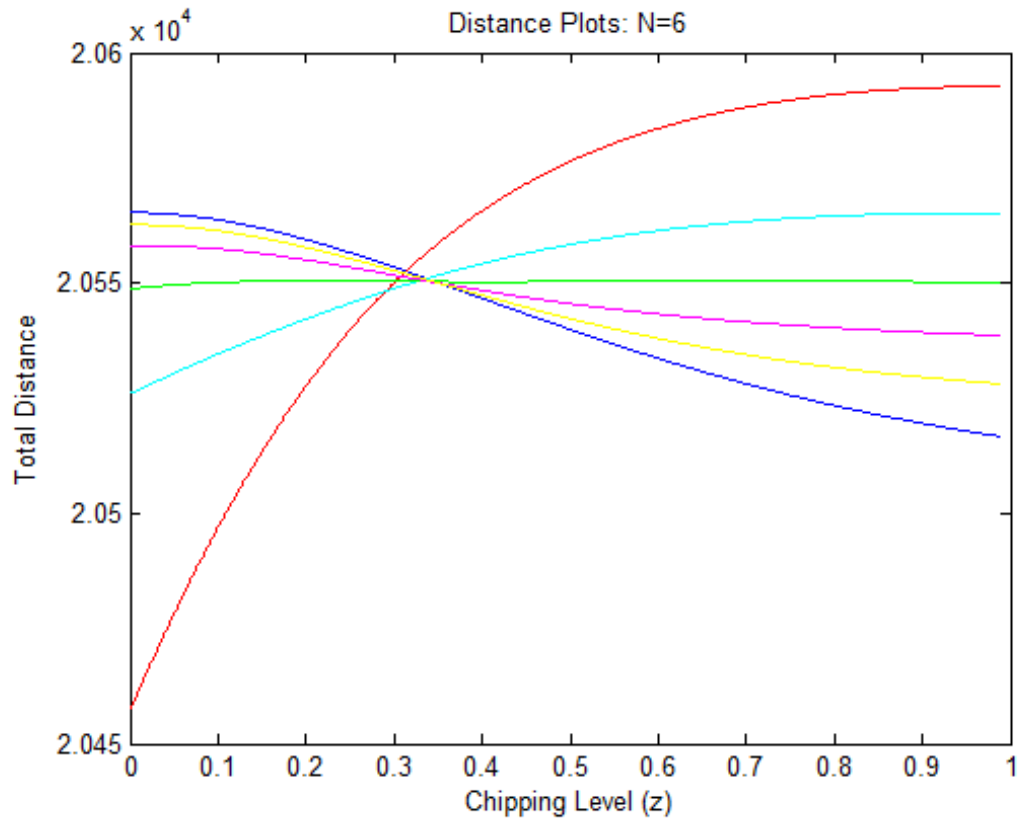


Figure 4.12: Intersections of Distance Plots for $N = 6$

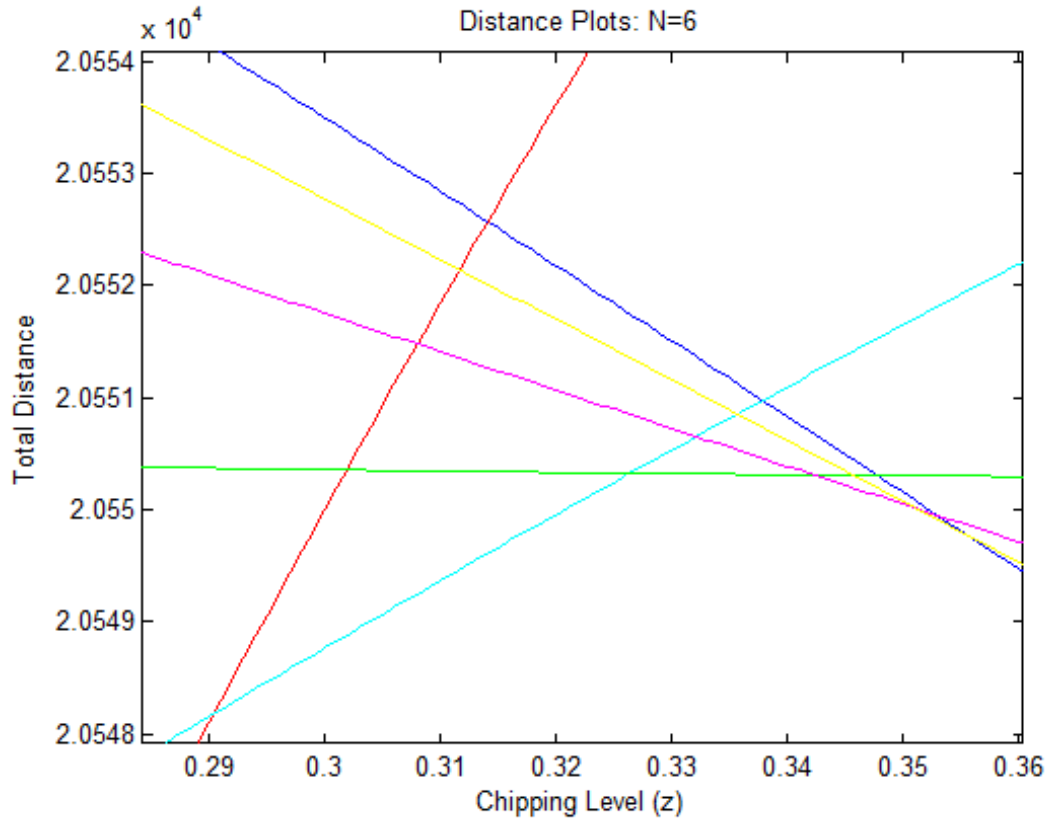


Figure 4.13: Close View of Intersections of Distance Plots for N=6

