On the Selection of Simulation Methods for Implementing Wireless Channel Models

by

Faith Morgan

A Thesis
Submitted to the Faculty
of the
WORCESTER POLYTECHNIC INSTITUTE
In partial fulfillment of the requirements for the
Degree of Master of Science
in
Electrical and Computer Engineering
by

Faith Morgan

January 2023

APPROVED:

_____

Professor Alexander Wyglinski, Major Advisor

_____

Professor Reinhold Ludwig

_____

Professor Seyed Zekavat

Abstract

In this thesis, we explore computer simulation of communication systems, specifically the selection of effective simulation methods for wireless channels. Computer simulation of receiver behavior in various channels is important for understanding the performance of a given system, and if it will meet the necessary reliability requirements. However, some channel simulations are more difficult to implement in software than others. This thesis addresses the methods which can be used to implement these simulations based on the channel being implemented. We begin by establishing a useful definition for simulation difficulty and consistent language with which we can discuss simulation methods and difficulty across different channels. We then analyze a set of channels and identify correlations between the general characteristics of channels and their corresponding simulation methods. Based on this analysis, we obtain a set of complexity characteristics and create a framework which can be used to guide the selection of simulation methods, identify simulation challenges early on, and identify what would need to be altered to mitigate these challenges.

Acknowledgements

I would like to express my deepest gratitude to my advisor Professor Alex Wyglinski for his guidance and support throughout both my undergraduate and graduate studies. I would like to thank Professor Reinhold Ludwig and Professor Seyed (Reza) Zekavat for their comments and suggestions while serving on my committee. I would also like to thank my husband, Seth, for his continual encouragement and his contributions during the revision process of this thesis. Finally, I would like to thank The MITRE Corporation for sponsoring my graduate studies and the initial phase of this research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Computer simulation of communication system receiver behavior in various channels is important for understanding the performance of a given system, and if it will meet the necessary reliability requirements. This research will focus on simulations following a Monte Carlo approach [1]. For some channels, the standard method of simulation is simple and effective. Take, for example, the estimation of the Bit Error Rate (BER) curve of a given system in an additive white Gaussian noise (AWGN) channel [2]. This is a relatively easy simulation to create and run, and does not typically take a long time. Simply choose a range of noise levels of interest and test uniformly spaced noise values along that range by generating AWGN samples for each of those values, adding them to the signal, and checking the bits at the output. This process can be repeated and averaged as many times as necessary to obtain the desired curve smoothness and can be completed in a reasonably short amount of time on a modern processor.

However, there are also channels for which the selection of simulation methods and the implementation of the simulation are more difficult. One example is a channel which has multiple performance parameters. Say we want to model a simplified Doppler rate of change effect, sometimes called Doppler rate or Doppler frequency rate [3], a basic Doppler shift over time based on velocity and acceleration, between two vehicles communicating with each other. Since the vehicles are both moving

across 3D space, the paths which they can take with respect to each other are nearly infinite. In addition, their speed and acceleration are variable. Despite the fact that the actual model for a single Doppler shift is relatively simple, this type of channel simulation presents a potential problem with simulation time constraints due to its unbounded and multidimensional nature.

Another example of a potentially challenging channel to simulate is one where there are "holes" in the performance curve, specific regions where the performance is worse. When the performance of a receiver is still unknown, we generally test a range of values and infer the "between" points to follow a curve. However, consider the case of a receiver in a simple uniform phase noise channel [4] which performs relatively poorly at specific values. If the typical simulation approach is taken by simulating evenly distributed test points across a phase range of 0 to 2π, it is possible that these holes will be missed. To detect these holes, the test points would have to be close enough together, they must possess sufficiently high resolution. Given that higher resolution testing will increase the simulation time, choosing the appropriate resolution for the test points in given simulation is a nontrivial task.

Finally, consider the example involving a high-performance channel where the BER is very low. In this case, it may take a significant number of simulation runs to get the desired precision, since errors are so rare.

From these examples, we see that different types of channels pose varying levels of challenge when implementing a simulation. However, while there are specific examples of difficult channels, it is not immediately clear what exact properties or characteristics lead to a given channel's simulation simplicity or difficulty. During the early research phase of this project, no existing research was found with respect to generalized classification of simulation complexity, or aspects of channels which would make them difficult to simulate. Most simulation methods discussed are very

specific and must be determined for each individual channel [5–10].

If these channels and simulation methods could be understood in a more general sense, there exists the potential to apply knowledge of one channel simulation to other channel simulations with similar characteristics. In addition, with this general knowledge of characteristics, it may be possible to manipulate and transform difficult channels to resemble simple channels, thus making them easier to simulate. In summary, this research asks the following questions: What makes simulations simple? What makes them difficult? How can we make difficult simulations simpler?

The goals of this project are the following:

- Create a concrete and useful definition of simulation difficulty: This will clarify the research questions and guide the subsequent steps.

- Establish unifying terminology to discuss simulation methods and complexity across a range of different channels: This terminology is important because it will allow consistent evaluation and comparison of channels during the analysis process.

- Identify correlations between the general characteristics of channels and the complexity of their simulation methods: The reason for identifying these characteristics is because there may be the potential to leverage simulation methods across different channels which share these found characteristics. The idea of manipulating these found characteristics to make channels with difficult simulation methods closer to channels with simple simulation methods is also of interest. Additionally, these characteristics will form the basis for the complexity framework.

- Create a framework to visualize and connect channel simulation characteristics to their effects on simulation methods: This complexity framework can be used

to identify simulation challenges early on and what would need to be altered to mitigate these challenges.

Please note that, without the loss of generality, we will only be considering a few channels to investigate, and a few existing simulation methods, to keep the scope of the project reasonable. There are numerous other existing channels and simulation methods than those discussed in this thesis. We will also not discuss channel modeling or how to select a model but will instead focus solely on how to implement a simulation in software after a system and channel model have already been selected. That is, given a system and a well-defined channel, how can we effectively implement a computer simulation to test the system's performance in the channel?

1.2 State of the art

The goal of this thesis is not to improve upon a single existing component of communication system simulation, but rather to bridge the gap between multiple components. By drawing connections between the various aspects of simulation, it will be easier to determine how a given system and channel can be simulated and what steps can be taken to simplify the simulation process. As such, this section will focus on existing work in multiple aspects of simulation and discussion of existing connections between these topics.

Simulation techniques:

The general approach for Monte Carlo (MC) simulation of communication systems has remained largely unchanged in recent years. There are a few accessible books on the subject. In particular, [11] and [12] provide a thorough treatment of communication system simulation and were very valuable in this project. There have been many other publications on the topic of communication system simulation, but

they often concern very specific approaches for simulation of a particular system or channel, or selection of the appropriate model to use for a given system [13, 14].

There is also content about methods for reducing simulation time. However, it is seldom specified when these methods can or should be used, but rather how they have been applied to specific systems.

Channel models:

Although this thesis will not focus on developing or selecting channel models, some channel models will be discussed throughout, specifically with respect to the simulation methods which can be used to implement these channels in a computer simulation.

There is a substantial amount of information available about channel models. There are channel models available for many different applications, such as AWGN, multipath, phase offset, Doppler effects, various fading channels, and more. The models themselves are generally well-defined. There is also an adequate amount of discussion on how to choose an appropriate model for a given scenario. Despite the fact that channel models are available and generally well-defined, we were unable to find any existing publication that classified or discussed channel models with respect to computer simulation methods specifically.

1.3 Challenges

Although simulation methods and channel models are both well researched independently, the bridge between them is not always clear. Channel models exist, and simulation methods exist, but we could not find any existing work which directly linked the two in a general sense such that simulation methods could be identified or selected based on a channel's characteristics. This means that, when implementing

a simulation for a given channel, determining what methods to use in implementing the simulation often requires research into existing work and methods for that exact channel. Additionally, existing works often do not include the reasoning behind the simulation methods used, which means any difference in channel or scenario specifics can lead to uncertainty as to whether the simulation methods in the existing work are also valid for the proposed scenario.

1.4 Thesis contributions

This work contains the following two primary contributions:

- Terminology to discuss simulation methods and complexity across a range of different channels: This terminology facilitates the analysis process which is used to identify complexity characteristics.
- A complexity framework which can be used to guide the selection of simulation methods, identify simulation challenges early on, and identify what would need to be altered to mitigate these challenges: This framework provides an understanding of the underlying characteristics associated with various simulation methods.

As previously discussed in Section 1.3, without a general understanding of how channel characteristics relate to simulation methods, determining simulation methods often requires research into existing work covering the exact channel of interest, and any differences between the channel in the existing work and the channel of interest can lead to uncertainty regarding the validity of applying the same simulation methods. A more general understanding of simulation methods as they relate to channel characteristics could lessen this problem by allowing the use of existing

works which have different channels or scenarios but share certain characteristics. In addition, having an understanding of the underlying characteristics which make a simulation method possible, as provided by the complexity framework, allows for increased confidence in the validity of a simulation method for a channel of interest.

1.5 Thesis organization

This thesis is organized as follows: Chapter 2 will provide a background on Monte Carlo simulation of communication systems and the representation of complex baseband signals and channels. Chapter 3 will introduce a definition of simulation difficulty and language for discussing simulation methods and complexity across a different channels. It will then present an analysis of six channel simulation examples and summarize the findings on implementation difficulty for these channels. Chapter 4 will propose a new framework for considering simulation complexity with respect to simulation and channel characteristics. Finally Chapter 5 will cover concluding thoughts and opportunities for future work.

Chapter 2

Overview for Implementing Channel Simulations

2.1 Monte Carlo simulation

Monte Carlo (MC) simulation is a technique where random sampling is used to predict the probability of an outcome. Rather than repeating a precise physical experiment, MC simulation uses random processes as inputs and estimates the properties of the output based on repeated experiments. In the field of communication systems, MC simulation is usually used to estimate the BER of the system.

Figure 2.1 illustrates a top level view of the MC simulation process, where we generate samples for each input process, run them through the communication system blocks (modeled in software) and channel, and observe the output.

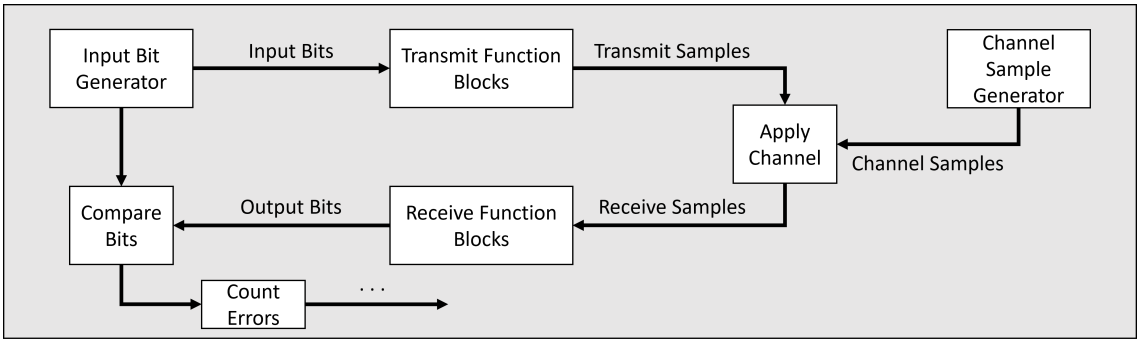


Figure 2.1: An overview of the Monte Carlo simulation process.

For example, suppose we define a communication system with input signal $A(t)$,

a random noise signal $N(t)$, output signal $Y(t)$, and output estimated by the receiver $\hat{Y}(t)$. The objective of this simulation is to find the expected value

$$E(g(\hat{Y}(t))) = \frac{1}{N}\sum_{i=1}^{N} g(\hat{Y}(i))$$

where N is the number of samples used in the simulation, $\hat{Y}(t)$ is the estimated output, and $g(\hat{Y}(t))$ is a function which determines bit errors [11].

A practical discrete implementation for this simulation is as follows:

1. Generate input bits $A(k)$ for $k = 1, 2, \ldots, N$.

2. Generate noise samples $N(j)$ for $j = 1, 2, \ldots, mN$, where m is the number of samples per bit.

3. Process the input bits through the transmit blocks, channel blocks (apply the generated noise samples), and receive blocks to obtain the estimated output bits $\hat{Y}(k)$.

4. Estimate the expected value using the expression:

$$E(g(\hat{Y}(k))) = \hat{P}_e = \frac{1}{N}\sum_{k=1}^{N} g(\hat{Y}(k))$$

where $g(\hat{Y}(k)) = 1$ if $\hat{Y}(k) \neq A(k)$ and $g(\hat{Y}(k)) = 0$ if $\hat{Y}(k) = A(k)$ [11]. The expected value, $\hat{P}_e$, is the estimated BER of this system.

MC simulation can be implemented in software in a variety of ways. One convenient option is a software such as MATLAB, which has many resources available on how to implement communications systems and MC techniques, and has many communications functions already implemented and available [15]. Another potentially

useful resource in this process is [16], which covers various signal processing algorithms which may be needed for simulating the communication system as a whole. For more information on MC simulation, please see [1, 11, 12].

2.2 Complex baseband channel representation

Communication system simulations are often implemented at baseband since passband simulations require more samples and take much longer to run. Throughout this thesis, we will assume simulations are being implemented using complex baseband signals. This section is largely borrowed from [17].

In this thesis, we will be using a Binary Phase Shift Keying (BPSK) modulation scheme for our communication signal. A BPSK signal can be represented in complex baseband as:

$$S_m(t) = S_{m1}\phi_1(t) + S_{m2}\phi_2(t)$$

where

$$S_{m1} = A\sqrt{E_g/2} \cdot \cos(\pi(m-1))$$

$$S_{m2} = A\sqrt{E_g/2} \cdot \sin(\pi(m-1)) = 0, \quad \forall m$$

$$\phi_1(t) = \sqrt{2/E_g} \cdot \cos(2\pi f_c t) g(t)$$

$$\phi_2(t) = -\sqrt{2/E_g} \cdot \sin(2\pi f_c t) g(t)$$

for $m = 1, 2$, signal pulse shape $g(t)$, bit duration T, and pulse shape energy E_g given by

$$E_g = \int_0^T g^2(t)\, dt$$

Thus, we obtain the following:

$$S_{11} = A\sqrt{E_g/2}$$

$$S_{21} = -A\sqrt{E_g/2}$$

$$S_{12} = S_{22} = 0$$

We have two possible symbols:

$$S_1(t) = A\cos(2\pi f_c t)g(t)$$

$$S_2(t) = -A\cos(2\pi f_c t)g(t)$$

For this thesis, we will assume $g(t)$ is a rectangular pulse shape with a duration of $[0, T]$. In this case, evaluating E_g gives $E_g = T$.

Channels can also be represented in this form. Since both the signal and the channel are functions of time, channel noise can be included in the equation as an additional noise term. For example, an additive noise such as AWGN can be added to the BPSK signal $S_m(t)$ as follows:

$$r(t) = S_m(t) + n(t), \quad m \in 1, 2$$

where $n(t)$ is the noise and $r(t)$ is the received signal.

2.3 Chapter summary

In this chapter, we gave a brief overview of the MC simulation process and recommended some useful resources for implementing simulations in software. We also

discussed the representation of communications signals and channels in complex baseband form.

Chapter 3

Proposed Channel Analysis

What makes simulations simple, what makes them difficult, and how can we make difficult simulations simpler? To answer the first part of this question, we need to consider two things:

1. What are the aspects of a particular simulation that cause it to be deemed difficult?

2. What characteristics of channels contribute to this difficulty?

Therefore, to start answering this part of our research question, we will first explore the MC simulation process and attempt to assemble a useful, concrete definition of simulation difficulty. Once we have a useful definition for difficulty, we will establish unifying terminology to discuss simulation methods and complexity across a range of different channels. We will then analyze various channel simulations and attempt to identify what characteristics contribute to these simulation difficulty levels.

The second part of the research question concerns making difficult channel simulations easier. Hopefully, once we identify what characteristics are making simulations difficult to implement, we can leverage that knowledge to identify ways of simplifying the simulation process.

3.1 Simulation difficulty definition

Some of the primary challenges in making an effective simulation are:

- Choosing the appropriate model for the scenario

- Balancing thoroughness with available computing power and time
- Being confident that the chosen amount of testing is sufficient

For this project, we will limit the scope to the last two aspects, which are closely related, and assume an appropriate model has already been chosen. There is a significant amount of existing content available on how to choose an appropriate model, so we will not concern ourselves with that aspect of simulation here. In practice, these two aspects can be broken down into more concrete choices which must be made during the simulation implementation process.

The first challenge, choosing how thorough to be in testing, comes down to choosing which possibilities to test. For example, if we want to evaluate the performance of a given system in an AWGN channel, we must choose which range of signal-to-noise ratio (SNR) values to test, and how to space them out. We could test a broad range of values, say [-30dB, -15dB, 0dB, 15dB, 30dB], or perhaps we want to look at performance in more detail in a much more narrow and targeted range, such as [-13dB, -12.5dB, -12dB, -11.5dB, -11dB]. In choosing these parameter values, we must consider both the bounds and the spacing of the values. In the AWGN case, using a higher resolution may not be particularly useful outside of creating a smoother performance curve. However, consider the case of a channel where the performance is low for certain values. In this case, it is vital to have sufficient resolution, otherwise important performance information may be missed.

The second challenge, determining if the amount of testing is sufficient, is closely tied to the first challenge, and concerns how many test points are needed to test a given parameter value. Because of the random nature of many channels we usually have to test a given value on the performance curve many times and average the result to get a better understanding of the performance. We then must ask ourselves,

"how do we know how many test points are needed for a given parameter value?" This question is much simpler to answer for some channels than others, as we will discuss below.

It seems, then, that the difficulty of simulating a system in a given channel is largely related to parameter bounds, spacing, and test points. There are certainly other complicating factors as well, but for the sake of limiting scope we will mainly be considering these three factors in our analysis during this project. Based on this, we will construct a definition of simulation difficulty for use in this project.

Definition: **Simulation Difficulty**—The level of complexity involved in choosing parameter bounds, spacing, and test points.

This definition is focused on difficulty caused to the user implementing the simulation. In addition to this difficulty, we will also consider effects on simulation time and available simulation methods as part of this analysis.

3.2 Analysis terms

In this section we will introduce seven terms which will be used throughout this thesis to discuss various parts of a simulation. We will use a simple AWGN channel simulation example to demonstrate their use.

Definition: **Performance Curve**—A graph which describes the performance of a system under some set of condition(s), such as a BER curve.

Definition: **Simulation Parameter**—A parameter which will be varied in order to produce a performance curve, and acts as an x-axis, such as the SNR.

Definition: **Parameter Values**—The set of possible values a parameter will be set to throughout the simulation.

Definition: **Performance Metric**—The measure by which we will describe the performance of a system, for example BER or symbol error rate (SER).

Definition: **Distribution**—The probability distribution used to model the channel being simulated.

Definition: **Test Points**—The set of actual values which the system will be tested against. For example, in an AWGN channel these would be the noise values which get added to the signal.

Definition: **Test Point Outcome**—The outcome of simulating the receiver under a specific test point. For example, in a BPSK system the outcome will be either a correctly decoded bit or an incorrectly decoded bit.

Suppose we want to visualize the BER of a BPSK modulated system in an AWGN

channel. To do that, we will use MC simulation to generate a **performance curve** such as the one shown in Figure 3.1 [18].

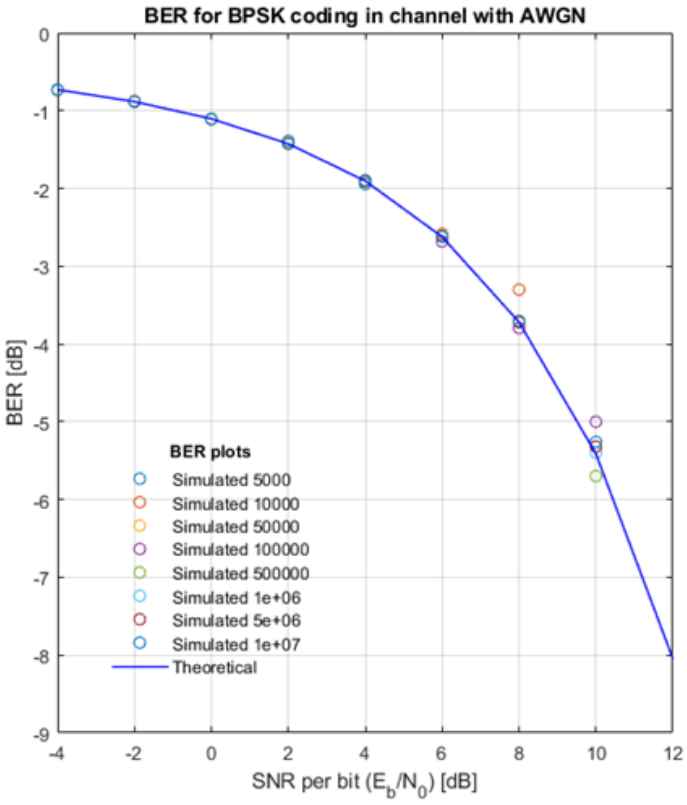
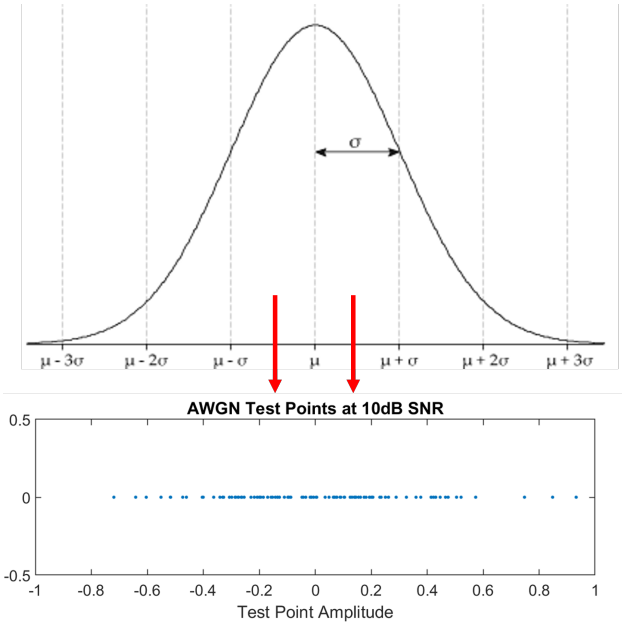


Figure 3.1: Performance curve for BPSK in an AWGN channel.

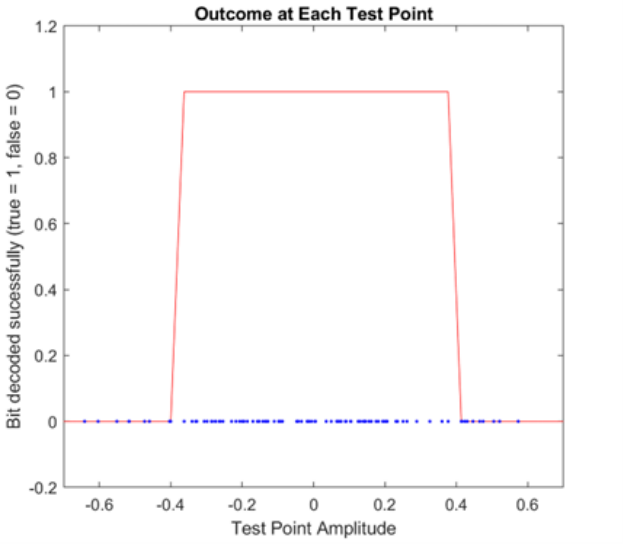
For this simulation, there is one **simulation parameter**, SNR per bit, which is marked on the x-axis. The parameter is **bounded**, with a lower bound of -4dB and an upper bound of 12dB. The parameter values are in the interval [-4dB, 12dB]. The BER is the chosen **performance metric** in this example case since it is commonly used.

For each parameter value (e.g. 10dB), we will generate a set of **test points** from the AWGN **distribution** and simulate the outcome at each of those points. This process is demonstrated in Figure 3.2.

As shown in Figure 3.2a, for each parameter value there will be an associated set



(a) Test point generation [19].



(b) Test point outcomes.

Figure 3.2: AWGN simulation process for BPSK

of test points. The receiver performance under that set of test points will be used to characterize the receiver performance under the parameter value. Figure 3.2b shows an example graph of the **test point outcomes** for the 10dB SNR value. Since the system is using BPSK, the test point outcomes are binary. Either the symbol is correctly recovered or not.

For a typical AWGN simulation, we then take the test point outcomes and average them to get the BER for that parameter value. Note that in Figure 3.2b the results are exaggerated to show the shape of the test point outcome graph. A real BPSK system would have better performance at 10dB and would only fail for test points at the edge of the graph (having an expected BER of around 3.9×10^{-6}).

3.3 Channel simulation analysis

In this section, we will analyze various channel simulations with respect to the difficulty definition above. For each channel, we will consider the complexity level of each aspect (bounds, spacing, and test points) and try to identify what characteristics contribute to these simulation complexity levels. We will look at both characteristics of the channel itself, as well as the expected characteristics of the performance curve and test point outcomes. We will also consider effects on simulation time and available methods as part of this analysis.

In order to focus on the simulation differences between channels as much as possible, we will assume a mostly "black box" receiver and make a few assumptions:

- The system being simulated is a binary communication system, specifically BPSK
- The receiver behavior is time invariant

3.3.1 Simulation of an additive white Gaussian noise channel

The first channel we will consider is the AWGN channel. The AWGN channel is likely the most widely used channel in the communication systems field. It is very well-defined and has been simulated frequently for many different systems, so it will serve as a good point of reference for the other channels we will discuss in this section.

A simulation of this channel will generate a performance curve that has one parameter (SNR). In the case of a BPSK system, this is energy per bit to noise ratio (Eb/N0). The performance metric is BER. For each SNR value, we will generate a set of test points or noise values which we will test the system against. These test

points will be randomly generated according to the probability distribution of the noise, which is a case of the Gaussian distribution.

The Gaussian distribution occurs when many independent noise sources are added together to create one overall noise source [20]. AWGN is a specific type of Gaussian noise, often used to model electrical or thermal noise, which is a stationary, zero mean, white, Gaussian process, typically the result of many individual random noise sources [20, 21]. The Gaussian probability density function (PDF) is uniquely specified by its first two moments [20]. The PDF is defined as:

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

where μ is the mean and σ^2 is the variance [22].