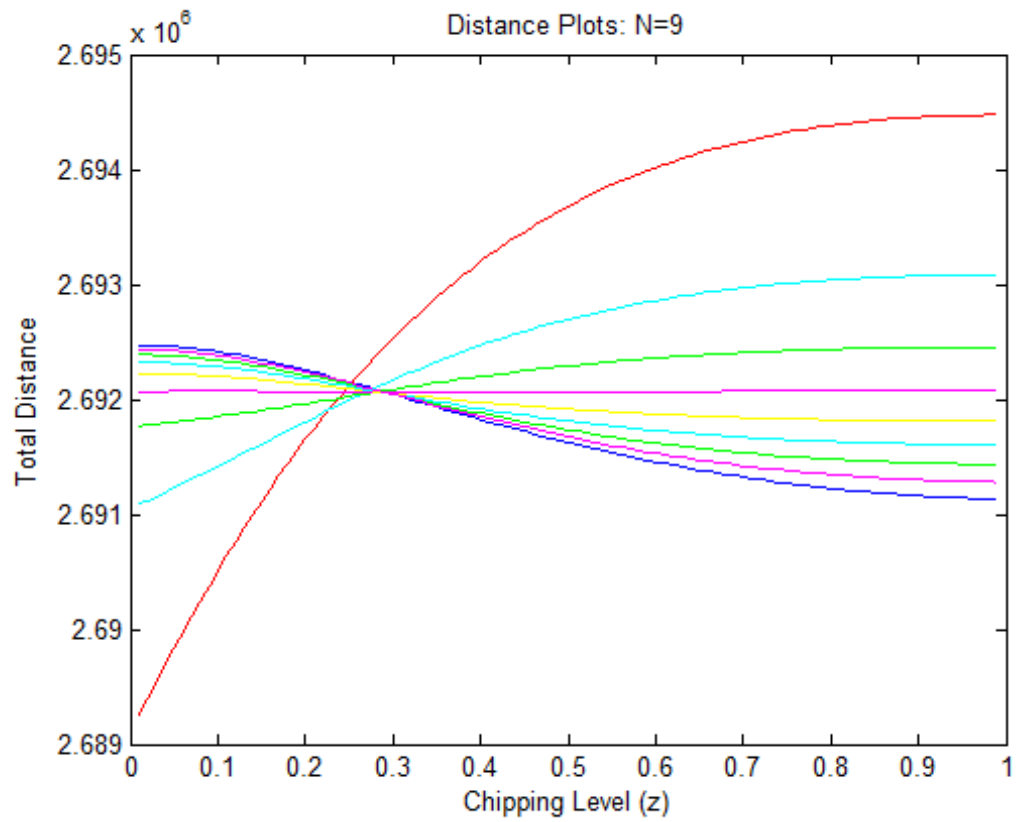


Figure 4.14: Intersections of Distance Plots for $N = 9$

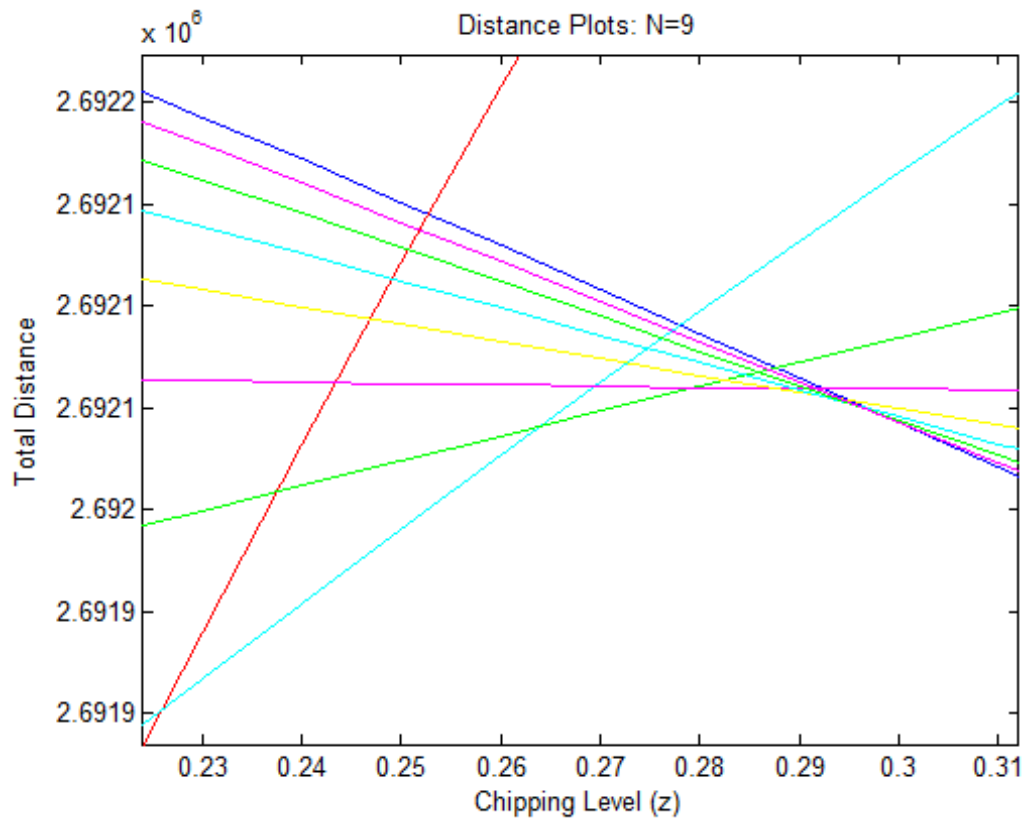


Figure 4.15: Close View of Intersections of Distance Plots for $N = 9$

N	First Intersection	Last Intersection
2	0.4142	0.4142
3	0.3732	0.4052
4	0.341	0.3916
5	0.3143	0.3748
6	0.2904	0.3561
7	0.2681	0.3365
8	0.2467	0.3171
9	0.2259	0.2983