Generating AWGN noise values in software is relatively simple, either using existing software functionality or implementing from scratch. For an overview of the AWGN generation process, see [2]. AWGN can be added either at passband as real noise or at complex baseband. Since passband simulations are not generally practical in software, we will focus on complex baseband simulations here. Since the real and complex components of the AWGN are independent and identically distributed (i.i.d.), we can consider this as adding two separate AWGN channels to the signal, one added to the real part of the signal, and one added to the complex part of the signal [23]. Using the complex baseband representation described in Chapter 2, we can represent the received signal when AWGN is added as follows:

$$r(t) = S_m(t) + n(t), \quad m \in 1, 2$$

When it comes to implementing the simulation described above, some decisions must be made regarding the simulation methods.

First, the parameter bounds must be chosen. There are no inherent bounds for this simulation, but reasonable bounds can be put into place based on the range of SNR values that can be expected given the application, and what values are useful. For example, we might choose not to simulate SNR values below -2dB since any BER below that point will be very high. Conversely, we might choose an upper SNR bound based on empirical channel measurements, or because the performance at the upper boundary value will meet performance requirements. We also have the advantage of being able to generate a theoretical BER curve for this channel (shown in Figure 3.3) which can be used as a tool to estimate where bounds should be. It is best to choose the smallest upper bound possible that still accomplishes the simulation goals and provides the necessary performance information. This is because high SNR regions will have lower error rates, which means that the simulation at those SNR values will take significantly more run time to obtain reliable results. In a BPSK AWGN simulation run until $10,000$ bit errors for each SNR value, we found that the required simulation time and test points increased faster than exponentially, inversely proportional to the BER. See Appendix B for the full results of this test.

Second, the parameter spacing must be chosen. The expected shape of the performance curve is smooth, without "performance holes", and monotonically decreasing. Because of this, the only concern with spacing of the parameter values is creating a good visual representation. Any spacing that conveys the performance curve overall is sufficient, so this is mainly based on preference. Ideally, the spacing will be such that the performance at any SNR value can be estimated visually from the curve, but there are not more values than necessary, since that would increase simulation time. A simulation with parameter values [0, 1, 2, 3, 4, 5] would require about twice as many total test points as one with parameter values [0, 2, 4], which

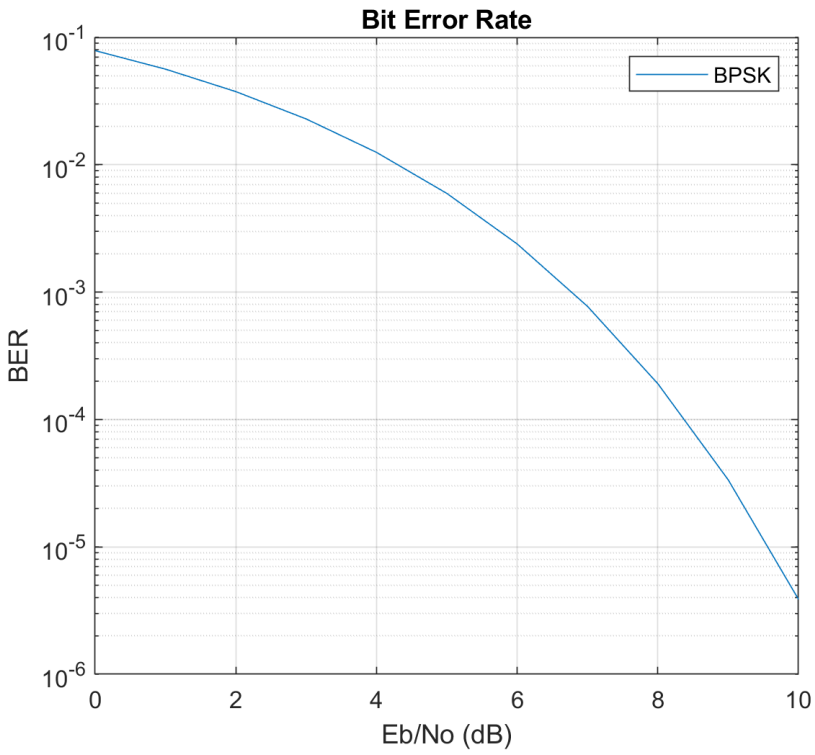


Figure 3.3: Theoretical BER curve for BPSK in an AWGN channel.

is roughly twice as much simulation time.

Finally, the test points themselves must be considered. During the simulation process, a set of test points will be generated for each parameter value. The number of test points in each set must be decided when the simulation is implemented. In the case of AWGN, although the distribution is not strictly bounded, it is light tailed, meaning there is little concern with the variance or missing low probability behavior with too few runs. A common approach in this case is to generate additional test points until a certain number of errors are reached, for example 100 errors. Running until a higher number of errors occur will result in a more reliable error rate, while running until a lower number of errors will be faster but less reliable.

One important factor of the test point selection for this channel is the expected test point outcomes. No specific test point values are expected to have significantly

worse performance than the surrounding region. In other words, there are no "holes" in the test point outcome graph. Rather, the test point outcomes follow a predictable pattern with one "pass" zone—a section of test points where the test point outcomes will be 1, indicating a correctly decoded bit—and the remaining regions being "fail" zones—a section of test points where the test point outcomes will be 0, indicating an incorrectly decoded bit. Because of this, we can randomly generate test points without any concern of missing important performance information.

Based on the above analysis, we will summarize the user difficulty of implementing this channel in Table 3.1.

Table 3.1: Implementation difficulty of the AWGN channel.

Additive White Gaussian Noise Channel			
	Parameter Bounds	Parameter Spacing	Test Points
Simple		X	X
Moderate	X		
Complex			

3.3.2 Simulation of a uniform phase noise channel

The next channel we will consider is the uniform phase noise channel. This channel may be used to model effects such as oscillator phase offset. For a typical communication system, the uniform phase noise channel would be a non-issue and would not need to be simulated because the phase would be tracked and thus would not cause errors. However, it is provided here as a straightforward example for analysis purposes. Therefore, for this example, assume the performance of the receiver in the presence of a phase offset is binary, and there is either no phase recovery or the phase recovery only works up until a fixed point.

We can consider the phase $\theta(t)$ of a periodic signal $s(t)$ as a periodic function with the same period as $s(t)$, so we do not have to consider phase values beyond a

single period, $-\pi$ to π. Thus, this channel follows a uniform distribution between $-\pi$ to π, which can be described by its probability density function (PDF) $f(x)$ or its cumulative distribution function (CDF) $F(x)$:

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{for } x \in [a, b] \\ 0 & \text{otherwise} \end{cases}$$

$$F(x) = \begin{cases} 0 & \text{for } x < a \\ \frac{x-a}{b-a} & \text{for } x \in [a, b] \\ 1 & \text{for } x > b \end{cases}$$

where $a = -\pi$ and $b = \pi$ [24].

We can represent a BPSK signal $S_m(t)$ as

$$S_m(t) = A\cos(2\pi ft + \theta_m)g(t), \quad m \in 1, 2, \quad \theta_1 = 0, \quad \theta_2 = \pi$$

where each symbol has phase θ_m. The phase noise from the channel, θ_n, would be added to the signal as follows:

$$S_m(t) = A\cos(2\pi ft + \theta_m + \theta_n)g(t)$$

Intuitively and from experience, testing a receiver's response in a uniform phase noise channel is relatively straightforward. In this section, we will attempt to quantify why the simulation process is straightforward and what properties of the channel make this simple simulation process possible. Overall, we will see that the simple simulation process is primarily due to the three difficulty aspects defined previously (parameter bounds, spacing, and test points) having low complexity, and that the

low complexity of these three difficulty aspects is due to certain properties of the uniform phase noise channel and the shape of the expected performance curve.

Unlike other channels which have input parameters (such as the AWGN channel, which has the parameter SNR), phase noise is just one set of uniformly distributed noise. This channel has no variable parameter; that is, there is no aspect of this channel which must be adjusted to various levels in order to get a complete understanding of the performance. The only part that varies is the phase offset value itself and that range of values is already built into the distribution, just as noise values are built into the AWGN distribution. Because of this, the performance curve is just a single point: the probability of error due to phase offset.

To implement this simulation, we still need to consider parameter bounds, spacing, and test points. This is a rather unique case where there is no variable parameter, making this the simplest case of parameter bounds possible. The parameter, probability of error, is bounded to a single point and it has no quantitative value because there is no x-axis. This means that the parameter bounds are effectively built into the channel itself and do not have to be selected. The parameter spacing follows suit—this is the simplest possible case because with only one parameter point, there is no spacing.

The only aspect of the simulation we then actually need to make decisions about is the set of test points used for the single parameter value. First, consider some characteristics of the distribution itself which is where the test points are drawn from:

- The distribution is strictly bounded from $-\pi$ to π
- The distribution is uniform, so there are no tails or low probability areas of concern

- The test point outcomes (shown in Figure 3.4) follow a predictable pattern with one "success" zone.

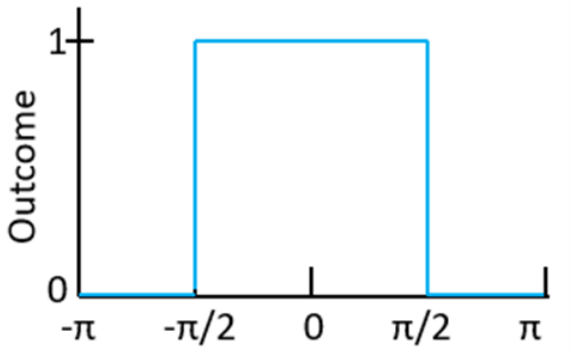


Figure 3.4: Test point outcomes of BPSK in a uniform phase noise channel.

Given these characteristics, we can confidently generate test points using the standard MC method. The only concern is testing enough points to get a reliable BER when we average across all test points. Because the test points follow a uniform distribution, there are no tails, which would normally increase the number of test points required, so even a reasonably small number of test points can represent the performance.

Although this method is perfectly viable, there are even more aspects of this channel which can be taken advantage of to reduce the required number of test points. Because the distribution is uniform and strictly bounded from $-\pi$ to π, we can simplify the simulation process further by skipping the random generation altogether and directly testing evenly spaced points. This way we can ensure we cover the full range evenly with a minimal number of test points (unlike random generation, which could take a larger number of points to fully represent the distribution). This drastically decreases the number of required test points. For example, in a software implementation of this simulation, testing with only 100 manually spaced test points consistently achieved a more accurate BER estimation than $100,000$

randomly generated test points. See Appendix A for this comparison.

Alternatively, we could simplify the problem even further by taking advantage of the binary test point outcomes and the easily tractable CDF of the uniform distribution. First, we can use manual test points in order to identify the threshold where the system fails. Once the threshold has been found, we directly calculate the probability that the offset will exceed that threshold. This threshold method is simple to implement for this channel because the test points are well bounded, being that the distribution itself is well bounded between $-\pi$ to π. Additionally, because our system is time invariant, we only need to run a given point once to understand the performance at that point, and the performance is pass or fail.

In this case, rather than testing uniformly spaced points, we can use a search method to find the threshold where the test outcomes change. There are many suitable search algorithms for such a task, such as a linear search, binary search, and exponential search [25, 26]. There are also search algorithms which take advantage of interpolation [25, 27], but since we are considering a case with binary test point outcomes, we will not discuss those here.

Since we want to quickly find a threshold value and are working with what is effectively a single dimensional array, we can take advantage of faster search algorithms than if we used a linear search. For algorithms such as binary or exponential, we will need to implement an upper bound on the test point values for the algorithms to work because they begin by breaking up the data into sections. However, since the algorithms go through large sections relatively quickly at first, picking a large value for the upper bound should be appropriate. There are also unbounded search algorithms we could consider using, such as a modified binary search, where we first establish the bound by doubling each guess and then carry out a standard bounded binary search [28].

Since there will be duplicate test point outcome values, if the search algorithm finds a point past the threshold, it will then need to go back to find the first of those values. Because of this, it may make sense to combine multiple search algorithms—one to get in the right area followed by one to find the precise threshold.

Rather than calculating test point outcomes beforehand and then searching for the threshold, we can calculate outcomes as we go for each test point we check during the search process. Searching in this way will significantly decrease computation versus trying to compute the outcome for the whole range of test points across a distribution.

Once the threshold has been found, we make use of the CDF of the uniform distribution. If we can find the threshold(s) where the phase offset results in a symbol error, we can readily calculate the probability that the noise will be above that threshold value and calculate the BER directly. For example, if we find that the system fails when the phase offset exceeds $\pm\pi/2$, we can calculate the BER as follows:

$$P(X < x) = F(x) = \frac{x-a}{b-a} = \frac{x+\pi}{2\pi}, x \in [-\pi, \pi]$$

$$\begin{aligned} P_e &= P(X < -\frac{\pi}{2}) + P(X > \frac{\pi}{2}) \\ &= P(X < -\frac{\pi}{2}) + (1 - P(X > \frac{\pi}{2})) \\ &= \frac{-\frac{\pi}{2} + \pi}{2\pi} + \left(1 - \frac{\frac{\pi}{2} + \pi}{2\pi}\right) = 0.5 \end{aligned}$$

We find a BER of 0.5, as might be expected for a BPSK system with effectively random phase without any phase correction.

Any of these three options (MC, manual test point selection, or the threshold

method) is a feasible way to get an understanding of the performance of a BPSK system in a uniform phase offset channel. There is no concern with relation to testing enough or requiring a large number of test points to obtain valid results. The first method is a pure MC simulation and there is a simple method to choose how many test points to generate. For the second option, we test a set of uniformly spaced points a single time. For the third, we mathematically calculate an exact probability based on the determined threshold value(s).