Figure 4.16: First and Last Intersection Points of Distance Plots for Various N

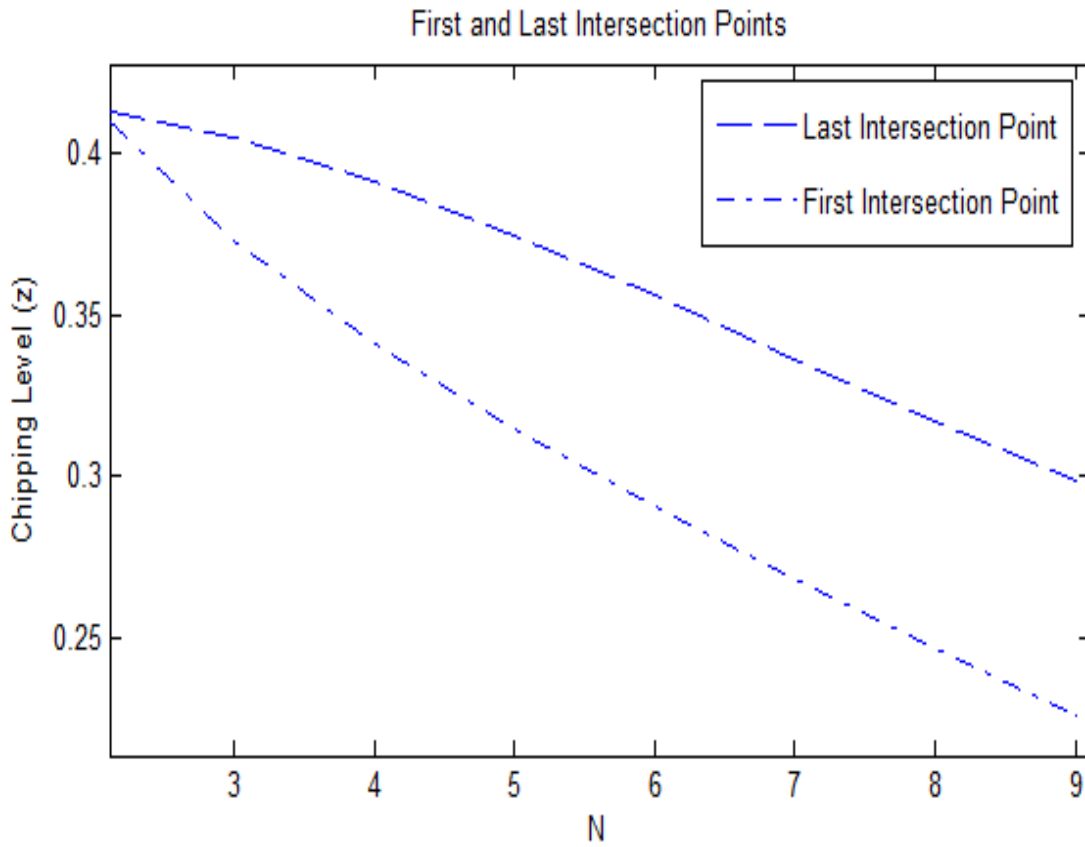


Figure 4.17: Plot of First and Last Intersection Plots vs. N

Chapter 5

Results and Conclusion

The aim of this thesis was to analyze the effect of altering the z -parameter on the worst-case error probability of regeneralized DSSS, and attempt to optimize this parameter. This was accomplished by looking at both the bounds on worst-case error probability, and the geometry of the system. This chapter will examine and compare the results of both approaches.

5.1 Comparison of Results

When examining the bounds on worst-case error probability, it was soon discovered that the upper bound was not sensitive to changes in z in a manner useful to our research. However, the lower bound was. It was found that, within an appropriate range of signal to noise, and signal to interference ratios, the optimal z value stayed consistent for a given N . As N was increased, the optimal z value decreased. This analysis was possible for values of N ranging from 3 to 20.

The second approach was based on geometry. According to [4], the system should be optimized under our channel conditions when the signalling vectors are uniformly distributed over the surface of an N dimensional sphere. A Matlab program was created to find this uniform distribution for a given N . It was found that under the constraints of the regeneralized DSSS signalling scheme, perfect uniform distribution was not possible for any N larger than 2. It was possible, however, to get very close to uniform distribution. For N larger than 2, a small range was found over which the vector distribution was nearly uniform. As N was increased, this range was became lower. The calculation of the vector distribution becomes exponentially larger as N is increased. Therefore the largest N that could be calculated was 9.

The two approaches to finding a z value that would optimize worst-case error performance were completely independent of one another. One relied on the bounds on an equation developed based on probability theory and the physics behind the system. The other was based purely on the geometry of the system.. The bound based approach yielded solid numbers for optimal z . The geometry based approach yielded small ranges of z over which the message vectors were very near uniform distribution. The results from both approaches are compared in Figure 5.1.

The results from the two independent approaches compare favorably. The optimal z values found using the lower bound fit within the range of intersections of the distance plots found through the geometric approach. The plots all seem to be fairly linear and share approximately the same slope. This can be seen in the extended comparison plot shown in Figure 5.2.

One curiosity that stems from this comparison is the fact that the optimal z values

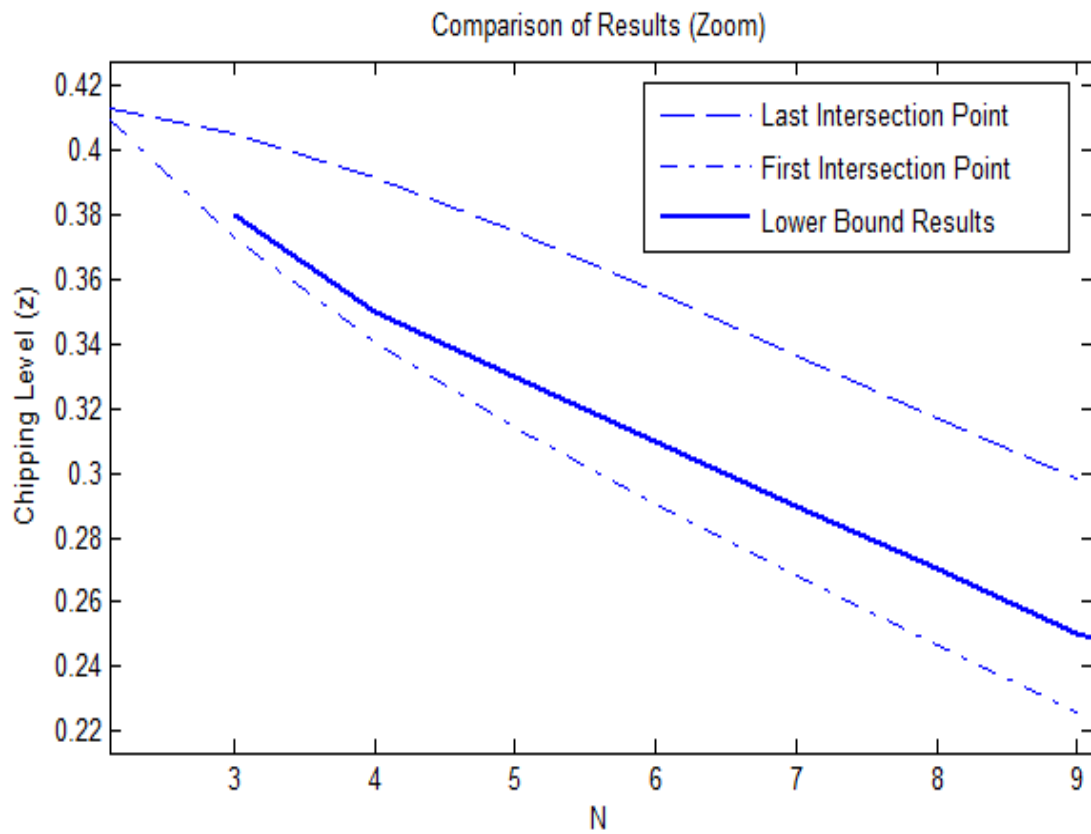


Figure 5.1: Comparison of Lower Bound Optimal z with Range of Intersection Points of Distance Plots