Based on the above analysis, we will summarize the user difficulty of implementing this channel in Table 3.2.

Table 3.2: Implementation difficulty of the uniform phase noise channel.

Uniform Phase Noise Channel			
	Parameter Bounds	Parameter Spacing	Test Points
Simple	X	X	X
Moderate			
Complex			

3.3.3 Simulation of a uniform phase noise channel with performance holes

We will now consider a system identical to the previous uniform phase noise channel system, but this time there is something internal in the black box receiver that is causing bit errors at specific phase offset values as shown in Figure 3.5. In other words, there are performance "holes," and the locations of these holes are unknown.

As with the uniform phase noise channel, there is no variable parameter in this system, so it is bounded to a single value by default. Parameter spacing is also irrelevant because there is only one parameter value. The only decision lies with the set of test points.

In the previous uniform phase noise case, we had the option of traditional MC

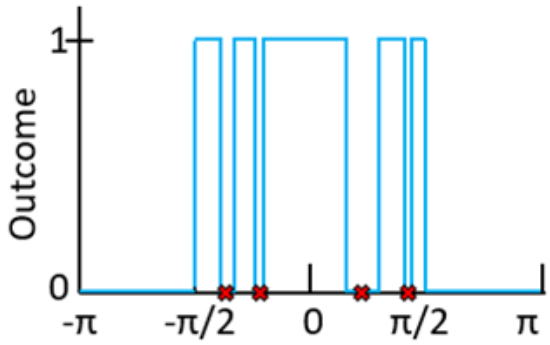


Figure 3.5: Test point outcomes of BPSK in a uniform phase noise channel with performance holes.

simulation, manual test point selection, or a performance threshold search method. In this case, because we have performance holes, the issue of test points becomes significantly more complicated. The distribution is unchanged, but the expected performance behavior is different.

For a traditional MC implementation of the uniform phase noise channel, the test point set only needs to be large enough to represent the distribution. However, in this case, we could generate a high number of errors and still completely miss the holes because we did not generate any test points in the exact regions where the system fails. The problem then becomes a matter of how many test points are required before we can be confident that any missed holes are so small that they can be neglected. This problem does not have any obvious solution based on existing literature. Additionally, because the test points are randomly generated, it may be more difficult to understand which regions do or do not have holes unless the test point outcome graph is examined directly. Even if a region appears to be hole free after some points are tested, if the size of the hole is unknown then there is no guarantee that any untested point does not have a hole.

Another simulation option would be manually selecting the test points. The goal with this method would be to have high enough resolution to be certain that if there

are holes, they are a small enough region that the probability of hitting them (if they exist) is below a certain level. This is not completely satisfactory, but it offers a general understanding of what the worst-case performance might be. Running the simulation with this approach is a tradeoff between resolution and time.

Unlike the case without holes, we cannot use the threshold method for this channel. The threshold method relies on calculating the probability of a region based on the failure threshold, but when there are performance holes, we no longer have a simple "pass" region, but many regions which pass or fail.

Although the parameter bounds and spacing are still simple, the test point selection in this case is very complex. It requires many test points to have a basic understanding of the performance, and it is unclear exactly how many points should be tested before the results are sufficiently reliable.

Based on the above analysis, we will summarize the user difficulty of implementing this channel in Table 3.3.

Table 3.3: Implementation difficulty of the uniform phase noise channel with performance holes.

Uniform Phase Noise Channel with Performance Holes			
	Parameter Bounds	Parameter Spacing	Test Points
Simple	X	X	
Moderate			
Complex			X

3.3.4 Simulation of a carrier frequency offset channel

The next simulation we will consider is a carrier frequency offset channel. This channel applies a constant frequency offset between the transmitter and receiver. For this simulation, we assume there is some black box frequency offset correction in the receiver, otherwise a frequency offset will gradually increase the phase offset

of each sample until the system breaks. In this case, we expect that the frequency offset correction will succeed up to a certain offset value.

This channel shares similarities with the uniform phase noise channel and AWGN channel, but there are some key differences. Frequency offset is not reflected in a single sample as is the case with phase offset or AWGN. This channel affects samples over time. Because of this, the test points in this case are not randomly distributed but must be manually generated based on the offset value. Then, for each offset value, we need to observe the performance under that value over time. Additionally, unlike phase noise which has binary performance, each frequency offset will have a failure rate. Based on this, we could use the frequency offset value as the parameter for the performance curve, and BER as the performance metric.

We should note that this channel could be considered in multiple ways, depending on what behavior was of interest. For example, instead of understanding the performance for each offset value, one might want to know the performance at a top level. In that case we would need to know the distribution of the offset values, then generate a set of test offsets from the distribution and check the bit error rate at each test offset. The result would be a single average BER due to carrier frequency offset. However, for this example, we will consider the case where the performance curve indicates BER based on frequency offset value.

Similar to AWGN, the carrier frequency offset parameter does not have inherent bounds, but we can apply reasonable bounds based on the application. If we are simulating a specific system, we can consider how inaccurate the oscillators of the transmitter and receiver may be. If we are simulating a general system, we can assume a conservative limit based on the frequency range we are considering.

We expect this performance curve to be monotonically increasing as the frequency offset increases with no performance holes. Because of this, the only concern

from spacing is to have a visually smooth curve. Even if the exact shape of the curve is unknown, we can confidently interpolate between points without worry of missing important system behavior.

The test points generated during this simulation will be increasing phase offsets over time which will be applied to the samples. The test point values will be fixed based on the parameter value (e.g., a 1Hz offset will shift each point by an additional $2\pi/F_s$, one full rotation divided by the number of samples per second). Because of this, although this channel is a frequency offset channel we can model its effects on a BPSK signal as follows:

$$S_m(t) = A\cos(2\pi ft + \theta_m + \theta_{cfo})g(t))g(t), \quad m \in 1,2, \quad \theta_1 = 0, \quad \theta_2 = \pi$$

where the phase shift due to carrier frequency offset, θ_{cfo}, is a non-random signal defined as:

$$\theta_{cfo}(k) = k\frac{2\pi f}{F_s} \text{ for } k = 0,1,...,N$$

The only decision is how many test points to generate for each offset value, essentially how much time to test the performance across for a single value. Since the effect of this channel is cyclical in nature, it likely will not require a large number of points to get a reasonable estimate. There is one caveat to simulating this channel—because the samples must be tested consecutively in order, there is less opportunity for parallel processing than in a channel where the effects are completely random.

Based on the above analysis, we will summarize the user difficulty of implementing this channel in Table 3.4.

Table 3.4: Implementation difficulty of the carrier frequency offset channel.

Carrier Frequency Offset Channel			
	Parameter Bounds	Parameter Spacing	Test Points
Simple		X	X
Moderate	X		
Complex			

3.3.5 Simulation of a heavy-tailed amplitude noise channel

The next channel we will analyze is an additive noise channel where the noise follows a heavy-tailed amplitude distribution. The term "heavy-tailed" generally refers to a probability distribution which is not exponentially bounded [29]. This type of distribution has been used to model atmospheric noise in the wireless channel, including distributions such as Student's t-distribution [30], the alpha-stable distribution [31, 32], and the lognormal distribution [33, 34], among others. For this scenario, we will use Student's t-distribution as an example for this heavy-tailed distribution.

The Student's t-distribution has a similar shape to the normal distribution, but it has heavy tails which means it is more likely to generate values far away from the mean than the normal distribution. It is a "heavy-tailed distribution that arises naturally in the construction of hypothesis tests for the expected value of a normally distributed random sample of observations with unknown variance" [35]. The PDF of the Student's t-distribution is defined as:

$$f(x) = \frac{\Gamma(\frac{\nu+1}{2})}{\sqrt{\nu\pi}\Gamma(\frac{\nu}{2})}\left(1+\frac{t^2}{\nu}\right)^{-(\frac{\nu+1}{2})}$$

where ν is the number of degrees of freedom and $\Gamma$is the gamma function [36, 37]. The number of degrees of freedom ν affects how heavy-tailed the distribution is, as

shown in Figure 3.6 [38].

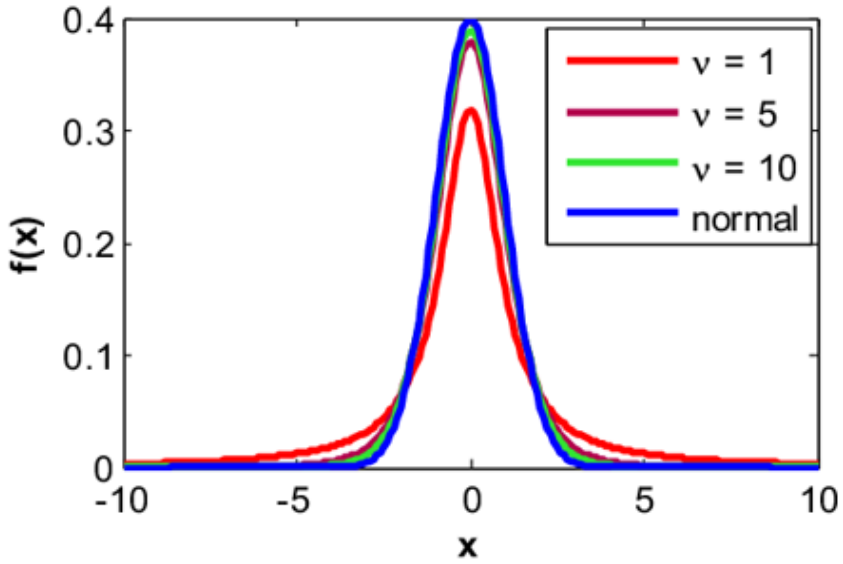


Figure 3.6: Comparison between the PDF of the Student's t-distribution with various degrees of freedom ν and the normal distribution.

There are methods to generate samples from this distribution so it can be used in MC simulation [39].

Thus, this channel is well-defined, and there are well known methods to generate noise samples from the distribution. Because of this, the heavy-tailed amplitude noise channel seems relatively simple to simulate at first glance, but one might encounter some difficulty when attempting to implement a simulation, as we will see below. This is because typical pure MC simulation relies on a limited number of test points (often just enough to get a certain number of errors) to capture the full behavior of a channel. However, when a distribution has heavy tails, the large variance in generated noise values can cause the performance results to be unreliable if the number of test points is insufficient. This can mean that the tradeoff between time and reliability becomes even more of an issue. Additionally, we will see that traditional variance reduction techniques might not always be possible nor practical.

Similar to AWGN, this channel is an additive noise channel. The received signal in this channel can be represented as

$$r(t) = S_m(t) + n(t), \quad m \in 1, 2$$

where $n(t)$ follows the Student's t-distribution. The performance metric is BER, and the parameter is SNR. There is no strict bound on the parameter values, but reasonable bounds can be put in place based on empirical data specific to the application. We expect the shape of the performance curve to be smooth and monotonically decreasing as SNR increases, and there are no performance holes expected. In terms of spacing of the parameter values, the only concern is creating a clear visual representation.

The difficulty of implementing this simulation lies in the test points. Specifically, the number of test points required for each parameter value when using standard MC simulation methods. When generating the performance curve, the BER will be lower for high SNR values. These low BER regions involve "rare event simulation," which majorly increases the number of test points required for those values.

Consider first the Gaussian distribution for comparison, which is a light-tailed distribution. With an AWGN channel, we do not always need to simulate the low BER portions of the performance curve because we may not care about performance past a certain threshold. But, if we do want to simulate those regions, although it will take many points, the variance is still not as great as the heavy-tailed case, so the number of points will be less than in the heavy-tailed case. Additionally, there are tools at our disposal called "variance reduction techniques" [11, 40]. One useful tool, known as importance sampling, is when the likelihood of error is artificially increased for the purpose of simulation. Importance sampling via exponential change

of measure can be used to greatly reduce the number of test points required for these rare event simulation regions [11, 41].

Now consider the heavy-tailed case. For a heavy-tailed distribution, we may encounter difficulty when trying to simulate low BER portions of the performance curve. Simulation of heavy-tailed distributions requires a particularly large number of test points to reflect the effect of the tail [42]. The values in the tail can have a major impact on the performance of a system despite their low probability of occurrence, so it is vital to capture that behavior in order to have a reliable understanding of the system performance as a whole [42]. However, some of the tools that can be used when the noise is Gaussian distributed are no longer possible when a distribution is heavy-tailed. When the underlying distribution is heavy-tailed, the typical method of importance sampling is, "intrinsically impossible because the required exponential moments do not exist" due to the nature of the distribution which is not exponentially bounded [43]. Methods of variance reduction appropriate for heavy-tailed distributions do exist, but they are generally more complex and specific than the methods typically used with light-tailed distributions [1, 42, 44]. We will not go into detail here, but the challenging nature of rare event simulation in the heavy-tailed case is known and is still a relevant topic of research [42, 45, 46].

One interesting idea in this scenario is to consider the threshold method we discussed in Section 3.3.2 with respect to uniform phase noise. If we can assume a few things about the system, we may be able to apply that same threshold method here:

- We do not expect the performance curve of this system to have any performance holes for specific noise values.
- The test point outcome graph will be monotonically decreasing from 0: In

this case, since the system is using BPSK and the performance is binary, there will be success near 0 amplitude, and then the performance will reach some threshold where the bit is incorrectly decoded.