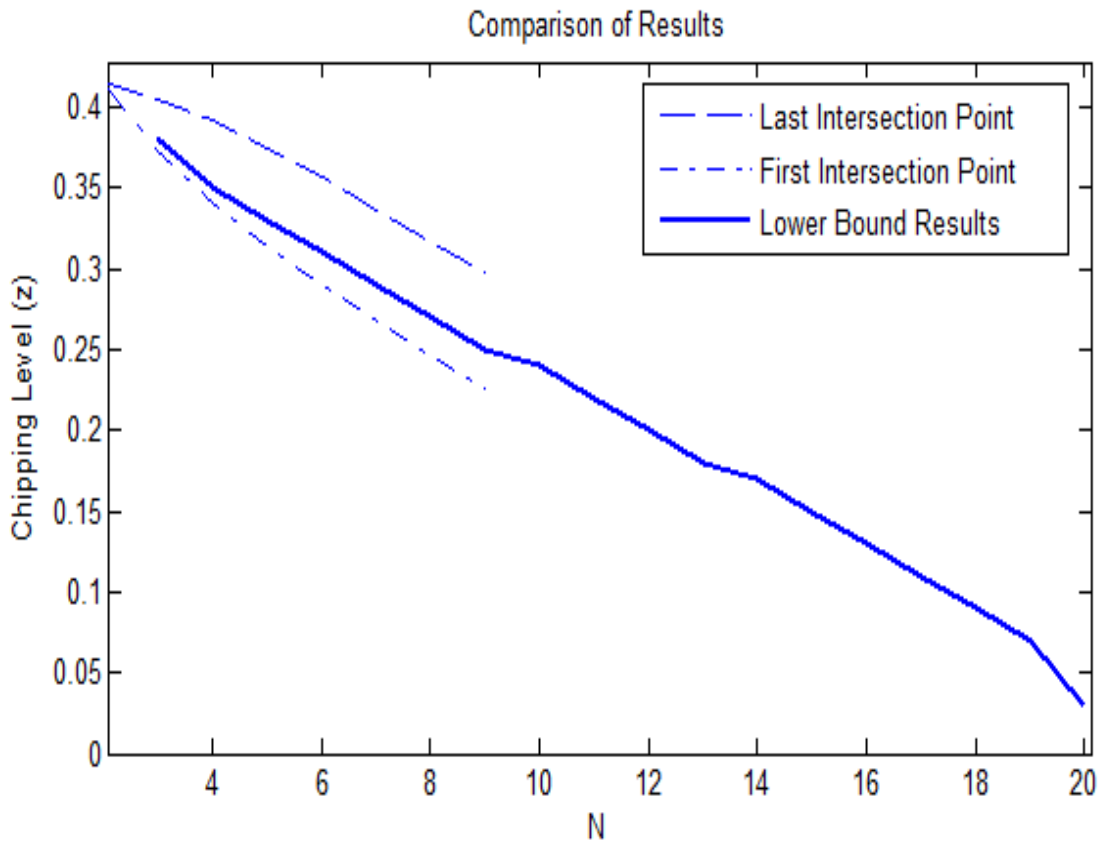


Figure 5.2: Extended Comparison of Bound Based and Geometric Optimal z Results

found using the bound based approach are slightly closer to the first intersection point than the last intersection point. In the distance plots from chapter 4, the intersection points seemed to cluster around the higher end of the range. Overall however, the results show that an optimal z value can be found by evaluating the lower bound on worst-case error probability, and that these optimal z values coincide with the range of found to distribute the message vectors nearly uniformly..

5.2 Suggestion for Further Research

This research leaves a few openings for future work. Consider the fact that we never examined the actual worst-case error probability of the system, only the bounds. One possibility for future research would be to find a closed form solution to the worst-case error probability, and use that to find the optimal chipping level. This has been attempted however, and became prohibitively complex. Another option would be to directly simulate the system instead. Using either of these methods would allow some realistic quantifiable measurements to be made. This thesis found optimal z values, but in order to do so, large amounts of thermal noise were added to the simulation. Plots of these results clearly show an optimal z value, but with all of the noise needed for the simulation to be possible, the actual level improvement is not really useful.

Another possibility would be to explore a 7 level DSSS scheme. This would involve a system similar to regeneralized DSSS, but with two signalling levels between 0 and 1. Bounds on the worst-case error probability would need to be derived, and a similar analysis on the chipping levels would be needed.

Finally, a third possibility would be to attempt to push the limits of the research in this thesis. The lower bound equation could only be examined up to $N=20$, and the geometric analysis only to $N=9$, due to computational limits. It might be possible to push these limits further through the use of a more efficient algorithm, or a very powerful computer. The results would be interesting because it seemed as if z was approaching 0 as N was increased.

REFERENCES

- [1]. M.K. Simon, J.K. Omura, R.A. Scholtz, and B.K. Levitt, *Spread Spectrum Communications Handbook*, McGraw Hill, NY 1994
- [2]. Bernard Sklar. *Digital Communications*, Second Edition, Prentice Hall PTR, 2001
- [3]. Haykin, Simon, *Communication Systems, Fourth Edition*, John Wiley and Sons, Inc., NY 2001
- [4]. B. Hughes, M. Hizlan. "An Asymptotically Optimal Random Modem and Detector for Robust Communication" *IEEE Transactions on Information Theory*, VOL. 36 NO. 4, July 1990.
- [5]. M. Hizlan. "A Generalization of Direct-Sequence Spread-Spectrum" Department of Electrical and Computer Engineering, Cleveland State University, Cleveland, OH.
- [6]. R. Kalakuntla. "Further Generalization of the Unique Spreading Sequence in Generalized Direct-Sequence Spread-Spectrum" Department of Electrical and Computer Engineering, Cleveland State University, Cleveland, OH.
- [7]. M. Hizlan, B. Hughes. "On the Optimality of Direct Sequence for Arbitrary Interference Rejection" *IEEE Transactions on Communications*, VOL. 39, NO. 8, August 1991.

Appendix A

Matlab Script for Evaluating Lower Bound

```
%This program will compute the lower bound for the worst-case error  
%probability for 5-level DSSS over a range of z values and signal to  
%interference ratios
```

```
clear all;close all;
```

```
tic;
```

```
N=3;          %set number of chips per bit
```

```
sndb=-10;    %set AWGN level (in dB)
```

```
z_step=.001; %set z step size
```

```
sidb_step=1; %set signal to interference ratio step size
```

```
sidb_max=10; %set max and min signal to interference ratio in dB
```

```
sidb_min=-10;
```

```

z_plot=[0:z_step:1];

sidb_plot=[sidb_min:sidb_step:sidb_max];

for sidb_level=sidb_min:sidb_step:sidb_max;

    index=1;

    for z=0:z_step:1;

        sn=10^(sndb/10);

        y=[];

        y1=[];

        % compute combination function for all values of N and saves in a
        % matrix for future use

        chs=zeros(N+1,N+1);

        for i=1:N

            chs(i,1)=1;

            for j=1:i-1

                chs(i,j+1)=chs(i,j)*(i-j)/j;

            end

        end

    end

    %-----

    for sidb=sidb_level

        yvec=[];

        si=10^(sidb/10);

        %summation over all variables

```

```

for l=1:N
    s=0;
    for k=1:N
        for r=0:N-k
            for a=0:min(k,l)
                for b=max(0,k+l-N-a):min(l,k)-a
                    for c=0:min(r,l-a-b)
                        for d=max(0,r+k+l-N-a-b-c):min(r,l-a-b)-c

s=s+0.5*erfc((sqrt(sn))*(1-(a-b+z*(c-d))/sqrt(l*si*(k+r*z^2))))*chs(l+1,a+b+1)
*chs(N-l+1,k-a-b+1)*chs(a+b+1,a+1)*chs(l-a-b+1,c+d+1)*chs(N-l-k+a+b+1,r-c-d+1)
*chs(c+d+1,c+1)*(2^(k+r-a-b-c-d)/(5^N-3^N));

                        end
                    end
                end
            end
        end
    end
end

%-----
%maximise s over l
yvec=[yvec;s];
end

```

```

        y1=[y1;max(yvec)];

    end

    %-----

    optimal(index)=y1;

    index=index+1;

end

figure(sidb+(sidb_min-2*sldb_min+1));plot(z_plot,optimal);

title(['N=',num2str(N), '      E_b/E_I [dB]=',num2str(sidb) , '
E_b/N_0 [dB]',num2str(sndb)]);

xlabel('Chipping Level (Z)');

ylabel('Worst-Case Error Probability');

min_error_prob(sidb+(sidb_min-2*sldb_min+1))=min(optimal);

[r_min c_min]=min(optimal);

z_min(sidb+(sidb_min-2*sldb_min+1))=z_plot(c_min);

end

figure(98);plot(sidb_plot,z_min);

title(['Minimizing Chipping Levels      N=',num2str(N), '
      E_b/N_0 [dB]=',num2str(sndb)]);

```

```
xlabel('E_b/E_I [dB]');  
ylabel('Optimal Chipping Level (Z)');  
  
figure(99);semilogy(sidb_plot,min_error_prob);  
  
title(['Minimal Worst-Case Error Probability    N=',num2str(N), '  
      E_b/N_0 [dB]=',num2str(sndb)]);  
  
xlabel('E_b/E_I [dB]');  
ylabel('Worst-Case Error Probability');  
  
z_min  
  
toc
```


Appendix B

Matlab Script for Geometric Analysis

```
%base 5 counter to generate complete set of vectors

%This program will evaluate the distance properties of an N-dimensional
%sphere by finding the total distance between a reference point and all
%other vectors. To run, change n to an appropriate value and comment out all
%portions and loops for higher n. 9 is the maximum n that can be processed

clear all; close all;

tic;

vec_check=[];

n=9;           %set n

counter=0;    %initialize counter

zz=1;
```