- The system is time invariant: This has been our assumption throughout this thesis.

- The system is memoryless: The performance of a sample under a single given test point is not dependent on any previous test points used.

If we can make these assumptions, then we can find the amplitude threshold where the noise causes the system to incorrectly decode a sample using the same search methods discussed in Section 3.3.2. Because the t-distribution has a tractable PDF, once we know where the system fails, we can directly calculate the probability that a noise value will be at or above that point, thus we can calculate the probability of failure of a single sample. Figure 3.7 illustrates this calculation for a t-distribution with degrees of freedom $\nu = 4$ [47]. This calculated probability then becomes the BER for the current parameter value. This threshold process can be repeated for each parameter value until the performance curve is complete.

One thing to note about this threshold concept is that it assumes we can evaluate the performance of a bit with one test point value. This would not be possible if the system was being simulated using multiple samples per bit; that is, in a system with multiple samples per bit, each sample would need its own corresponding test point. The issue here is that a given noise value may not "make or break" a bit, because the ability to correctly decode the bit will depend on many samples and test points. This also leads to the question of whether the selected probability distribution is reflective of how the noise behaves on a smaller scale—each point is drawn randomly, but in reality, we may see one impulse of noise spread across multiple samples.

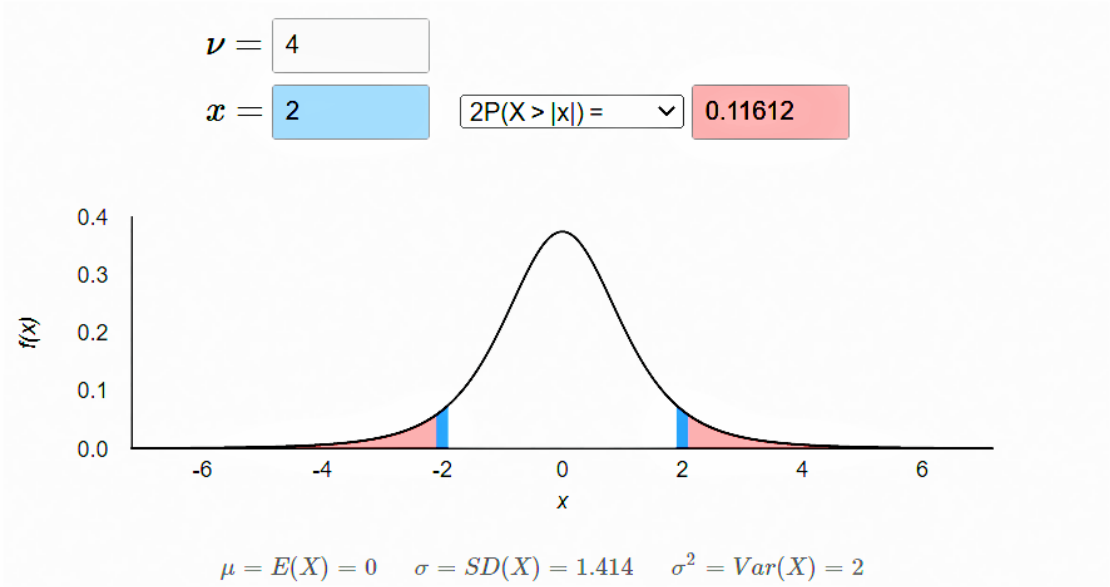


Figure 3.7: Probability that the magnitude of a noise sample generated from the Student's t-distribution exceeds an amplitude of 2.

For example, a lightning strike may affect a whole contiguous "chunk" of samples, but drawing each point individually from a distribution likely will not reflect that behavior. Finding the correct method for expanding this concept to multiple samples per symbol, then, may be non-trivial. However, this threshold method is still a potentially useful concept, so we will continue to discuss it in the context of one sample per bit as we have been assuming throughout this paper. The multi-sample case may be of interest for future study.

Based on the above analysis, we will summarize the user difficulty of implementing this channel in Table 3.5. Note that in the case where the described threshold method can be used, the complexity of choosing how many test points to use may be reduced.

Table 3.5: Implementation difficulty of the heavy-tailed amplitude noise channel.

Heavy-Tailed Amplitude Noise Channel			
	Parameter Bounds	Parameter Spacing	Test Points
Simple		X	
Moderate	X		
Complex			X

3.3.6 Simulation of a multi-parameter channel

Each channel we have analyzed so far has had only one parameter for the performance curve. The next channel we will consider is one which has multiple parameters. For this example simulation, we will use a simplified Doppler rate of change effect as the channel. We will begin by introducing standard Doppler effects and move on to the simplified Doppler rate of change channel from there.

There is a large amount of existing literature on basic Doppler effects [48–51]. That is, direct Doppler shift of a signal as well as the multipath effects of Doppler spread. The Doppler shift of a single wave is described as:

$$\omega_d = \beta v \cos(\theta)$$

where v is the velocity, θ is the angle, $\omega_d = 2\pi f_d$, and $\beta = 2\pi/\lambda$, λ being the wavelength of the carrier frequency [49].

As we look beyond a single wave to the effects of Doppler in a physical environment, it is desirable to use probabilistic models for Doppler effects. This is where the Rayleigh and Rician models come in [23]. It should be noted that the probabilistic models for Doppler spread are less accurate to reality than channels such as AWGN because the mechanism that generates Doppler spread channel effects is not probabilistic in the same way [23]. However, we need models even if they are inaccurate, so that systems can be compared and evaluated at a basic level [23]. There is plenty

of literature surrounding the subject and these models can be useful, but it is important to keep in mind that these channels are not truly characterized by probabilistic models and the results using these models are not necessarily robust [23].

Now, let us consider the following scenario: There are two aircraft communicating with each other. Either aircraft can follow any path at any time and their speed and acceleration are variable. The Doppler models discussed so far have been based on a constant velocity, but how will a system react if the velocity is changing? And furthermore, if the acceleration, the rate at which the velocity changes, is also variable over time?

We will consider the classic Doppler effect to be in place, only dependent on the velocity and not the acceleration or higher order derivatives. In the case of a single wave, when the velocity is constant, there is a constant frequency shift. When the velocity is variable and the acceleration is constant, it follows that we would expect a linear velocity and frequency shift. Finally, when velocity and acceleration are variable and jerk is constant, we would expect acceleration to be linear and the frequency shift to follow a second order curve as shown in Figure 3.8. In reality, acceleration of the transmitter with respect to the receiver will not be linear and could take any number of complicated curves. This effect would be magnified in the velocity curve, as shown in Figure 3.9. However, for the sake of creating a tractable problem space, we will consider acceleration to be a constant value. The code for generating Figures 3.8 and 3.9 can be found in Appendix C.

Given the already very generalized models we have for basic Doppler effects, it seems unlikely that an existing probabilistic model such as Rayleigh or Rician will serve us well for this scenario. However, using a probabilistic model which accounts for both velocity and acceleration may complicate this example more than necessary. Since the reason for this simulation example is simply to explore the

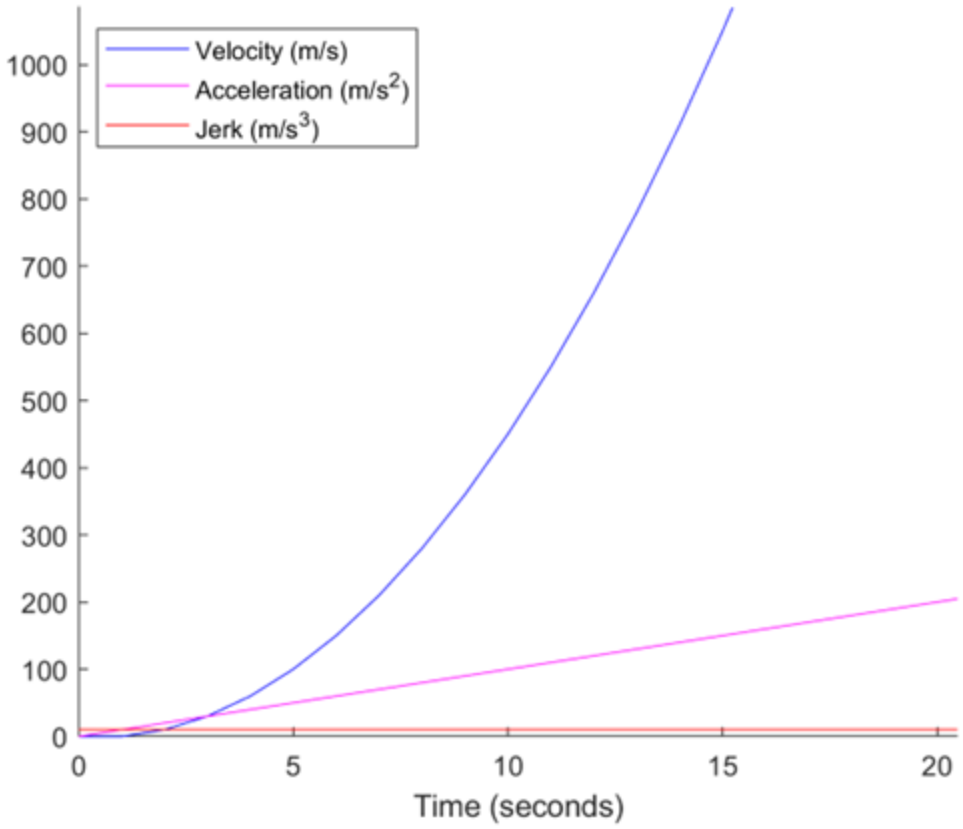


Figure 3.8: Constant jerk effect on acceleration and velocity.

multi-parameter case, an extremely simplified model is suitable. For this discussion, we will instead consider a more direct non-probabilistic approach, a basic Doppler shift over time based on velocity and acceleration, with the goal of determining the velocity and acceleration limits of our communication system. Depending on what tools our system has to handle frequency shifts, it may perform differently depending on the characteristics of the frequency shift. For example, it may be able to handle a constant frequency shift (the same as the carrier frequency offset discussed in Section 3.3.4), but will not be able to correct when the frequency shift is changing over time.

Before we can simulate this system, we must define the performance curve we

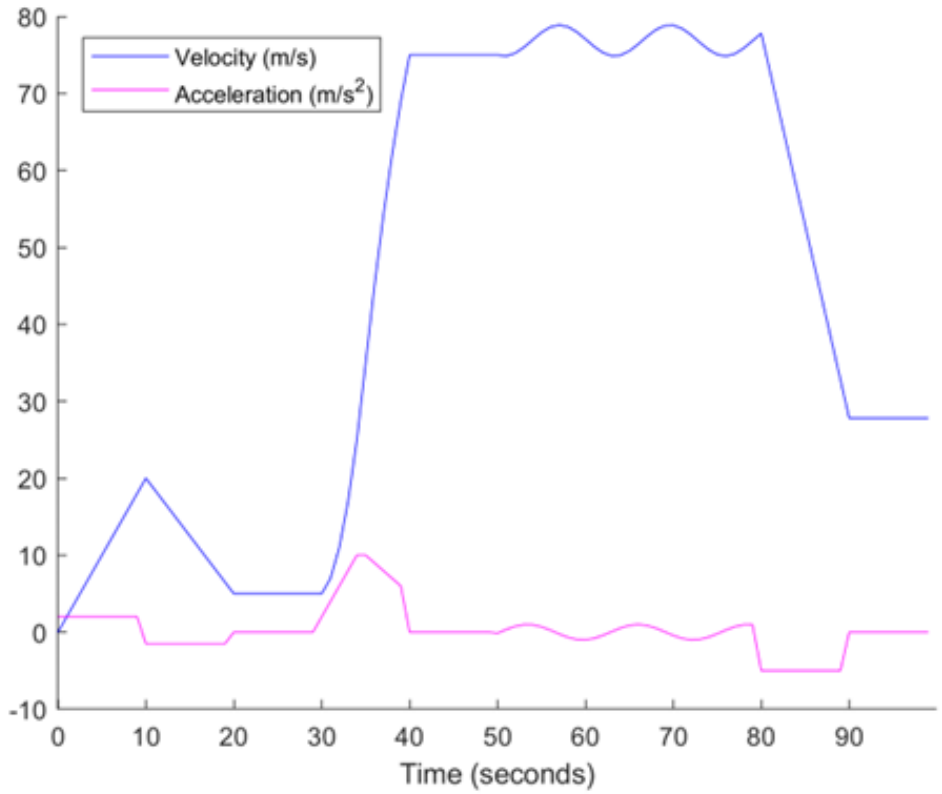


Figure 3.9: Nonlinear acceleration effect on velocity.

want to obtain through simulation. The goal is to determine how our system performs in the presence of a basic Doppler effect, which in this case is a function of both velocity and acceleration. This is the first case we have explored where the performance curve has more than one parameter.

At the top level, we need to evaluate the performance under a range of acceleration values, but this is where we encounter some complexity. Say, for example, we would like to test an acceleration value of 5m/s2. When we generate the test points, should we generate points from 0m/s to 5m/s over one second, or 5m/s to 10m/s, or some other set of values? The performance may be different for any of these sets. Because of this, we need to test a range of velocity values for each acceleration value. One approach to this would be to test a range of initial velocity values for

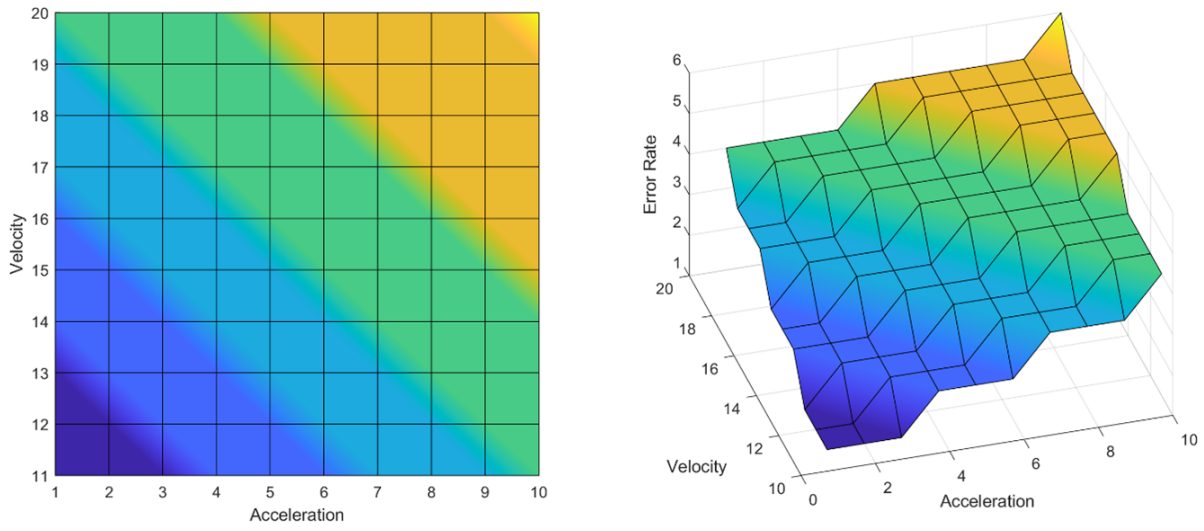
each acceleration value. Thus, our performance curve has two parameters—velocity and acceleration.

The two parameters for this simulation are not inherently bounded, so we will need to add bounds based on the physical limitations of the system. The maximum velocity will occur when the two vehicles, such as aircraft, are travelling either directly towards or directly away from each other and will peak at the combined total of maximum velocity of the two planes (and likewise for acceleration). However, due to the large number of test points required for each parameter value (as we will discuss later), simply choosing the absolute limit as a bound may not be the ideal option. We will need to select bounds based on a balancing act between simulation time and coverage. Since this channel is essentially a time varying frequency shift, the performance curve will most likely be a smooth shape with no unexpected holes. Spacing can be selected based on desired smoothness.

Test points for this channel simulation will need to be generated in two dimensions. For each acceleration parameter value on the performance curve, multiple sets of test points must be generated, one for each initial velocity within the velocity bounds. Then, the performance must be determined for each initial velocity value.

Once each set of test points have been tested, there are multiple options for viewing the results. One possibility would be to view the performance curve in three dimensions to show the performance relationship between acceleration and velocity, such as Figures 3.10. Alternatively, we could average the velocity results for each test set to get a view of the average performance at that acceleration value and plot the performance curve in the usual two-dimensional way.

Having two parameters greatly increases the time needed to evaluate this system and would only increase more if additional parameters were needed. Each parameter



(a) Top view

(b) Side view

Figure 3.10: Conceptual two-dimensional BER curve as a function of both acceleration and velocity.

will increase the number of test points multiplicatively based on the number of parameter values. For example, for a one parameter simulation with 10 parameter values, and approximately $5,000$ test points per value, there would be a total of $10 \times 5,000 = 50,000$ test points. If that same simulation had an additional parameter with 15 values, the total number would become $(10 \times 15) \times 5,000 = 750,000$ test points. Adding just one more value to the second parameter would increase the number of test points by $(10 \times 1) \times 5,000 = 50,000$. Thus, each additional parameter will severely impact simulation time, and each additional value will add multiple sets of test points to the simulation.

Even though this channel has an extremely simple model, a single Doppler shift over time based on velocity and acceleration, it may actually be a difficult channel simulation to implement. The user must select parameter bounds and spacing which cover the full performance area of interest, but as small as possible because

each additional parameter value adds not one, but many sets of test points to the simulation. This is a difficult tradeoff since the penalty for each parameter value is so high. For some cases with a large number of values, the number of test points may be prohibitively large.

Based on the above analysis, we will summarize the user difficulty of implementing this channel in Table 3.6. Note that, although the large number of test points for this channel is an issue, the root cause of this is actually the parameter bounds and spacing, and the test points themselves are simple to determine once the bounds and spacing are selected. This is in contrast to the heavy-tailed amplitude noise channel, where the selection of test points for a single parameter value is the direct cause of difficulty.

Table 3.6: Implementation difficulty of the multi-parameter channel.

Multi-Parameter Channel			
	Parameter Bounds	Parameter Spacing	Test Points
Simple			X
Moderate			
Complex	X	X	

3.4 Chapter summary

In this chapter, we proposed a definition for simulation difficulty, established unifying terminology to discuss simulation methods and complexity across a range of different channels, and finally analyzed an array of channel simulations with respect to their characteristics and simulation methods. We found that certain channels are more difficult to simulate than others due to a range of characteristics of not only the channel model itself, but also the performance curve and test points being simulated. These characteristics affect user difficulty in selecting and implementing simulation methods, simulation time, and the available methods which can be used.

The user difficulty for the analyzed channels is summarized in Table 3.7, where 1 is simple, 2 is moderate, and 3 is complex.

Table 3.7: Summary of user implementation difficulty across channels.

User Difficulty Summary	Parameter Bounds	Parameter Spacing	Test Points
Additive White Gaussian Noise	2	1	1
Uniform Phase Noise	1	1	1
Uniform Phase Noise with Holes	1	1	3
Carrier Frequency Offset	2	1	2
Heavy-Tailed Amplitude Noise	2	1	3
Multi-Parameter	3	3	1

Chapter 4

Proposed Complexity Framework

The difficulty chart in Chapter 3 shows an approximate level of complexity for implementing each aspect of a channel simulation. However, though it may give a benchmark idea of complexity, it is not practically useful for actually deciding how to simulate a given channel or choosing which techniques to consider, and it does not account for the other two difficulty aspects of simulation time and available methods.

We propose the following framework in Tables 4.1, 4.2, and 4.3 as a more practical and useful way of considering simulation complexity. There are three categories of characteristics: performance curve characteristics, test point outcome characteristics, and channel characteristics. For each characteristic, the complexity framework defines what the simple, moderate, and complex case are. It also describes what component of simulation will be affected by this characteristic, either user difficulty, simulation time, or available simulation methods. Finally, for each affected simulation component, it describes what the positive or negative effects will be.

A user could use this framework to quickly identify potential simulation challenges early in the process and identify what would need to be altered to mitigate these challenges. A channel simulation could be simplified by transforming one of the characteristics from the complex case to the moderate case. For example, one could identify bounds for a channel with previously unknown bounds, split up manually selected test points resulting in monotonic segments of outcomes to enable the threshold method, or split up independent parameters into separate simulations as much as possible to reduce the number of parameters in any one single simulation,

thus reducing overall simulation time.

Table 4.1: Complexity framework—performance curve.

Characteristic	Simple Case	Moderate Case	Complex Case	Affected Simulation Component
Number of Parameters:	Less parameters		More parameters	Time: Additional parameters increase time multiplicatively
Parameter Bounds:	Inherent bounds	Bounds based on empirical channel data or existing research	Bounds unknown	User: Bounds are required to simulate, must cover area of interest but be also minimize rare event simulation and simulation time
Parameter Spacing:	Wide spacing based on resolution preference		Narrow or strategic spacing due to performance holes	Time: Spacing affects number of parameter values required and thus simulation time, need to balance resolution with time User: Resolution is an issue if there are performance holes
Monotonicity:	Monotonic curve	Non-monotonic but smooth curve	Performance holes	Time: Affects parameter spacing

Table 4.2: Complexity framework—test point outcomes.

Characteristic	Simple Case	Moderate Case	Complex Case	Affected Simulation Component
Performance Holes:	No		Yes	Time: Increased time due to higher resolution required User: Must choose number of test points or manually select test points
Monotonicity:	Monotonic	Low order curve	High order curve or performance holes	Methods: Threshold method is possible for monotonic, may be possible in segments for a low order curve, but is likely not practical for any higher order curves
Time Variance:	Time invariant		Time varying	Time: If time varying, number of tests required for each test point value increases User: If time varying, user must choose how many times to test each point based on how the outcome varies Methods: Parallel processing is easier to implement for time invariant outcomes, if combined with memoryless characteristic then test points can be tested in any order and the threshold method may be possible
Memoryless:	Memoryless		Memory	Methods: If test point outcomes are memoryless, parallel processing is easier to implement, if combined with time invariance the threshold method may be possible

Table 4.3: Complexity framework—channel.

Characteristic	Simple Case	Moderate Case	Complex Case	Affected Simulation Component
Distribution Well-Bounded:	Strictly bounded	Bounded (ex. Exponentially)	Unbounded	Time: If unbounded, number of test points required per parameter value increases due to variance User: Strict bounds allow simple manual test point selection (assuming no performance holes) Methods: Heavy tails make some importance sampling techniques, which mitigate the time required for low BER regions, impossible
Non-Random Time-Varying Effects:	No		Yes	User: Must generate channel effects strategically across time Methods: Parallel processing more difficult to implement

Chapter 5

Conclusion

In this thesis, we proposed a more general understanding of simulation methods as they relate to channel characteristics. To identify these characteristics, we began by reviewing the implementation of channel simulations in Chapter 2. In Chapter 3, we applied this knowledge to analyze a set of channel simulations and identify characteristics which affect simulation difficulty. Finally, in Chapter 4, we compiled the findings into a practical framework to be used during the selection and implementation of simulation methods. The complexity framework can be used to guide the selection of simulation methods, identify simulation challenges early on, and identify what would need to be altered to mitigate these challenges

Without a general understanding of how channel characteristics relate to simulation methods, determining simulation methods often requires research into existing work covering the exact channel of interest, and any differences between the channel in the existing work and the channel of interest can lead to uncertainty regarding the validity of applying the same simulation methods. The complexity framework proposed in this thesis could allow the use of existing works which consider different channels or scenarios but share certain characteristics. Most importantly, this understanding of the underlying characteristics which make a simulation method possible allows for increased confidence in the validity of a simulation method for a channel of interest.

5.1 Future work

There are many potential avenues for future work based on this research. First, it is of interest to identify additional complexity characteristics. Channels have many characteristics not considered in this thesis, and these characteristics could be useful in further categorizing channels and their simulation methods. Second, it may be useful to identify additional existing simulation methods and their correlations to the found characteristics. In particular, there are many simulation simplification methods and methods of shortneing the simulation process which are not discussed here. Finally, it is of interest to expand this analysis to higher order modulation schemes. This thesis was limited to binary communication systems, but some of the principles may be applicable to higher order systems as well.

Bibliography

[1] D. P. Kroese, *Handbook of Monte Carlo methods*. Wiley series in probability and statistics ; 706, Hoboken, N.J: Wiley, 2011.

[2] M. Viswanathan, "Simulate additive white gaussian noise (AWGN) channel." `https://www.gaussianwaves.com/2015/06/how-to-generate-awgn-noise-in-matlaboctave-without-using-in-built-awgn-function`, 2015.

[3] Z. Chenggong, C. Xi, and H. Zhen, "A comprehensive analysis on Doppler frequency and Doppler frequency rate characterization for GNSS receivers," in *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, pp. 2606–2610, 2016.

[4] A. Piemontese, G. Colavolpe, and T. Eriksson, "Phase noise in communication systems: from measures to models," *CoRR*, vol. abs/2104.07264, 2021.

[5] A. Behlouli, P. Combeau, and L. Aveneau, "MCMC Methods for Realistic Indoor Wireless Optical Channels Simulation," *Journal of lightwave technology*, vol. 35, no. 9, pp. 1575–1587, 2017. Place: New York Publisher: IEEE.

[6] M. Patzold, A. Szczepanski, and N. Youssef, "Methods for modeling of specified and measured multipath power-delay profiles," *IEEE transactions on vehicular technology*, vol. 51, no. 5, pp. 978–988, 2002. Place: New York, NY Publisher: IEEE.

[7] A. Minja and V. Senk, "Quasi-Analytical Simulation Method for Estimating the Error Probability of Star Domain Decoders," *IEEE transactions on communications*, vol. 67, no. 5, pp. 3101–3113, 2019. Place: New York Publisher: IEEE.

[8] T. Sakai and K. Shibata, "A study on quick simulation for estimation of low FER of LDPC codes," *2009 IEEE 9th Malaysia International Conference on Communications (MICC), Communications (MICC), 2009 IEEE 9th Malaysia International Conference on*, pp. 468–473, Dec. 2009.

[9] H. A. Ahmed, A. I. Sulyman, and H. S. Hassanein, "Bit error rate performance of orthogonal frequency-division multiplexing relaying systems with high power amplifiers and Doppler effects," *Wireless communications and mobile computing*, vol. 13, no. 8, pp. 734–744, 2013. Publisher: Blackwell Publishing Ltd.

[10] P. Zhao, X. Wang, K. Zhang, Y. Jin, and G. Zheng, "Doppler Modeling and Simulation of Train-to-Train Communication in Metro Tunnel Environment.," *Sensors (Basel, Switzerland)*, vol. 22, June 2022. Place: Switzerland Publisher: MDPI.