114

```
dist1_total1(zz)=0;

step=.001;      %set z step

for z=step:step:.99;    %set z range

    dist1_total(zz)=0; %initialize distance totals

    dist2_total(zz)=0;

    dist3_total(zz)=0;

    dist4_total(zz)=0;

    dist5_total(zz)=0;

    dist6_total(zz)=0;

    dist7_total(zz)=0;

    dist8_total(zz)=0;

    dist9_total(zz)=0;

    dist10_total(zz)=0;

    %%%%for loops for counter. 1 per n

    for i=1:5;

        for ii=1:5;

            for iii=1:5;

                for iiii=1:5;

                    for iiii=1:5;

                        for iiiiii=1:5;

                            for iiiiii=1:5;
```

```

for iiiiiii=1:5;
    for iiiiiii=1:5;
        for iiiiiii=1:5;
%begin individual operations on vectors
vec=[i ii iii ivv iiii iiii iiii
iiiiii iiiiiii iiiiiiiiii iiiiiiiiii];
        pass=0;
vec_geo=[];
for j=1:n
    if vec(j)==1
        vec_g=-1;
    elseif vec(j)==2
        vec_g=-z;
    elseif vec(j)==3
        vec_g=0;
    elseif vec(j)==4
        vec_g=z;
    elseif vec(j)==5
        vec_g=1;
    end
    vec_geo=cat(2,vec_geo,vec_g);
end
for jj=1:n;

```

```

        if vec_geo(jj)==1 || vec_geo(jj)==-1
            pass=1;
        end
    end
end
if pass;
    %vec_check=cat(1,vec_check,vec_geo);
    %counter=counter+1;
    vec_norm=vec_geo/(sqrt(sum(vec_geo.^2))); %Normalize
    %Begin distance calculations

ref_point1=ones(1,n)*(1/sqrt(n));
dist1=(sqrt(sum(((ref_point1-vec_norm).^2)'))));
dist1_total(zz)=dist1_total(zz)+dist1;

ref_point2=cat(2,ones(1,1),zeros(1,n-1));
dist2=(sqrt(sum(((ref_point2-vec_norm).^2)'))));
dist2_total(zz)=dist2_total(zz)+dist2;

ref_point3=cat(2,ones(1,2),zeros(1,n-2))*(1/sqrt(2));
dist3=(sqrt(sum(((ref_point3-vec_norm).^2)'))));
dist3_total(zz)=dist3_total(zz)+dist3;

ref_point4=cat(2,ones(1,3),zeros(1,n-3))*(1/sqrt(3));

```

```
dist4=(sqrt(sum(((ref_point4-vec_norm).^2)'))));
dist4_total(zz)=dist4_total(zz)+dist4;

ref_point5=cat(2,ones(1,4),zeros(1,n-4))*(1/sqrt(4));
dist5=(sqrt(sum(((ref_point5-vec_norm).^2)'))));
dist5_total(zz)=dist5_total(zz)+dist5;

ref_point6=cat(2,ones(1,5),zeros(1,n-5))*(1/sqrt(5));
dist6=(sqrt(sum(((ref_point6-vec_norm).^2)'))));
dist6_total(zz)=dist6_total(zz)+dist6;

ref_point7=cat(2,ones(1,6),zeros(1,n-6))*(1/sqrt(6));
dist7=(sqrt(sum(((ref_point7-vec_norm).^2)'))));
dist7_total(zz)=dist7_total(zz)+dist7;

ref_point8=cat(2,ones(1,7),zeros(1,n-7))*(1/sqrt(7));
dist8=(sqrt(sum(((ref_point8-vec_norm).^2)'))));
dist8_total(zz)=dist8_total(zz)+dist8;

ref_point9=cat(2,ones(1,8),zeros(1,n-8))*(1/sqrt(8));
dist9=(sqrt(sum(((ref_point9-vec_norm).^2)'))));
dist9_total(zz)=dist9_total(zz)+dist9;
```



```
plot(step:step:.99,dist5_total,'m');  
plot(step:step:.99,dist6_total,'y');  
plot(step:step:.99,dist7_total,'c');  
plot(step:step:.99,dist8_total,'g');  
plot(step:step:.99,dist9_total,'m');  
plot(step:step:.99,dist10_total,'y');  
  
toc
```