[11] Michel C. Jeruchim, Philip Balaban, and K. Sam Shanmugan, *Simulation of Communication Systems : Modeling, Methodology and Techniques*, vol. 2nd ed of *Information Technology: Transmission, Processing, and Storage*. New York: Springer, 2000.

[12] K. Shanmugan, T. Rappaport, W. Tranter, and K. Kosbar, *Principles of Communication Systems Simulation with Wireless Applications. [electronic resource]*. Pearson, 1st edition ed., 2003.

[13] R. . . Feng, 2), C.-X. . . Wang, 3), J. . . Huang, 3), X. . . Gao, 3), S. . . . Salous, and H. . . . Haas, "Classification and Comparison of Massive MIMO Propagation Channel Models," *IEEE Internet of Things Journal*, vol. 9, pp. 23452–23471, Dec. 2022. Num Pages: 23471 Publisher: Institute of Electrical and Electronics Engineers Inc. 23452.

[14] Z. K. Adeyemo, D. O. Akande, F. K. Ojo, and H. O. Raji, "Comparative Evaluation of Fading Channel Model Selection for Mobile Wireless Transmission System," *International Journal of Wireless & Mobile Networks*, vol. 4, no. 6, pp. 127–138, 2012.

[15] J. Proakis, M. Salehi, and G. Bauch, *Contemporary Communication Systems Using MATLAB and Simulink*. BookWare companion series, Thomson–Brooks/Cole, 2004.

[16] Arie Dickman, *Verified Signal Processing Algorithms in MATLAB and C : Advised by Israel Greiss*. Cham, Switzerland: Springer, 2022.

[17] S. Zekavat, "Wireless comm notes." Lecture Notes.

[18] N. Jovanovic, "BPSK system modeled and benchmarked against BER (SNR)." `https://github.com/etfovac/bpsk-ber/releases/tag/v1.2`, 2021.

[19] AMSI, "Normal distribution." `https://amsi.org.au/ESA_Senior_Years/SeniorTopic4/4f/4f_2content_3.html`.

[20] F. G. Stremler, *Introduction to communication systems*. Addison-Wesley series in electrical engineering, Reading, Mass: Addison-Wesley Pub. Co., 1977.

[21] J. G. Proakis, *Digital communications*. Boston: McGraw-Hill, 5th ed. ed., 2008. Publication Title: Digital communications.

[22] Wlodzimierz Bryc, *The Normal Distribution : Characterizations with Applications*, vol. 1995 of *Lecture Notes in Statistics*. New York, NY: Springer, 2012. Issue: Vol. 100.

[23] David Tse and Pramod Viswanath, *Fundamentals of Wireless Communication*. Cambridge: Cambridge University Press, 2005.

[24] A. Leon-Garcia, *Probability, Statistics, and Random Processes for Electrical Engineering*. Pearson/Prentice Hall, 2008.

[25] W. Anggoro, *C++ Data Structures and Algorithms. [electronic resource]*. Packt Publishing, 1st edition ed., 2018.

[26] R. Sedgewick and K. Wayne, *Computer Science: An Interdisciplinary Approach. [electronic resource]*. Addison-Wesley Professional, 1st edition ed., 2016.

[27] A. S. Mohammed, E. Amrahov, and F. V. Çelebi, "Interpolated binary search: An efficient hybrid search algorithm on ordered datasets," *Engineering Science and Technology, an International Journal*, vol. 24, pp. 1072–1079, Oct. 2021. Publisher: Elsevier B.V.

[28] J. L. Bentley and A. C.-C. Yao, "An Almost Optimal Algorithm for Unbounded Searching," *Inform. Proc. Lett.*, vol. 5, pp. 82–87, 1976.

[29] "Steady-State Properties of of GI/G/1," in *Applied Probability and Queues* (S. Asmussen, ed.), pp. 266–301, New York, NY: Springer New York, 2003.

[30] H. M. Hall, "A new model for impulsive phenomena: Application to atmospheric-noise communication channels," Tech. Rep. SEL-66-052, Stanford, Aug. 1996.

[31] H. Hao, H. Wang, L. Chen, W. Jun, L. Qiu, and L. Rong, "Initial results from SQUID sensor: Analysis and modeling for the ELF/VLF atmospheric noise," *Sensors*, vol. 17, p. 371, Feb. 2017.

[32] D. A. Chrissan and A. C. Fraser-Smith, "A comparison of low-frequency radio noise amplitude probability distribution models," *Radio Science*, vol. 35, no. 1, pp. 195–208, 2000.

[33] J. Omura and P. Shaft, "Modem Performance in VLF Atmospheric Noise," *IEEE Transactions on Communication Technology*, vol. 19, no. 5, pp. 659–668, 1971.

[34] J. K. Omura, "Statistical Analysis of LF/VLF Communication Modems," tech. rep., 1969.

[35] Bronius Grigelionis, *Student's T-Distribution and Related Stochastic Processes*. SpringerBriefs in Statistics, Heidelberg: Springer, 2013.

[36] S. Hurst, "The characteristic function of the student-t distribution, financial mathematics," *Statistics Research, SRR044-95*, 1995.

[37] E. W. Weisstein, "Student's t-distribution." `https://mathworld.wolfram.com/Studentst-Distribution.html`. From MathWorld, A Wolfram Web Resource.

[38] H. Yeganegi, P. Salami, and m. r. Daliri, "A Template-Based Sequential Algorithm for Online Clustering of Spikes in Extracellular Recordings," *Cognitive Computation*, vol. 12, May 2020.

[39] R. W. Bailey, "Polar Generation of Random Variates with the t-Distribution," *Mathematics of Computation*, vol. 62, no. 206, pp. 779–781, 1994. Publisher: American Mathematical Society.

[40] G. Rubino and B. Tuffin, *Rare Event Simulation Using Monte Carlo Methods*. Hoboken, UNITED KINGDOM: John Wiley & Sons, Incorporated, 2009.

[41] P. Smith, M. Shafi, and Hongsheng Gao, "Quick simulation: a review of importance sampling techniques in communications systems," *IEEE Journal on Selected Areas in Communications, Selected Areas in Communications, IEEE Journal on, IEEE J. Select. Areas Commun.*, vol. 15, pp. 597–613, May 1997. Place: USA Publisher: IEEE.

[42] H. Liu, *Rare events, heavy tails, and simulation*. Ph.D., University of Colorado at Boulder, Ann Arbor, 2006. ISBN: 978-0-542-94258-7 Publication Title: ProQuest Dissertations and Theses 3239435.

[43] S. Asmussen, K. Binswanger, and B. Højgaard, "Rare Events Simulation for Heavy-Tailed Distributions," *Bernoulli*, vol. 6, no. 2, pp. 303–322, 2000. Publisher: International Statistical Institute (ISI) and Bernoulli Society for Mathematical Statistics and Probability.

[44] S. Asmussen and D. Kroese, "Improved Algorithms for Rare Event Simulation With Heavy Tails," *Advances in Applied Probability*, vol. 38, June 2006.

[45] M. Alqahtani and T. Grafke, "Instantons for rare events in heavy-tailed distributions," *Journal of Physics A: Mathematical and Theoretical*, vol. 54, p. 175001, Apr. 2021. Publisher: IOP Publishing.

[46] L. Rojas Nandayapa, "A review of conditional rare event simulation for tail probabilities of heavy tailed random variables," *Boletín de la Sociedad Matemática Mexicana*, pp. 159–182, Oct. 2013.

[47] M. Bognar, "Student's t-distribution applet." `https://homepage.divms.uiowa.edu/~mbognar/applets/t.html`, 2021.

[48] P. Hirschausen, L. Davis, D. Haley, and K. Lever, "Identifying key design parameters for Monte Carlo simulation of Doppler spread channels," *2014 Australian Communications Theory Workshop (AusCTW), Communications Theory Workshop (AusCTW), 2014 Australian*, pp. 33–38, Feb. 2014. Publisher: IEEE.

[49] William C. Jakes, *Microwave Mobile Communications.* John Wiley & Sons, Jan. 1994.

[50] P. Dent, G. E. Bottomley, and T. Croft, "Jakes fading model revisited," *Electronics Letters*, vol. 29, pp. 1162–1163(1), June 1993. Publisher: Institution of Engineering and Technology.

[51] J. Gibson, *Mobile Communications Handbook, 3rd Edition. [electronic resource].* CRC Press, 3rd edition ed., 2017.

Appendix A

Comparison of Test Point Generation Methods for Uniform Phase Noise

A.1 rand_vs_manual.m

```
clear all
close all

M = 2; % Modulation order
num_reps = 100; %repeat simulation num_reps times

%% Random Generation
N = 100000; %number of test points

%get cluster of estimates
BER_rand = zeros(1,num_reps);
for k = 1:num_reps
    txBits = randi([0 M-1],N,1);
    txSig = pskmod(txBits,M,0);
    theta = -pi + (2*pi)*rand(N,1);
    rxSig = txSig.*exp(1i.*theta); %apply random phase shift to each sample
    rxBits = pskdemod(rxSig,M,0);
    error_count = 0;
    for i = 1:N
        if txBits(i) ~= rxBits(i)
            error_count = error_count + 1;
        end
    end
    BER_rand(k) = error_count/N;
end

%% Manual Generation
N = 100; %number of test points

%get cluster of estimates
BER_manual = zeros(1,num_reps);
for k = 1:num_reps
    txBits = randi([0 M-1],N,1);
    txSig = pskmod(txBits,M,0);
    theta_manual = (-pi:(2*pi)/(N-1):pi)';
    rxSig = txSig.*exp(1i.*theta_manual); %apply phase shift to each sample
```

```
    rxBits = pskdemod(rxSig,M,0);
    error_count = 0;
    for i = 1:N
        if txBits(i) ~= rxBits(i)
            error_count = error_count + 1;
        end
    end
    BER_manual(k) = error_count/N;
end

figure()
plot(BER_rand);
hold on
plot(BER_manual);
title("BER estimates: 100,000 random test points vs. 100 manual test
    points")
ylabel("BER Estimate")
xlabel("Simulation Instance")
legend("Random test point BER estimation", "manual test point BER
    estimation")
```

A.2 Results

For 100 instances of this simulation (with $100,000$ random test points and 100 manual test points in each simulation instance), manual generation was found to obtain a more accurate estimate with less test points as shown in Figure A.1

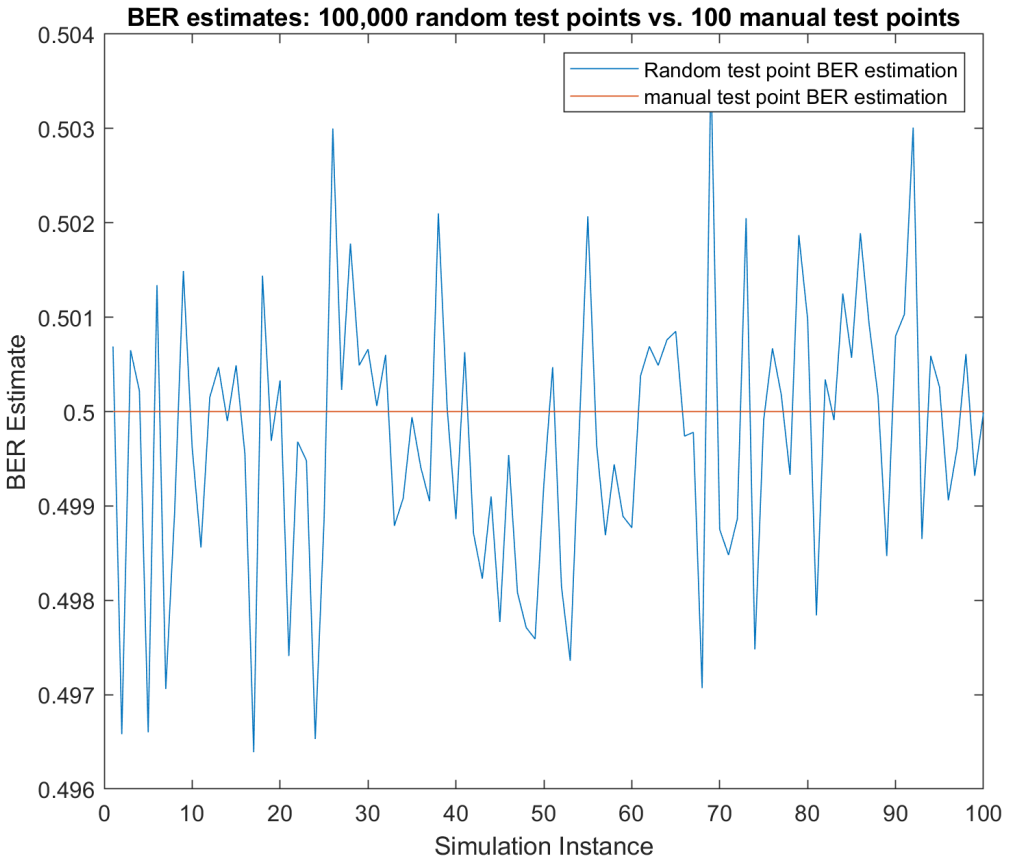


Figure A.1: Comparison of BER estimates: 100,000 random test points vs. 100 manual test points.

Appendix B

Low BER Simulation Time

B.1 awgn_ber.m

```
clear all
close all

M = 2; % Modulation order
EbN0 = (0:10);
ber_vec = zeros(1,length(EbN0));
bits_per_frame = 100;
req_errors = 100; %Run until 100 bit errors are found

timer = zeros(1,length(EbN0));
test_point_totals = zeros(1,length(EbN0));
for(k = 1:length(EbN0))%Go through the specified Eb/No range
    num_errors = 0;
    num_bits = 0;
    tic
    while(num_errors < 10000)
        txBits = randi([0 M-1],bits_per_frame,1);
        txSig = pskmod(txBits,M,0);
        rxSig = awgn(txSig,EbN0(k),'measured');
        rxBits = pskdemod(rxSig,M,0);
        for i = 1:bits_per_frame
            if txBits(i) ~= rxBits(i)
                num_errors = num_errors + 1;
            end
        end
        num_bits = num_bits + bits_per_frame;
    end
    timer(k) = toc;
    test_point_totals(k) = num_bits;
    %Add BER for this Eb/N0 value to the BER Vector
    ber_vec(k) = num_errors/num_bits;
end

berQ = berawgn(EbN0,'psk',M,'nondiff');

semilogy(EbN0,berQ)
hold on
semilogy(EbN0,ber_vec)
```

```
xlabel('Eb/N0 (dB)')
ylabel('BER')
legend('BPSK Theoretical','BPSK Simulated')
title("Bit error rate")
grid

figure()
bar(EbN0,timer)
xlabel('Eb/N0 (dB)')
ylabel('Run Time (seconds)')
title("Run time until 10,000 bit errors")
grid

figure()
bar(EbN0,test_point_totals)
xlabel('Eb/N0 (dB)')
ylabel('Test Points')
title("Test points required for 10,000 bit errors")
grid

figure();
semilogy(EbN0,test_point_totals);
xlabel('Eb/N0 (dB)')
ylabel('Test Points')
title("Test points required for 10,000 bit errors")
grid

figure();
semilogy(EbN0,test_point_totals/1000000);
xlabel('Eb/N0 (dB)')
ylabel('Test Points (Scaled) and BER')
title("Test points required for 10,000 bit errors compared to BER")
grid
hold on
semilogy(EbN0,ber_vec)
legend('Test Points (Scaled)','BER')
```

B.2 Results

This simulation produced the BER curve of a BPSK system in AWGN for SNR values $[0, 10]$dB. We chose to run each SNR value until we had obtained $10,000$ errors. The resulting BER curve is shown in Figure B.1 along with the theoretical curve for this channel.

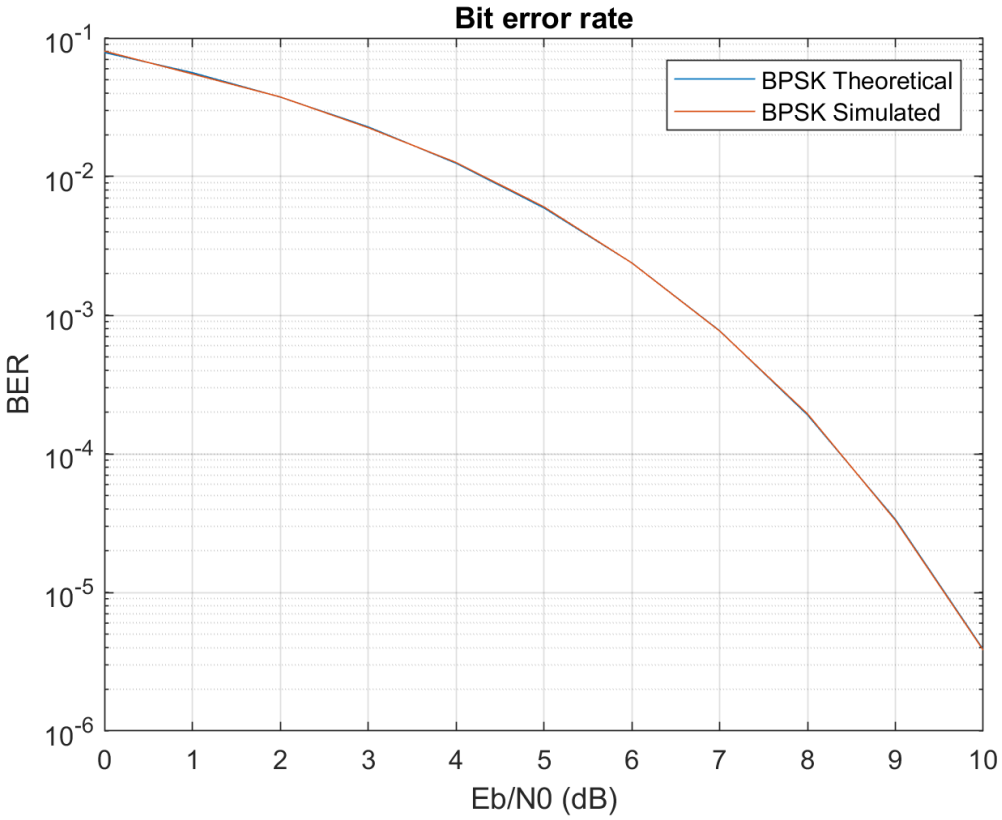


Figure B.1: BPSK BER in an AWGN channel.

The time required to obtain $10,000$ errors increased rapidly for each SNR value. Figures B.2 and B.3 show the number of test points required to obtain $10,000$ errors and the time required to run those test points, respectively.

When plotted on a logarithmic scale as shown in Figures B.4 and B.5, it is clear that the number of test points required is inversely proportional to the BER.

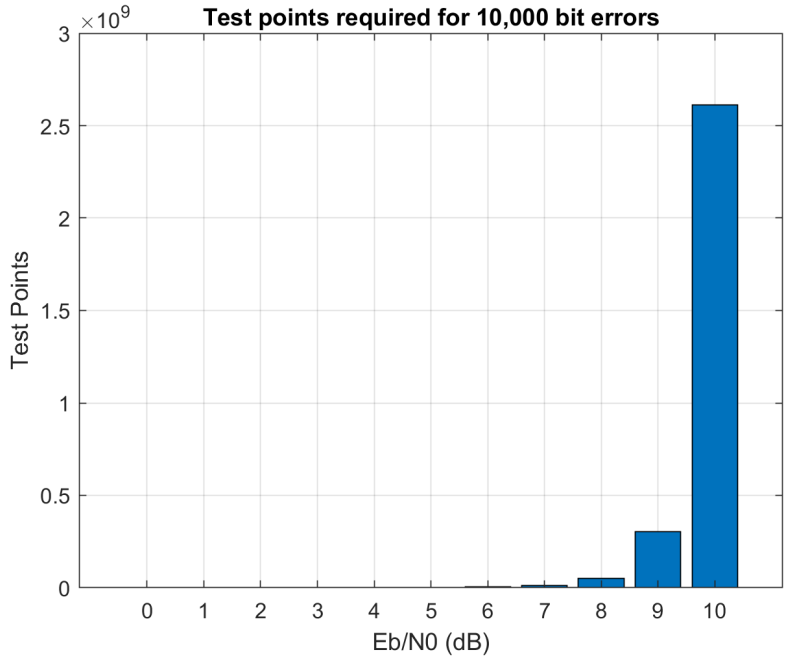


Figure B.2: Test points required for 10,000 bit errors.

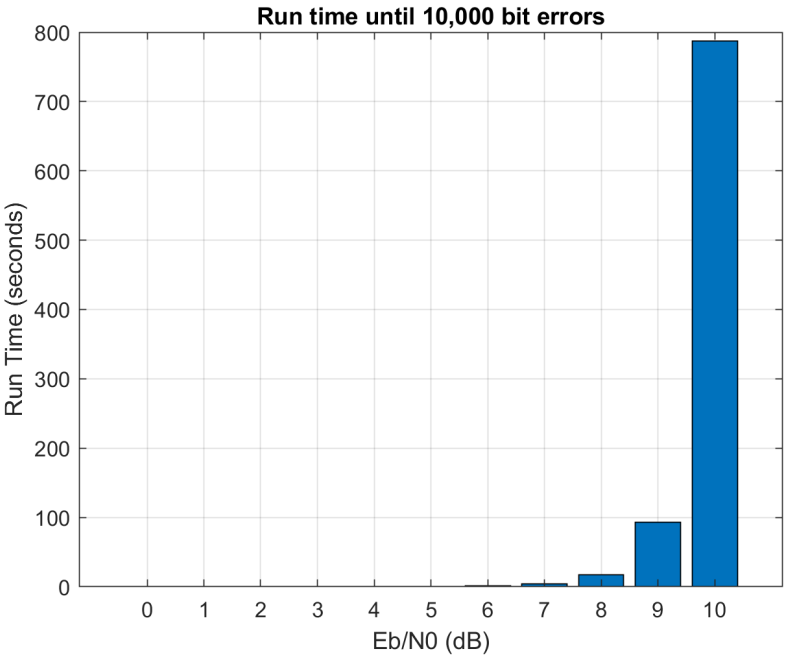


Figure B.3: Run time until 10,000 bit errors.

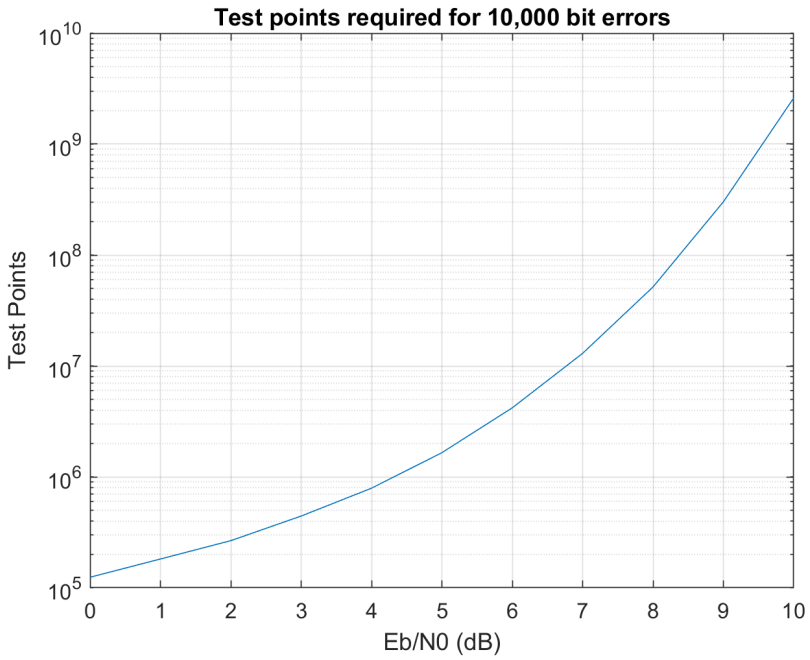


Figure B.4: Test points required for 10,000 bit errors (log scale).

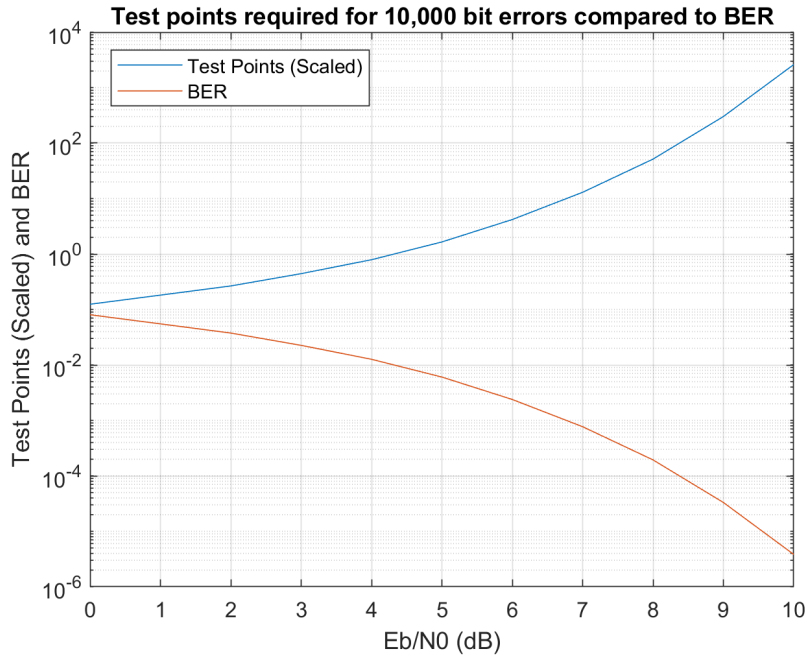


Figure B.5: Test points required for 10,000 bit errors compared to BER.

Appendix C

Velocity and Acceleration Curves

C.1 velocity_effects.m

```
t = 0:1:99;

jerk = 10*ones(1,length(t));

a(1) = 0;
for i=2:100
    a(i) = a(i-1) + jerk(i-1);
end

v(1) = 0;
for i=2:100
    v(i) = v(i-1) + a(i-1);
end

figure();
hold on
plot(t,v,'b');
plot(t,a,'m');
plot(t,jerk,'r');

%% shifting

t = 0:1:99;

a = [2*ones(1,10) -1.5*ones(1,10) 0*ones(1,10) 2:2:10 10:-1:6 0*ones(1,10)
    sin(t(51:80)*1/2) -5*ones(1,10) 0*ones(1,10)];

v(1) = 0;
for i=2:100
    v(i) = v(i-1) + a(i-1);
end

figure();
hold on
plot(t,v,'b');
plot(t,a,'m');
```